Transforming Indoor Air Quality Management at Butterfly Air through AI: Neural Prophet Forecasting and RAG-based AI Agent Integration

Links:

Project Code (request access):

`https://github.com/es1221/Masters-code.git`

Video Demo: `https://youtu.be/wuJruw6eamA`

Graph figures: `https://github.com/es1221/Matsters_images.git`

Master's Dissertation

Eesha Sondhi

CID: 02019308

Supervisor: David Boyle

Abstract

This dissertation addresses the critical gap between Indoor Air Quality (IAQ) monitoring and proactive management through an integrated forecasting and information retrieval system for Butterfly Air, a UK-based IAQ monitoring company. Poor IAQ significantly impacts health, cognitive function and productivity, yet current solutions focus on monitoring, not prediction. The project develops two complementary components: a Neural Prophet forecasting model for 12-hour IAQ prediction, and a Retrieval-Augmented Generation (RAG) AI Assistant for intuitive information access. The forecasting system achieves 94.8% average accuracy across all IAQ parameters, exceeding the 90% target defined by Butterfly, whilst the RAG system demonstrates near-perfect retrieval metrics after iterative improvements. The implementation enables clients to anticipate air quality issues before they affect occupants, addressing a critical problem where current solutions only react after poor air quality has already impacted health and productivity. This has the potential to transform reactive monitoring into proactive management, whilst remaining financially viable for Butterfly Air at scale.

Contents

1 Introduction

Indoor Air Quality (IAQ) has emerged as a critical factor in building management affecting occupant health, cognitive function and productivity. Current solutions focus on monitoring rather than prediction, creating a gap between data collection and actionable insight. This project addresses this gap by developing an integrated forecasting and information retrieval system for Butterfly Air, a UK-based IAQ monitoring company.

Studies show CO_2 levels above 1000 ppm can reduce cognitive function by up to 15% (Allen et al., 2016), while elevated $PM_{2.5}$ increases cardiovascular risks (Carre & Hill, 2021). Poor IAQ costs UK businesses £13.2 billion annually through decreased productivity and increased absenteeism (Royal College of Physicians, 2020).

Without predictive capabilities, building managers can only respond to poor air quality after it occurs. By forecasting IAQ parameters 12 hours in advance, this project enables ventilation adjustments before problems arise, maintaining optimal conditions continuously.

Current IAQ solutions in the commercial market, including those from competitors like Kaiterra (Kaiterra, 2024) and Awair (Awair, 2023), focus primarily on real-time monitoring without predictive capabilities. Research efforts have attempted to address this limitation, with recent studies achieving forecasting accuracies between 85-88% using various machine learning approaches (Liu et al., 2023; Wang & Shen, 2022). However, these solutions often face implementation challenges in real-world building management systems due to computational complexity, lack of user-friendly interfaces, and limited integration with existing building management workflows.

Butterfly Air's existing platform provides real-time IAQ monitoring but lacks the predictive capabilities and accessible information retrieval that would transform data into actionable insights for clients. This project aims to enhance their platform with two complementary components: a machine learning forecasting model for accurate 12-hour IAQ predictions, and an intelligent AI Assistant for intuitive access to IAQ information. The integration of these components represents a novel approach in the IAQ field, moving beyond monitoring to enable proactive management.

1.1 Research Questions

The project addresses three key research questions:

1. Can Neural Prophet with autoregression achieve accurate (>90%) forecasting for diverse IAQ parameters while maintaining computational efficiency suitable for commercial deployment?

2. Can a Retrieval-Augmented Generation (RAG) system provide factually accurate responses about IAQ data and contextually appropriate recommendations?

3. How can these technical components be integrated to deliver tangible value for both Butterfly Air and its clients?

1.2 Success Criteria

Success will be determined by three criteria established in consultation with stakeholders:

1. The ML model must achieve >90% accuracy across all IAQ parameters.

2. The AI Assistant must provide factually correct responses with no hallucinations, as defined by Butterfly Air's CEO.

3. The solution must provide competitive differentiation for Butterfly Air within the IAQ market while remaining financially viable at scale.

The key contribution is an integrated AI system bridging the gap between data collection and action by combining accurate forecasting with accessible information retrieval, advancing both Butterfly Air's commercial offering and the wider IAQ field with this novel approach.

2 Related Work

2.1 Indoor Air Quality Monitoring and Prediction

Indoor Air Quality (IAQ) monitoring has evolved significantly in recent years, with smart sensor technology enabling more comprehensive data collection. Carre and Hill (2021) established that poor IAQ contributes to numerous health issues, including respiratory conditions, cognitive impairment and reduced productivity, driving increased interest in monitoring solutions. Current commercial systems from manufacturers like Kaiterra and Awair provide real-time monitoring but lack predictive capabilities, creating a significant technological gap (Zhang et al., 2022). Recent research has attempted to address this gap. Liu et al. (2023) developed LSTM neural networks for CO_2 prediction, achieving >85% accuracy over 6-hour forecasts. Similarly, Wang and Shen (2022) applied ensemble methods combining gradient boosting with ARIMA models for IAQ parameter prediction, reporting >88% accuracy for temperature and humidity but lower performance (>76%) for particulate matter. Most notably, Rahman et al. (2024) demonstrated that comprehensive IAQ prediction requires models that can:

1. Handle multivariate data across diverse parameters

2. Capture both linear and non-linear relationships

3. Process temporal dependencies at different time scales

4. Adapt to varying environmental conditions

2.2 Predictive Modelling Approaches for Time Series Data

Time series forecasting involves predicting future values based on previously observed data points ordered by time. This field has witnessed significant advancements with the emergence of hybrid statistical-neural models. Traditional approaches such as ARIMA (AutoRegressive Integrated Moving Average) and exponential smoothing models have demonstrated effectiveness for linear patterns but struggle with complex non-linear relationships (Box et al., 2019).

The Facebook Prophet model, developed by Taylor and Letham (2018), represented a breakthrough by decomposing time series into trend, seasonality and holiday components using an additive model framework as shown in Equation 1:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t \qquad (1)$$

Where:

- $g(t)$ represents the trend component (the overall direction)
- $s(t)$ captures seasonality (recurring patterns at fixed intervals)
- $h(t)$ accounts for holiday effects (irregular events)
- ϵ_t is the error term

It's important to note that Prophet is fundamentally a statistical model using Bayesian curve-fitting techniques rather than a machine learning approach. This distinction is significant because Prophet identifies patterns using predefined structural components and mathematical formulas rather than learning patterns directly from the data. While this statistical approach works well for data with clear seasonal patterns and trends, it struggles with complex non-linear relationships and strong temporal dependencies, like that characterise IAQ parameters.

2.3 Neural Prophet: Advancing the State of the Art

Neural Prophet (Triebe et al., 2021) represents a significant advancement by incorporating neural network capabilities into time series forecasting. Unlike standard Prophet, Neural Prophet is fundamentally a machine learning model that learns patterns directly from data rather than simply fitting predefined components. This machine learning foundation enables it to capture complex non-linear relationships even before adding autoregression.

The model extends Prophet's conceptual architecture but implements it through a PyTorch neural network framework, as shown in Equation 2:

$$y_t = g(t) + s(t) + h(t) + \sum_{i=1}^{AR(p)} \beta_i \cdot y_{t-i} + \epsilon_t \quad (2)$$

Where the additional term $\sum_{i=1}^{AR(p)} \beta_i \cdot y_{t-i}$ represents the autoregressive component of order p.

The autoregressive component further enhances Neural Prophet's capabilities by focusing on short-term temporal dependencies. In an autoregressive (AR) model, the future value is predicted based on a combination of its past values, essentially "zooming in" on recent data points that often have the strongest influence on immediate future values. The number of past values considered is defined by the parameter `n_lags`.

For example, if you have an AR model with `n_lags` = 3, it predicts the next time step (y_{t+1}) based on the last 3 time steps (y_t, y_{t-1}, y_{t-2}). Each past value is weighted by a learned coefficient (β_i) that determines its influence on the prediction.

This autoregressive component is particularly valuable for IAQ prediction because:

- IAQ parameters exhibit strong temporal dependencies where future values depend on recent measurements.
- The relationship between consecutive measurements often follows patterns that can be captured through weighted historical values.
- Autoregression allows the model to incorporate feedback mechanisms inherent in indoor environments.

Neural Prophet's implementation of machine learning with autoregression within a PyTorch framework provides three key advantages over alternative approaches:

- **Flexibility:** The architecture combines the interpretability of Prophet's decomposition approach with the expressiveness of neural networks.
- **Scalability:** Unlike pure deep learning solutions (e.g., RNNs, LSTMs), Neural Prophet requires less training data and computational resources.
- **Interpretability:** It maintains the component-wise decomposition while adding autoregressive capabilities, enabling better analysis of forecasting factors.

The selection of Neural Prophet over alternatives such as DeepAR (Salinas et al., 2020) or N-BEATS (Oreshkin et al., 2019) was driven by its balance of predictive power and deployment practicality. While deep learning models may offer marginally higher theoretical accuracy, they present significant scaling challenges for Butterfly Air's use case. With each client requiring multiple fine-tuned models (one per IAQ metric), Neural Prophet offers an optimal balance, delivering excellent predictive accuracy (average of 94.8% from the experimental results) without the computational overhead that would compromise commercial scalability in production environments.

2.4 RAG Systems for Domain-Specific Applications

Retrieval-Augmented Generation (RAG) systems combine large language models with external knowledge databases to produce more factual and contextually relevant responses. Unlike traditional AI systems that rely solely on pre-trained knowledge, RAG systems first retrieve relevant information from external sources before generating responses. This approach functions similarly to how a human might consult reference materials before answering a complex question.

Karpukhin et al. (2020) established key foundations for RAG systems through their Dense Passage Retrieval approach, which transforms both questions and potential answers into numerical vector representations (embeddings) that capture semantic meaning. This embedding-based approach significantly outperforms traditional keyword searches for complex queries by understanding conceptual relationships rather than just matching terms. Building on this work, Lewis et al. (2021) further developed the RAG framework, demonstrating that by retrieving relevant documents first and then generating answers based on this contextual information, AI systems could produce responses that are both more accurate and better supported by evidence. This approach has proven particularly valuable in specialist domains where factual precision is crucial.

In building management and environmental monitoring contexts, RAG applications remain underexplored. The most relevant precedent comes from Chen and Zhao (2023), who applied a basic RAG architecture to building efficiency diagnostics. However, their implementation lacked the multi-vector store design (separating different types of knowledge into distinct databases) and chunking optimisations (breaking information into more searchable segments)

that this project implements to improve retrieval precision and accuracy for a personalised AI Agent.

2.5 Positioning Within the Current Research Landscape

This work advances the state of the art in two key dimensions:

- **IAQ Prediction:** By demonstrating that Neural Prophet with autoregression can achieve **94.8% accuracy** across diverse IAQ parameters, exceeding previous related work benchmarks of 85–88% while maintaining computational efficiency suitable for deployment in building management systems.
- **Domain-Specific AI Agent:** By establishing an effective architecture for IAQ-specific Retrieval-Augmented Generation (RAG) systems with optimised vector stores and data preprocessing approaches. These enable accurate factual responses about historical measurements and contextually appropriate recommendations.

This dual contribution addresses a critical gap in the current literature: the integration of accurate predictive modelling with accessible, context-aware interfaces that transform technical predictions into actionable insights for building managers and occupants.

3 Methodology

This project involved developing two complementary components: an accurate machine learning forecasting model for IAQ parameters, and an intelligent AI Assistant to deliver IAQ related information in a user-friendly manner. This section details the methodological approaches used for both components.

3.1 Machine Learning Forecasting Approach

3.1.1 Data Source and Collection Environment

The data used in this study was collected from a Morpho device, a specialised IAQ monitoring system developed by Butterfly Air, installed in a client office building in London. The Morpho uses multiple sensors to gather comprehensive IAQ data, including an NDIR sensor for CO_2 measurements, an OPC-R2 Optical Particle Counter for particulate matter ($PM_{2.5}$, PM_{10}), a metal oxide sensor (MOX) for volatile organic compounds (VOC), and an electrochemical sensor for formaldehyde (HCHO). Temperature and humidity sensors are also integrated into the system, measuring these parameters are critical inputs for the prediction model. All sensors work together to provide a holistic view of indoor air conditions, with data transmitted to Butterfly Air's cloud platform where it becomes accessible for analysis and model training via their API. This particular client's data was chosen, due to them being Butterfly's longest-standing client with continuous measurements, providing access to over 6 months worth of uninterrupted data for this project.

3.1.2 Model Selection and Implementation Overview

The study began with the Neural Prophet model, a time-series forecasting model that combines the strengths of traditional statistical methods with modern neural

networks. Unlike standard statistical models that follow predefined patterns, Neural Prophet can learn complex relationships directly from data. The methodology involved:

1. Retrieving one month of historical IAQ data (temperature, humidity, CO_2, $PM_{2.5}$, PM_{10}, HCHO, VOC) from the client API.
2. Preprocessing the data to handle missing values and prepare it for time series forecasting.
3. Training individual Neural Prophet models for each IAQ metric.
4. Implementing unrolling prediction to generate 12-hour forecasts.
5. Evaluating model performance using standard accuracy metrics: MAE, RMSE, and MAPE.
6. Performing k-fold cross-validation for in-depth evaluation.
7. Fine-tuning model parameters based on cross-validation results.

3.1.3 Data Preprocessing

Algorithm 1 IAQ Data Preprocessing

Input: Raw IAQ data D from Butterfly Air API
Output: Preprocessed dataset D_p suitable for Neural Prophet

function PREPROCESS_DATA(D)
  for each record r in D **do**
    Parse datetime: r.timestamp $\leftarrow$ ISO8601_TO_DATETIME(r.dateTime)
    Extract measurements: r.value $\leftarrow$ r.measurements[parameter]
    if r.value is missing **then**
      r.value $\leftarrow$ INTERPOLATE(previous_value, next_value)
  $D_p \leftarrow$ empty dataset
  D_p.ds $\leftarrow$ all timestamps from D
  D_p.y $\leftarrow$ all values from D
  return D_p

Algorithm 1 outlines the IAQ data preprocessing pipeline after retrieving one month of hourly data. It transforms raw sensor data into a format suitable for time series analysis. The process begins by parsing timestamps from ISO 8601 format and extracting relevant parameter measurements (temperature, humidity, CO_2, etc.) from each record. Missing values, which commonly occur in sensor networks due to connectivity issues or maintenance periods, are addressed through interpolation to ensure data continuity. The algorithm produces a clean, two-column dataset with standardised timestamps (`ds`) and corresponding parameter values (`y`), which serves as the foundation for all subsequent modelling steps.

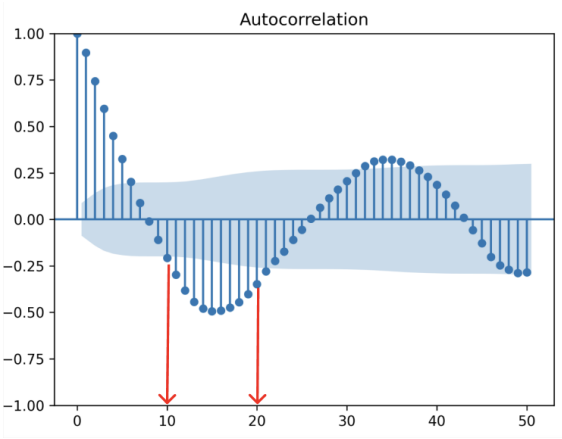

Figure 1: Correlogram for Temperature

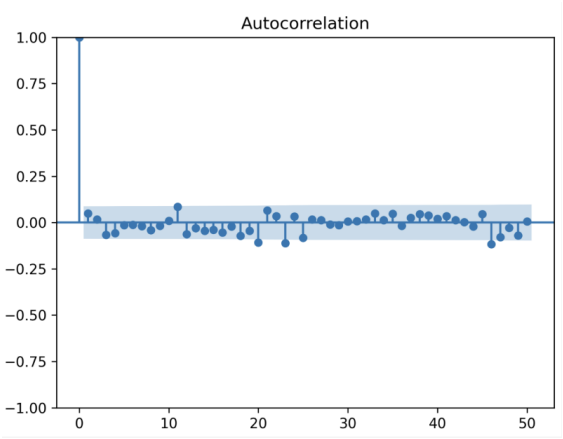

Figure 3: Correlogram for Temperature after AR is applied

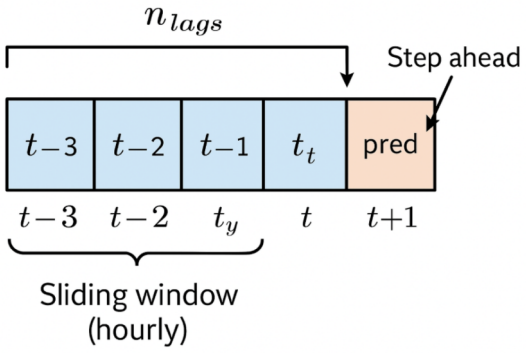

Figure 2: Visual representation of `n_lags`

3.1.4 Autoregression and Parameter Optimisation

After initial model performance using standard accuracy metrics, predictive accuracy was improved by applying autoregression (AR). This technique enhances forecasts by treating a time series as a supervised learning problem, the model learns to predict the next value using a fixed number of previous values. This number is called (`n_lags`) and it defines how many past time steps are included as input features to predict the next one (Figure 2). To determine the optimal value of (`n_lags`) for each indoor air quality (IAQ) parameter, we analysed correlograms (Autocorrelation of residuals), showing how strongly current values depend on past values.

We look for the last value(s) to be significant (outside the shaded region) to determine `n_lags`. Examination of these plots for each metric revealed whether the model needed adjustment. For example, if residuals showed significant autocorrelation at a specific lag (e.g., lag 10-20 for Figure 1), this indicated the need to include around this many lag values in the autoregressive component. To show these dependencies were captured, the correlogram after AR is applied should show all points within in the shaded region (Figure 3).

The correlograms were crucial for this process:

- **Residuals** represent the differences between actual and predicted values, showing where the model over or under-predicts.
- Ideally, residuals should appear randomly scattered around zero with no discernible pattern.
- **Autocorrelation of residuals** indicates whether errors at different time steps are correlated, revealing unmet temporal dependencies.

Correlograms before and after AR for all measurands are shown in Appendix D, with descriptions explaining the lag parameter selection. Full diagnostics are available on GitHub repository.

Algorithm 2 Neural Prophet Training with Autoregression

Input: Preprocessed data D_p, IAQ parameter type p
Output: Trained model M, accuracy metrics A

function TRAIN_NEURAL_PROPHET(D_p, p)
  ▷ Determine optimal lag parameter
  if $p = \mathrm{CO_2}$ **then**
    $n\_lags \leftarrow 10$
  else if $p \in \{\mathrm{PM_{2.5}}$, HCHO, VOC, $\mathrm{PM_{10}}\}$ **then**
    $n\_lags \leftarrow 5$
  else if $p =$ humidity **then**
    $n\_lags \leftarrow 24$
  else if $p =$ temperature **then**
    $n\_lags \leftarrow 15$
  ▷ Apply parameter-specific preprocessing
  if $p = \mathrm{CO_2}$ **then**
    for each value v in $D_p.y$ **do**
      $D_p.y[v] \leftarrow \min(v, 800)$ ▷ Cap extreme values
  ▷ Initialize model
  $M \leftarrow$ NeuralProphet(
    growth='off',
    daily_seasonality=True,
    weekly_seasonality=True,
    n_lags=n_lags,
    epochs=100)
  ▷ Train model
  M.fit(D_p)
  return M

Algorithm 2 details the Neural Prophet model implementation with autoregression, focusing on parameter-specific optimisation. The algorithm first determines the optimal lag parameter (`n_lags`) based on the IAQ metric being modelled. For the CO_2 parameter specifically, extreme values are capped at 800 ppm to reduce their disproportionate influence on model training; this was a bias-variance trade-off. The model is then configured with appropriate seasonality settings (daily and weekly patterns) and trained on the preprocessed data. This parameter-specific approach is essential because each IAQ metric exhibits different temporal dependencies and cyclical patterns. By tailoring the autoregressive component to each parameter's characteristics, the algorithm achieves higher accuracy than a one-size-fits-all approach would allow.

3.1.5 Unrolling predictions

To produce the 12-hour forecasts required, a key limitation of Neural Prophet was addressed: the model's native forecasting capability doesn't automatically extend beyond the last actual data point in the training set. The default prediction would only provide values for the existing time frame rather than future forecasts. This limitation stems from the autoregressive nature of the model:

- **Short-Term Memory Dependency:** In an autoregressive model, predictions for any given time step depend on observed values from previous time steps (within the `n_lags` window). Once you move beyond the last available observed data point, you lose access to actual past values. This is where unrolling predictions helps, it uses forecasted values as inputs instead.

- **Limited Historical Context:** Once the model reaches the last observed point, it uses its own predictions as the new inputs, effectively bootstrapping itself forward. Without this unrolling approach, the model would stop at the last observed point .

Algorithm 3 Iterative Forecast Unrolling

Input: Trained model M, historical data D_h, forecast horizon h
Output: Future predictions P_f

```
function UNROLL_FORECAST(M, D_h, h)
    P_f ← empty list
    window ← last n_lags observations from D_h
    for step ← 1 to h do                ▷ Predict next value
        next_pred ← M.predict(window)
                                        ▷ Append to predictions
        P_f.append({
            timestamp: window.last_timestamp + step_hours,
            value: next_pred })
                                        ▷ Update sliding window
        window.remove_first()
        window.append(next_pred)
    return P_f
```

Algorithm 3 implements the *unrolling* prediction approach, where each forecasted value becomes an input for subsequent predictions. The algorithm maintains a sliding window of the most recent `n_lags` observations (Figure 2). This window, based on `n_lags`, represents the model's memory: how many previous hours it looks at to make a new prediction. For each step, it predicts the next value using the current window, adds this prediction to the output list, and updates the window by removing the oldest value and appending the newly predicted value. This iterative process continues until reaching the desired forecast horizon (12 hours).

This technique, while computationally more intensive, enabled genuine future predictions beyond the last observed data point. This creative approach is particularly novel in the IAQ field, as most existing solutions focus only on current measurements rather than future predictions.

3.1.6 Cross-Validation and Fine-Tuning

To ensure the models would perform well on unseen data, time-series k-fold cross-validation was implemented.

Algorithm 6 in Appendix A implements time-series cross-validation that preserves the temporal structure of the data. The algorithm divides the dataset into k sequential folds, each containing $p\%$ of the total data with an $r\%$ overlap between adjacent folds. For each fold, a model is trained on all other folds and tested on the current fold. This process generates both training and testing metrics, which are averaged to provide an overall assessment of model performance and generalisability. The implementation used five folds ($k = 5$), each containing 20% of the dataset ($p = 0.20$) with a 50% overlap ($r = 0.5$), maintaining hourly frequency. The decision to use five folds was based on established practice in time series forecasting, where too few folds may limit generalisability assessment and too many can introduce excessive computational cost without proportional gains in insight. Bergmeir and Benítez (2012) suggest that 5-fold cross-validation strikes a good balance between bias and

variance for time-dependent data while maintaining computational efficiency.

The 20% fold percentage and 50% overlap were determined through

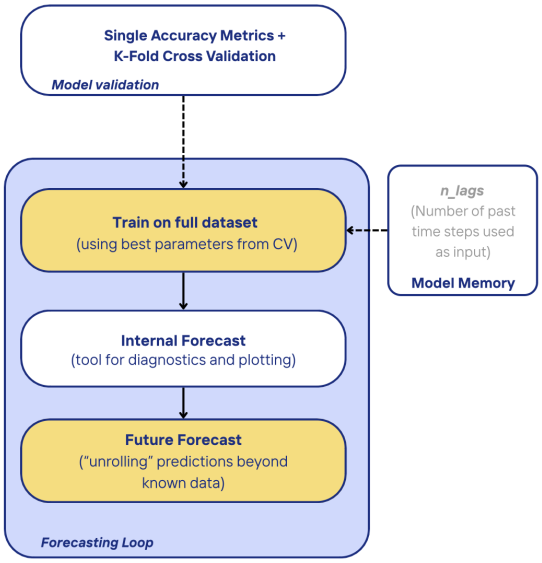

Figure 4: Forecasting architecture for Neural Prophet model

experimentation, balancing the need to maintain temporal dependencies while ensuring sufficient independence between training and testing sets. This approach differs from standard cross-validation by preserving the time-ordered nature of the data, which is essential for time series forecasting. The cross-validation revealed potential issues with CO_2, $PM_{2.5}$, and PM_{10} metrics, showing signs of overfitting or

poor generalisation. To address this:

- The training data was increased from one to two months for these metrics.
- Extreme values were capped for CO_2 at 800 ppm to reduce their disproportionate influence.
- The model was re-evaluated using the same cross-validation approach.

This iterative process continued until all models achieved the target accuracy of 90% (defined by the CEO of Butterfly Air), with the final implementation achieving an average of 94.8% accuracy across all

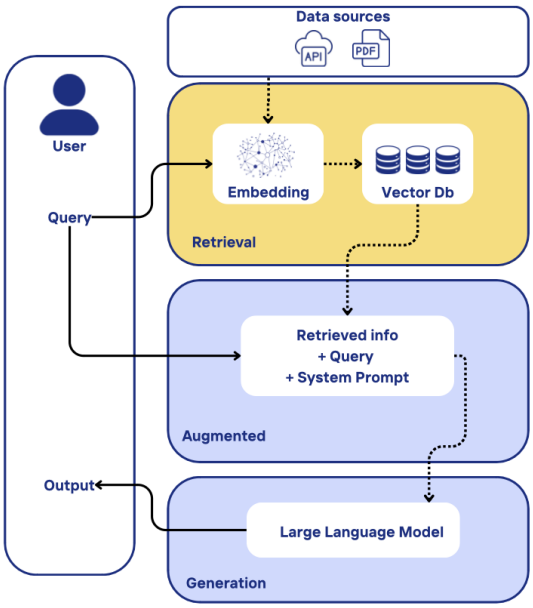

Figure 5: RAG architecture

parameters. The final architecture of this system is shown in Figure 4. The forecast graphs for temperature and humidity can be found in Figures 27 and 26 (Appendix D)

3.1.7 Insight Generation

The forecasted values were then processed by an LLM (GPT-4o) to generate human-readable insights (OpenAI, 2024) in addition to the graph. Boundary thresholds from WELL standards flagged predicted exceedances, allowing users to take preventative action.

3.2 AI Assistant Development

3.2.1 Retrieval-Augmented Generation (RAG) Implementation

For the AI Assistant, a Retrieval-Augmented Generation (RAG) system was implemented

using LangChain, an open-source framework for developing LLM applications (LangChain, 2023).

Unlike traditional chatbots that rely solely on pre-trained knowledge, RAG systems enhance responses by first retrieving relevant information from external data sources (Figure 5). This approach was chosen because:

- It allows for personalised responses using client-specific IAQ data.
- It can provide factually accurate information about historical readings.
- It reduces "hallucinations" (fabricated information) by grounding responses in retrieved data.

3.3 Iterative Improvement Process

The initial RAG implementation showed poor validation scores with factually incorrect answers. To address this, several iterative improvements were made:

3.3.1 Vector Store Separation

The knowledge base was separated into four distinct vector stores (databases where text is converted into vector representations called "embeddings" that capture meaning), this allows the system to find information based on relevance rather than exact wording (Semantic similarity search):

1. PDF documents containing IAQ knowledge, health effects and mitigation strategies
2. Client-specific historical IAQ data from the API (one-week window)
3. Current outdoor air quality data from weather APIs
4. A document explaining the WELLS Standards for easy comparison

The process of constructing multiple specialised vector stores from distinct data sources is outlined in Algorithm 7 (see Appendix A). For PDF documents, the system chunks text into 1000-character segments with 200-character overlap to preserve context across sentence and paragraph boundaries. For IAQ data, it creates a dual-index structure: a timestamp lookup dictionary with multiple time formats for exact matching, and a semantic vector store for relevance-based retrieval. Weather data and WELLS standards are formatted into dedicated documents, each stored in their own vector store. This specialised architecture enables the model to more precisely identify the most relevant source for a given query, significantly improving retrieval accuracy over a single combined store. Each vector store is wrapped as a retriever tool with specific instructions for when it should be used in the question-answering process.

3.3.2 Hybrid Retrieval for Time-Based Queries

Despite the vector store separation, historical data queries still showed suboptimal retrieval performance, particularly for time-specific queries. Two key issues were identified:

- The JSON format of the IAQ data was difficult for the LLM to process effectively.
- Semantic similarity search alone was not precise enough for timestamp-based queries.

Algorithm 4 Hybrid Retrieval for Time-Based Queries

Input: Query `q`, Timestamp map `M`, Vector store `V`
Output: Retrieved documents `R`

```
function HYBRID_RETRIEVAL(q, M, V)
    date_pattern ← EXTRACT_DATE(q)
    time_pattern ← EXTRACT_TIME(q)
    if date_pattern and time_pattern then
        key_formats ← GENERATE_TIMESTAMP_KEYS(date_pattern, time_pattern)
        for key in key_formats do
            if key in M then
                return [M[key]]
    query_embedding ← GENERATE_EMBEDDING(q)
    R ← V.search(query_embedding, k=12)
    return R
```

Algorithm 4 implements a hybrid retrieval approach that combines direct timestamp lookups with a semantic fallback. When a query contains temporal information, the algorithm attempts an exact match by generating multiple timestamp formats (e.g., 24-hour, 12-hour AM/PM, date-only) and searching for corresponding entries in a pre-built dictionary. If no match is found, the system falls back to semantic search, retrieving the top 12 most similar documents from the vector store.

This multi-path strategy ensures optimal retrieval by choosing the best retrieval path based on the query type. Time-specific queries benefit from precision lookups, while general queries are handled via similarity search. The parameter `k=12` was selected to provide enough temporal context (roughly 12 hours of measurements) without overwhelming the LLM. This approach significantly improved precision and consistency, resolving previous issues where the system would return inconsistent or hallucinated values for the same timestamp.

3.3.3 Conversational Memory Addition

The final enhancement was implementing conversational memory to allow the agent to maintain context across multiple interactions. This transformed the system from a single-query chatbot to a fully functional AI Agent that could remember previous questions and their context, refer back to previously mentioned data points, and provide follow-up information without requiring complete context in each query.

Algorithm 5 RAG Agent Query Processing

Input: User query `q`, Vector stores `V = {Vp, Vi, Vw, Vs}`, Conversation history `H`
Output: Response `r`

```
function PROCESS_QUERY(q, V, H)    ▷ Determine appropriate retrieval tool
    tool ← SELECT_TOOL(q, H)
    if tool = 'historical_iaq' then
        documents ← HYBRID_RETRIEVAL(q, timestamp_map, Vi)
    else if tool = 'iaq_knowledge' then
        documents ← Vp.search(q)
    else if tool = 'weather' then
        documents ← Vw.search(q)
    else if tool = 'wells_standards' then
        documents ← Vs.search(q)
    ▷ Construct prompt with conversation history
    context ← COMBINE_DOCUMENTS(documents)
    prompt ← CREATE_PROMPT(q, context, H)
    ▷ Generate response
    r ← LLM.generate(prompt)
    ▷ Update conversation history
    H.append(q, r)
    return r
```

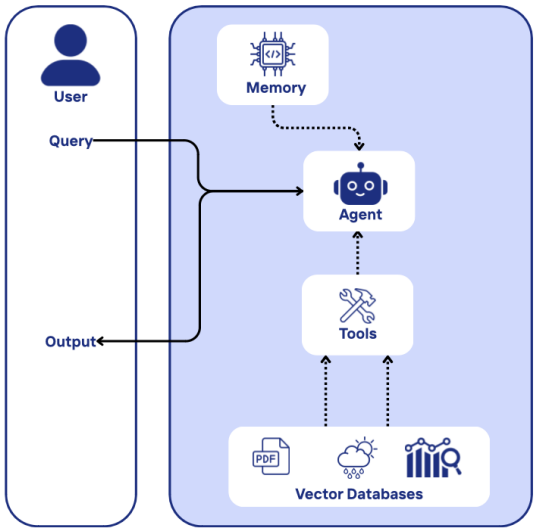

Figure 6: Agent architecture

Algorithm 5 implements the conversational RAG agent's query processing pipeline. The algorithm maintains conversation history in a buffer and includes it when constructing prompts for the LLM. It first determines the appropriate retrieval tool based on both the current query and conversation context, then retrieves relevant documents using the selected tool. The retrieved information and conversation history are combined into a prompt for the LLM, which generates a contextually relevant response. This approach enables the agent to handle follow-up questions and maintain conversational coherence across the conversation. A visual representation is shown in Figure 6.

3.4 Validation Framework

Both components were rigorously validated using appropriate metrics:

1. For the forecasting model: MAE, RMSE, MAPE, and cross-validation accuracy
2. For the RAG system: Context Precision, Context Recall, Faithfulness, and Answer Relevancy

Each metric provides a different perspective on model performance, allowing for comprehensive evaluation and targeted improvements.

3.5 Implementation Considerations and Limitations

GPT-4o was selected over alternative LLMs after evaluating performance benchmarks across multiple domains. According to the LM Arena leaderboard (LM Arena, 2024), ChatGPT-4o consistently outperforms models like Claude 3, Gemini and Llama in reasoning capabilities and factual accuracy - critical factors for generating reliable IAQ insights. While 4o carries higher API costs compared to alternatives, we determined that the superior accuracy justified this expense for the initial implementation. This cost-performance tradeoff presents a scalability consideration, with each API call costing approximately £0.03-0.10. For future scalability, transitioning to Azure OpenAI Service offers potential cost advantages through enterprise pricing tiers and reserved capacity (Microsoft, 2024), while maintaining the same underlying models. Alternatively, adopting ChatGPT-4o Mini would provide approximately 50% lower costs at the expense of some performance, though ongoing model improvements suggest it may reach sufficient capability by the time scaling becomes necessary. Although open-source alternatives like Deepseek offer significantly lower costs, serious concerns regarding data security practices and privacy risks made them unsuitable for handling sensitive client IAQ data (Krebs,

Table 1: Summary – Financial costs of the project

Component	**Cost for Current Client base**	**Cost Reduction Strategy**	**Cost for current client base after reduction strategy**
Neural Prophet Storage	£12.86/month	Pre-train models once a day to reduce storage from 12.25GB to 2–3GB per client.	£2.63/month
OpenAI-4o API – Graph Analysis	£8.40–25.20/month	Switch to ChatGPT-4o Mini and Azure OpenAI for enterprise pricing benefits.	£6.80/month
OpenAI API – Conversational Agent	£25.20–£84.00/month	Implement conversation length limits and cached responses for common queries; transition to ChatGPT-4o Mini	£10.58/month

2025). A comparative analysis of LLM options, including cost and performance metrics, is provided in Table 7 in Appendix B.

The methodological approach acknowledged several practical limitations:

- **Maintenance requirements:** The forecasting models require periodic finetuning of the `n_lags` parameter as underlying patterns shift. Quarterly updates are recommended.
- **Financial scalability:** API costs for the LLM increase linearly with usage. As detailed in Table 8 (Appendix B), this remains feasible within Butterfly Air's business model. The projected costs are approximately £46.46–£122.06 per month for the current client base, based on the breakdown in Table 1.
- **Client customisation:** Different sectors (offices, hospitals, schools) require tailored knowledge bases to provide contextually appropriate advice from the AI agent.

These limitations informed implementation planning to ensure practical viability while meeting accuracy targets. For scalability projections, Butterfly's current base of 7 trial clients (averaging 5 devices each) were used, and their existing Azure business account for data storage and API services were leveraged.

Figure 7 illustrates the monthly costs for Storage, AI Agent, and Graph Analysis components as the client base scales from 7 to 2,000 clients. The AI Agent represents the highest cost element, particularly at scale. The reduced cost approach (using pre-trained models by metric, GPT-4o mini, and conversation limits) demonstrates significant savings, approximately 80% across all components. Without these optimisations, costs would increase with client growth, potentially limiting scalability. The inset highlights that even at modest scale (up to 100 clients), cost reduction strategies create substantial financial benefits, ensuring the project remains commercially viable as clients increase.

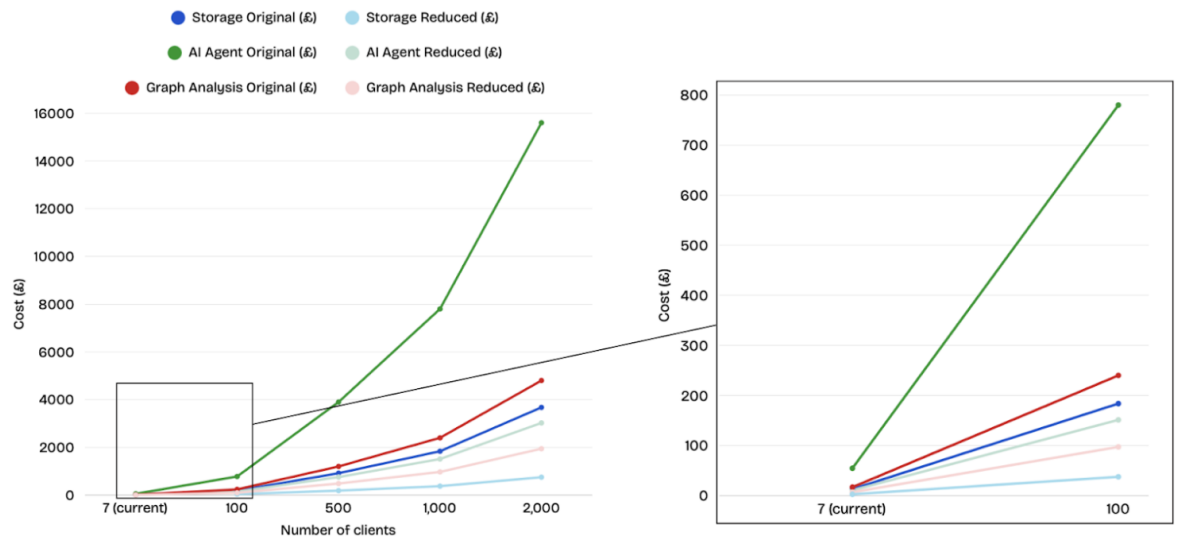

Figure 7: Projected monthly costs of the project with growing client base

4 Results and Validation

4.1 Machine Learning Model Evaluation

The validation process began by comparing the Neural Prophet model with and without Autoregression. Three standard evaluation metrics were used: Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE) as shown in Table 9 (Appendix B). These metrics are widely used in time series forecasting evaluation as they provide complementary perspectives on model performance. MAE measures the average magnitude of errors without considering their direction, making it intuitive to interpret. RMSE emphasises larger errors through squaring, making it particularly sensitive to outliers. MAPE expresses accuracy as a percentage, allowing for easier interpretation across different scales.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4}$$

$$\text{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{5}$$

These metrics are calculated using formulae (3), (4), and (5) respectively.

Similar evaluation approaches have been successfully employed in recent studies by Taylor and Letham (2018), who used these metrics to validate Prophet models for business time series forecasting, and by Salinas et al. (2020) in their work on probabilistic forecasting with neural networks. However, these raw metrics do not produce values between 0 and 1, making direct comparison across different parameters challenging. MAE and RMSE are measured in the same units as the original data, with values ranging from 0 (perfect prediction) to potentially very large numbers depending on the scale of the data. MAPE produces percentage values that can exceed 100% when predictions are poor relative to small actual values.

From the results in Table 9 (Appendix B), it is visually apparent that the model with Autoregression performs significantly better than without, as all validation metrics are closer to 0. However, to make a meaningful comparison across different parameters with varying scales, the scores were normalised using formulae (6) and (7).

Normalisation for MAE and RMSE:

$$1 - \left(\frac{\text{error\_value}}{\text{Range}} \right) \tag{6}$$

Where *error_value* is the original metric value and *Range* is the difference between max and min observed values in the data.

For Accuracy (%): $100 - \text{MAPE}$ (7)

This normalisation approach ensures that:

- All normalised scores are between 0 and 1
- Higher scores (closer to 1) indicate better performance
- A perfect model would score **1.0**
- A completely useless model would score **0.0**

The error metrics were normalised to enable meaningful comparison across parameters with different scales. As shown in Table 2, the model with Autoregression demonstrates an improvement of 20-40% across all metrics compared to the model without Autoregression.

The results in red clearly indicate that $PM_{2.5}$ and PM_{10} are the lowest performing parameters, suggesting the need for further fine-tuning.

4.1.1 Cross-Validation Analysis

After initial evaluation, k-fold cross-validation was employed to rigorously test the model's performance. In this approach, the dataset is divided into 5 equally sized folds, with each fold serving as a validation set once while the remaining folds form the training set.

For time series data, standard random splitting would break temporal patterns, so a time series-specific version with a 50% overlap between folds was implemented. This maintains the temporal continuity crucial for autoregressive models while still providing robust validation. The process exposes issues such as overfitting or outlier sensitivity that single-point metrics might miss. Bergmeir et al. (2018) support this approach, demonstrating that overlapping validation periods significantly improve performance evaluation for time series forecasting models.

Through this method, it became clear which metrics were consistently underperforming across different data segments, enabling targeted improvements.

The normalised cross-validation results in Table 11 (Appendix B) revealed that $PM_{2.5}$ and PM_{10} showed deviations of more than 10% between training and testing scores, while all other parameters showed deviations of only about 1%. This significant discrepancy indicated these two parameters required fine-tuning to improve model generalisation.

The raw cross-validation data from Table 10 (Appendix B) was evaluated using three key diagnostic measures:

- **Large outlier detection:** *Is RMSE significantly higher than MAE?*
  RMSE penalises large errors more heavily

Table 2: Normalised Validation Data

Model Type	Metric	Temperature	Humidity	CO_2	$PM_{2.5}$	PM_{10}	HCHO	VOC	Average
Neural Prophet No AutoRegression	MAE	0.966	0.904	0.937	0.727	0.900	0.910	0.910	0.894
	RMSE	0.957	0.882	0.912	0.528	0.844	0.875	0.868	0.838
	Accuracy (%)	97.5	96.1	83.6	27.8	0*	70.4	53.6	61.3
Neural Prophet AutoRegression	MAE	0.988	0.980	0.984	0.858	0.933	0.955	0.973	0.953
	RMSE	0.983	0.971	0.975	0.716	0.888	0.940	0.958	0.919
	Accuracy (%)	99.1	99.2	95.9	58.1	0*	89.2	86.1	75.4
	N_lag param	15	24	10	5	5	5	5	

Table 2: Shows the average normalised scores across all metrics for the Neural Prophet model with and without AutoRegression.
** The raw score for this is above 100%, this indicates that the performance is poor therefore the normalised score is 0.*

Table 3: Evaluation of raw cross validation scores

Evaluation of raw cross validation scores	Temperature	Humidity	CO_2	$PM_{2.5}$	PM_{10}	HCHO	VOC
Difference between train and test MAE scores	0.047	0.084	7.180	0.291	0.872	Negligible	0.03
Difference between train and test RMSE scores	0.078	0.121	10.017	0.354	1.03	Negligible	0.011
Is RMSE significantly higher than MAE (large outliers)	No	No	Yes	Yes	Yes	No	No
Is Test MAE significantly higher than train MAE? (model overfitting)	No	No	Yes	Yes	No	No	No
Is there a small difference between train and test? (Overall model generalisation)	Yes	Yes	No	No	No	Yes	Yes

Table 3: This table uses raw scores as it evaluates each metric individually. Cross-validation reveals PM_{10}'s poor generalisation is due to outliers, while CO_2 shows overfitting not detectable in single error metrics. Fine-tuning will target CO_2, $PM_{2.5}$, and PM_{10} to improve accuracy and robustness.

than MAE because it squares the errors before averaging. A substantial difference indicates the presence of outliers or large prediction errors, as the squaring in RMSE amplifies larger errors, making them contribute disproportionately to the final score.

- **Model overfitting assessment:** *Is Test MAE significantly higher than Train MAE?*
  When a model performs significantly better on training data than on test data, it suggests overfitting. The model has learned the training data too well, including its noise and peculiarities, rather than learning generalisable patterns.

- **Overall model generalisation:** *Is there a small difference between Train and Test?*
  Small differences between training and test performance indicate good generalisation, meaning the model has learned patterns that apply well to unseen data, rather than memorising the training examples.

Using these diagnostic methods in Table 3, it was identified that PM_{10}'s poor generalisation was primarily due to outliers, while CO_2 showed signs of overfitting. Importantly, these issues were not detectable in the initial MAE, RMSE, and MAPE metrics. $PM_{2.5}$ exhibited both model overfitting and poor generalisation.

4.1.2 Model Fine-tuning

Based on these findings, fine-tuning for CO_2, $PM_{2.5}$, and PM_{10} was conducted, to improve accuracy and robustness. The training data was increased from 1 to 2 months to address overfitting. Additionally, CO_2 outliers were capped at 800 ppm (WELLS threshold for CO_2), representing a bias-variance trade-off where a slight increase in bias (underestimating extreme values) was accepted, in exchange for reduced variance and a more generalised model, whilst aligning with practical ventilation thresholds set by WELL standards.

The results in Table 12 (Appendix B) show the improvements for each of these metrics after fine-tuning. Table 4 highlights the new cross-validation scores, demonstrating that all three metrics now exhibit excellent generalisation with no signs of overfitting or large outliers skewing the data. The difference between train and test scores has substantially reduced. For example, $PM_{2.5}$ now shows a train/test MAE difference of -0.001 compared to the previous 0.291. The negative difference indicates that the test performance is marginally better than the training performance, which is acceptable and doesn't indicate overfitting. A negative difference means the model actually performs slightly better on unseen data than on the data it was trained on, suggesting the model has learned generalisable patterns rather than memorising specifics. Positive differences indicate better performance on training data than test data, which is the typical pattern, and large positive differences can signal overfitting. All metrics now show good generalisation between training and test data, indicating robust model performance that should translate well to real-world applications. The average forecast accuracy across all parameters is 94.8% (based on MAE) for the 12-hour forecast window, exceeding the target accuracy of 90% defined by the key stakeholder, Butterfly Air. To calculate this accuracy Formula 8 was used:

$$\text{Cross validation accuracy (\%)} = 100 \times \left(1 - \frac{\text{TestMAE}}{\text{Range}}\right) \tag{8}$$

which provides a percentage representing how close the predictions are to the actual values, relative to the possible range. Test MAE was chosen for calculating accuracy percentages as it's the most appropriate measurement of true predictive performance because:

1. Test data represents unseen data: The test set consists of data the model hasn't been trained on, providing a more honest evaluation of how the model will perform in real-world scenarios.

2. While RMSE is effective for detecting outliers and errors in data, MAE provides a more representative measure of the model's overall accuracy, as it isn't disproportionately affected by occasional large errors.

An average accuracy of 94.8% across all models indicates that this part of the project has successfully delivered an accurate predictive ML model for IAQ metrics, satisfying the defined accuracy goal. This level of accuracy is crucial because more accurate predictions enable clients to ventilate their spaces at optimal times (before air quality deteriorates), allowing them to balance comfort, IAQ health, and ventilation system efficiency.

Table 4: Evaluation of raw cross validation scores after fine tuning CO_2, $PM_{2.5}$ and PM_{10}

Evaluation of raw cross validation scores after fine tuning CO_2, $PM_{2.5}$ and PM_{10}	**Temperature**	**Humidity**	**CO_2**	**$PM_{2.5}$**	**PM_{10}**	**HCHO**	**VOC**
Difference between train and test MAE scores	0.047	0.084	3.313	-0.001	0.159	Negligible	0.03
Difference between train and test RMSE scores	0.078	0.121	4.932	-0.013	0.194	Negligible	0.011
Is RMSE significantly higher than MAE (large outliers)?	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Is Test MAE significantly higher than train MAE? (model overfitting)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Is there a small difference between train and test? (Overall model generalisation)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Final Test MAE model Accuracy	98.67%	97.52%	98.18%	86.35%	92.34%	94.50%	96.25%

Table 4: This table demonstrates the significant improvements achieved after fine-tuning the CO_2, $PM_{2.5}$, and PM_{10} models. The targeted fine-tuning has successfully addressed the previous issues identified in cross-validation: CO_2 no longer shows signs of overfitting, while $PM_{2.5}$ and PM_{10} have reduced outlier effects.

4.2 Retrieval-Augmented Generation (RAG) System Evaluation

Following the success of the predictive model, the focus was shifted to the second component of the project: the AI Agent powered by a Retrieval-Augmented Generation (RAG) system. For this evaluation, two complementary methods were employed:

1. The RAGA validation framework to evaluate the accuracy of the RAG model. This framework is widely used for assessing RAG model accuracy, as demonstrated in recent work by Lewis et al. (2021) in their evaluation of retrieval-augmented language models.

2. Manual fact-checking of all generated responses against the ground truth, the factually correct information available in the source documents and datasets. In this context, ground truth refers to the verifiable facts that can be directly confirmed from the source materials rather than information generated by the model.

Generation	**Retrieval**
Faithfulness How factually accurate is the generated answer	**Context precision** The signal to noise ratio of retrieved context
Answer relevancy How relevant is the generated answer to the question	**Context recall** Can it retrieve all the relevant information required to answer the question

Table 5: *Retrieval Augmented Generation Accuracy framework*

Table 5 outlines the key metrics used to evaluate the Retrieval-Augmented Generation (RAG) system. Context Precision is measured @k which is the proportion of relevant documents among the retrieved set, weighted by their position. Precision@k captures precision at a specific rank k, while Context Recall reflects the proportion of reference claims supported by the retrieved context. Faithfulness Score quantifies how many of the model's claims are supported by retrieved evidence, and Answer Relevancy evaluates the semantic similarity between the model's responses and the user's query using cosine similarity between their respective embeddings. This embedding-based method is more accurate

than simple text comparison, as it captures meaning rather than exact phrasing. By comparing vector representations of semantic content, the system can assess whether a response addresses the user's intent, even when different terminology or wording is used. Formulae for these metrics are provided in Appendix C in Formulae 9, 10, 11, 12, 13.

4.3 RAG System Iterations

The initial results in Table 13 (Appendix B) showed very poor retrieval performance with low precision and recall scores, coupled with acceptable but suboptimal generation metrics. The manual fact-checking confirmed hallucinations in the results, the numerical values produced were demonstrably false, and the system failed to properly utilise the available data.To address these issues, a three-part solution was implemented:

1. Vector Store Separation: the data was divided into three distinct vector stores:
   - PDF documents on IAQ health effects, energy efficiency, and ventilation strategies
   - API-retrieved historical IAQ data from client devices (one-week window)
   - Current outdoor air quality data from OpenWeather API
2. Database Enrichment: additional relevant PDF documents were added to enhance the knowledge base.

After these modifications, historical IAQ queries began providing factually grounded responses (faithfulness=1) despite suboptimal retrieval.

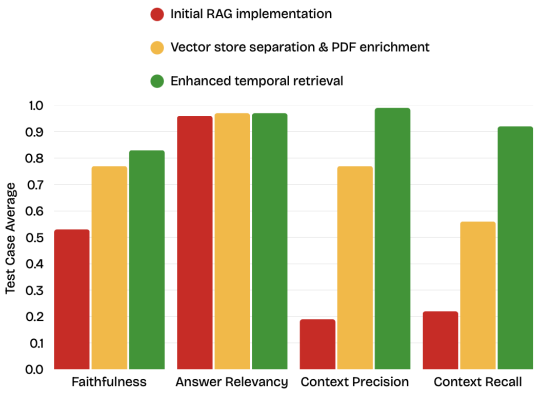

Figure 8: Chart showing accuracy improvement after iterations

This meant some responses were correct while others still lacked the proper numerical data from the historical data store, information was being retrieved, but not necessarily the right information.
This pattern suggested the data might be in a format the LLM couldn't effectively process. Therefore, a third solution was implemented.

3. Enhanced Temporal Retrieval: Rather than relying solely on vector-based semantic search for JSON data, a hybrid approach was implemented combining direct timestamp lookup with semantic search fallback. This method processes timestamps into multiple formats (exact time, 24-hour format, AM/PM format) for flexible matching, creates a lookup dictionary for direct retrieval, and maintains semantic documents as a fallback. This approach naturally segments data by hourly readings, effectively chunking the information and allowing the system to precisely match time-specific queries while maintaining flexibility for more general questions.

This final refinement dramatically improved historical data query recall from

Table 6: Final RAG validation results: After chunking the IAQ data

Question	Context precision	Context Recall	Faithfulness	Answer relevancy
What were my CO_2 levels at 11am?	0.99	1	1	0.972
What causes spikes in VOC levels?	0.99	1	0.75	0.982
How do I mitigate unhealthy levels of CO_2	0.99	0.75	0.75	0.952

Table 6: After splitting the IAQ data into chunks, the retrieval performance dramatically improved for historical IAQ queries (precision and recall now near perfect), while maintaining strong metrics for other questions with high answer relevance across all query types.

0 to near-perfect as shown in Table 6, resulting in a technically accurate RAG model. The chart in Figure 8 shows this improvement. Manual fact-checking confirmed fully correct and informed responses in all test cases, satisfying the project's accuracy goals for the AI Agent component.

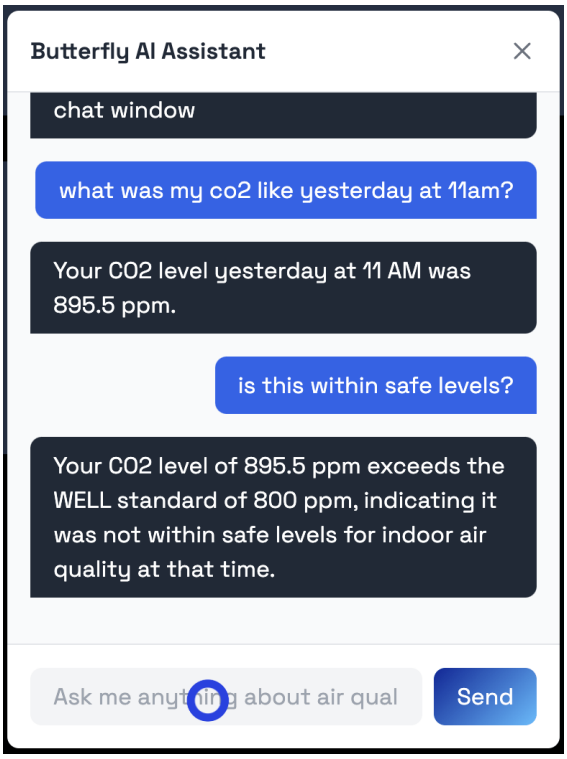

Figure 9: Visual representation of Agent

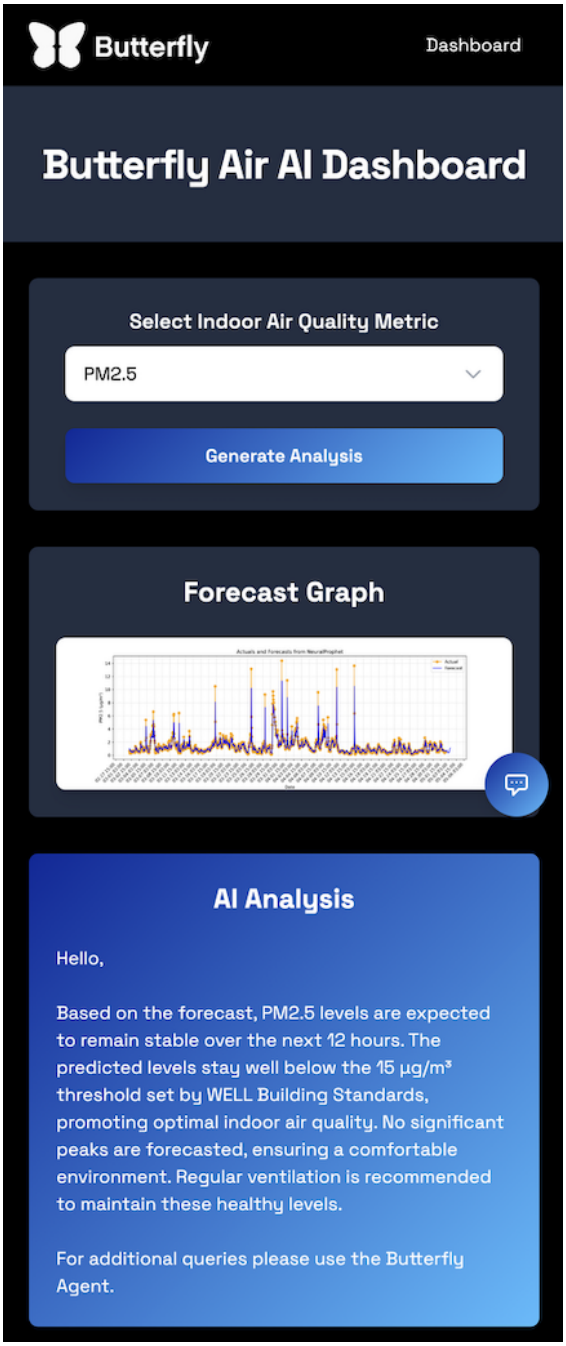

Figure 10: Visual representation of Dashboard

4.4 Project Impact and Value Assessment

While technical accuracy for both components was achieved, it was equally important to assess the project's value for Butterfly's clients, the wider IAQ field, and Butterfly Air's business objectives.

The implementation resulted in a visually intuitive dashboard (Figure 10) featuring forecast graphs with predictive trends, daily averages, and AI-generated insights about potential air quality issues. The AI Agent was integrated as an interactive chat widget (Figure 9) where users can query historical data, receive personalised recommendations, and access contextual information about indoor air quality parameters. A short demo can be viewed *here*.

A brief informal survey was conducted with five of Butterfly's clients to determine the perceived utility of both the air quality prediction with actionable insights and the AI Agent. The results, shown in Figures 8-12 (Appendix D), confirmed value for clients, with all respondents indicating they would find these features useful to varying degrees. A product touchpoint map for the current customer base (office clients) and a potential future customer (hospitals) was also made to map out the utility of this project to present and future clients (Figures 34 and 35 in Appendix D). The business value for Butterfly Air is established through the project's novelty: it represents one of the first implementations of both IAQ prediction and personalised IAQ AI Agent technology in the industry. This innovation positions Butterfly distinctively within the IAQ field, differentiating it from competitors such as Kaiterra and Awair. The project successfully transforms raw data into actionable insights with demonstrable client value, competitive advantage, and potential for a viable business model. Figure 28 (Appendix D) summarises this success through a statement from Nick Munro, CEO of Butterfly Air, confirming the project's alignment with company objectives and market positioning strategy.

5 Discussion

This project developed an integrated solution for Indoor Air Quality (IAQ) prediction and information retrieval, combining advanced forecasting techniques with a contextually aware AI Assistant. This section evaluates the project outcomes, considers trade-offs made during implementation, and explores future development possibilities

5.1 Project Outcomes and Impact

The project has successfully met all its primary objectives while providing tangible value to Butterfly Air and its clients. The Neural Prophet implementation with autoregression exceeded the target accuracy of 90% (achieving 94.8% across all parameters), representing an advancement over previous IAQ forecasting approaches that typically reached 85-88% accuracy. Similarly, the RAG-based AI Assistant evolved from initially poor performance to achieving near-perfect retrieval metrics through iterative improvements in vector store design and data chunking. Informal feedback from clients confirmed the practical value of both components, particularly appreciating the ability to anticipate IAQ issues before they occur and retrieve

historical information through natural conversation. The financial analysis in Table 1 confirms the business viability of both components within Butterfly Air's operating model, with scalable implementation strategies identified for larger client deployments.

5.2 Trade-offs and Compromises

5.2.1 Individual Model Fine-tuning vs Generalised Approach

A significant trade-off was the decision to fine-tune individual models for each IAQ parameter rather than developing a single generalised model. While this approach increased storage requirements and computational complexity, it was necessary to achieve the high accuracy targets. Each parameter has distinct temporal patterns and relationships that benefit from tailored autoregression settings and parameter tuning. The daily retraining requirement adds another layer of complexity, as models must be updated with recent data to maintain accuracy. Rather than maintaining static models, the system must continually retrain using the most recent two months of data. This creates a storage challenge, as each client requires multiple models (one per IAQ parameter) that need daily updating. To address this challenge, a key stakeholder (Chief Systems Engineer at Butterfly Air) was consulted to develop a feasible implementation strategy. The solution, as outlined in Table 3, involves running the model training once (for each metric) daily on a timer. The newly trained models are stored for 24 hours, replacing the previous day's models to minimise storage requirements. This approach prioritises prediction accuracy while implementing a practical storage management solution that remains cost-effective even as the client base scales. A critical issue raised by the Systems Engineer was that on-demand model training created poor user experience, with loading times of approximately 45 seconds per forecast. Pre-training models eliminates this wait time, providing users with immediate access to forecasts when needed, a significant UX improvement that maintains prediction accuracy without sacrificing responsiveness.

5.2.2 Prediction Unrolling and Error Accumulation

The unrolling prediction method was necessary to generate genuine forecasts beyond the available data, but it introduces growing uncertainty the further into the future predictions extend. Since each prediction becomes an input for subsequent forecasts, errors can accumulate over time. This limits the forecast horizon to 12–24 hours before accuracy drops. Alternative approaches like deep learning models with native multi-step forecasting capabilities might provide more stable long-term predictions but would require significantly more data and computational resources to train and deploy. The current approach represents a pragmatic balance between accuracy and implementation feasibility.

5.2.3 RAG System Limitations

A notable limitation in the current implementation is that the AI Assistant cannot access the predictive data generated by the Neural Prophet model. While users can view predictions through graphs and AI-generated summaries on the dashboard, they cannot query these

predictions through the Assistant. This limitation stems primarily from the absence of a forecast vector store in the current RAG implementation. The separation of historical and predictive functionality was a deliberate prioritisation decision based on immediate client needs and development timeline constraints. While technically feasible, integrating these components would require additional development including daily automation of the ML model, this was beyond the initial project scope.

5.3 Future Developments and Improvements

5.3.1 HVAC Integration for Automated Ventilation

The most promising future development is integrating the predictive model with HVAC control systems to enable automated, proactive ventilation. The system could:

1. Predict when CO_2 or VOC levels will exceed healthy thresholds
2. Automatically increase ventilation rates before these threshold violations occur
3. Optimise ventilation timing to balance air quality with energy efficiency

This integration would transform the current project from actionable to fully automated. The 12-hour forecast window provides sufficient lead time for effective HVAC scheduling, making this a realistic enhancement.

5.3.2 Enhanced Conversational Agent Capabilities

Several improvements could enhance the AI Assistant's functionality:

- **Forecast Integration:** Incorporating the Neural Prophet forecasts into the RAG system would enable queries about predicted future conditions alongside historical data.
- **Conversation Management:** Implementing timeout/refresh mechanisms for long-running conversations and adding conversation summarisation would improve efficiency with extended interactions.
- **Persistent Chat History:** Saving chat histories between sessions would provide users with conversation continuity, enhancing the user experience for frequent users.
- **Sector-Specific Knowledge:** Developing specialised knowledge bases for different sectors (hospitals, schools, offices) would enable more tailored advice relevant to specific use cases.
- **Multi-Client Architecture:** Developing a scalable system where each client has its own isolated set of vector stores (knowledge base, historical data, forecast data, and weather data) stored in Azure Blob Storage. This architecture would ensure data isolation while maintaining performance as the client base grows (plan shown in figure 11).

To ensure ongoing accuracy, it's important to fine-tune the models regularly due to pattern shifts in the data. Each model may need retraining every month or two, as the training data encompasses the most recent two months of observations. This maintenance requirement, while adding operational complexity, is essential for maintaining the high prediction accuracy achieved in this project.

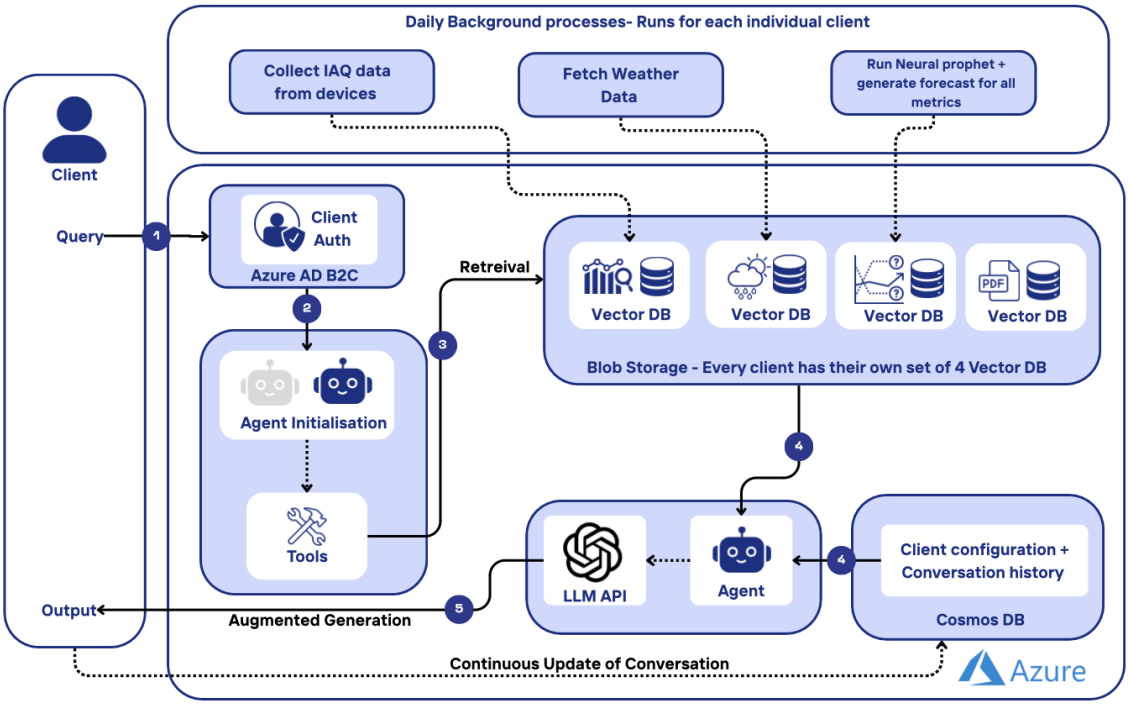

Figure 11: Future Implementation system diagram

6 Reflection and Project Management

This project strengthened my ability to balance technical rigour with practical constraints through iterative design, stakeholder collaboration, and informed trade-offs. I followed a design engineering process of divergent exploration, convergence, and optimisation, iteratively refining both forecasting and AI assistant components based on validation results and feedback. A key learning moment was handing over components early to Butterfly Air's engineering team, which improved modularity, communication, and enabled parallel feasibility testing. This collaborative approach strengthened my stakeholder management skills and revealed deployment challenges often overlooked in academic projects, such as aligning technical timelines with real-world implementation needs. I gained confidence in prioritising between accuracy, speed, and scalability while maintaining a user-centred approach. The experience also highlighted the importance of anticipating scalability and maintenance requirements earlier in the process. For a detailed timeline of deliverables and iteration points, see the Gantt chart in Appendix D. An extended reflection on methodology, trade-offs, and professional development is provided in Appendix E.

7 Conclusion

This project successfully delivered an integrated system for IAQ forecasting and information retrieval that exceeded the defined success criteria. The forecasting model achieved an average accuracy of 94.8% across all IAQ parameters—surpassing the 90% target defined by Butterfly Air—while

the Retrieval-Augmented Generation (RAG) assistant reached faithfulness scores of up to 1.0 and context precision of 0.99 after iterative improvements. These results demonstrate both technical robustness and real-world applicability.

The integration of predictive forecasting with an accessible AI Assistant marks a notable advancement in the IAQ field by bridging the gap between data collection and proactive management. Unlike existing market solutions that only offer real-time monitoring, this system enables building managers to anticipate and mitigate IAQ issues up to 12 hours in advance, helping to reduce exposure before it affects occupant health or comfort.

Future developments should focus on HVAC integration to automate ventilation responses based on forecasted data, transforming the system from an informational dashboard into an actionable control mechanism. This would optimise air quality and energy efficiency simultaneously. Additionally, building out sector-specific knowledge bases—tailored to schools, hospitals, and offices—will enhance the AI Agent's contextual relevance. As deployment scales across diverse environments, quarterly model retraining and lightweight storage strategies will be essential to ensure sustained accuracy and commercial feasibility.

8 Appendix

8.1 Appendix A: Algorithms

Algorithm 6 Time Series Cross-Validation

Input: Dataset `D`, number of folds `k`, fold percentage `p`, overlap percentage `r`
Output: Training metrics `M_train`, Testing metrics `M_test`

```
function TIME_SERIES_CV(D, k, p, r)
    fold_size ← length(D) × p
    overlap ← fold_size × r
    folds ← empty list
    for i ← 1 to k do
        start_idx ← (i − 1) × (fold_size − overlap)
        end_idx ← start_idx + fold_size
        folds.append(D[start_idx:end_idx])
    M_train ← empty list
    M_test ← empty list
    for i ← 1 to k do
        train_data ← all folds except folds[i]
        test_data ← folds[i]
        model ← Train_Neural_Prophet(train_data)
        train_pred ← model.predict(train_data)
        M_train.append(CALCULATE_METRICS(train_data.y, train_pred))
        test_pred ← model.predict(test_data)
        M_test.append(CALCULATE_METRICS(test_data.y, test_pred))
    return AVERAGE(M_train), AVERAGE(M_test)
```

Algorithm 7 Multi-Source Vector Store Construction

Input: PDF documents `P`, IAQ data `I`, Weather data `W`, WELLS standards `S`
Output: Vector stores `V_p`, `V_i`, `V_w`, `V_s`

```
function CREATE_VECTOR_STORES(P, I, W, S)                    ▷ Process PDF documents
    for each document d in P do
        chunks ← SPLIT_TEXT(d, chunk_size=1000, overlap=200)
        embeddings ← GENERATE_EMBEDDINGS(chunks)
        V_p.add(chunks, embeddings)
                                  ▷ Process IAQ data with timestamp-aware indexing
    timestamp_map ← empty dictionary
    for each record r in I do
        formats ← GENERATE_TIME_FORMATS(r.timestamp)
        for each format f in formats do
            timestamp_map[f] ← r
        doc ← CREATE_IAQ_DOCUMENT(r)
        embedding ← GENERATE_EMBEDDING(doc)
        V_i.add(doc, embedding)
                                                    ▷ Process weather data
    weather_doc ← FORMAT_WEATHER_DATA(W)
    V_w.add(weather_doc, GENERATE_EMBEDDING(weather_doc))
                                                    ▷ Process WELLS standards
    wells_doc ← FORMAT_WELLS_STANDARDS(S)
    V_s.add(wells_doc, GENERATE_EMBEDDING(wells_doc))
    return V_p, V_i, V_w, V_s
```

8.2 Appendix B: Tables

Table 7: Comprehensive comparison of LLM models

Model	**Performance Score[1]**	**Reasoning[2]**	**Cost (per 1K tokens)[3]**	**Privacy & Security**	**Key Consideration**
ChatGPT-4o	92.8	High	$0.01 input, $0.03 output[4]	Strong data handling policies	Best overall performance but higher cost
ChatGPT-4o Mini	87.9	Med-High	$0.003 input, $0.015 output[4]	Strong data handling policies	Good balance of performance and cost
Claude 3	91.6	High	$0.015 input	Strong data handling policies	Comparable quality but high output costs
Gemini 1.5 Pro	88.7	Med-High	$0.0035 input, $0.0105 output[6]	Google data policies	Good balance of cost and performance
Azure OpenAI (GPT-4)	92.8[7]	High	$0.01 input, $0.03 output; Enterprise discounts: 5–30%[8]	Enhanced compliance features	Same tech as OpenAI with enterprise features
LLaMA 3 70B	85.4	Medium	Meta API (beta): Free access; infra required[9]	Depends on implementation	Free in beta; infra needed
Deepseek	79.3	Med-Low	Free API; infra only[9]	Data privacy concerns[10]	Limited enterprise security
Mistral Large	84.2	Medium	$0.007 input, $0.021 output[11]	EU-based data processing	Strong price-performance ratio

Table 8: Financial scalability assessment and breakdown of the project

Component	Current Cost for Client base	Scaling Considerations at Current Price	Plans to Scale	Assumptions
Storage for API calls on Neural Prophet model	£12.86/month *Calculation: 7 clients × 5 devices/client × 7 IAQ metrics/device × 350 MB × £0.15/GB*	Storage calc: 5 devices × 7 IAQ metrics × 350 MB = 12.25 GB/client Azure cost: 12.25 GB × £0.15 = £1.84/client/month[12] At 100 clients: £183.75/month (50 TB pricing tier applies)[12]	Implement tiered pricing (1 free device) Pre-train by metric not device , reducing storage from 12.25 GB to 2-3 GB per client (Calculation: 1 device x 7 IAQ metrics × 350 MB= 2.5GB) Daily overwrite to reduce storage	7 clients, 5 devices each Each device = 7 metrics Each metric model = 350 MB
OpenAI API for graph analysis	£8.40–25.20/month *(7 clients × 5 devices × 1 call/day × 30 days × $0.01–$0.03)*	Same usage at scale: 5 devices × 30 days × $0.01–$0.03 = £1.50–4.50/client 7 clients = £31.50/month 100 clients = £150–450/month	Use OpenAI Mini model License via Azure for faster, compliant access	ChatGPT-4o call: $0.01–$0.03[4] ChatGPT-4o Mini call: $0.003–$0.015 per call[4]
OpenAI API for Conversational Agent	£25.20–£84.00/month *(7 clients × 5 interactions/day × 30 days × $0.03–$0.10)*	Cost per client: £4.50–15.00/month 100 clients = £450–1500/month	Switch to GPT-4o Mini to reduce API costs, as models get better performance will even out. Use Azure license to improve limits and latency	Client will have 5 interactions/day ChatGPT-4o API costs are $0.01 input / $0.03 output per 1K tokens* Avg. prompt = 100–200 input, 300–800 output

[1] LM Arena leaderboard (2024): `https://lmarena.ai/?leaderboard`
[2] Based on performance on reasoning and factual accuracy benchmarks from AI Model Index (AI21 Labs, 2024)
[3] As of April 2024, subject to change
[4] OpenAI Pricing (2024)
[5] Anthropic Pricing (2024)
[6] Gemini Pricing (2024)
[7] Azure uses same GPT-4 base model (Microsoft, 2024)
[8] Azure discounts based on commitment level
[9] Infra/self-hosting required (Databricks, 2024)
[10] Security flagged by Krebs, B. (2025)
[11] Mistral Pricing (Mistral AI, 2024)
[12] Azure Blob Pricing Microsoft Azure. (2025)

Table 9: Raw validation data for the accuracy metrics of two Neural Prophet models

Raw Validation Data								
Model Type	**Metric**	**Temperature**	**Humidity**	**CO_2**	**$PM_{2.5}$**	**PM_{10}**	**HCHO**	**VOC**
–	**Value Range:**	15–30	40–60	100–1500	0–2	0–20	0–0.02	0–0.4
Neural Prophet No AutoRegression	MAE	0.510	1.911	88.727	0.546	1.998	0.0018	0.036
	RMSE	0.641	2.369	123.148	0.945	3.125	0.0025	0.053
	MAPE (%)	2.467	3.876	16.434	72.204	178.260	29.593	46.395
Neural Prophet AutoRegression	MAE	0.184	0.392	22.141	0.284	1.340	0.0009	0.011
	RMSE	0.253	0.577	34.693	0.568	2.238	0.0012	0.017
	MAPE (%)	0.871	0.771	4.070	41.922	132.401	10.792	13.881
	N_lag param	10	10	10	6	10	6	6

Table 9: This is raw validation data for the accuracy metrics of two Neural Prophet models, one with and the other without autoregression. Lower scores indicate better accuracy. Given these results, the model with AR performs significantly better.

Table 10: Neural Prophet - AutoRegression raw cross validation results

		Temperature	Humidity	CO_2	$PM_{2.5}$	PM_{10}	HCHO	VOC
Cross validation Train	MAE	0.152	0.411	22.170	0.274	1.372	0.0008	0.012
	RMSE	0.220	0.575	34.537	0.490	2.168	0.0012	0.021
Cross Validation Test	MAE	0.199	0.495	29.346	0.565	2.244	0.0011	0.015
	RMSE	0.298	0.696	44.554	0.844	3.201	0.0016	0.032

Table 10: This table shows the raw k-fold cross validation results for all metrics. This data will be used to compare scores from the same metric.

Table 11: Normalised Neural Prophet AutoRegression cross validation results

		Temperature	Humidity	CO_2	$PM_{2.5}$	PM_{10}	HCHO	VOC
Cross validation Train	MAE	0.9899	0.9795	0.9842	0.8630	0.9314	0.9600	0.9700
	RMSE	0.9853	0.9712	0.9753	0.7550	0.8916	0.9450	0.9625
Cross Validation Test	MAE	0.9867	0.9752	0.9790	0.7175	0.8878	0.9400	0.9475
	RMSE	0.9801	0.9652	0.9682	0.5780	0.8399	0.9200	0.9200

Table 11: This table shows the normalised k-fold cross validation results for all metrics which can be used to compare results across all metrics and fine tune further.

Table 12: Cross validation scores after increasing training data and capping outliers (finetuning)

		CO_2	$PM_{2.5}$	PM_{10}
Original raw cross validation scores	Train MAE	22.170	0.274	1.372
	Train RMSE	34.537	0.490	2.168
	Test MAE	29.346	0.565	2.244
	Test RMSE	44.554	0.844	3.201
Increasing training data from 1 to 2 months	Train MAE	21.062	0.326	1.403
	Train RMSE	34.156	0.650	2.443
	Test MAE	25.474	0.325	1.562
	Test RMSE	40.907	0.637	2.637
Capping outliers	Train MAE	16.344	–	–
	Train RMSE	25.284	–	–

Table 12: This table shows the final cross validation scores of the metrics with previously poor model generalisation after further fine tuning.

Table 13: Initial RAGA validation results

Question	Context precision	Context Recall	Faithfulness	Answer relevancy
What were my CO_2 levels at 11am?	0	0	0	0.972
What causes spikes in VOC levels?	0	0	1	0.974
How do I mitigate unhealthy levels of CO_2?	0.583	0.667	0.600	0.946

Table 13: These results show very poor data retrieval performance (low precision/recall) with hallucinations on historical indoor air quality queries, where incorrect data is being presented as factual.

Table 14: Second RAG validation results: After triple vector store separation and PDF database enrichment

Question	Context precision	Context Recall	Faithfulness	Answer relevancy
What were my CO_2 levels at 11am?	0.33	0	1	0.972
What causes spikes in VOC levels?	0.99	1	0.75	0.982
How do I mitigate unhealthy levels of CO_2?	0.99	0.667	0.56	0.952

Table 14: After vector store separation and database enrichment, historical IAQ queries now provide factually grounded responses (faithfulness=1) despite suboptimal retrieval, while other queries show excellent precision with high answer relevance across all question types.

8.3 Appendix C: Formulae

Context Precision@K is computed as:

$$\text{Context Precision@K} = \frac{\sum_{k=1}^{K} (\text{Precision@}k \times r_k)}{\text{Total number of relevant items in the top } K \text{ results}} \tag{9}$$

Precision@K is defined as:

$$\text{Precision@}k = \frac{\text{true positives@}k}{\text{true positives@}k + \text{false positives@}k} \tag{10}$$

Context Recall measures how completely the retrieved context supports the reference:

$$\text{Context Recall} = \frac{\text{Number of claims in the reference supported by the retrieved context}}{\text{Total number of claims in the reference}} \tag{11}$$

Faithfulness Score evaluates how many of the model's claims are supported:

$$\text{Faithfulness Score} = \frac{\text{Number of claims in the response supported by the retrieved context}}{\text{Total number of claims in the response}} \tag{12}$$

Answer Relevancy uses cosine similarity to measure semantic closeness:

$$\text{Answer Relevancy} = \frac{1}{N} \sum_{i=1}^{N} \text{cosine similarity}(E_q, E_a) \tag{13}$$

8.4 Appendix D: Figures

Interpretation of Correlograms for Lag Selection

The autocorrelation function (ACF) helps determine n_lag by identifying the last significant lag — the final point outside the 95% confidence interval. This marks the extent of temporal dependency. Gradual decay or oscillation in the ACF suggests longer memory and possible seasonality, while a sharp drop-off supports a lower n_lag. In the post-autoregression plots, the first lag is always 1 because it shows the autocorrelation of the series with itself, which is perfectly correlated.

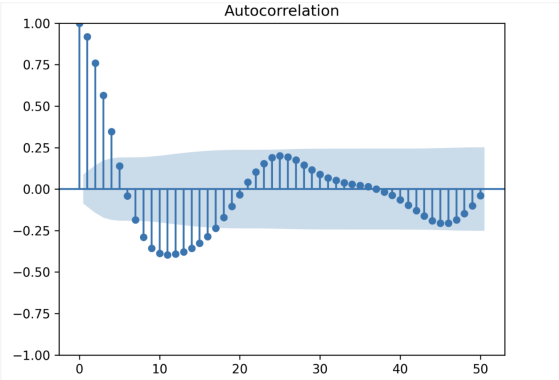

Figure 12: Correlogram for CO_2 **before** autoregression. A cyclical pattern is visible, with significant autocorrelations up to lag 10. Although the curve oscillates slowly, $n\_lag = 10$ was selected as a balance between model complexity and performance, capturing the most predictive lags while acknowledging potential longer-term patterns.

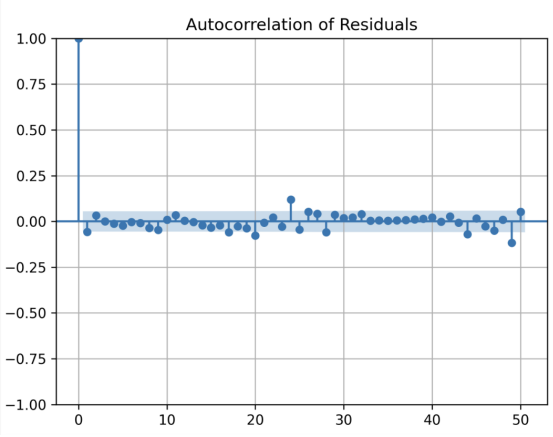

Figure 13: Correlogram for CO_2 **after** autoregression ($n\_lag = 10$). The model successfully reduces autocorrelation, with all lags falling within the 95% confidence interval, indicating effective short-term temporal decorrelation.

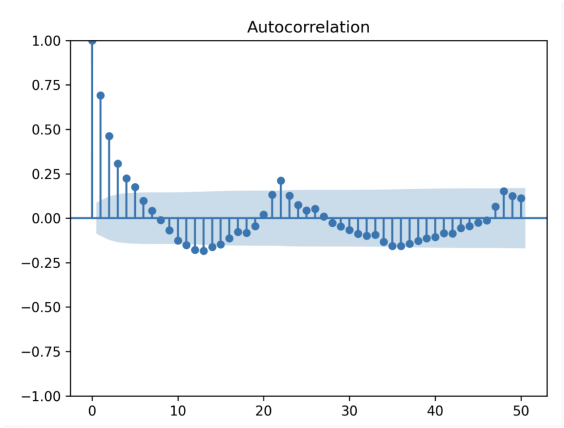

Figure 14: Correlogram for $PM_{2.5}$ **before** autoregression. Significant lags end at lag 5, where autocorrelation drops within the confidence bounds. This suggests limited short-range dependency, making $n\_lag = 5$ appropriate.

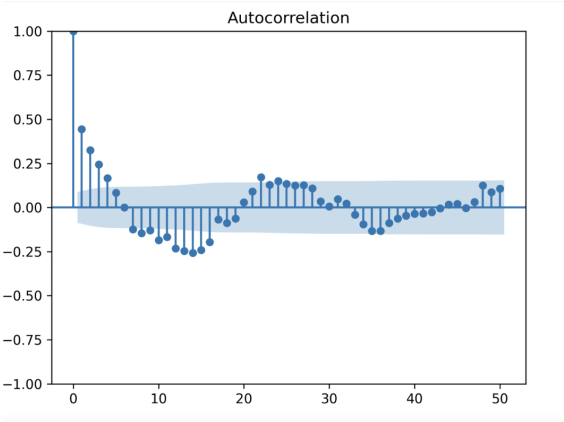

Figure 15: Correlogram for PM_{10} **before** autoregression. The strong autocorrelation mostly ends around lag 5–6. After that, autocorrelation exists but is weaker and falls within the confidence interval. This justifies selecting $n\_lag = 5$ to capture the core temporal structure while avoiding noise from insignificant lags.

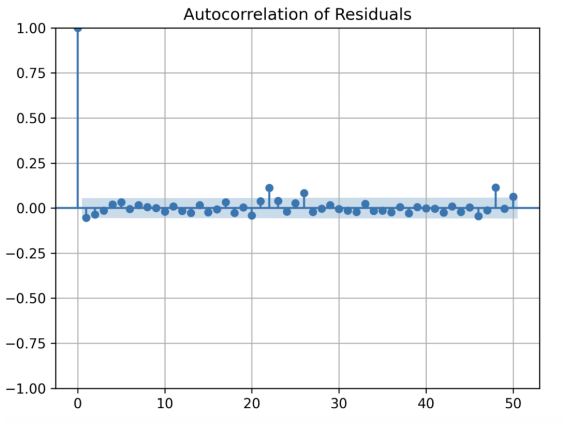

Figure 16: Correlogram for $PM_{2.5}$ **after** autoregression ($n\_lag = 5$). All residuals lie within the confidence band, indicating the autoregressive structure was well captured.

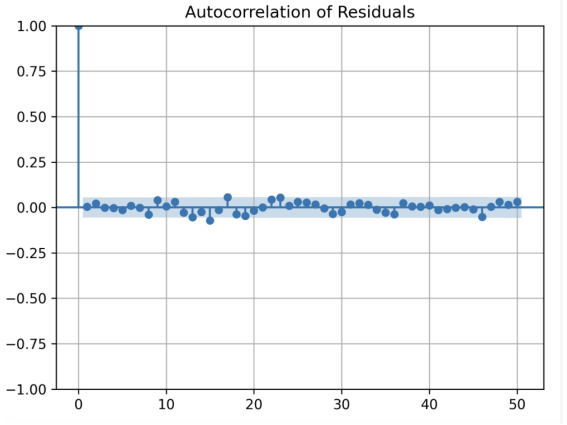

Figure 17: Correlogram for PM_{10} **after** autoregression ($n\_lag = 5$). The residual autocorrelation is removed, with all lags now within the 95% confidence interval. This suggests the model successfully captured the relevant lagged structure.

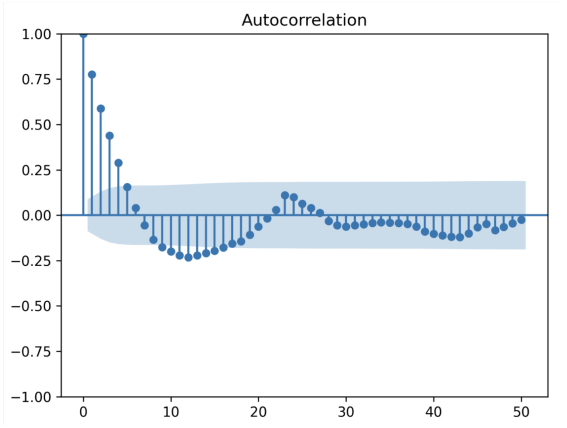

Figure 18: Correlogram for VOC **before** autoregression. Autocorrelations become negligible after lag 5, indicating minimal memory. $n\_lag = 5$ efficiently captures the relevant structure.

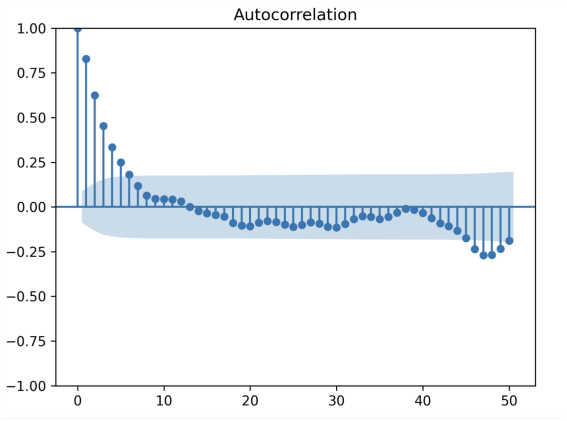

Figure 19: Correlogram for HCHO **before** autoregression. The ACF shows dependency up to lag 6. Beyond that, values stabilise, justifying $n\_lag = 5$.

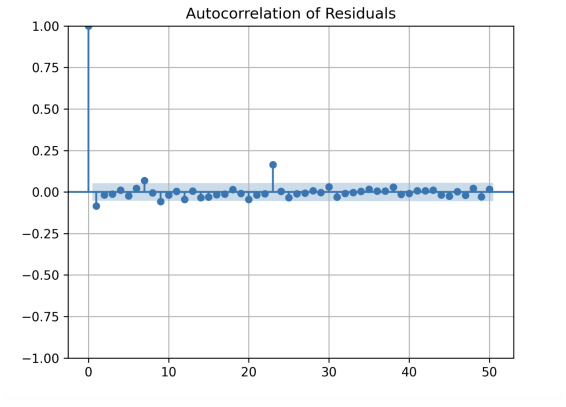

Figure 20: Correlogram for VOC **after** autoregression ($n\_lag = 5$). All points fall within the confidence bounds, confirming effective residual whitening.

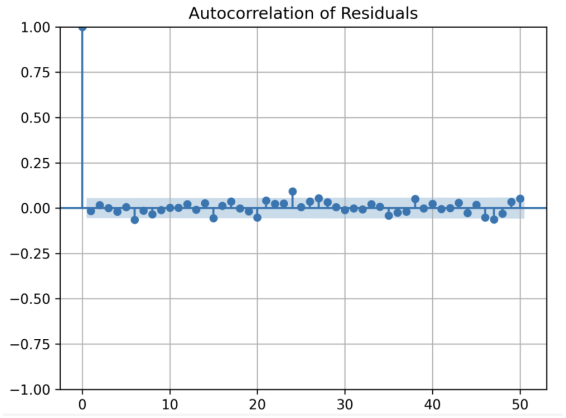

Figure 21: Correlogram for HCHO **after** autoregression ($n\_lag = 5$). Temporal structure is successfully removed, and residuals fall entirely within confidence limits.

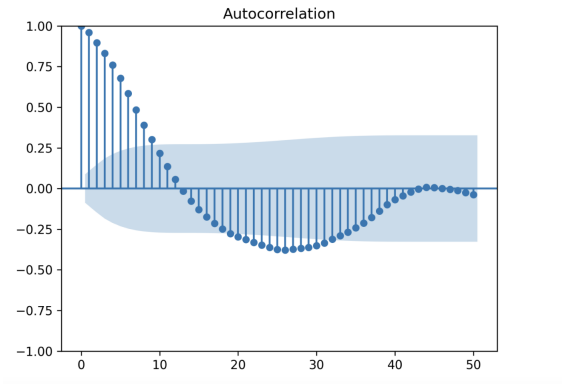

Figure 22: Correlogram for Humidity **before** autoregression. A distinct cyclical pattern is evident, with significant peaks near lag 24 — reflecting daily seasonality. To capture this behaviour, $n\_lag = 24$ was selected.

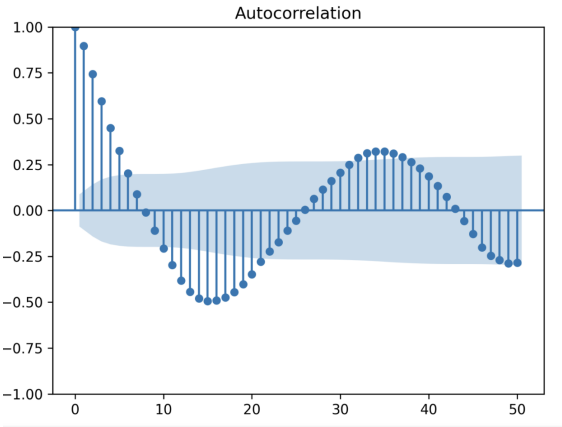

Figure 23: Correlogram for Temperature *before* autoregression. The gradually decaying and oscillating pattern suggests underlying periodicity. Autocorrelation becomes statistically insignificant begins around lag 21, $n\_lag = 15$ was selected as a more conservative compromise to avoid overfitting

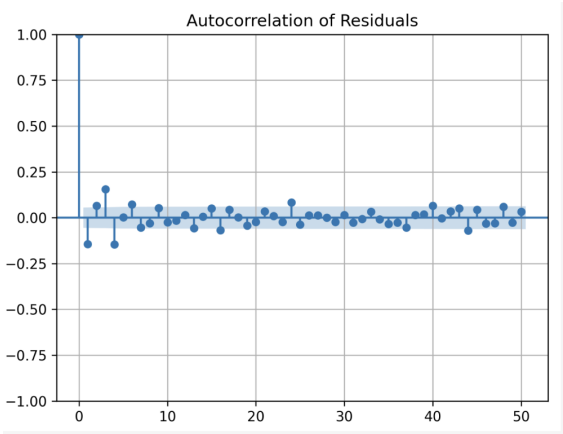

Figure 24: Correlogram for Humidity **after** autoregression ($n\_lag = 24$). The model removes the periodic component, with most lags now inside the confidence bounds, confirming successful handling of diurnal cycles.

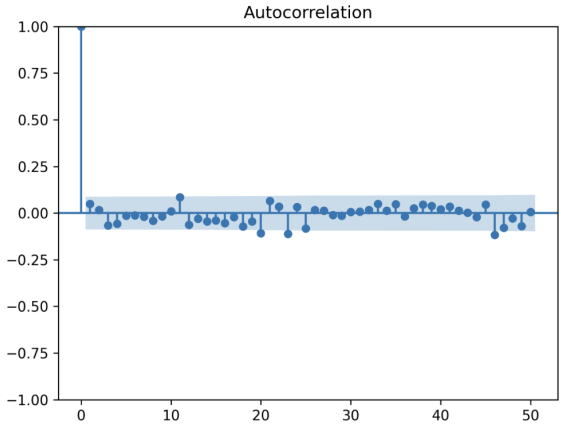

Figure 25: Correlogram for Temperature **after** autoregression ($n\_lag = 15$). The ACF is flattened, and temporal dependencies are largely removed.

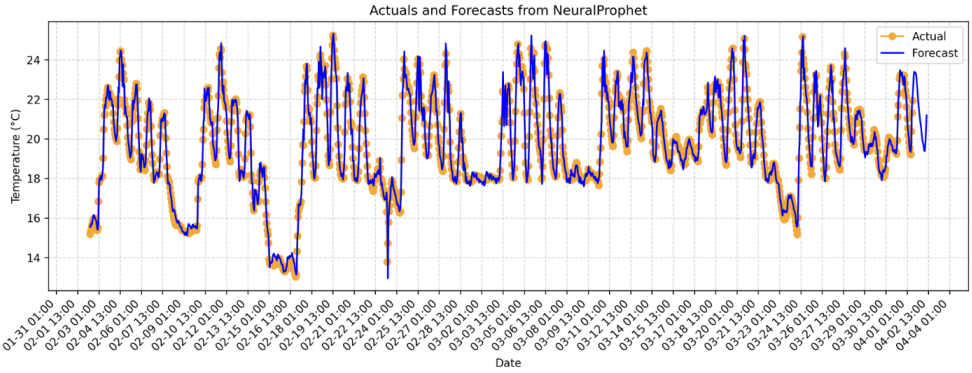

Figure 26: Forecast graph for temperature

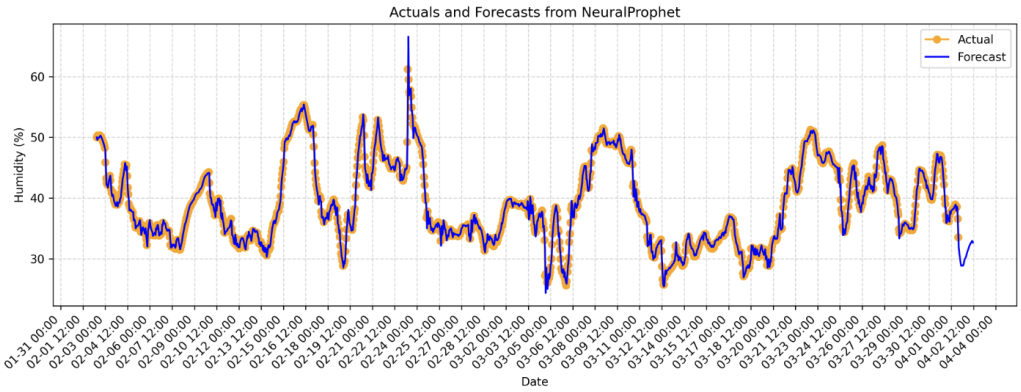

Figure 27: Forecast graph for humidity

Regarding your Master Solo Project this is just to clarify how valuable the work you are doing is to the development of IAQ monitoring as a new industry.

I would identify 3 keys aspects which I believe are capable of driving change in the arena of international property management as follows:

- Forecast Graphs: Using AI to predict IAQ metrics so clients can perform an actuation to reduce any predicted peaks.

- Threshold Detection: AI analysis with built in threshold detection (defined by the WELLS standards) to allow the user to know if high levels of a metric are expected to they can act accordingly.

- ChatBot: built in with outdoor air quality, Clients historical data and information on IAQ on the impact of energy efficiency and health. The client can ask any query and get a response on these subjects, with the purpose of great user centred design which is compact, easy to use and can answer a variety of questions.

I would therefore strongly encourage you to pursue these aspects as priority and I am sure the outcome of your academic activities will align with the developing needs of the commercial world to drive understanding of this crucial area of innovation.

Without doubt your pioneering work will be of great value to our competitive drive towards Butterfly becoming the leading IAQ brand internationally.

Signature: Nick Munro

Figure 28: Project Success Statement from CEO of Butterfly Air

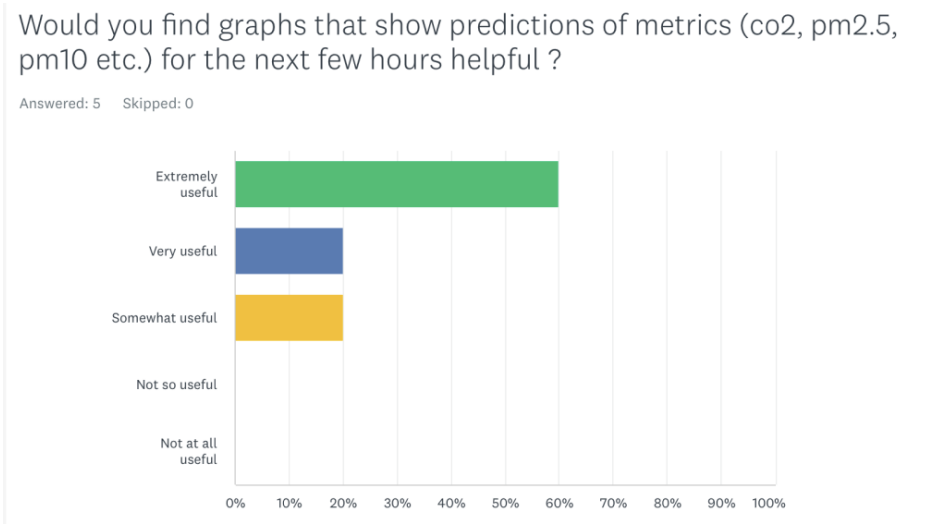

Figure 29: Client survey responses 1

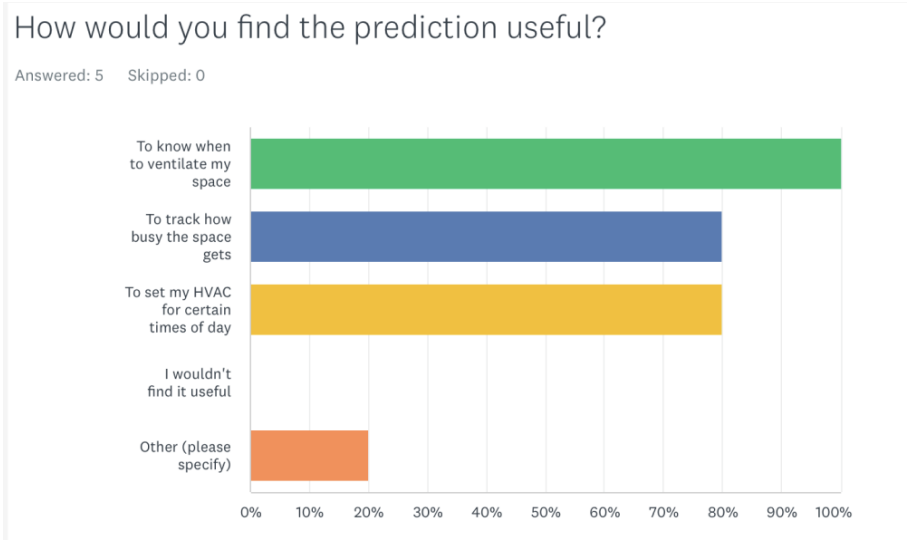

Figure 30: Client survey responses 2

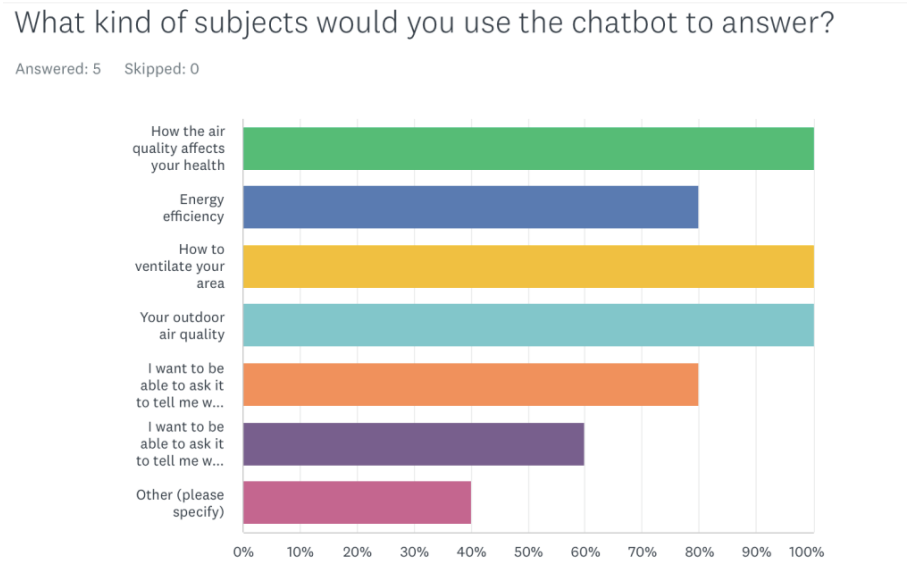

Figure 31: Client survey responses 3

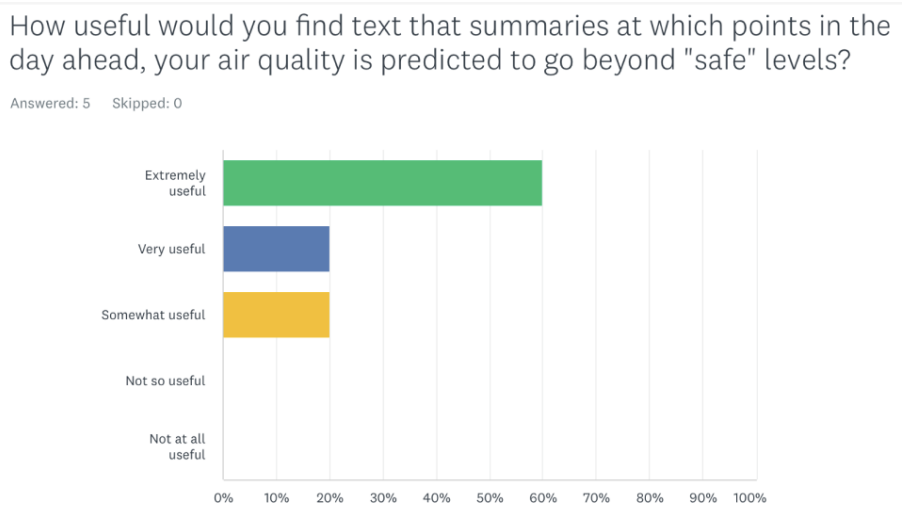

Figure 32: Client survey responses 4

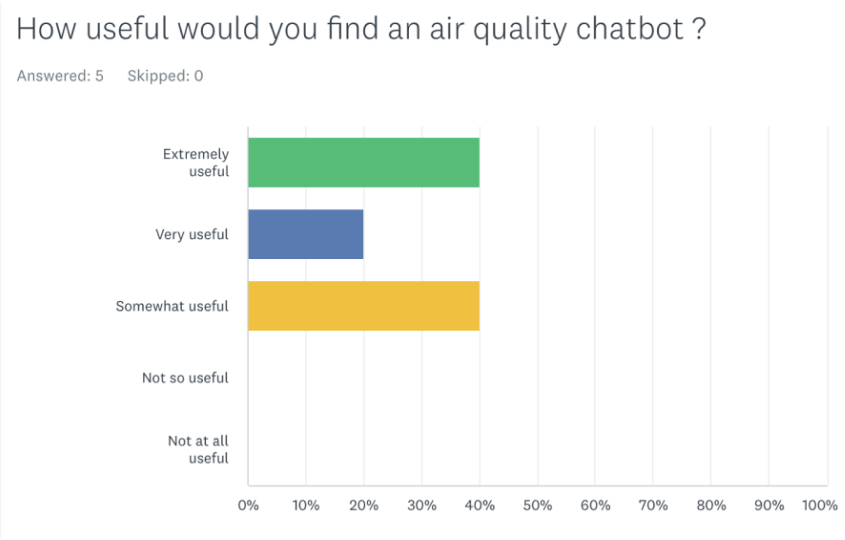

Figure 33: Client survey responses 5

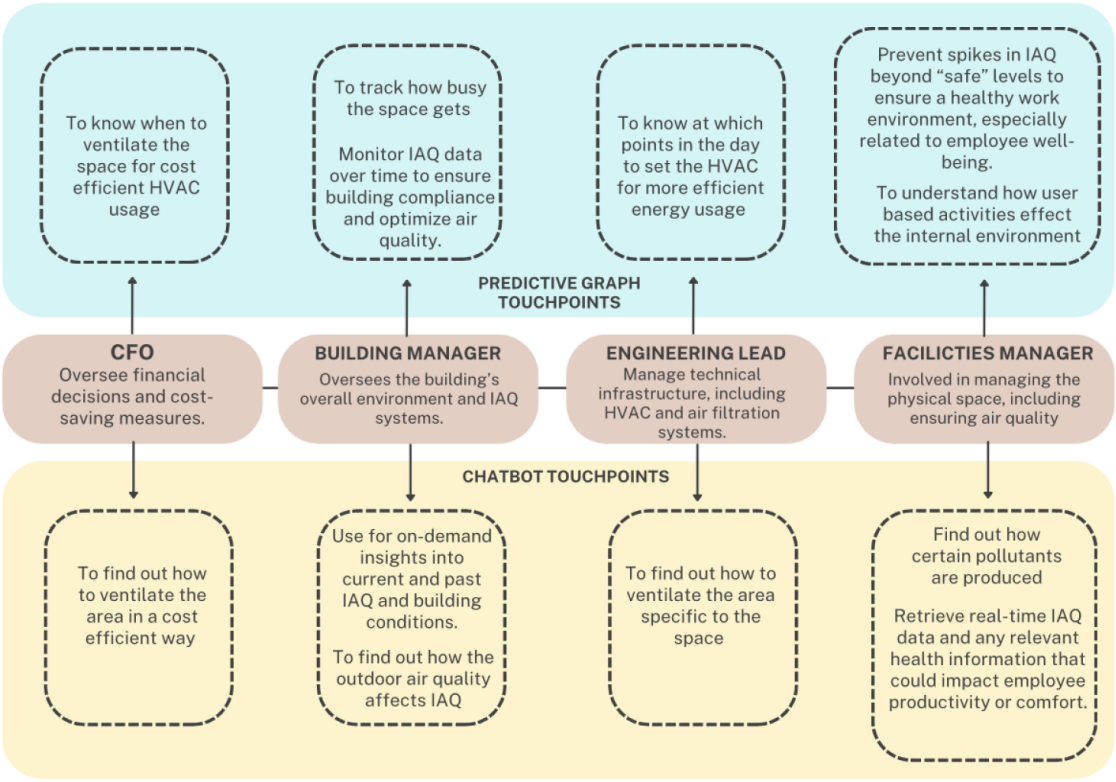

Figure 34: Product touchpoint map - Office

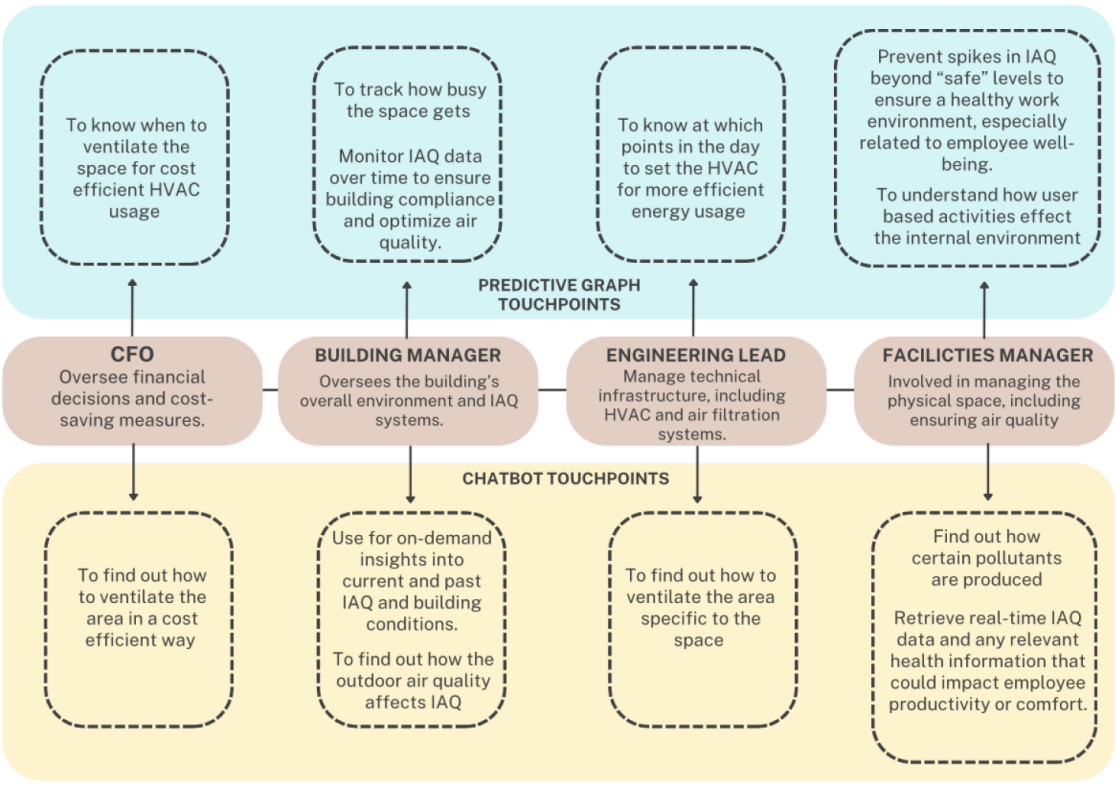

Figure 35: Product touchpoint map - Hospital

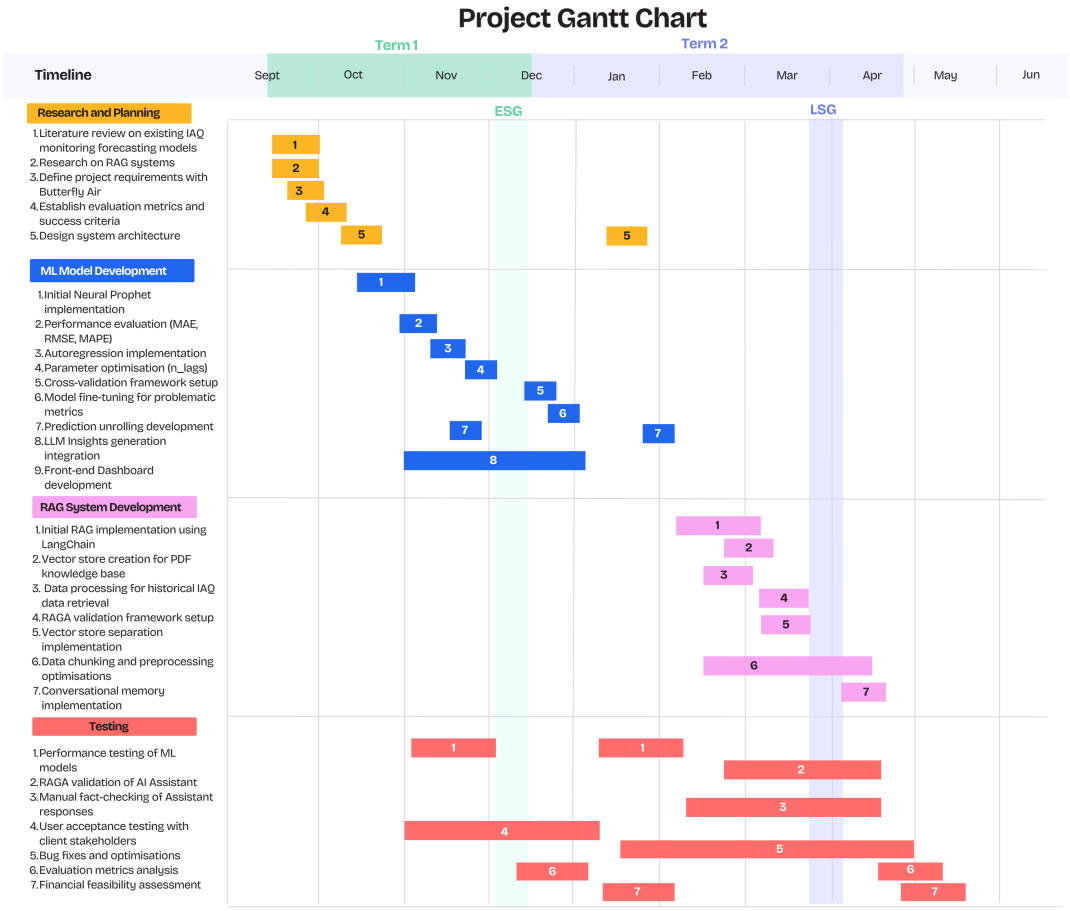

Figure 36: Project Gantt chart

8.5 Appendix E: Reflection

This project provided an invaluable opportunity to apply the design engineering process in a real-world context, balancing academic rigour with commercial viability. It required me to navigate ambiguity, make decisions under constraints, and iteratively develop a system that delivered both technical accuracy and stakeholder value.

From a methodological perspective, I followed a design engineering approach of divergent exploration followed by convergent decision-making. At the outset, I explored multiple forecasting methods—ARIMA, Prophet, LSTM, DeepAR—before converging on NeuralProphet based on a balance of performance, interpretability, and scalability. Similarly, in developing the RAG assistant, I experimented with different vector store structures, chunking strategies, and retrieval methods before converging on a multi-store hybrid retrieval pipeline. These processes reflect the creative generation of alternatives, followed by critical selection based on contextual constraints.

Iteration was central throughout. I repeatedly refined both systems in response to validation results, stakeholder feedback, and scalability considerations. In the forecasting component, I implemented autoregression, unrolling, and fine-tuning techniques, iteratively optimising each model until it met the 90%+ accuracy threshold within the compute limitations defined by Butterfly. For the AI assistant, performance improved significantly through targeted debugging of hallucinations, format adaptation of the JSON data, and hybridising timestamp lookups with semantic retrieval. These iterative refinements mirror the design engineering principle of optimisation within constraints, where effectiveness is maximised without compromising feasibility.

A particularly formative moment came when I decided to hand over parts of the implementation to Butterfly's engineering team earlier than planned. This required me to package my code and rationale clearly to enable independent testing and highlighted the importance of modular, documented systems. Doing so allowed for parallel feasibility evaluation and deepened my understanding of stakeholder dynamics and time-sensitive development cycles.

Reflecting on my professional development, this project strengthened my systems thinking, technical communication, and strategic decision-making. I gained confidence in identifying when to prioritise accuracy, speed, or maintainability based on stakeholder goals and learned how to scope work that remains extensible. At the same time, I recognised areas for growth: earlier planning for scalability, stronger timeboxing of exploratory phases, and improved documentation of design rationale for future users.

To benchmark my work, I compared the final solution against both academic standards (e.g. model validation metrics, reproducibility) and industry benchmarks (e.g. scalability, UX, deployment feasibility). This allowed me to critically evaluate strengths, like predictive accuracy and retrieval faithfulness, and identify remaining gaps, such as integration of predictive data into the chatbot.

In summary, this project embodied the full arc of the design engineering process: from contextual exploration and method selection, through iterative development and

optimisation, to stakeholder collaboration and reflection on impact. It provided a strong foundation for delivering future AI-enabled tools that are both technically robust and aligned with real-world needs.

For a structured timeline of work progression and milestones, see the Gantt chart in Appendix D.

9 Bibliography

Allen, J. G., MacNaughton, P., Satish, U., Santanam, S., Vallarino, J., & Spengler, J. D. (2016). Associations of cognitive function scores with carbon dioxide, ventilation, and volatile organic compound exposures in office workers: A controlled exposure study of green and conventional office environments. *Environmental Health Perspectives*, 124(6), 805–812.

AI21 Labs. (2024). *AI Model Index for Reasoning Performance.* `https://www.ai21.com/blog/ai-model-index`

Anthropic. (2024). *Claude API Documentation and Pricing.* `https://www.anthropic.com/api`

Awair. (2023). *Enterprise Indoor Air Quality Monitoring Solutions.* Retrieved April 15, 2025, from `https://www.getawair.com/business`

Benidis, K., Rangapuram, S. S., Flunkert, V., Wang, B., Maddix, D., Turkmen, C., Gasthaus, J., Bohlke-Schneider, M., Salinas, D., Stella, L., Callot, L., & Januschowski, T. (2020). Neural forecasting: Introduction and literature overview. *arXiv preprint arXiv:2004.10240.* `https://arxiv.org/abs/2004.10240`

Bergmeir, C., & Benítez, J. M. (2012). *On the use of cross-validation for time series predictor evaluation. Information Sciences*, 191, 192–213. `https://doi.org/10.1016/j.ins.2011.12.028`

Bergmeir, C., Hyndman, R. J., & Koo, B. (2018). A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis*, 120, 70–83. `https://doi.org/10.1016/j.csda.2017.11.003`

Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2019). *Time series analysis: Forecasting and control* (5th ed.). John Wiley & Sons.

Butterfly Air. (2024). *Butterfly Air – Indoor Air Quality Monitoring.* Available at: `https://www.butterfly-air.com`

Carre, A., & Hill, J. (2021). Indoor air quality and public health outcomes: A comprehensive review. *Environmental Health Perspectives*, 129(9), 096001.

Chen, L., & Zhao, T. (2023). Applying retrieval-augmented generation to building efficiency diagnostics. *Energy and Buildings*, 278, 112673.

Databricks. (2024). *Total Cost of Ownership for On-Premise LLM Deployment.* `https://www.databricks.com/resources/ebook/llm-deployment-cost`

Google. (2024). *Gemini API Pricing*. https://ai.google.dev/pricing

Kaiterra. (2024). *Commercial Indoor Air Quality Monitors & IAQ Management*. Retrieved April 10, 2025, from https://www.kaiterra.com/en/commercial

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W. (2020). Dense Passage Retrieval for Open-Domain Question Answering. *Proceedings of EMNLP*, 6769–6781. https://aclanthology.org/2020.emnlp-main.550/

Krebs, B. (2025). Experts Flag Security, Privacy Risks in Deepseek AI App. *KrebsOnSecurity*. https://krebsonsecurity.com/2025/02/experts-flag-security-privacy-risks-in-deepseek-ai-app/

LangChain. (2023). *LangChain: Building applications with LLMs through composability*. GitHub. https://github.com/langchain-ai/langchain

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2021). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *NeurIPS*, 33, 9459–9474. https://proceedings.neurips.cc/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf

Liu, J., Wong, C. K., & Li, K. (2023). Long short-term memory networks for indoor CO2 prediction in smart buildings. *Building and Environment*, 226, 109860.

LM Arena. (2024). *Language Model Performance Leaderboard*. https://lmarena.ai/?leaderboard

Microsoft. (2024). *Azure OpenAI Service pricing*. https://azure.microsoft.com/en-gb/pricing/details/cognitive-services/openai-service/

Microsoft Azure. (2025). *Azure Blob Storage Pricing – Standard Tier, LRS Redundancy*. Retrieved April 20, 2025, from https://azure.microsoft.com/en-gb/pricing/details/storage/blobs/

Mistral AI. (2024). *Pricing*. https://mistral.ai/pricing/

OpenAI. (2024). *ChatGPT-4o [Large language model]*. https://openai.com/index/chatgpt-4o/

OpenAI. (2024a). *OpenAI API Pricing*. https://platform.openai.com/docs/pricing

Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2019). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*.

RAGAS. (2024). *RAGAS: Evaluation framework for your RAG pipelines.* https://docs.ragas.io/en/stable/

Rahman, A., Singh, P., & Martinez, J. (2024). Comprehensive indoor air quality prediction: Challenges and opportunities. *Journal of Building Engineering*, 85, 105421.

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *Proceedings of EMNLP-IJCNLP*, 3982–3992. https://aclanthology.org/D19-1410/

Royal College of Physicians. (2020). *The inside story: Health effects of indoor air quality on children and young people.* Royal College of Physicians, London.

Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3), 1181–1191. https://doi.org/10.1016/j.ijforecast.2019.07.001

Taylor, S. J., & Letham, B. (2018). Forecasting at scale. *The American Statistician*, 72(1), 37–45. https://doi.org/10.1080/00031305.2017.1380080

Triebe, O., Hewamalage, H., Pilyugina, P., Laptev, N., Bergmeir, C., & Rajagopal, R. (2021). NeuralProphet: Explainable forecasting at scale. *arXiv preprint arXiv:2111.15397*.

Wang, H., & Shen, L. (2022). Ensemble methods for indoor air quality prediction in smart buildings. *Journal of Computational Design and Engineering*, 9(3), 1026–1041.

Zhang, Y., Li, D., & Ma, Y. (2022). Commercial indoor air quality monitoring systems: A market analysis. *Sensors*, 22(4), 1488.