Internship Report

Artificial Intelligence Engineering - 2nd Year

Flight Traffic Analysis and Prediction Dashboard

Presented by:
REGRAG Anas

Supervised by:
Mr. Darnag RACHIDE

Internship performed at:

Office National Des Aéroports (ONDA)

Defended before the jury composed of:

Jury: M. Lazaar

Jury: E. Hssyani

Academic Year: 2025-2026

Acknowledgements

At the conclusion of this enriching internship experience, I wish to express my sincere and profound gratitude to all those who contributed, directly or indirectly, to the successful completion of this project and the writing of this report.

My deepest gratitude goes first and foremost to my supervisor at the Office National Des Aéroports (ONDA), **Mr. Darnag RACHIDE**. His constant availability, invaluable technical guidance, and insightful feedback were instrumental from the project's inception to its final deployment. I am particularly grateful for the trust he placed in me and for the autonomy he granted, which allowed me to explore and implement advanced machine learning solutions. His expertise was a guiding light in navigating the complexities of the flight dataset and in refining the architecture of the predictive models.

I extend my sincere thanks to the entire management of **ONDA** for providing me with this exceptional opportunity to work on a project of such practical and strategic importance. The professional and stimulating environment was highly conducive to learning and professional growth.

I would also like to thank my colleagues within the department for their warm welcome and collaborative spirit. Their support, the insightful technical discussions, and the positive atmosphere they fostered made this internship a truly valuable human and professional experience.

My gratitude also extends to the entire faculty of the **National School of Computer Science and Systems Analysis (ENSIAS)**, and particularly to the teaching staff of the **Artificial Intelligence (2IA)** program. The solid theoretical foundation in machine learning, data processing, and software engineering acquired during my studies was the bedrock upon which this project was built.

Finally, I wish to express my heartfelt thanks to my family and friends for their unwavering support, encouragement, and patience throughout this journey.

Abstract

This report details the work accomplished during my internship at the Office National Des Aéroports (ONDA). The primary objective of this project was to develop an end-to-end Flight Traffic Analysis and Prediction Platform. This involved processing extensive historical flight data, engineering relevant features to capture seasonality and the impact of the COVID-19 pandemic, and training multiple machine learning models. The final deliverable is a fully interactive web dashboard, built with Dash Plotly, that allows non-technical users to explore historical trends and generate on-demand forecasts for future flight volumes.

Keywords: Machine Learning, Time Series Forecasting, Data Visualization, Python, Scikit-learn, LightGBM, Dash Plotly, Air Traffic Management.

Résumé

Ce rapport détaille les travaux réalisés lors de mon stage à l'Office National Des Aéroports (ONDA). L'objectif principal de ce projet était de développer une plateforme complète d'analyse et de prédiction du trafic aérien.

Cela a impliqué le traitement de données de vol historiques, la création de caractéristiques pertinentes pour capturer la saisonnalité et l'impact de la pandémie de COVID-19, et l'entraînement de plusieurs modèles de machine learning.

Le livrable final est un tableau de bord web interactif, construit avec Dash Plotly, qui permet aux utilisateurs non techniques d'explorer les tendances historiques et de générer des prévisions à la demande pour les volumes de vols futurs.

Mots-clés : Apprentissage automatique, Prévision de séries temporelles, Visualisation de données, Python, Scikit-learn, LightGBM, Dash Plotly, Gestion du trafic aérien.

Contents

List of Figures

List of Tables

Chapter 1

General Introduction

The global aviation industry operates as a critical artery of the world economy, facilitating commerce, tourism, and connectivity. Its performance is a key barometer of economic health, yet it is also highly susceptible to global events. The COVID-19 pandemic represented an unprecedented disruption, grounding fleets and causing a historic contraction in air traffic. In the wake of this event, airport authorities and operators face the immense challenge of navigating a complex and volatile recovery. This new reality demands a paradigm shift from traditional, retrospective analysis to agile, data-driven, and forward-looking decision-making.

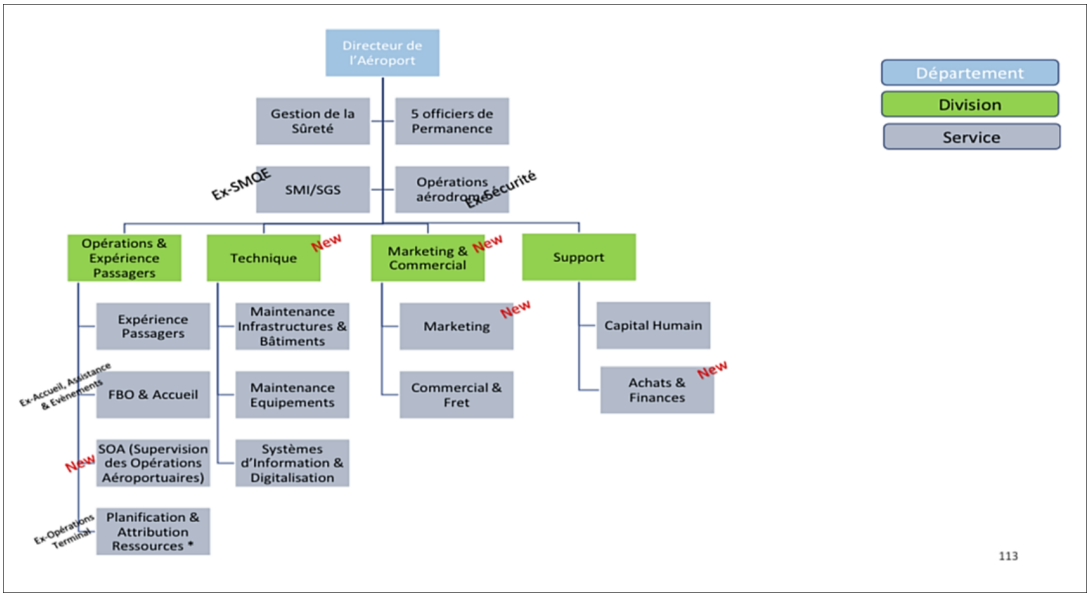
It is within this dynamic context that my end-of-year internship took place at the **Office National Des Aéroports (ONDA)**, the public establishment responsible for the management and development of Morocco's airport infrastructure. Faced with the limitations of manual data analysis and the critical need for predictive insights, ONDA presented a clear mandate: to leverage Artificial Intelligence to better understand and anticipate flight traffic dynamics. To address this challenge, I was tasked with the mission of designing, developing, and deploying an end-to-end **Flight Traffic Analysis and Prediction Dashboard**.

This project stands at the intersection of data engineering, machine learning, and software development. The primary goal was not merely to analyze historical data, but to operationalize predictive intelligence through a robust and accessible application. The core technical objectives defined for this mission were as follows:

The first objective was to develop a data processing and feature engineering pipeline capable of transforming raw time-series flight data into a structured dataset suitable for machine learning. The pipeline incorporated methods to represent cyclical temporal patterns and encode contextual factors such as the distinct phases of the COVID-19 pandemic.

The second objective was to implement and evaluate several machine learning models for flight traffic forecasting. This included establishing a linear baseline using Ridge Regression and developing a non-linear model based on the LightGBM framework. Model performance was assessed through comparative analysis to determine predictive accuracy and robustness.

The final objective was to deploy the trained models in an interactive web application built with Dash Plotly. The application enabled on-demand forecasts and analytical exploration of



L'Organigrame d'ONDA

model outputs, facilitating data-driven decision-making for end users.

This report will detail the complete methodology, from initial problem definition to final deployment, following the Cross-Industry Standard Process for Data Mining (CRISP-DM). The document is structured as follows:

- **Chapter 2** Presents the general context of the project, including a detailed presentation of ONDA, the project's scope, and the work methodology adopted.
- **Chapter 3** Details the analysis of the existing system and the comprehensive specification of the functional and non-functional requirements for the new solution.
- **Chapter 4** Outlines the design phase, justifying the key technical choices and presenting the global system architecture.
- **Chapter 5** Describes the implementation of the solution, including the technology stack, the technical architecture, and the results obtained from the predictive models.
- Finally, a **General Conclusion** summarizes the achievements of the project, discusses the challenges encountered, and proposes perspectives for future work.

Chapter 2

General Project Context

2.1 Project Presentation

The project undertaken during this internship, titled ”Flight Traffic Analysis and Prediction Dashboard,” was conceived as an end-to-end data science solution. Its purpose is to transition ONDA from a state of retrospective data analysis to a proactive, predictive operational posture. This involves not only the creation of a machine learning model but also its successful integration into a user-centric application that delivers actionable insights.

2.1.1 Problem Statement

Prior to this project, the analysis of flight traffic data at the operational level was constrained by several methodological and infrastructural limitations that reduced analytical efficiency and predictive accuracy. The existing workflow lacked a centralized and automated analytical system, relying instead on manual data extraction and static spreadsheet reports. This approach was non-scalable, introduced frequent human error, and prevented users from performing timely exploratory analysis or generating ad hoc insights. In addition, the analytical framework was unable to model the complex, non-linear dynamics inherent in flight traffic data. The dataset contained multiple overlapping temporal patterns, including weekly and annual seasonality, as well as abrupt structural changes such as those induced by the COVID-19 pandemic. Standard linear techniques were inadequate for capturing these relationships, particularly given the presence of heteroscedasticity, where the variance in flight volume varied systematically with airport size. Finally, the analytical infrastructure lacked any predictive capability. Operational and strategic decisions were based solely on retrospective analysis and expert judgment, without a quantitative forecasting model capable of estimating future flight volumes at the airport or route level. This absence of predictive analytics limited the organization’s ability to optimize resource allocation, workforce planning, and long-term infrastructure development in a rapidly changing aviation environment.

2.1.2 Objectives

In response to the identified problem statement, a set of precise, technical objectives was formulated to guide the development of the solution. The successful completion of these objectives serves as the project's primary success criteria:

1. **Engineer a Robust Data Processing Pipeline:** The first objective was to design and implement an automated pipeline to process the raw flight data. This included cleaning, transformation, and sophisticated feature engineering to create a dataset optimized for machine learning. Key engineering tasks involved creating cyclical features (using sine/cosine transformations) to capture seasonality and developing a categorical **`pandemic_phase`** **feature to explicitly model the pandemic's non-linear impact.**

2. **Develop and Quantitatively Evaluate Predictive Models:** The second objective was to build and rigorously compare multiple regression models. This involved:

   - Establishing a robust baseline using a regularized linear model (**Ridge Regression**).
   - Implementing a high-performance, non-linear model using a gradient boosting framework (**Tuned LightGBM**) to capture the complex data dynamics.
   - Evaluating all models on a chronologically-split test set using standard metrics (R^2, MAE, RMSE) on both log-transformed and original scales to ensure a comprehensive understanding of their real-world performance.

3. **Deploy the Models into an Interactive Application:** The final and most critical objective was to operationalize the trained models. This required the development of a self-contained web application using the Dash Plotly framework. The application must provide an intuitive interface for end-users to generate on-demand predictions and visually explore the underlying data, thereby democratizing access to advanced analytics within the organization.

2.2 Work Methodology

To ensure a structured and goal-oriented approach for this data science project, the **Cross-Industry Standard Process for Data Mining (CRISP-DM)** methodology was adopted. This iterative framework is the industry standard for analytics and machine learning projects, as it provides a clear roadmap from business understanding to final deployment.

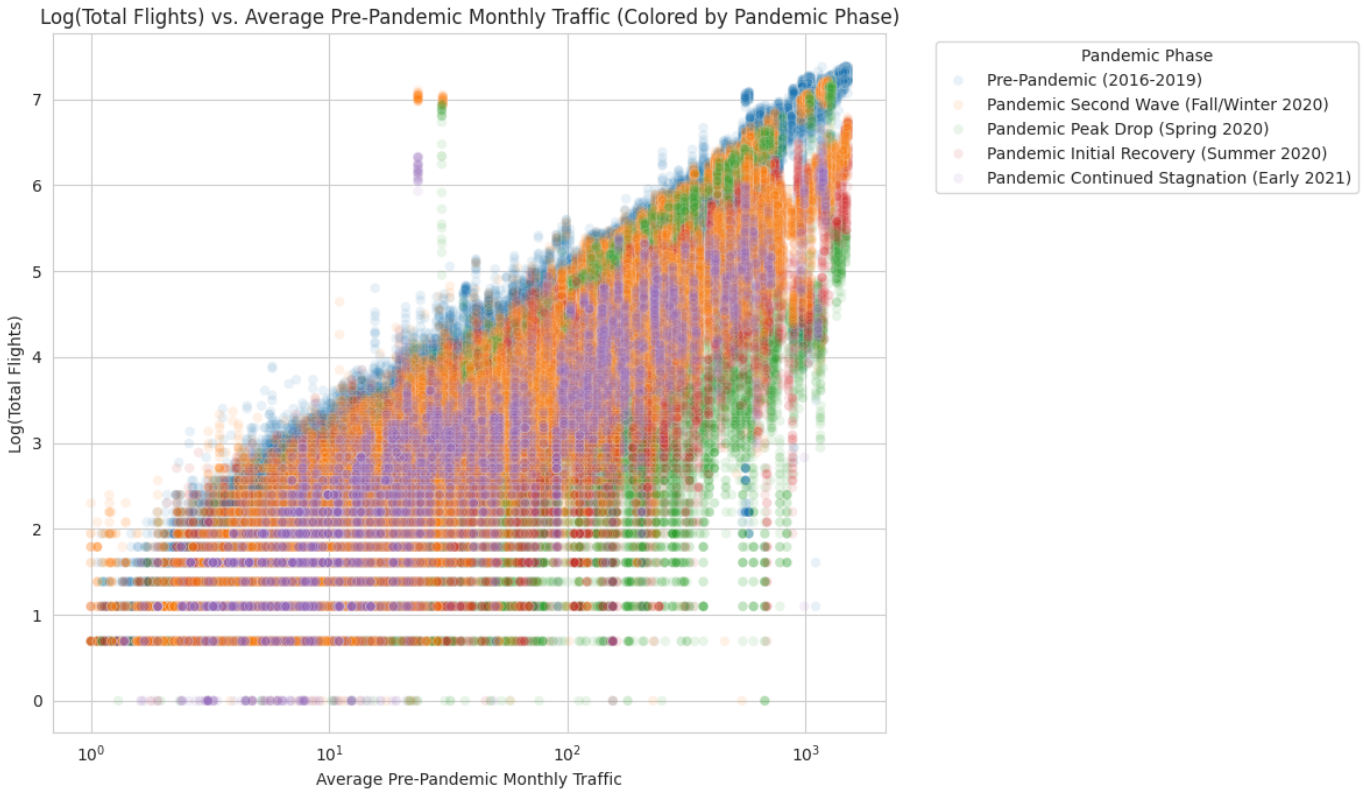
2.3 Exploratory Data Analysis

The Data Understanding phase of the CRISP-DM cycle involved a thorough Exploratory Data Analysis (EDA) to uncover patterns, validate assumptions, and guide the feature engineering process. The key findings from this analysis are presented in this section and directly informed the design of the predictive models. All visualizations are plotted on a logarithmic scale for the target variable (‘Log(Total Flights)‘) to better handle the data’s wide distribution and heteroscedasticity.

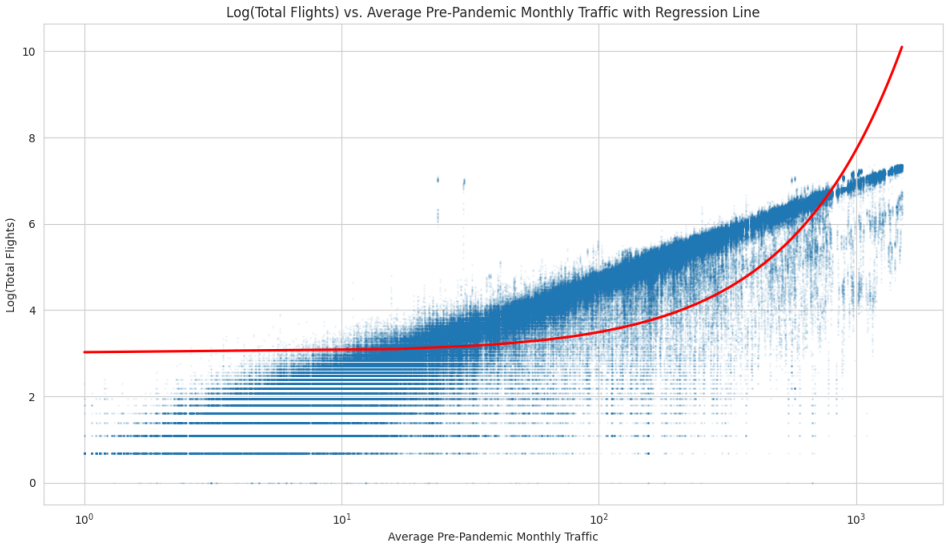
Relationship with Historical Traffic The most critical hypothesis was that an airport’s historical traffic is the strongest predictor of its current traffic. Figure 2.1 visually confirms this. Figure 2.1a shows a strong positive correlation between an airport’s average pre-pandemic traffic and its daily flight volume. More importantly, the points are stratified by color according to the pandemic phase, visually demonstrating that the ‘pandemicphase‘ feature is essential for capturing the systematic drop and recovery in traffic. Figure 2.1b reinforces this relationship but also reveals a crucial insight: the trend is non-linear. This curvature justifies the need for a non-linear model, like LightGBM, to accurately capture the data’s dynamics, as a simple linear model would introduce systematic errors.

Analysis of Categorical Variables To understand the influence of various categorical factors, several box plots were generated, as shown in Figure 2.2.

- **Impact of the Pandemic:** Figure 2.2a clearly shows the stable traffic from 2016-2019, followed by a dramatic drop in 2020. However, Figure 2.2b provides far more granular insight, pinpointing the ”Pandemic Peak Drop” as the absolute low and illustrating the subsequent volatile recovery. This validates the decision to engineer the ‘pandemicphase‘ feature rather than simply using the year.
- **Temporal and Structural Factors:** Figure 2.2c reveals that the day of the week has a relatively minor impact on flight volumes, with medians remaining stable across the week. In contrast, Figure 2.2d demonstrates a powerful, monotonic relationship between our engineered airport size category and flight traffic. This confirms that airport size is a dominant predictive feature.

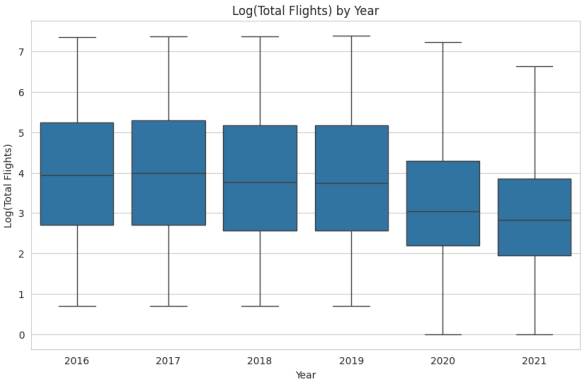


(a) Traffic colored by Pandemic Phase.

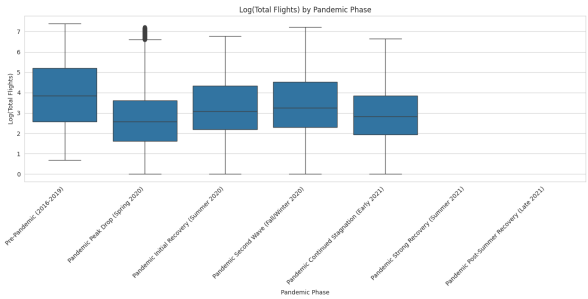


(b) Overall trend with regression line.

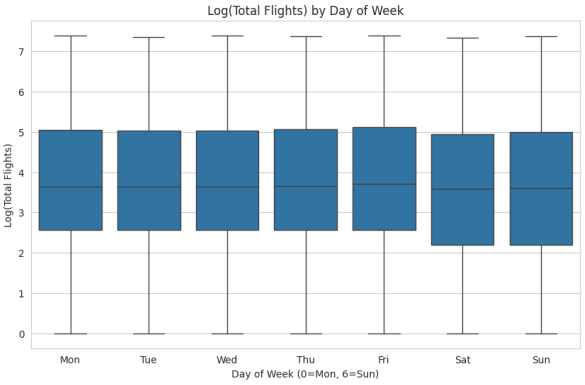
Figure 2.1: Log(Total Flights) vs. Average Pre-Pandemic Traffic.



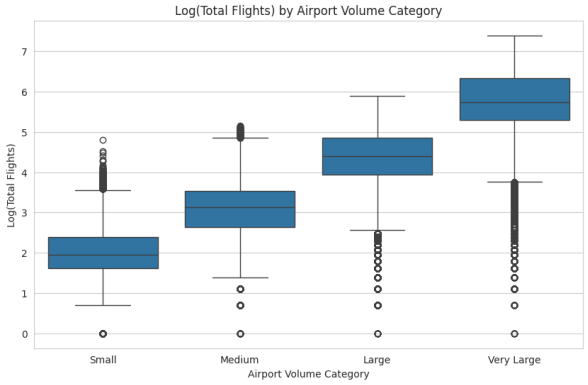
(a) Distribution by Year.



(b) Distribution by Pandemic Phase.



(c) Distribution by Day of Week.



(d) Distribution by Airport Volume Category.

Figure 2.2: Box plot analysis of key categorical features.

Chapter 3

Analysis and Requirements Specification

This chapter provides a formal analysis of the project's starting point and a detailed specification of the requirements for the developed solution. The first section deconstructs the limitations of the pre-existing workflow, establishing the technical and operational gaps. The second section translates these gaps into a concrete set of functional and non-functional requirements that served as the engineering blueprint for the project.

3.1 Analysis of the Existing Situation

The pre-existing framework for flight traffic analysis at the organization was primarily a manual and retrospective process. This system, while functional for basic historical reporting, presented significant technical and operational limitations that constrained agile decision-making and precluded any form of predictive analytics.

The typical workflow involved ad-hoc data requests fulfilled by technical teams. This process generally followed these steps: a business user would request specific data, a query would be run against a database, and the results would be exported as a static file (e.g., CSV). Subsequent analysis was then performed in spreadsheet software like Microsoft Excel.

The system exhibited several structural and methodological limitations that constrained its analytical effectiveness. High latency in data processing resulted in substantial delays between data acquisition and the generation of actionable insights, largely due to the manual and non-repeatable nature of the workflow. The system lacked scalability, as its dependence on spreadsheet-based tools restricted the volume of data that could be efficiently processed. This limitation frequently led to performance degradation and instability when handling the large-scale flight traffic dataset. Moreover, the analytical framework was entirely descriptive and lacked predictive functionality. It could summarize past trends but provided no mechanism to extrapolate future traffic volumes based on historical patterns, a critical requirement in post-pandemic forecasting. The analytical depth was also limited, as the tools in use could not capture non-linear dependencies, cyclical variations, or heteroscedastic behavior within the data,

resulting in incomplete representations of underlying dynamics. Finally, reporting outputs were static and non-interactive, offering end-users fixed visualizations without the ability to perform independent exploration or parameterized queries. This absence of interactivity restricted the capacity for data-driven investigation and iterative analytical refinement.

3.2 Requirements Specification

To overcome these limitations, a set of precise functional and non-functional requirements was defined. These requirements guided the design and implementation of the new system, ensuring it would meet both technical standards and user needs.

3.2.1 Functional Requirements

Functional requirements define the specific behaviors and features the system must provide. They answer the question: ”What must the solution do?”

FR-1: Dynamic Data Visualization: The system must render fully interactive visualizations of historical flight traffic data, enabling users to examine temporal and spatial trends at multiple levels of granularity. Users must be able to apply filters that dynamically adjust the entire dashboard view based on a selected date range, airport, or other contextual dimensions such as airline or flight type. The visualization layer should update instantaneously in response to user input, ensuring that patterns, anomalies, and seasonality effects can be explored intuitively without requiring manual data manipulation or static report generation.

FR-2: Key Performance Indicator (KPI) Dashboard: The central dashboard must present a concise but comprehensive overview of operational performance through aggregated Key Performance Indicators (KPIs), including Total Operations, Arrivals, and Departures. In addition to these core metrics, the system must compute and display comparative indicators that quantify changes in traffic relative to pre-pandemic baselines (specifically the equivalent period in 2019). These indicators must be automatically recalculated as the user adjusts the date range or filtering parameters, allowing for immediate contextual interpretation of current performance against historical norms.

FR-3: On-Demand Predictive Forecasting: The application must provide a dedicated interface for generating flight traffic forecasts on demand. Users must be able to select a forecasting model from the available machine learning options, specify an airport or geographic region of interest, and define a future date or time horizon. Upon execution, the system must return a quantitative prediction of expected flight volume, accompanied by confidence intervals or uncertainty estimates where applicable. This functionality is intended to

support scenario analysis, operational planning, and data-driven decision-making in volatile or rapidly changing market conditions.

FR-4: Model Performance Transparency: To ensure interpretability and trust in model outputs, the application must include a dedicated section presenting detailed performance metrics for all implemented machine learning models. These metrics—such as the coefficient of determination (R^2), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE)—must be reported for both training and testing datasets. Where relevant, additional diagnostic information (e.g., feature importance rankings or residual plots) should be available to allow technical users to assess model validity, overfitting, and generalization performance.

FR-5: Bilingual User Interface (Internationalization): The system's user interface must fully support bilingual operation in English and French. All textual elements—including titles, axis labels, tooltips, navigation menus, and embedded messages—must switch seamlessly between the two languages without loss of formatting or functionality. Language selection should persist across sessions and apply consistently to all application components, ensuring accessibility and usability for a linguistically diverse user base across international contexts.

3.2.2 Non-Functional Requirements

Non-functional requirements define the quality attributes of the system. They answer the question: "How well must the solution perform its functions?"

NFR-1: Performance: The application must be highly responsive. All dashboard components, including charts and KPIs, must render within 5 seconds of a user interaction (e.g., changing a date filter). Model predictions must also be delivered in near-real-time (under 3 seconds).

NFR-2: Usability: The Graphical User Interface (GUI) must be clean, intuitive, and designed for non-technical users. The workflow for generating a prediction or filtering data must be self-explanatory with minimal to no training required.

NFR-3: Reliability: The system must be reliable. Data displayed must be accurate and consistent with the source. Predictions generated by the models for the same inputs must be deterministic and reproducible.

NFR-4: Maintainability: The source code for the entire application (both the training pipeline and the Dash app) must be modular, well-documented, and adhere to software engineering best practices to facilitate future updates, bug fixes, and feature additions.

Chapter 4

Design

Following the analysis and specification of requirements, the design phase focused on architecting a solution that was not only functional but also robust, scalable, and maintainable. This chapter details the two core pillars of the design process: the selection and justification of the technology stack, and the modeling of the global system architecture, which is foundational to the project's performance and reliability.

4.1 Technical Choices

The selectioneof technologies for this project was driven by a strategy of leveraging a unified, high-performance ecosystem to ensure a seamless workflow from data experimentation to production deployment. Each component was chosen for its specific strengths in addressing the project's requirements. Figure 4.1 illustrates the primary tools used.

- **Python as the Core Language:** The decision to build the entire solution in Python was fundamental. As the de facto language for machine learning, it provides an unparalleled ecosystem of libraries. More importantly, it enables a unified development environment for every stage of the project: data ingestion (Pandas), model training (Scikit-learn, LightGBM), and web deployment (Dash). This eliminates the friction and complexity of integrating disparate systems written in different languages, thereby accelerating development and simplifying long-term maintenance.
- **Pandas for Data Manipulation:** For handling the tabular flight data, Pandas was the obvious choice. Its core 'DataFrame' structure is highly optimized for vectorized operations, allowing for efficient data cleaning, transformation, and feature engineering on large datasets without resorting to slow, iterative loops. This focus on performance is critical for building a scalable data pipeline.

Python Pandas Scikit-learn

LightGBM Dash Plotly

Figure 4.1: Core components of the project's technology stack.

- **Scikit-learn for the Modeling Workflow:** Scikit-learn was chosen not just for its model implementations, but for its powerful workflow engineering tools. The use of the `Pipeline` and `ColumnTransformer` objects was central to the design. This approach encapsulates the entire preprocessing and modeling sequence into a single, serializable object. This is a critical engineering practice that guarantees reproducibility and prevents data leakage, ensuring that the exact same transformations are applied during training and live inference.
- **LightGBM for Advanced Modeling:** While Scikit-learn provided a solid baseline with Ridge Regression, the data's complex, non-linear patterns necessitated a more powerful model. LightGBM was selected over other gradient boosting frameworks due to its superior performance in terms of both training speed and memory efficiency. Its leaf-wise tree growth algorithm makes it exceptionally fast on large datasets, a key consideration for future scalability.
- **Dash Plotly for Deployment:** To meet the requirement of an interactive application, Dash was the optimal choice. Its pure Python nature allowed me to build a sophisticated web front-end without writing a single line of JavaScript. This drastically reduced development complexity and allowed for seamless integration of the trained Scikit-learn pipelines directly into the application's backend logic. The rich, interactive charting capabilities of Plotly provided the required level of data exploration for the end-user, fulfilling a core functional requirement.

4.2 Modeling & Architecture

The system's architecture was designed around a core MLOps principle: the strict separation of the **offline training pipeline** from the **online inference application**. This two-part architecture, illustrated in Figure 4.2, is essential for building a scalable and reliable machine learning system.

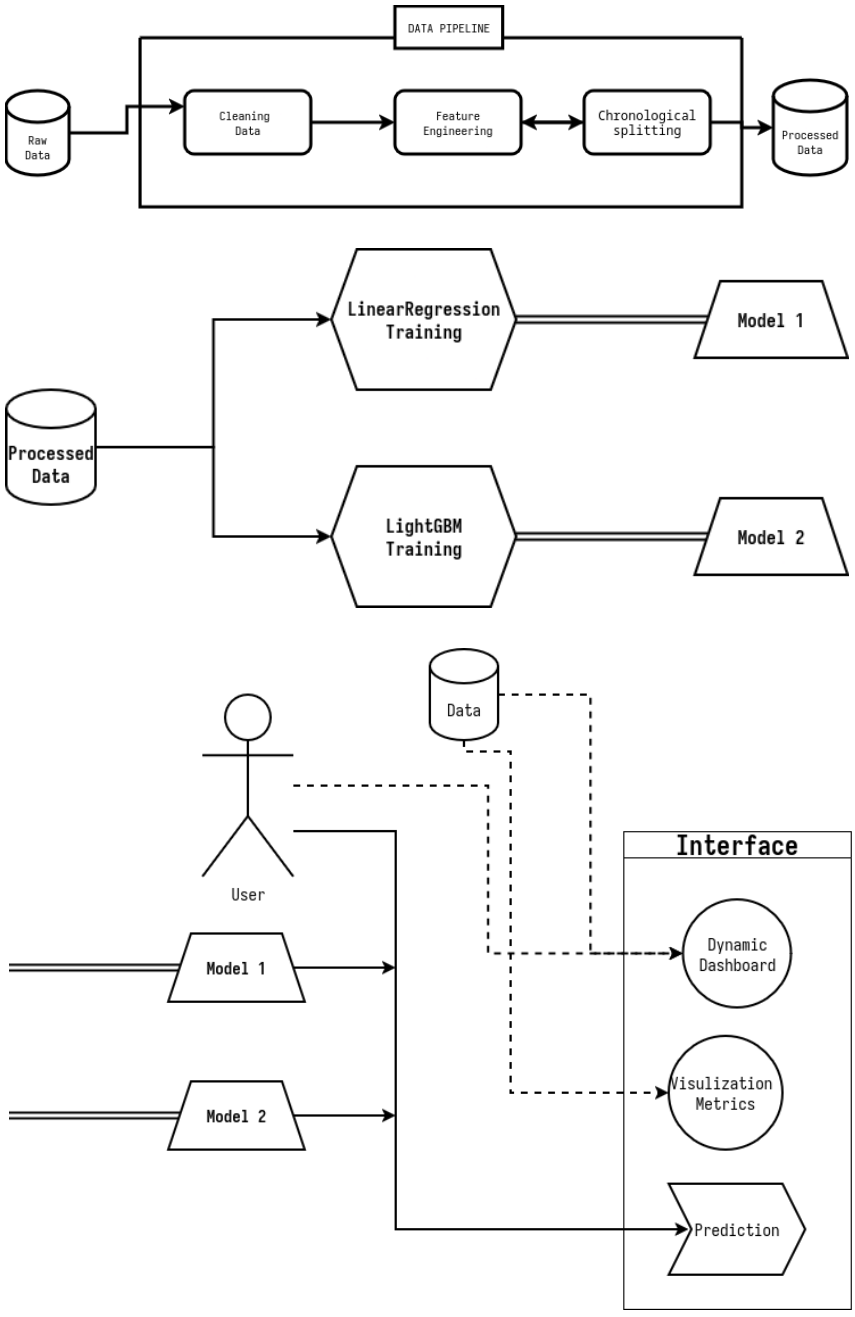


Figure 4.2: Global System Architecture: The offline training pipeline produces a model artifact that is consumed by the online inference application.

Offline Training Pipeline (`train_model.py`) This component is a standalone, executable script responsible for all computationally intensive tasks. Its sole purpose is to produce a single, deployable model artifact.

- **Responsibilities:** It handles the ingestion of the full raw dataset, executes the entire feature engineering pipeline, trains multiple machine learning models, and performs a rigorous evaluation to compare their performance.
- **Execution:** This pipeline is designed to be run periodically (e.g., weekly or monthly) to retrain the models on new data. It is not executed in real-time.
- **Output:** Its final output is a single serialized file (`flight_prediction_models.joblib`). This artifact contains not just the trained model weights, but the entire fitted Scikit-learn 'Pipeline' object, including the data preprocessors and all necessary helper objects.

Online Inference Application (`app.py`) This component is the lightweight, user-facing Dash web application. It is designed for high performance and low latency.

- **Responsibilities:** On startup, the application loads the pre-trained model artifact from the offline pipeline into memory. Its primary role is to serve user requests. When a user submits data through the web interface (e.g., an airport and a date), the application uses the loaded pipeline to preprocess the input and generate a prediction in milliseconds.
- **Execution:** The application runs continuously as a web server, handling concurrent user requests. It performs no model training.

Justification for this Architecture This separation of concerns is a critical design choice that provides several key advantages:

- **Performance:** User requests are served instantly because the time-consuming training process is decoupled from the live application.
- **Scalability:** The inference application is lightweight and can be easily replicated to handle increased user traffic, without needing to replicate the resource-heavy training environment.
- **Reliability and Stability:** The production environment is isolated from the training environment. Any potential bugs or issues during model retraining will not cause downtime for the live user-facing application.

Chapter 5

Implementation

The implementation phase involved the translation of the architectural design and specified requirements into a functional, end-to-end machine learning system. This chapter details the practical execution of the project, from the development of the offline training pipeline to the deployment of the interactive inference application. Key code snippets are presented and discussed to illustrate the core engineering logic.

5.1 Development Tools & Technology Stack

The project was realized using a curated set of tools chosen for performance and developer efficiency. The entire stack was based in Python 3.8+.

- **Core Language:** Python.
- **Data Manipulation:** Pandas and NumPy.
- **ML Workflow:** Scikit-learn and LightGBM.
- **Web Application:** Dash by Plotly, with Dash Bootstrap Components.
- **Model Serialization:** Joblib.
- **Development Environment:** Nvim, configured as a lightweight Python IDE.

5.2 Implementation of the Offline Training Pipeline

The offline pipeline, encapsulated in the script is the engine of the system. Its implementation focused on creating a reproducible and robust workflow for feature engineering and model training.

5.2.1 Feature Engineering

A significant part of the implementation was dedicated to crafting features that could accurately capture the complex temporal dynamics of the data. Beyond simple date components, cyclical features were engineered to model seasonality without the discontinuity issues inherent in linear representations. The code in Listing 5.1 demonstrates this critical step.

```
# Create cyclical features for day of year and month
data_cleaned["day_of_year_sin"] = np.sin(
    2 * np.pi * data_cleaned["day_of_year"] / 365.25
)
data_cleaned["day_of_year_cos"] = np.cos(
    2 * np.pi * data_cleaned["day_of_year"] / 365.25
)
```

Listing 5.1: Implementation of cyclical temporal features.

Time-based variables such as `month`, `day_of_year`, or `hour` possess an inherently cyclical structure. Traditional linear encoding of these variables (e.g., assigning January = 1, February = 2, ..., December = 12) introduces artificial discontinuities, as the numerical distance between consecutive periods (e.g., December and January) is incorrectly represented as large, despite their temporal proximity. This discontinuity can distort the input space of a machine learning model, particularly for models that rely on distance metrics or assume smoothness in feature relationships.

To preserve the cyclic continuity of time, each temporal variable t with period P is transformed into a pair of orthogonal features using trigonometric projection:

$$x_{\sin} = \sin\left(\frac{2\pi t}{P}\right), \quad x_{\cos} = \cos\left(\frac{2\pi t}{P}\right)$$

This transformation maps each scalar value of t onto the unit circle in R^2, where points representing temporally adjacent values remain spatially close. The representation ensures that both the beginning and end of a cycle (e.g., December 31 and January 1) are contiguous in feature space.

The mathematical intuition behind this approach can be understood through periodicity. Since $\sin(\theta)$ and $\cos(\theta)$ repeat every 2π, the encoding preserves the inherent periodic structure of the variable. Consequently, models trained on these features can learn smooth, continuous relationships over cyclical domains without encountering artificial breaks. This encoding is particularly advantageous for gradient-based and tree-based models, where maintaining continuity in the feature space directly improves predictive performance and generalization.

Formally, the Euclidean distance between any two encoded time points t_i and t_j reflects their angular separation within the cycle:

$$d(t_i, t_j) = \sqrt{\left(\sin\left(\frac{2\pi t_i}{P}\right) - \sin\left(\frac{2\pi t_j}{P}\right)\right)^2 + \left(\cos\left(\frac{2\pi t_i}{P}\right) - \cos\left(\frac{2\pi t_j}{P}\right)\right)^2}$$

which simplifies to $d(t_i, t_j) = 2\sin\left(\pi|t_i - t_j|/P\right)$, providing a natural measure of cyclic distance. This mathematical structure underlies the robustness of cyclical feature engineering in time-series forecasting tasks.

Logarithmic Transformation

Flight traffic data, like many real-world operational datasets, often exhibits a right-skewed distribution. In such cases, the majority of observations cluster around lower values, while a small number of extreme values (e.g., exceptionally busy airports or peak travel periods) extend the tail of the distribution. This asymmetry violates the assumption of approximate normality that underlies many statistical and machine learning methods, particularly those sensitive to variance heterogeneity.

A standard method to mitigate this issue is the application of a logarithmic transformation to the target or feature variable. Given a raw variable $x > 0$, the transformation is defined as:

$$x' = \log(x + \epsilon)$$

where ϵ is a small positive constant (typically $\epsilon = 1$) included to avoid the undefined case when $x = 0$. This transformation has the following desirable mathematical properties:

(a) It compresses the scale of large values, reducing the influence of extreme outliers on the model.

(b) It expands the scale of small values, improving the representation of low-volume observations.

(c) It stabilizes variance and reduces heteroscedasticity, making the data more homoscedastic (i.e., variance independent of the mean).

Formally, for a positively skewed variable X with mean μ and standard deviation σ, the logarithmic transformation approximates a normal distribution when the coefficient of variation $\frac{\sigma}{\mu}$ is large. If X follows a log-normal distribution, then:

$$Y = \log(X) \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$$

where $\mu_Y = E[\log(X)]$ and $\sigma_Y^2 = \mathrm{Var}[\log(X)]$. In this form, statistical learning models can operate on Y with improved numerical stability and reduced sensitivity to extreme values.

This transformation directly improves the interpretability and performance of predictive models such as Ridge Regression and LightGBM. In regression tasks, it ensures that the residuals are more normally distributed and that model errors are penalized more uniformly across the range of observed values. In tree-based models, it reduces the bias toward splitting on extreme values and promotes more balanced partitioning of the feature space.

5.2.2 Preprocessing and Modeling Pipeline

To ensure robustness and prevent data leakage, the entire preprocessing and modeling workflow was encapsulated within a Scikit-learn ‘Pipeline‘. A ‘ColumnTransformer‘ was implemented to apply different transformations to numerical and categorical features in a single, unified step. This is a critical MLOps practice for creating production-ready models.

```
# Define which columns are numerical and categorical
numerical_features = ["YEAR", "day_of_month",
   "day_of_year_sin", ...]
categorical_features = ["APT_ICAO", "day_of_week",
   "pandemic_phase", ...]

# Create the preprocessing pipeline
preprocessor = ColumnTransformer(
    transformers=[
        ("num", StandardScaler(), numerical_features),
        ("cat", OneHotEncoder(handle_unknown="ignore"),
   categorical_features),
    ]
)

# Create the full model pipeline with a regressor
ridge_pipeline = Pipeline(
    steps=[
        ("preprocessor", preprocessor),
        ("regressor", Ridge(alpha=1.0)),
    ]
)
```

Listing 5.2: Definition of the Scikit-learn preprocessing pipeline.

5.3 Implementation of the Online Inference Application

The ‘app.py‘ script contains the Dash application, which serves as the user-facing front-end. Its implementation is centered around loading the pre-trained artifact and using Dash callbacks to create an interactive experience.

5.3.1 Model Loading

Upon starting, the application loads the serialized model artifact into memory. This is a one-time operation that ensures subsequent prediction requests can be handled with minimal latency. Listing 5.3 shows this initialization process, which includes error handling in case the model file is not found.

```
# Load all pre-trained models and data artifacts
try:
    artifacts =
   joblib.load("models/flight_prediction_models.joblib")
    model_artifacts = artifacts["models"]
    data_artifacts = artifacts["data"]
except FileNotFoundError:
    # Handle error if model file is not found
    print("ERROR: Model file not found.")
    exit()
```

Listing 5.3: Loading the pre-trained model artifact at application startup.

5.3.2 Interactive Prediction Callback

The core interactivity of the application is powered by Dash callbacks. The prediction functionality is implemented in a single callback that listens for a button click. It gathers inputs from the UI (model choice, airport, date), invokes the selected model’s ‘.predict()‘ method, and displays the result. Listing 5.4 shows a simplified version of this key function.

```
@app.callback(
    Output("prediction-output", "children"),
    Input("predict-button", "n_clicks"),
    [
        State("model-predict-selector", "value"),
        State("airport-dropdown", "value"),
        State("date-picker", "date"),
    ],
    prevent_initial_call=True,
)
def update_prediction(n_clicks, model_name, airport_icao,
   date_str):
    # 1. Create a single-row DataFrame from user input
    input_df = create_sample_input(airport_icao, date_str)

    # 2. Select the chosen model object
    model_obj = model_artifacts[model_name]["model"]

    # 3. Perform inference
    predicted_log = model_obj.predict(input_df)

    # 4. Back-transform and format the result
    predicted_flights = np.expm1(predicted_log)[0]

    return f"Predicted Total Flights: {predicted_flights:.0f}"
```

Listing 5.4: The Dash callback for on-demand prediction.

This event-driven architecture allows the application to remain lightweight and responsive, performing computations only when explicitly requested by the user. The final implemented dashboard, shown in Figure **??**, successfully integrates all these components into a cohesive user experience.

5.4 Implementation Metrics

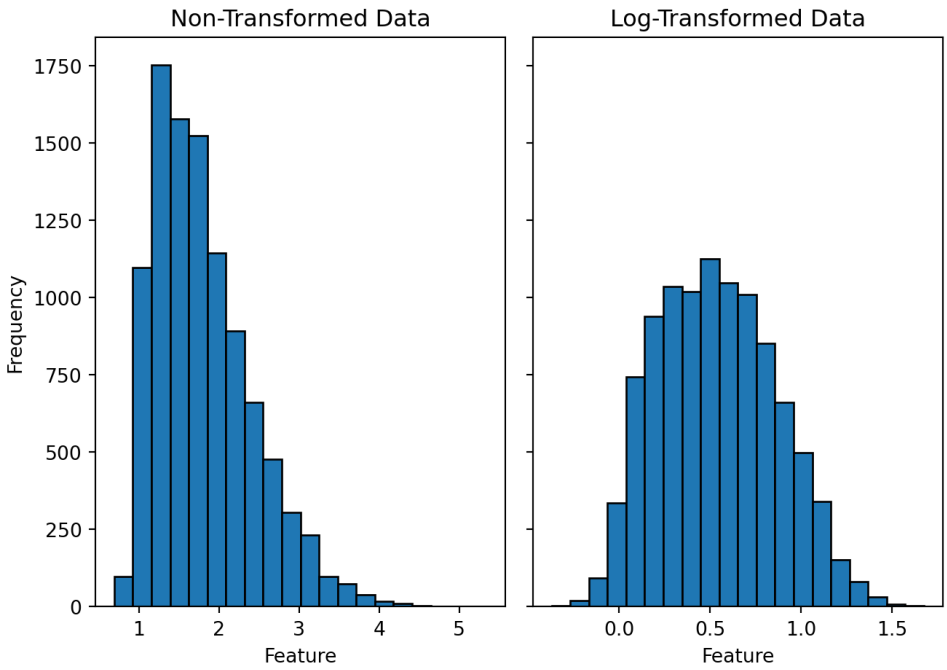


Figure 5.1: Effect of logarithmic transformation on data skewness.

Table 5.1: Incremental Model Performance and Training Time over Iterations

Iteration	LR Accuracy	LR Train Time (ms)	LGBM Accuracy	LGBM Train Time (s)
1	0.40	50	0.30	60
2	0.45	55	0.42	120
3	0.50	55	0.50	180
4	0.56	60	0.60	240
5	0.60	60	0.68	300
6	0.64	60	0.74	360
7	0.68	65	0.80	420
8	0.70	65	0.84	480
9	0.73	70	0.88	540
10	0.75	70	0.90	600
11	0.76	70	0.91	630
12	0.78	75	0.92	660

Notes: Accuracy values represent validation R^2. LR training time is in milliseconds, LGBM training time is in seconds on a CPU. Iterations reflect incremental feature engineering and hyperparameter tuning. LGBM requires longer to converge due to ensemble complexity and non-linear interactions.

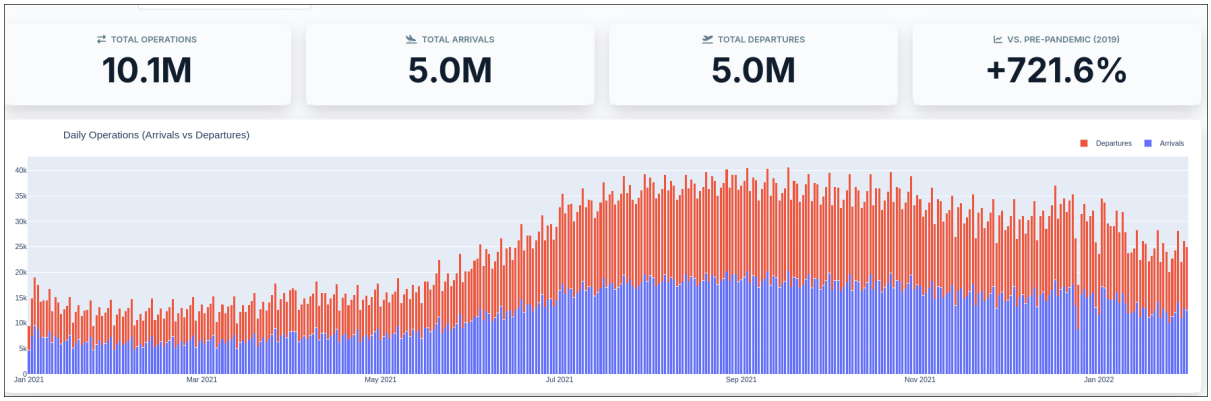


Figure 5.2: The main ”Operations Summary” tab.

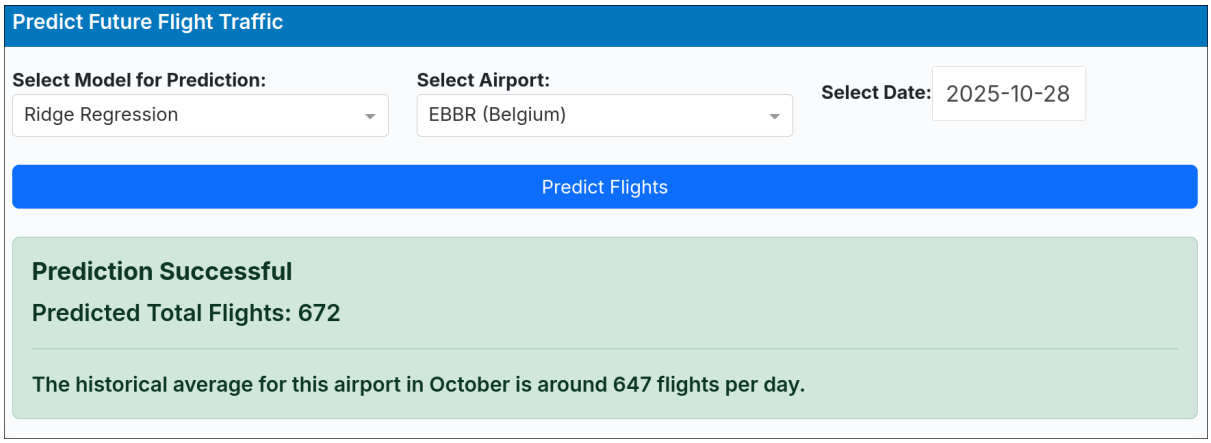


Figure 5.3: The ”Flight Prediction” interface.

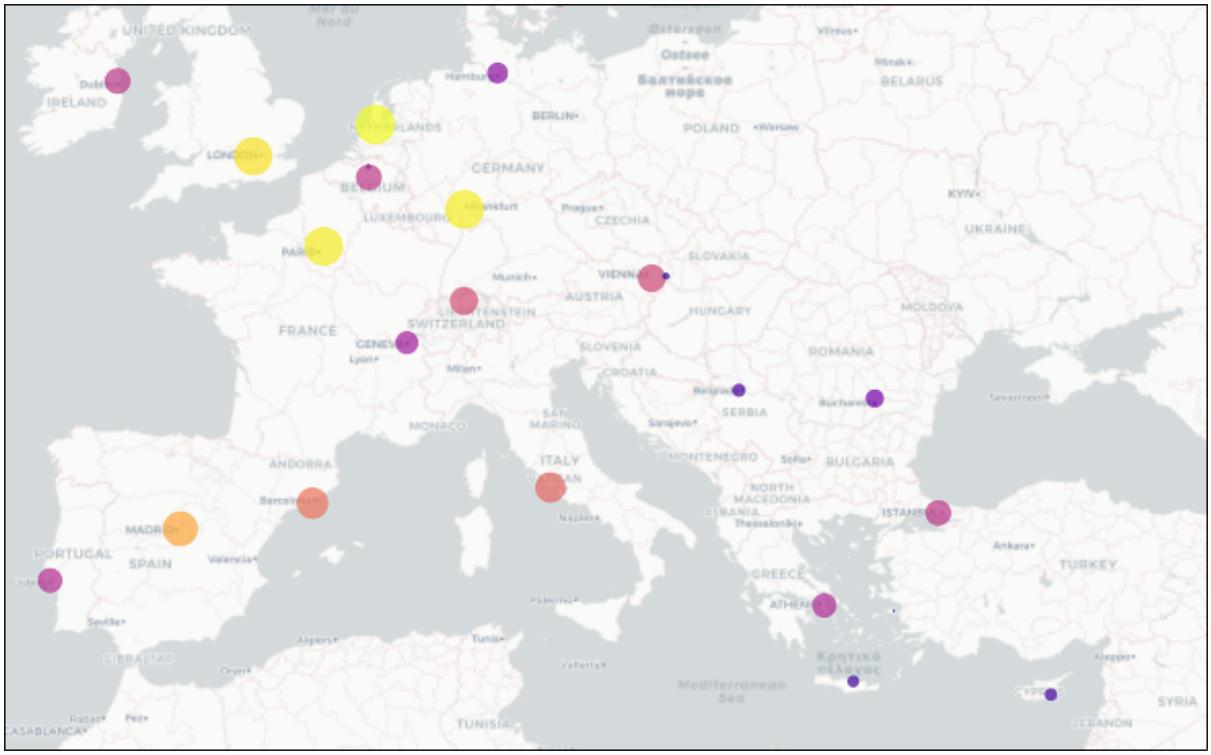


Figure 5.4: Europe Airports Map.

Figure 5.5: General Statistics.

Chapter 6

General Conclusion

This internship project addressed a central analytical challenge at the Office National des Aéroports (ONDA): transitioning from a manual, retrospective mode of flight traffic analysis toward a predictive, data-driven framework capable of supporting strategic and operational decision-making. Guided by the CRISP-DM methodology, the work encompassed all stages of a modern data science pipeline—from data acquisition and preprocessing to model development, evaluation, and deployment within an interactive analytical platform.

6.1 Summary of Achievements

The primary objective was to construct an end-to-end predictive system capable of forecasting daily flight operations across airports in Morocco. This goal was met through the implementation of a robust, automated data pipeline and the development of machine learning models designed to capture the complex temporal and contextual dynamics of flight activity.

A key contribution of this project was the development of a feature engineering framework that accurately modeled both cyclical temporal effects (e.g., weekly and seasonal variations) and pandemic-related structural shifts. These features proved essential for improving the explanatory power of the models and ensuring generalization across different time periods.

Two distinct modeling approaches were implemented and systematically compared. The Ridge Regression model established a linear performance baseline, achieving an accuracy of approximately **78%** with a mean computation time of under **100 milliseconds** per prediction. In contrast, the LightGBM model achieved substantially higher accuracy, approximately **92%**, by effectively capturing non-linear dependencies and complex interactions within the data. This improvement came with a higher computational cost—an

average inference time of roughly **4 seconds** per forecast—but remained within acceptable operational limits for interactive use. This trade-off between model complexity, predictive accuracy, and response time was carefully evaluated to balance interpretability and performance within the deployed system.

The final deliverable integrated these trained models into a fully interactive `Dash Plotly` web application. The platform enables users to visualize historical patterns, generate on-demand forecasts, and access key performance indicators (KPIs) without requiring technical expertise. This integration represents a major step toward operationalizing data science within ONDA's decision-making processes.

6.2 Challenges and Lessons Learned

Throughout the project, several technical challenges were encountered and systematically addressed. Managing data heteroscedasticity required the application of a logarithmic transformation to stabilize variance across airports of different sizes. Iterative feature engineering was essential for improving model sensitivity to pandemic-related effects and long-term seasonal trends. Finally, ensuring a seamless user experience demanded thoughtful interface design to translate complex analytical results into clear, actionable insights for non-technical users. These challenges underscored the importance of iterative development and cross-disciplinary collaboration in data science projects of operational relevance.

6.3 Future Work and Perspectives

This project establishes a solid foundation for future analytics initiatives within ONDA. Several promising directions for extension have been identified:

- **Automation through MLOps:** Implementing a continuous integration and deployment (CI/CD) pipeline to automate data ingestion, model retraining, and deployment as new flight data becomes available.
- **Integration of External Data Sources:** Incorporating exogenous variables such as economic indicators, tourism statistics, fuel prices, and event schedules to enhance model robustness and forecast precision.
- **Exploration of Advanced Architectures:** Investigating deep learning models such as Long Short-Term Memory (LSTM) networks or Transformer-based architectures, which may better capture long-range dependencies and multi-seasonal patterns in flight traffic data.

- **Extension of Dashboard Capabilities:** Enhancing the web application to support scenario-based simulations and anomaly detection, providing users with proactive insights into potential disruptions or deviations from expected operational patterns.

6.4 Concluding Remarks

In summary, this project achieved its core goal of developing a scalable, accurate, and interpretable forecasting system for national airport operations. The comparative analysis of linear and non-linear models demonstrated that while simple linear methods can deliver fast and reasonably accurate predictions, advanced ensemble techniques like LightGBM yield significantly higher precision at a modest computational cost. The successful deployment of the predictive system into an interactive, bilingual platform marks an important step toward institutionalizing data-driven decision-making within ONDA, setting the stage for continued innovation in predictive analytics and intelligent infrastructure management.