UHASSELT
KNOWLEDGE IN ACTION

# Faculty of Business Economics

## Master of Management

***Master's thesis***

***Evaluation of modern tools for data scientists***

**Jozef Ivano**

Thesis presented in fulfillment of the requirements for the degree of Master of Management, specialization Data Science

**SUPERVISOR :**

Prof. dr. Koenraad VANHOOF

2024
2025

UHASSELT
KNOWLEDGE IN ACTION

**www.uhasselt.be**
Universiteit Hasselt
Campus Hasselt:
Martelarenlaan 42 | 3500 Hasselt
Campus Diepenbeek:
Agoralaan Gebouw D | 3590 Diepenbeek

# Faculty of Business Economics

Master of Management

***Master's thesis***

***Evaluation of modern tools for data scientists***

**Jozef Ivano**

Thesis presented in fulfillment of the requirements for the degree of Master of Management, specialization Data Science

**SUPERVISOR :**

Prof. dr. Koenraad VANHOOF

## Declaration of Originality

This project is all my own work and has not been copied in part or in whole from any other source except where duly acknowledged. As such, all use of previously published work (from books, journals, magazines, internet, etc) has been acknowledged within the main report to an item in the References or Bibliography lists.

I also agree that an electronic copy of this project may be stored and used for the purposes of plagiarism prevention and detection.

Signed: 

Date: 27th May 2025

# Preface

This master's thesis is the result of months of research, experimentation, and critical reflection, completed as part of the final requirements for the Master's program in Management in Data Science at Hasselt University. The work explores the usability and effectiveness of AI-powered chatbots in guiding non-experts through the development of predictive models for flight delay forecasting—an increasingly relevant topic in the intersection of artificial intelligence and aviation analytics.

The journey of writing this thesis has been both intellectually stimulating and technically challenging. I have had the opportunity to engage deeply with emerging technologies, experiment with real-time data workflows, and critically assess the strengths and limitations of AI in practice.

I would like to express my sincere gratitude to my supervisor, Koen Vanhoof, for their valuable guidance, constructive feedback, and consistent support throughout this process. I also extend my appreciation to Jaroslav Ivančo, Dana Ivančová, Tereza Ivančová and Sofia Čorbová for their encouragement, support and insights during various stages of this project.

Lastly, I am thankful for the technological resources and tools that made this research possible—including the AI platforms evaluated—without which this study could not have taken its current shape.

Jozef Ivančo
Hasselt, May 2025

# Abstract

The quick adoption of artificial intelligence (AI) across various sectors has opened new doors in democratizing the complicated data science workflows, especially through their AI-powered chatbots. This thesis, therefore, seeks to explore whether mainstream AI chatbots (ChatGPT, DeepSeek, Google Gemini, Microsoft Copilot, and Meta AI) can really help laymen in developing predictive models of flight delays forecasted using real-time weather and traffic data. Usability, contextual memory, and multi-step task handling were taken as the criteria for determining how well each chatbot could possibly guide users through the data preprocessing, feature engineering, model training, and result interpretation aspects of the process. Comparative results show ChatGPT as the most reliable assistant, offering structured, iterative guidance and contextual awareness, while others have proven significantly limited in memory loss, error propagation, and inconsistency of output quality. They demonstrate the power of AI chatbots in minimizing technical thresholds in high-stakes domains like aviation, while also underscoring the importance of human oversight, transparency, and ethical safeguards. This research also intends to share the evolving discourse concerning AI usability as well as practical models in implementing chatbot-assisted machine learning for operational decision-making contexts.

# Table of Contents

# List of Tables



# List of Abbreviations

**AI –** Artificial Intelligence

**ANN –** Artificial Neural Network

**API –** Application Programming Interface

**CART –** Classification and Regression Trees

**COWRF –** Coyote Optimization Weighted Random Forest

**DBSCAN –** Density-Based Spatial Clustering of Applications with Noise

**DIP –** Digital Information Platform

**EDA –** Exploratory Data Analysis

**EWR –** Newark Liberty International Airport

**FAA –** Federal Aviation Administration

**FDPP-ML –** Flight Delay Path Previous-based Machine Learning

**GUI –** Graphical User Interface

**JFK –** John F. Kennedy International Airport

**LGA –** LaGuardia Airport

**LLM –** Large Language Model

**LR –** Linear Regression

**MAE –** Mean Absolute Error

**MBPP –** Mostly Basic Python Programming

**METAR –** Meteorological Aerodrome Report

**MJLS –** Markov Jump Linear Systems

**ML –** Machine Learning

**MSE –** Mean Squared Error

**R –** Coefficient of Determination

**RF –** Random Forest

**TCR –** Task Completion Rate

**XGBoost –** Extreme Gradient Boosting

# 1. Introduction

The use of artificial intelligence has become embedded in fields such as healthcare, education, and customer service, and it is now found more than ever in decision making, administration processes, and personalizing experiences. For example, AI chatbots have the potential to work in triaging patients, academic support, or clients, besides many other functions within this competence. With the evolution of such systems, it is becoming apparent that these systems might become useful even as an assistant for data science tasks, especially now when able to handle large text inputs or interact with uploaded files. Interest for evaluation in more complex applications, such as predictive modelling and machine learning workflows, has been created due to the phenomenal rapid development of these systems.

Constant delays in the aviation industry are a design fault for all aircraft, leading to substantial economic losses, inefficient resource usage, and customer dissatisfaction. Therefore, it becomes essential to optimally predict and control flight delays to better optimize air traffic operations. Various machine learning models have generated promising results in predicting flight delays. Implementation of such technologies, however, requires a technical skill set which still forms a barrier for non-technical personnel working in aviation.

## 1.1 Problem Statement

While air traffic delay prediction models have shown promising results using machine learning models, their execution usually requires specialized skills, thereby posing barriers to non-expert aviation stakeholders. Other conventional approaches to prediction have a hard time incorporating real-time data, such as weather updates and live air traffic feeds, owing to their integration complexities (Reitmann & Schultz, 2022; Eurocontrol, 2024). While AI chatbots like ChatGPT, DeepSeek, and Google Gemini can potentially help in democratizing complex predictive analytics, there remains little literature on their usability in guiding non-experts through the entire ML workflow, particularly in safety-critical domains such as aviation (FAA, 2022).

## 1.2 Focus and Scope

This thesis studies whether AI chatbots could function as practical, non-expert-oriented tools for the construction of predictive models on air traffic delay. This includes chatbot-assisted model building on flight delay prediction from publicly available datasets and real-time weather feeds. The assessment focuses on five widely available AI chatbots: ChatGPT, DeepSeek, Google Gemini, Microsoft Copilot, and Meta AI, according to usability, performance, contextual memory, and multi-step capabilities in machine learning workflows.

## 1.3 Relevance and Importance

There has been literature covering the contributions of AI to machine learning workflows and its predictive analytics, mostly lacking in probing how these chatbots serve as end-to-end guides for non-experts in highly risky, domain-specific areas like aviation. It becomes all the more necessary in this day and age of democratized AI tools to know their practical limits and strengths. This research fills a significant gap: to examine whether AI chatbots lower the technical barrier for entry into predictive model construction for stakeholders such as airline operations managers, research personnel, and air traffic controllers who do not have specialized knowledge in machine learning or Python coding skills.

The results of this study will have a substantial impact on both AI usability research and operational issues within aviation analytics by providing an alternative path to accessing and applying predictive technologies in practice.

## 1.4 Research Questions and Objectives

The primary question posed in this study is as follows:

Can AI chatbots, such as ChatGPT, DeepSeek, Google Gemini, Microsoft Copilot, and Meta AI, effectively guide a data analyst in developing and evaluating a machine learning model for predicting air traffic delays using real-time weather, traffic, and resource data?

The following objectives arise from this:

- To evaluate the usability and contextual intelligence of popular AI chatbots in guiding users through data preprocessing, model selection, and result interpretation.
- To evaluate the performance of chatbot-assisted models in terms of accuracy, explainability, and generalization.
- To discover the limitations and risks that may arise from the reliance on AI chatbots for predictive modelling in safety-critical domains such as aviation.
- To evaluate whether and how chatbot-assisted workflows can meaningfully democratize machine learning technologies.

## 1.5 Structure of the Thesis

The thesis is structured as follows:

Chapter 2: The Literature Review chapter provides a historical perspective through a review of different research carried out on AI chatbots for data science, predictive modelling in aviation, and the usability of AI tools.

Chapter 3: Methodology covers the research design, which encompasses the framework for evaluating the performance of chatbots and the datasets used.

Chapter 4: Analysis and Results presents a comparative analysis of five of the AI chatbots tested in this study.

Chapter 5: Discussion interprets the results and discusses implications, limitations, and future research directions.

Chapter 6: Conclusion summarizes the key contributions from this research and provides recommendations for practitioners and future scholars.

# 2. Literature Review

An extraordinary transformation has set in motion with numerous AI tools gradually integrated into data science, predictive modelling, and aviation. Chatbots powered by AI, techniques of machine learning for predicting flight delays, AI-assisted model development, usability and effectiveness of AI-based chatbots, comparative benchmarking of chatbot performances - the study of all these brings into focus appreciating the holistic role that AI is playing for advanced analytics in the democratic sense. Therefore, the present literature review seeks to provide a critical evaluation of existing literature and theories across these domains that identify gaps in knowledge and provide the underpinning basis for assessing the ability of AI-driven chatbots to empower novices for real-time decision-making in the construction of predictive models in high-stakes environments such as aviation. The discussion then draws upon the relevant literature across five thematic areas as primed as per the research focus and objectives of this thesis.

## 2.1 Overview of AI-Powered Chatbots in Data Science

Chatbots like ChatGPT, DeepSeek, Microsoft Copilot, Google Gemini and Meta AI are transforming data science by making fundamental tasks automatic such as coding, data analysis, and predictive modelling. Their integration into workflows has optimized efficiency, precision, and access for experts and non-experts. Daython3 (2024) identifies the practical use of chatbots like ChatGPT and Perplexity in data science, where their ability to optimize code, generate reports, and enable data-driven decision-making is noteworthy. Similarly, Bani (2025) discusses the role of AI chatbots in enhancing data visualization, automating monotonous tasks, and increasing analytical capability. These publications point to the revolutionary impact of chatbots while acknowledging their limitations, particularly their dependence on high-quality data and difficulties with domain-specific tasks.

Besides general workflow automation, chatbots are also becoming intelligent assistants in data preprocessing, which is a crucial task in data science. Zhang et al. (2024) explain how LLMs like GPT-4 can assist in error detection, data imputation, schema matching, and entity matching, showing their potential to preprocess raw data before analysis. Their findings indicate that GPT-4 achieved near-perfect accuracy on some datasets, suggesting its potential for improving data quality in predictive modelling. However, the study also highlights limitations, including computational inefficiencies, token constraints, and difficulties in handling highly specialized datasets. This concurs with Chia (2024), whose capability of code generation, debugging, and constructing artificial data through AI chatbots are mentioned in his article but also reference deficiencies in hallucination (fabrication) and bias compromising the veracity of AI-augmented information.

Despite these technologies offering indications of heightened participation by AI in the automation of data science pipelines, their capability within predictive modelling under real-time scenarios has yet to be realized. Mishra et al. (2024) check the capability of GPT-3.5 as a "Language Data Scientist" (LDS), assessing its performance in answering zero-shot, natural language questions for data science activities. Their experiment realizes partial success (32.89% accuracy) in forecasting sets of data and generating correct code but also raises questions such as improper code execution, no response to complex questions, and token constraints. From the results, one can deduce that even if AI chatbots are prone to assist inexperienced personnel in preliminary exploration and data preprocessing, they have questionable credibility for building and calibrating forecasting models.

Aside from these, security, ethics, and environmental aspects are becoming an ever-growing issue as well. Malec (2024) describes the power-hungry process of training large language models (LLMs) and carbon footprint with widespread use. Synoptek (2023) recognizes threats in terms of security such as risk of noncompliance, personal data concern, and the vulnerability to intrusion or hacking risks that pose chief ethical concerns to businesses deploying AI chatbots. These perspectives complement the technical arguments with a wider vision for AI chatbot adoption in data science.

While restricted, AI chatbots offer promise for non-experts to develop machine learning models. However, existing research primarily focuses on general applications rather than domain-specific, real-time predictive modelling, like air traffic delay prediction. This gap offers room for this study to evaluate the potential of AI chatbots in enabling non-experts to develop and test real-time predictive models, addressing key issues like data integration, accuracy, and usability in high-risk domains.

In summary, a clear potential exists for chatbots to automate key activities in data science, such as programming, data manipulation, and analysis. As of now, challenges remain in terms of computational efficiency; a solution still needs to be proposed for faster processing by the software; issues cantered around domain expertise; or even some ethical issues that lie within the domain; lastly, and most importantly, real-time predictive ability. On the other hand, current work focuses mainly on broader general-purpose applications of data science, with minimal work applied to specific, real-time applications such as flight delay prediction. These gaps indicate a clear need for further inquiry into how AI chatbots might help improve the accessibility divide for non-expert users - in particular, for topics within the critically important, time-sensitive domains where high accuracy and interpretability are of utmost importance.

As the literature under review shows, while AI chatbots have changed the landscape of data science workflows, their applicability needs to be better studied in specialized real-time predictive modelling tasks. The Analysis and Results section that follows will investigate whether, in particular, AI chatbots can help non-expert users create flight delay prediction models, taking into account chatbot usability, performance, and real-time data capabilities.

## 2.2 Machine Learning for Air Traffic Delay Prediction

### *2.2.1 Evolution of Machine Learning in Flight Delay Prediction*

Machine learning in the prediction of air traffic delays has developed immensely in the last ten years, and several models have been tried and tested to increase the accuracy of predictions. An early comparison between Markov Jump Linear Systems (MJLS), Artificial Neural Networks (ANNs), Classification and Regression Trees (CART), and Linear Regression (LR) for predicting air traffic delays was presented by Gopalakrishnan and Balakrishnan (2017). Their findings indicated that MJLS models worked better for regression-type prediction tasks, while ANNs worked better for classification-type predictions. These models were not real-time adaptive, and therefore their use in making operational decisions was restricted.

Likewise, with a bid to improve predictive models, Li (2018) came up with a Random Forest-based classification approach, integrating a delay propagation model to track the influence of an initial delay on subsequent flights. While the study indicated that chained delay prediction improved forecast accuracy, it did not incorporate real-time data streams, which are crucial in airline operations.

### *2.2.2 Advancements in Machine Learning Models*

Later research has come up with more sophisticated ensemble learning techniques for greater predictive power. Mamdouh et al. (2023) designed the Flight Delay Path Previous-based Machine Learning (FDPP-ML) model, which relied primarily on historical flight schedules rather than live operational data. The study showed that historical flight delays and turnaround times had strong predictions of future delays. However, the model's failure to include real-time traffic congestion or weather conditions limited its application for real-time decision-making by airlines.

Similarly, Alfarhood et al. (2024) tried out CatBoost, XGBoost, and LightGBM models for Saudi Arabian Airlines delay prediction. They found CatBoost to be superior to other models, with a 76% accuracy rate for classifying delayed vs. on-time flights. Unlike most previous studies, this one focused on data for one airline's operations, demonstrating the effectiveness of specially designed AI models for delay prediction in a specific airline. Its potential across different airlines and airspace conditions remains to be found out.

The other novel approach introduced Dai (2024) presented the combination of Density-Based Spatial Clustering of Applications with Noise (DBSCAN) with Coyote Optimization Algorithm-based Weighted Random Forest (COWRF). The combination of clustering and ensemble learning made the paper achieve accuracy level of 97.2% and even surpassed normal model accuracy. All this may have surpassed expectations, but one was concerned with computational requirement and simplicity levels of such model.

### *2.2.3 The Role of Real-Time Data in Predicting Delays*

While static historical data has been more than sufficient for delay prediction, real-time data streams have been advanced by more recent research to enable increased accuracy levels. Rodriguez (2024) analysed the performance of NASA's Digital Information Platform (DIP) in air traffic control using machine learning and established the importance of processing real-time data when it is a matter of rerouting aircraft and delay avoidance. As opposed to earlier models that were founded on historical data only, the study demonstrated that real-time machine learning models allow airlines to react dynamically to disruptions. However, the study failed to consider whether such models would be feasible for non-machine learning experts.

Similarly, Durgut (2025) explained big data velocity in aviation and concluded real-time processing challenges and proposed edge computing, stream processing platforms (Apache Kafka, Apache Flink), and AI predictive models as a solution. Such conclusions complement Rodriguez's (2024) real-time adaptability discussion, adding that live data streams need to be integrated into machine learning models. Neither of the studies mentioned, however, how AI chatbots or AI assistants can be utilized to aid non-technical aerospace professionals in implementing such models.

### *2.2.4 AI-Driven Operational Benefits for Airlines*

Aside from increasing predictive accuracy, machine learning models have yielded tangible financial and operational benefits to airlines. Alphaidze and Renart (2024) investigated the use of real-time analysis of data to optimize resource allocation, cost savings, and minimization of flight disruption. The study introduced an event-driven architecture based on MongoDB and Google Cloud's Vertex AI and established that machine learning could reduce financial losses in delay propagation.

In the same vein, Brooker (2024) explored how JetBlue, Delta, and United Airlines use AI-fuelled predictive analytics to streamline flight operations. These airlines have collaborated with weather intelligence suppliers and invested in AI-based forecasting models, which has resulted in large cost savings. For instance, JetBlue's partnership with Tomorrow.io has led to between $300,000 and $600,000 annual savings per hub. Though both research illustrate the operational advantages of AI, they fail to acknowledge the accessibility of such technologies to non-technical professionals.

### *2.2.5 Challenges in AI Implementation for Flight Delay Prediction*

While useful, AI-powered flight delay prediction faces challenges in its implementation, particularly in infrastructure integration, data privacy, and industry-wide coordination. Brooker (2024) pointed out that while AI significantly enhances delay prediction, its adoption requires significant investment in technology along with staff training. Additionally, data privacy and security concerns must be addressed through rigorous governance frameworks.

Other than privacy concerns, Rodriguez (2024) and Durgut (2025) also noted the difficulty of AI integration into the legacy aviation system. Most airlines operate on outdated infrastructures, and the real-time deployment of AI is complicated without pervasive system upgrades. Data interoperability among airlines, airports, and regulating bodies remains also a serious challenge, which calls for industry-wide collaboration in terms of formulating standardized AI integration protocols.

### *2.2.6 Identified Research Gaps and the Role of AI Chatbots in Democratizing Machine Learning*

While existing research has been successful in improving model accuracy, real-time response, and cost savings, not many studies address the usability aspect—how airline professionals, air traffic controllers, or researchers who do not have AI background can use machine learning models for delay prediction.

This thesis seeks to bridge this gap by considering whether AI chatbots such as ChatGPT, DeepSeek, Microsoft Copilot, Google Gemini and Meta AI can assist predictive modelling non-specialists. Earlier research does not consider whether AI chatbots are capable of guiding users through preprocessing data, selecting models, and interpreting outcomes—essence for aviation staff looking to leverage AI without the need for intense technical learning. Whether or not AI chatbots can link real-time streams of data and propose dynamic optimizations is another path not taken previously.

Through an assessment of the usability, precision, and performance of AI chatbots in facilitating flight delay modelling, this research can not only contribute to aviation machine learning but also AI-assisted research processes. Closing this unexplored intersection of usability and forecasting modelling can possibly deliver a new framework for simplifying complex AI technology to reach a wider array of stakeholders within the aviation industry.

On the one hand, the machine learning approaches score rather well in promising flight delay prediction, while on the other hand, there are still some open challenges pertaining to real-time data integration and accessibility for the non-technical user. This study will further expand on these findings through testing how AI-based chatbots would lead non-technical end-users through the steps of predictive model development using aviation datasets, as outlined in the Methodology chapter.

## 2.3 AI-Assisted Model Development

Artificial intelligence is increasingly being integrated into various stages of machine learning model development to facilitate greater automation, improved accuracy, and reduced development time. Recent advancements in AI-based tools, particularly no-code and low-code platforms, have revolutionized the development, training, and deployment of models to a great extent. These tools aim to normalize AI model construction, even for non-experts, by making repetitive tasks such as feature selection, hyperparameter adjustment, model evaluation, and deployment easier. Nevertheless, in spite of these developments, there are still gaps in areas such as interpretability, bias elimination, and efficient integration with domain-specific applications.

A multitude of studies have approached exploring the capabilities and limitations of AI-assisted model development. Rao et al. (2023) introduced LowCoder, a hybrid low-code tool that integrates visual programming with natural language-based AI assistance to aid in AI model development. Their work illuminated how AI-powered assistants aid users in discovering relevant machine learning functions and assist in building pipelines without extensive coding knowledge. Tufano et al. (2024), in a parallel study, developed AutoDev, an AI-powered software development framework that generates software code, tests it, and refines it independently. It is showcased among the first studies demonstrating the possibility of an AI agent autonomously performing iterations of model development tasks. The mentioned research address how AI-assisted tools can greatly enhance efficiency, and yet there remains the challenge of interpretability with model creation done without explicit human intervention.

Literature over the past years has also highlighted the emergence of so-called no-code AI platforms. A study by Law (2023) on ten no-code AI tools about which users can build and deploy machine-learning models assuming no programming expertise, includes tools such as PyCaret, DataRobot, and Google AutoML. It was noted that such tools lower the barrier for AI adoption; however, they tend not to be customizable and unsuitable for very niche applications. Similar problems were raised by Gupta (2020) concerning feature selection, with trust within automated methods to accomplish performance metrics rather than interpretability and domain-specific relevance. Integration of AI during model development helps to cut training time and bolster efficiency, but one of the major concerns raised deals with AI-generated models being far removed from human decision-making principles.

AI chatbots are known for their extensive role in the AI-assisted model development process by guiding users through machine-learning workflows. Lung (2024) elaborates that these AI chatbots are built by training on large datasets using deep learning technology to assist the user in performing tasks like debugging or code generation. Similarly, Rao et al. (2023) put through evidence that AI-powered coding assistants, such as GitHub Copilot, streamline software development by suggesting relevant code snippets. Nevertheless, these investigations have suggested that chatbots and AI assistants, while capable of efficiently automating repetitive tasks, tend to struggle with problems requiring subtle difference and human intervention is often needed to validate the correctness of generated outputs.

The limits of AI-assisted model development regarding bias detection and mitigation are the most stringent. As Merigan (2023) explains, when biased feature selection is done within machine learning, that AI-driven automation may serve to merely enhance existing biasedness rather than eliminating it through careful biases to mitigate. This finding from Dataquest Labs (2024) further shows how often AI chatbots simply reproduce any present bias in their training sources, raising ethical concerns around content generated by AI. Both studies emphasize greater transparency and regulatory oversight toward ensuring AI-assisted tools' generation of fair and unbiased models.

All these concerns notwithstanding, AI-assisted model development provides an efficient workflow improvement and normalizes the ability to implement machine learning technologies. The work of Shaikh (2023) on chatbot training illustrated that, among other functions, AI is capable of improving data preprocessing and model training by automating tokenization, entity recognition, and feature selection. Weka (2023) also explained the ever-increasing influence of AI into enterprise solutions out there. Automated model development tool brings an improved decision-making ability to industries such as healthcare and finance towards achieving this. These studies lend credence to the argument that AI-enabled model-making may extraordinarily simplify the machine learning workflow, while at the same time demonstrating that human over-sight will always be essential in such endeavours.

Such a research gap stands to emerge from the above studies about how much AI chatbots can independently lead non-experts into the full machine learning pipeline. Existing research have devoted much attention to AI-assisted coding tools with no analysis of AI chatbots as comprehensive guides for users without machine learning knowledge. They either automate parts of the model development process or need some level of understanding and knowledge to function correctly. This gap can be filled by research that evaluates how effective, in principle, AI-power chatbots could be in guiding the user in the complete machine-learning process, from preparing the data to evaluating the model.

This thesis investigates a gap on whether AI chatbots such as ChatGPT, DeepSeek, Microsoft Copilot, Google Gemini and Meta AI can actually assist a layperson in developing predictive models. The research investigates how well they can guide users through data cleaning, feature selection, model training, and interpretation of results, determining the possible reliability of AI chatbots as alternative tools for traditional AI assisted model-making. This may enrich the now-widening discourse on the usability of AI in model development and provide fresh avenues to optimize this type of chatbot-based AI assistance for actual real-world deployment and application.

Much of the promise of AI-aided design tools focuses on the automation of machine learning, but it is unclear how properly these tools can help non-expert users. This research is going to test the validity of AI chatbots in the various steps of the complete model-building workflows, including data cleaning, feature selection, model training, and evaluation, as defined in the Methodology.

## 2.4 Usability and Effectiveness of AI Chatbots

AI chatbots have emerged being used in a variety of domains such as education, healthcare, scientific research, and data analytics. While AI-powered conversational interfaces can afford efficiency and automation, their usability and effectiveness are still much debated. Researchers have examined user engagement, decision-making, and insight generation regarding both pros and cons. This part will critically assess recent literature in order to find out whether AI chatbots are useful in various situations and what needs to be improved.

### *2.4.1 User Engagement and Usability in Scientific and Educational Contexts*

Various reports highlight the usefulness of AI chatbots in academia and research, especially for assisting research workflows. Pividori (2024) mentions that ChatGPT makes scientific writing, literature reviews, and coding more productive. His conclusion pointed out that chatbots mainly work for tasks that put the human user in control of the final text, thus diminishing the chance of spreading untruthful information. However, when it came to literature review tasks, these chatbots came up with hallucinations and incomplete interpretations which caused apprehensions among researchers concerning the accuracy. Paharia (2024) reached a similar conclusion in a survey of medical researchers wherein 61% of the respondents declared that they knew about the relevance of the AI chatbots to research, while only 52% said that they had ever used them for any scientific purposes. These have been increasingly adopted, yet the low levels of formal AI training in academic institutions observed in the study threaten research integrity.

Nguyen et al. (2024) have studied chatbot usability for health data collection and found that chatbot-gathered interface usability scored higher than traditional form-based data entry. According to the findings, the chatbot mode offered a lower cognitive load and increased engagement; however, the final AI-generated reports were sometimes more difficult to comprehend than reports developed manually. These findings corroborate those of Veras et al. (2023), who assessed the effectiveness of ChatGPT for health sciences students. They found that chatbots were helpful for writing assistance and administrative tasks, yet many students struggled with the outputs of AI that lacked contextual understanding. The conclusion drawn from both studies points to a disturbing trend: while chatbots improve usability and engagement, they present challenges of lacking fidelity and interpretability.

### *2.4.2 Effectiveness in Decision Support and Analytical Tasks*

In detail, Laoudai (2024) has reported on the effectiveness of AI chatbots in the field of data analysis and consultancy. He made a comparison with traditional data analysis and stated that these chatbots are excellent in processing huge datasets, finding trends, and making predictive recommendations. Interpretability still poses a challenge, as the lack of transparency in decision-making comes with deep learning models. A similar issue was observed in the work of Chen et al. (2024), who evaluated multimodal AI chatbots in oncology case assessments. While these chatbots were very accurate in multiple-choice diagnostic determination tasks, they did considerably worse at free-text explanation generation. These findings indicate that, while the AI nature may assist in structured decision-making, challenging tasks exist with more abstract reasoning tasks that require some sort of context.

Hultberg et al. (2024) also had research into chatbot competence in economics education. ChatGPT and a host of other similar models were tested against multiple-choice and open-ended economics problems. They revealed that AI chatbots did very well at problem solving with structure but failed miserably at tasks requiring analytical reasoning and justification. A common strand throughout several studies reveals that, while insights are generated by AI, they often lack depth and require supervision and correction by humans for authenticity and accuracy.

### *2.4.3 Ethical Concerns, Transparency, and Interpretability*

One of the most enduring problems regarding AI chatbots is that they are decisionally opaque. Paharia (2024) reported that 77% of researchers worried about AI-created content with no clear reason behind it. Similarly, Chen et al. (2024) reported that clinicians were sceptical about chatbot-recommended diagnoses due to the black-box nature of AI models. Transparency issues were also raised by Laoudai (2024), stating that deep learning models used in AI analytics are not simple to interpret, therefore not as reliable for compliance-driven industries such as finance and healthcare.

Ethical issues of plagiarism, data protection, and bias have also been popularly debated. Veras et al. (2023) cautioned against the possibility of AI perpetuating bias in student examinations, while Hultberg et al. (2024) identified systematic mistakes in chatbot-made economic projections. Pividori (2024) additionally noted that scientific conclusions may be skewed by AI chatbots, highlighting the role of human checking in AI-powered research. These studies collectively suggest that chatbots must be backed by strong ethical regulation and user training to mitigate risks of disinformation and prejudice.

### *2.4.4 Bridging the Gap: Areas for Improvement and Future Research*

While AI chatbots are taking over some responsibilities in academia, healthcare, and analytics, they still exhibit limitations in reasoning, transparency, and ethical governance. The deficiency of proper formal training in AI within universities could be understood from the point of view that according to Paharia (2024), appropriate educational programs must embed training in AI literacy so that researchers and students use effective chatbots. Doing great, as pointed out again by Laoudai (2024) and Chen et al. (2024), better interpretability in AI-generated insights would eventually lead to even more reliability in high-stakes decision-making domains like biomedical and financial domains with hybrid AI-human systems.

Contextual awareness is yet another extremely critical area in which a lot could be improved in chatbot response. Nguyen et al. (2024) and Veras et al. (2023) revealed that users found difficulty in understanding the reports generated by the chatbot, which means that any future design of a chatbot must incorporate feedback loops from users to enhance contextual accuracy. In addition, multimodal AI models, which combine text, image, and structured data analysis, should play a good role in closing the gap of reasoning in chatbots, as posited by Chen et al. (2024).

Last but, not least; ethical concerns regarding AI-generated misinformation, plagiarism, and bias would have to be taken care of through enhancement in AI auditing systems. The ethical AI framework is essential given that AI chatbots become more embedded into scientific research, education, and business analytics. It will allow storing trust and credibility in these AI-driven insights.

Previous research has shown the strengths of use and weaknesses of interpretation in AI chatbots in science and education. The study expands these findings by evaluating chatbot usability, context understanding, and error handling during the predictive model development, with results to be shared in the Analysis and Results chapter.

## 2.5 Comparative Analysis of AI Chatbots

The field of AI chatbots has enhanced greatly, and many methods have been developed to measure their performance, usefulness, and application in different domains. This section evaluates critically the processes taken in benchmarking and evaluation for AI chatbots, emphasizing their methodology, efficacy in coding, and overall performance of the model. The studies researched provide insights into the benefits and limitations of AI propelled conversational agents and coding assistants. That is, through comparing these methodologies and findings, this review identifies research gaps in the current evaluation methods and suggests future experimental lines.

### *2.5.1 Methodologies for Assessing and Comparing AI Chatbots*

Chatbots or evaluation methodologies include user-experienced evaluation methodologies and task completion-, efficiency-, or engagement-driven benchmarks. Uramis (2023) presents an architecture for chatbots that are tested beforehand and afterward, including evaluation of functionality, usability, and conversational flow, while significantly emphasizing performance metrics such as correctness of responses, user satisfaction, and cross-platform independence for determining the usefulness of a chatbot. In similar lines, ChatBees (2024) concentrates on chatbot evaluation metrics, most importantly task completion rate (TCR), user satisfaction, and efficiency for self-service. So, in conclusion, these studies suggest that a good-performing chatbot must be able to fully engage the user while limiting human intervention.

The authors of Freshworks (2024) took a more data-centric route, analyzing the performance of chatbots in a data-driven manner. This paper proposes that, by tracking interactions and real-time performance metrics, deeper insights into chatbot effectiveness will enable continuous refinements. However, the limitation shared by most studies is to focus solely on customer-facing uses while downplaying the consideration of chatbots designed for technical or data science-oriented workflows.

Going beyond the ordinary evaluation of chatbots, Banerjee et al. (2023) give benchmarking approaches in a more broadened sense, which observe the LLM-powered chatbots through standardized methodologies. Their study proposed performance measurements measuring code generation accuracy, logical reasoning, and contextual understanding, all of them essentials for AI-driven coding assistants. This study differs from all others because it has incorporated comparative evaluation for relevant assessment of AI chatbots in technical domains.

The analysis of Smutny and Bojko (2024) further tightens the constraints on chatbot assessment and performs a more specific evaluation of LLM-based chatbots in web development tasks. Their methodology directly compares several chatbots through their ability to generate, debug, and optimize code snippets. A major insight from this study is that LLM-based chatbots can differ greatly in capabilities, depending on the task complexity, suggesting the necessity for a domain-specific evaluation metric.

### *2.5.2 Benchmarks for Evaluating Chatbot-Generated Code and Model Performance*

In AI-assisted code generation, benchmarking is an essential part of evaluation. Ghosh et al. (2024) provide a comprehensive commentary on various benchmarks for code generation evaluation, discussing their advantages and disadvantages. The paper shows that old benchmarks, such as

HumanEval or MBPP, consider syntactic correctness and functional accuracy, but pay little attention to readability, maintainability, and security attributes of generated code. Such a gap is highlighted by Yetiştiren et al. (2023), who compare GitHub Copilot, Amazon CodeWhisperer, and ChatGPT, evaluating them for their ability to generate high-quality code fragments. Results indicate that AI-aided coding tools increase productivity but produce poor-quality code that requires human intervention due to issues like redundancy, inefficiency, and non-compliance with best practices.

Quan et al. (2025) propose a more competitive approach to benchmarking via CodeElo, a ranking method aimed at assigning Elo ratings to LLMs according to their competency in competitive coding tasks. The methodology thus becomes relevant since it assesses AI coding assistants through the lens of actual programming tasks and not solely unit test passage rates. The opposite of this is what classical evaluation metrics tend to do, as they hardly ever consider the scale and adaptability of AI-generated code.

The demand for a multi-directional evaluation is thus reinforced by Reda (2024), stating that different chatbots excel at different evaluation criteria. In contrast, some models perform well on logical reasoning tasks but struggle with conversational coherence, others output syntactically correct code, yet they cannot handle contextual refinement of their outputs in response to user feedback. This opens a glaring need for holistic frameworks of evaluation that factor both task-specific and user-centric metrics.

Samuylova (2025) takes a more domain-focused approach and reviews 20 LLM evaluation benchmarks in depth, ranging from the perspective of language understanding, reasoning, and code generation. The paper notes that, fundamentally, the benchmarks must evolve alongside the LLMs since many traditional benchmarks become outdated as models improve. Future benchmarking efforts are thus considered to incorporate assessments of multi-step reasoning, efficiency in handling errors, and adaptability of the model to problems solved in a dynamic manner.

### *2.5.3 Identified Gaps and Future Research Directions*

While studies present a strong foundation for evaluating AI chatbots, many crucial gaps remain unfilled. One major shortcoming is the lack of a comprehensive evaluation framework for AI chatbots in scientific computing and data science workflows. Most benchmarks focus on general conversational AI or code generation without assessing AI chatbots' capabilities in very intricate analytical tasks such as data pre-processing, statistical modelling, and machine learning experimentation.

Another consideration that is significant to the implication of a real-world scenario is that most benchmark scores do not assess AI tools in coding on collaborative works. For example, instead, AI tools for coding should be assessed in the context of whole project workflows rather than broken down into isolated tasks, such that debugging efficiency, scalability, and integration within a developer's environment could play a major role.

Current benchmarks used for chatbots neglect some complex analytical tasks, such as data modeling. Hence, this study presents a comparative framework to evaluate the performance of chatbots in an entire machine-learning pipeline, but towards real-life applications for the end-users, which will be presented in the Analysis and Results.

## 2.6 Ethical Considerations and Limitations

### *2.6.1 Risks of Relying on AI-Generated Code and Guidance*

The incorporation of AI-generated code in the process of software development has brought crucial gains in productivity and effectiveness. This, however, brings about ethical questions regarding security and reliability. Khoury et al. (2023) evaluate the security risks posed by the ChatGPT-generated code, revealing it suffers from common vulnerabilities such as user inputs not being correctly handled, the use of nonsecure means for cryptographic implementation, and susceptibility to injection attacks. This means that AI models may generate working code that, however, does not pass security best practices, thus laying software applications wide open to different cyber threats. Ma et al. (2023) perform an integrated audit of the AIs generating codes and emphasize that often, public AI systems are not going to clarify their reasoning behind code suggestions, meaning the integrity of any output may be very hard for developers to assess in a transparent way. Beyond the security questions, Atemkeng et al. (2024) maintain that the use of code generated by AI could undermine human agency in programming, with software developers blindly relying on AI solutions without much understanding of the logic behind such actions. Coding has been cited as a potential decline among developers, as misuse of AI help can create knowledge voids in debugging and algorithm optimization. Timbó (2024) makes a similar argument, adding that excessive reliance on the use of AI-generated code could lead to a diminishing human creativity in the problem-solving process and a lack of accountability in cases of software failures. Despite the gains in productivity from adopting AI-generated code, these studies argue that such adoption should be matched by a strict verification process, human oversight, and ethical programming guidelines for the development of secure and responsible software.

### *2.6.2 Accuracy and Reliability Concerns When Using Chatbots for Research*

Although chatbots based on large-language models have helped revolutionize how researchers work, their validity and ethical perspectives are still questionable. As Lawton (2024), these risks pose a serious menace to the AI-generated misinformation and bias from its responses. He also discusses factually incorrect or misleading information in a confident voice: this may well concern scientific and technical research. It could affect research integrity in such fields.

Another limitation is transparency about AI decision-making. According to Ma et al. (2023), outputs generated from AI systems do not have a clear indication of how they arrive at their conclusions. This has made it very challenging to claim verification of research findings by AI because users may not tell if the chatbot's suggestion is based on reason or flawed training data. Dsouza (2023) adds that AI models trained on biased datasets may lead to propagation of wrongful information in line to their prejudices in its research and industry application.

Despite all challenges, AI chatbots are still embraced in research because they automate literature review, summarize complex topics, and assist in technical writing. However, it is the user who will shoulder ethical responsibility of verification of AI-assisted research, thus understanding needs stringent validation mechanisms before any integration into study scientific outputs of AI-derived insights.

### *2.6.3 Ethical Considerations in AI-Assisted Decision-Making in Aviation*

Artificial intelligence faces several ethical dilemmas as applied to aviation decision-making. According to Dsouza (2023), the major ethical concerns in this industry include bias and discrimination, privacy, transparency, and job displacement. Bias in an AI system is an important source of risk: a poorly biased dataset may lead to decisions biased against a particular body such as flight operations, maintenance prioritization, or passenger screening. Fairness and inclusivity need to be ensured in all AI models within the aviation context to help avoid unjust or biased decision-making.

Privacy can also be deemed an ethical consideration for AI used in aviation, especially with passenger data collection, flight monitoring in real time, and biometric verification systems. The AI models accruing passenger-related sensitive data need to comply with data protection regulations and avoid any unauthorized access or misuse. In the same context, transparency and accountability are also important: AI systems making decisions on flight optimization or maintenance scheduling should carry auditable explanations acceptable to all operators and agents working in trust with them.

In conclusion, AI poses an ethical challenge to jobs in aviation. With AI-powered automation displacing human functions in air traffic control, maintenance diagnostics, and even piloting, concerns regarding job displacement are rising. The industry needs to engage proactively with the workforce transition by ensuring reskilling initiatives and ethical policies that balance automation against human oversight, according to Dsouza (2023).

### *2.6.4 Gaps and Opportunities for Further Research*

Even with such comprehensive literature on most of the key ethical areas in AI-generated code and chatbots driving decision-making, there are no studies that have looked into how such risks might be mitigated in high-stake industries, like aviation. One of the critical areas for future investigation is developing standardized ethical frameworks for AI validation in those areas of application from which the recommendation of an AI might impact safety and security. Also, there is little empirical evidence on the long-run effects of AI on programming skills and research quality.

Recognizing the ethical risks associated with AI-generated outputs, this study focuses on a critical evaluation of the quality, reliability, and transparency of chatbot-developed predictive models. Identification of hallucinations, biases, and usability barriers will be one of the key aspects of this research study, with analyses of the findings to be presented in the Analysis and Results chapter and Discussion chapters.

# 3. Methodology

## 3.1 Introduction

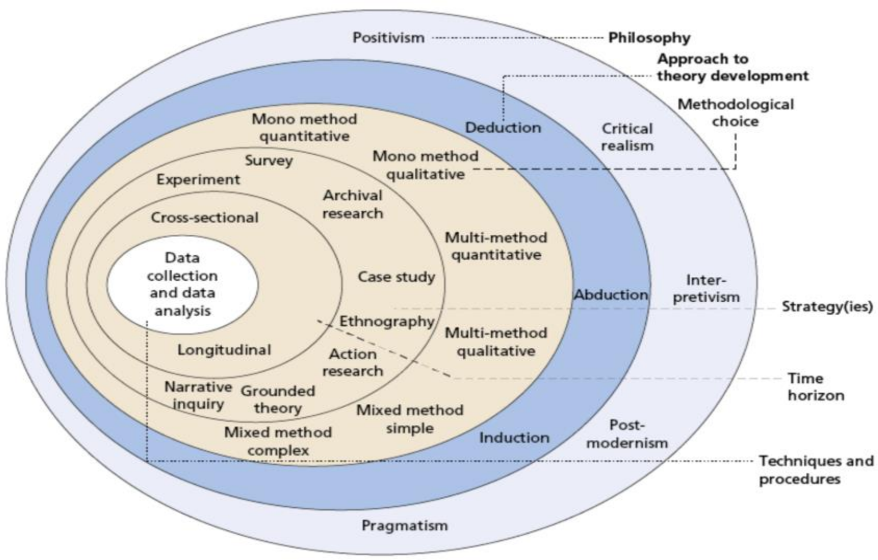
This section describes the research approach of the study, including the theoretical framework, research strategy, methods of data collection, and method of analysis. In their studies, O'Gorman & Macintosh (2014) argued that having a strong research methodology helps ensure that the study has clear logic and structure with the aim of the research. This research attempts to evaluate the usefulness of AI-powered chatbots: ChatGPT, DeepSeek, Google Gemini, Microsoft Copilot and Meta AI in the development of a flight delay prediction model. The methodology is structured according to Saunders et al.'s (2019) Research Onion model so that there is an orderly way of gathering, organizing, and analysing data.

## 3.2 Research Process

This study follows the Saunders' Research Onion model so that the decisions on methodology would include the relevant research methods for achieving the research objectives. The layers of the Research Onion describe each layer of the research philosophy, approach, methodological decisions, strategies, time scope, and techniques for gathering information. By peeling these layers towards the innermost, the study seeks to achieve consistency and dependability in posing the problem of the effectiveness of AI chatbots in predictive modelling in air traffic delays.

**Figure 1:**

*The research onion*




*Note.* From Research Methods for Business Students, by Saunders et al., 2019.

## 3.3 Research Philosophy

The methodology used on a specific research topic defends and explains how knowledge is created and rationalized. In Saunders et al. (2019), epistemology, ontology and axiology are listed as the core philosophies of a given research problem. This study is based on reality and uses a pragmatic philosophy that assumes knowledge and reality interact with one another. A pragmatic approach is most suitable here because it emphasizes the real-world application of AI chatbots in assisting with model development instead of just contemplating concepts. Because the study sets out to assess the usability and performance of AI chatbots in the aviation industry, pragmatism allows for the flexibility that is required to evaluate the AI chatbots' role in data science processes (Saunders et al., 2019).

## 3.4 Theory Development

This research uses an abductive reasoning method, which is a blend of deductive and inductive approach. Abduction stems from theorizing an explanation for an event or phenomena and then retesting it. Because this research claim engages with a specific problem (AI-driven flight delay prediction) and utilizes components of machine learning, it also looks into how AI chatbots can work towards solving the problem, which is why abduction is appropriate here. This approach gives freedom of using established machine learning theories while bringing in fresh concepts from AI chatbot interaction.

## 3.5 Research Methodology

This study uses a single qualitative method relying on the secondary data technique of data analysis. The purpose of this research is to assess the capability of AI chatbots to provide assistance in flight delay prediction through analysis of recorded interactions, responses, and code created by the bots. This clear step approach includes:

• Data cleaning: Assisting in the identification and resolution of missing data, including feature construction, and cleaning pertaining to flight delays.

• Data mining: Evaluating chatbot suggestions for the selection of important predictive factors.

• Teaching the model: How AI chatbots propose and realize machine learning models such as Random Forest, XGBoost, Neural Networks, etc.

• Code Improvement: Evaluation of the provided answers by the chatbots and the changes they wish to make to the coding issues.

## 3.6 Research Strategy

The methodology is based on the analysis of the effectiveness of the chats in determining the assistance of the prediction of flight delays. Secondary data sources are aviation delay datasets from the Kaggle, OpenSky Network, and METAR weather data. This process includes:

1. Interfacing with the chatbot and providing it prompts with a specific structure for machine learning operations.

2. Collecting the responses the chatbot provides along with any suggestions or commands it is able to code.

3. Evaluation of the bot's answers regarding the predicted value, usability, and interpretability of the offered results.

4. Evaluation of the difficulties and constraints encountered in modelling undertaken with the help of chatbots.

## 3.7 Time Horizon

Study focuses on the interaction and evaluation of chatbots within a limited point in time which doesn’t change over time (Saunders et al., 2019). This is done in consideration with the scope of the study which concentrates on analysing the effectiveness of AI chatbots within a particular timeframe.

## 3.8 Techniques and Procedures

To complete the study, the following techniques are used:

• Document Analysis: Looking into previously published works about AI chatbots, prediction of flight delays, and the application of various techniques from artificial intelligence and machine learning.

• Chatbot Testing: Testing ChatGPT, DeepSeek, Google Gemini, Microsoft Copilot and Meta AI in the model development activities.

• Comparative Performance Assessment: Reviewing the performance of the chatbots’ results based on set parameters such as accuracy of information, efficiency, interpretability, error rate, and reproducibility of the output.

With this methodology, a comprehensive evaluation of AI chatbots as research assistants in flight delay prediction is done to establish their usability, effectiveness, and shortcomings.

# 4. Analysis and Results

This chapter focuses on a critical and comprehensive analysis of the process of building a flight delay prediction model using Python in Visual Studio with the use of AI chatbots. It compares the experiences across several chatbot platforms such as ChatGPT, DeepSeek, Google Gemini, Microsoft Copilot, and Meta AI. In contrast to the results emerging from the academic literature discussed in Chapter 2 (Literature Review), the comparison evaluates how far AI chatbots meet their predicted promises of automating, simplifying, and assisting machine learning tasks for non-expert users.

## 4.1 Introduction to the Analytical Framework

In order to maintain objectivity in the evaluation, each chatbot was given the same starting query to ask for help in developing a machine learning model for predicting flight delays from weather-related data, date, and time. The entire process involved data loading, cleaning, feature selection, exploratory analysis, model training (classification and regression), and performance evaluation. The evaluation criteria included usability, understanding the context, error handling, adaptability, quality of output, integration of real-time API, and final performance metrics for the model.

## 4.2 ChatGPT: Iterative Guidance and Contextual Intelligence

According to all indications, ChatGPT showcased the most nicely structured yet user-friendly support across all platforms. It took the process through gradual, logical steps, with waiting for confirmation of input before proceeding. Noteworthy in this dialogic approach was the approach for non-expert users, as illustrated by Lung (2024), who noted the value of such interactivity of chatbots in pedagogical styles of model development with AI assistance.

ChatGPT cleaned the provided data by itself upon upload, showcasing the cleaned datasets formatted as tables within the chat interface. The user could select specific rows or columns and ask for further analysis, proving schema comprehension and manipulation features. This further resonates with Zhang et al. (2024), who acknowledged that GPT-4 exhibited superiority in schema matching and data imputation tasks. Moreover, ChatGPT correctly identified important features for the model, recommending dropping some, such as unique order numbers and airline carrier codes. Such an addition was integral because it ensured that the prediction model focused only on the weather variables. The literature of Chia (2024) pointed out that AI chatbots might hallucinate or misidentify important data attributes, but ChatGPT served this case with consistency and relevance in feature selection. The platform addressed missing values and provided descriptions on processing instructions for categorical variables and normalization of continuous ones. This aligns well with Rao et al. (2023) on how AI coding assistants are reputed for performing a preprocessing task with minimal input from a user.

ChatGPT also conducted exploratory data analysis (EDA), providing distribution patterns, delay trends by hour, and weather impact. All these findings comply with Bani (2025), who noted the improvement of analytical capacity through AI tools.

The development work with ChatGPT began by using a multi-class classification model that predicts the extent of flight delay. The prediction model created 4 classes from historical delays, ranging from none to severe delays. The implementation of this model shows an overall accuracy of 61.72%, which is fairly acceptable from the data but clearly indicates performance imbalances. The table below shows the classification report:

*Table 1: ChatGPT Classification Model Performance*

| Class | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| **0 (On-time/Early: <= 0 minutes)** | 0.27 | 0.16 | 0.20 | 3663 |
| **1 (Minor Delay: 1-15 minutes)** | 0.34 | 0.23 | 0.27 | 2753 |
| **2 (Moderate Delay: 16-60 minutes)** | 0.70 | 0.87 | 0.78 | 13114 |
| **3 (Severe Delay: > 60 minutes)** | 0.49 | 0.29 | 0.37 | 1710 |

*Note.* Results reflect ChatGPT's performance in classifying flight delays across four categories. Metrics are influenced by dataset imbalance favouring Class 2 (Moderate Delay).

The performance of the model was heavily skewed towards Class 2, the class that represents the majority of the dataset. Both macro and weighted averages showed that this class had the highest recall and precision, illustrating bias from the side of the classifier towards overrepresented outcomes. The degree of imbalance was reinforced by a relatively low macro-averaged F1 score of 0.41, denoting weak generalization over the underrepresented classes.

After multiple iterations, it became evident that the model predictions were virtually identical regardless of input changes. This led to the conclusion that the model was not dynamically responding to changing feature values. When this issue was raised within the ChatGPT interface, it was able to correctly identify the problem as dataset imbalance - the larger number of samples belonging to Class 2.

ChatGPT then offered some possible solutions: collecting a more balanced dataset or emphasizing the importance of weather-related features in the training process. The ability of the system to learn iteratively and self-correct is thus demonstrated, which reinforces the argument put forward by Tufano et al. (2024) that advanced AI development tools can indeed revise their outputs in light of issues raised by the user.

When requested to shift to a regression model, ChatGPT most effectively rewrote its code and demonstrated flow and contextual memory. It generated a graphical user interface that showed the feature importance as well. The final regression metrics were:

- **Mean Absolute Error (MAE):** 20.15
- **Mean Squared Error (MSE):** 1481.66
- **$R^2$ Score:** 0.145

While not groundbreaking, these results were functional, demonstrating the overall reliability of the chatbot in terms of producing usable predictive models. Although the classification model was not uniformly strong across all categories, the interaction yielded useful diagnostic feedback and possible strategies for improvement. Such experiences indicate that ChatGPT serves not only as a model-generating tool but also as an impact analytic reflection, hence acting as a co-pilot to those who are not familiar with or skilled enough to perform the machine learning workflows themselves.

**Friendliness and Conversational Quality**

Since the start, ChatGPT has been very pleasant and cooperative in tone. The instruction set was broken into small steps that were easy to follow, with the chatbot waiting for the user to proceed, making it a bit interactive and less overwhelming. The responses were polite, explanatory, and supportive throughout, even in the case of an error. Instead of returning an error message, ChatGPT graciously followed with some genuine explanations and fixes to boost user confidence.

**Conversation Evolution Over Time**

When interacting with ChatGPT, it demonstrated excellent conversational memory by accurately retaining previous inputs and adapting its recommendations as the task went on. It was also very coherent and contextually aware. It demonstrated the capacity to build on previous context, for example, by pointing up issues with the initial classification model, starting to modify the code, creating a GUI element, and then switching to regression modelling in accordance with previous instructions. The bot's ability to learn from past errors and produce better results created the appearance of a genuine dialogue, where its thinking improved with each interaction.

## 4.3 DeepSeek: Excessive Detail and Memory Gaps

DeepSeek had multiple important limitations. The replies were full of unnecessary processes and did not take user context or readiness into account. This was counter-productive to the adaptive interaction model proposed by Veras et al. (2023), which stated that chatbots should balance information density and user understandability.

DeepSeek could provide assistance in the initial set-up, but the bot could not clean the data autonomously and seriously mismanaged iterative error correction. Each time there was an error, the bot would regenerate the steps leading to that error, which was inefficient and frustrating. This failure to maintain state mirrors what Mishra et al. (2024) have discussed - an LLM has poor performance in handling multi-turn complex queries in technical fields.

DeepSeek's replies to error messages were accurate but too repetitive, like giving documentation-oriented quotations without considering the user's context. After five attempts, it refused to proceed with the prediction due to accumulated inconsistencies, which made the user revert to clean datasets from ChatGPT.

It also failed to implement the API, providing very general or sometimes missing suggestions regarding the use of real-time data. This goes against the real-time integration in Rodriguez (2024), who highlighted real-time model flexibility in modern aviation applications.

In the end, it became quite impossible for DeepSeek to create any predictive model due to the length limit of the chat, exposing prominent usability and continuity issues that further undermine the tool's viability for complex workflows.

**Friendliness and Conversational Quality**

DeepSeek's tone was neutral and informative but at times rather impersonal and overwhelming. It sometimes shared long passages of information or very many steps without checking if the user was ready for them or even needed them. Responses poured the conversation with technical information, which would be correct, however, what were missing were the conversational balance for non-experts to remain in the conversation.

**Conversation Evolution Over Time**

The flow of the conversation with DeepSeek worsened with the passage of time. It was unable to maintain any concentration on the current work and frequently lost track of previous messages, particularly when debugging code. On several occasions, the chatbot followed its instructions regardless of changes to the earlier problems it had observed, repeating earlier efforts to fix an error. The level of continuity was interrupted when the user had to restart the inquiry procedure after multiple failed efforts at repair because the conversation finally reached a chat length limit. In the face of such intricate tasks, this resulted in a chaotic and ineffective experience.

## 4.4 Google Gemini: Speed Over Depth

Gemini provided speedy responses with a lack of depth and context in its reasoning. Early on, it hallucinated column names (a common fault pointed out by Chia, 2024) and failed to detect redundant variables like Hourly Altimeter Settings and Tail Numbers. Although it eventually provided executable data-cleaning code, the errors continued due to outdated assumptions regarding the structure of the dataset. Feature engineering hit memory issues, an observation that matches hardware-sensitive problems noted by Rao et al. (2023). Attempts to resolve this gave rise to new errors, with the chatbot showing a failure to maintain awareness of its earlier outputs. Gemini's first classification model was very worrisome, as it created just two classes (delayed vs. not delayed), Class 0 precision being 0.44, its recall being 0.49, and its F1-score 0.46. It also had an overall very high reported accuracy (0.998); however, the performance was misleading due to a severe class imbalance, through which a dominating class, that of non-delayed entries, was found within the data (support = 99405 vs. 100). Therefore, the following outcomes reaffirm the findings of Hultberg et al. (2024) that chatbots can give high-level metrics that disguise poor subgroup performance. It preserved the same structure after switching to the XGBoost model, but it was able to increase the precision of delayed predictions. It eventually presented the classification model with the following classifying performance:

- **Precision (Class 0):** 0.44
- **Accuracy:** 0.9989
- **Precision (Overall):** 0.9994

- **Recall:** 0.9993
- **F1 Score:** 0.9994

In regression mode, Gemini delivered surprisingly strong metrics:

- **MSE:** 1636.59
- **MAE:** 10.35
- **$R^2$ Score:** 0.98

Despite the promisingly high performance indicators, the model predicted a extreme value of a 16-hour delay, which showed inconsistency and the need for cautious interpretation.

**Friendliness and Conversational Quality**

Gemini responded quickly and efficiently, but lacked warmth and patience in its tone. It would often provide code snippets quickly, but would make assumptions that were not based in fact. Its tone was unsupportive when it came to users making an error. In later stages, issues with its responses would quite easily be diverted back to users with suggestions for manual debugging, which came across as dismissive rather than helpful.

**Conversation Evolution Over Time**

Its impairment in conversation quality was largely due to bad memory and weak contextual tracking. It started hallucinating non-existent column names and then quickly forgot some of its own suggestions for earlier steps within the conversation. While troubleshooting errors, it provided solutions that were entirely disconnected from prior steps, causing confusion and redundancy. Although some of the model's performance improved, there was no adaptability demonstrated in the conversation, and often, memory lapsing disrupted the continuity of the workflow.

## 4.5 Microsoft Copilot: Incomplete Yet Functional

Copilot had a very disappointing analysis. It failed to process large inputs exceeding the token limit (8000 characters) and could neither automate the cleaning of data, wrongly assuming which columns to drop in the dataset and which to keep as features for model training. One-hot encoding gave rise to dimensionality problems with repetitive errors that it couldn't eliminate alone. Being stuck in a loop, the user was forced to restart the whole process. After indicating which columns to use as model training features, Copilot provided correct Python code. When dealing with real-time data, Copilot provided useful steps, which guided the API-based live weather-data integration. At first, there was confusion, but it finally ended up being able to produce a working regression model:

- **MAE:** 19.67
- **$R^2$:** 0.20

This was the best regression performance for reducing errors. However, even multiple attempts were useless for Copilot as it could not produce a working classification model, noted by Pividori (2024), which stated that it was unsuitable for different modelling requirements.

**Friendliness and Conversational Quality**

The Copilot provided some useful directions, but had a terribly mechanical tone and a rather rigid feel to it. Its performance was sluggish, and it was prone to making incorrect assumptions, especially in the early stages of the process, such as determining which columns to use. Even after providing steps for API integration, large input handling failures and persistent unresolved errors made the experience extremely unpleasant. User support was sparse when errors appeared, and the replies were often shallow, lacking the depth to guide user through troubleshooting.

**Conversation Evolution Over Tim**e

As soon as tasks became complicated, Copilot found it difficult to maintain a dialogue. Incoherent interactions and frequent restarts were caused by input limits and cross-memory limitations. When the functional regression model was eventually created, it was unable to provide a classification model and lacked a mechanism to enhance its performance based on previous discussions. For the duration of the project, this discrepancy hindered the chatbot's ability to establish an effective or learner-friendly work ethic.

## 4.6 Meta AI: Limited Input Handling and Syntax Errors

Upon evaluation, Meta AI, as a chatbot-assisted model development tool, revealed critical shortcomings making it unfit for developing a flight delay prediction model. Similarly to other evaluated chatbots, Meta AI was unable to process or clean datasets on its own. This shortcoming contradicts what Zhang et al. (2024) mentioned about LLMs being able to assist with preprocessing features such as schema matching and data imputation.

While Meta AI was prompted to provide Python code for data cleaning and analysis, the code it provided was syntactically incorrect, due to the excessive presence of non-functional comments and unclear formatting. Such comments were probably written for the sake of readability but caused syntax errors in the development environment. This was similar to Ghosh et al.'s (2024) understanding that AI-generated codes do not conform to functional coding standards, thereby introducing redundancy and inefficiency.

Another major problem was the acute input token limitation present in Meta AI, which prevented it from accepting longer error messages generated within the Visual Studio Code environment or even extended Python code snippets. This unnatural drawback took a heavy toll on debugging interactivity. The inability to communicate full error messages or receive multi-line code corrections hindered progress, as this was an iterative process while developing a machine learning pipeline. The same limitations were pointed out by Mishra et al. (2024), who stated that LLM-based assistants inadequately addressed more complex tasks due to token limitations.

The technical drawback made Meta AI virtually unworkable except during the initial stages of interaction. With syntax issues found and messages being too long to paste, further model development could not proceed. Such a scenario stands in stark contrast to the expectations of Lung (2024) and Rao et al. (2023), who focused on the great promise that AI chatbots hold in supporting iterative software development. Here, the restrictions on conversation and functionality eroded Meta AI's usability.

Meta AI also lacked the capability for self-correction or alternative troubleshooting suggestions, as is available with other chatbots. This absence of self-correction is a significant shortcoming, given that Laoudai (2024) and Chen et al. (2024) have both stressed the relevance of explainability and responsiveness for high-stakes analytical tasks.

**Friendliness and Conversational Quality**

Although characterized by a polite tone, the apparent utility of Meta AI was hindered by various technical limitations. It returned Python code decorated with excessive comments, causing syntax errors, while crude limitations around the length of inputs rendered interactions tedious. There was no direct impoliteness or unhelpfulness in its tone; however, the absence of any usable content was irritating.

**Conversation Evolution Over Time**

Meta AI engagement was very brief and unproductive. The chatbot could not engage in any iterative improvement due to input-length restrictions and failed syntaxes. It could not build upon any previous steps, nor did it remember previous codes that might help in carrying on the development. This made dialogue evolution impossible, reducing the role of the chatbot to rather an incomplete one-directional dialogue.

## 4.7 Comparative Performance Overview

*Table 2: AI Chatbot Performance Comparison*

| Chatbot | Version | Data Cleaning Ability | Classification Performance | Regression Performance | Error Handling | Usability & Memory | Friendliness | Conversation Evolution |
|---|---|---|---|---|---|---|---|---|
| **ChatGPT** | Paid | Automated with feature selection, missing value handling, and merging | Accuracy: 61.72%, Macro F1: 0.41, Bias towards class 2, iterative improvement | MAE: 20.15, MSE: 1481.66, $R^2$: 0.145 | Excellent, interactive debugging with iterative improvement | High contextual awareness and memory within session | Very friendly and step-by-step | Adapted and evolved with feedback |
| **DeepSeek** | Not aplicable | Basic dropna, failed to clean full dataset, required external help | Failed to complete classification, memory issues | Did not complete | Lost track during debugging, confusing replies | Verbose, lacks continuity in responses | Neutral but overwhelming | Fragmented and lost memory |
| **Google Gemini** | Free | Partial, hallucinated columns, caused errors, high memory usage | Accuracy: 99.88% (overfit), Macro F1: 0.73, Precision 0.44 for class 0 | MAE: 10.35, MSE: 1636.59, $R^2$: 0.98 (overfit) | Provided faulty code multiple times, context lost easily | Low, limited to short prompts, easily forgets context | Fast but impersonal | Quickly forgot earlier context |
| **Microsoft Copilot** | Free | Incorrect assumptions, required manual column selection | Failed to deliver classification model | MAE: 19.67, $R^2$: 0.20 | Handled some errors but needed manual correction and resetting | Poor memory, frequent restarts needed | Mechanical and slow | Disconnected conversations |
| **Meta AI** | Not aplicable | Not capable; provided unusable code with syntax errors | Not evaluated due to input length limits and syntax errors | Not supported due to interface limits | Failed due to token limit, could not handle long inputs | Extremely limited message length, no continuity | Courteous but unhelpful | No real conversation possible |

*Note. Summary of five AI chatbots' performance in predictive modelling. Differences may reflect varying access levels (e.g., free vs. paid versions).*

## 4.8 Summary

The analysis revealed that the chatbots were highly divergent in their performance in guiding non-experts towards the development of data-driven prediction models. ChatGPT ranked as the most reliable and capable, which was in alignment with literature that proves the adaptability and usability of AI chatbots (Zhang et al., 2024; Lung, 2024). Copilot might work well under structured environments and with unambiguous prompts, but it lacks the autonomy. Although Gemini provided the most robust metrics from its prediction models, it still has some unpredictable outputs as a result of hallucinations and poor contextual grounding. The two DeepSeek and Meta AI could not produce usable results. DeepSeek had memory issues and overloaded irrelevant information, while Meta AI had issues related to the message length and clarity of code.

This shows that domain knowledge and human supervision are important in ensuring meaningful, accurate results when working with AI chatbots (Ma et al., 2023; Merigan, 2023). Thus, while promising, AI chatbots still require refinement and improvements in contextual awareness before completely replacing human guidance in predictive modelling tasks.

# 5. Discussion

## 5.1 Summary of Key Findings

This study aimed to assess whether AI chatbots, primarily ChatGPT, DeepSeek, Google Gemini, Microsoft Copilot, and Meta AI, could provide useful assistance to non-experts in developing real-time predictive models for forecasting flight delays. The results showed a significant disparity in platform performance among the evaluated AI. ChatGPT consistently achieved the highest performance, encompassing usability, context awareness, and iterative feedback, while DeepSeek and Meta AI were limited by memory and input constraints, respectively. On the other hand, some platforms, including Gemini or Copilot, indicated some promise through regression metrics or partial API integration but were inconsistent and lacking in adaptability throughout the entire modelling pipeline.

## 5.2 Interpretation of Results

The results indicate several clear trends:

- Contextual Retention and Memory Handling: ChatGPT's conversational memory enabled it to build on earlier conversations and continue the modeling process in a coherent manner. Similarly to Lung (2024), whose research stressed the pedagogical value of chatbots that assist iterative user-guided development.
- Feature Selection and Data Preprocessing: While ChatGPT excelled in autonomous data cleaning and schema detection, Gemini and Copilot struggled with hallucinated columns and inappropriate feature selections, confirming Chia's (2024) findings regarding these hallucination risks.
- Classification vs. Regression Performance: Notwithstanding high regression $R^2$ scores, Gemini's binary classification model exhibited severe overfitting, indicative of class imbalance. Balanced solution of this issue by ChatGPT confirms Rao et al.'s (2023), opinions regarding the use of chatbots in feature weighting and model diagnostic utility.
- DeepSeek failed to support entire predictive pipelines owing to the presence of repetition loops and chat length limitation. This adds weight to Mishra et al.'s (2024) assertion on the current state of large language model difficulty for such multi-step, domain-specific workflows.
-

## 5.3 Implications of the Research

The findings bring strong notions about rankings concerning the democratization of machine learning in high-stake real-time applications such as aviation:

- Widening Accessibility: Chatbots, like ChatGPT, can lower the access threshold for non-technical users to engage in exploratory modeling and hypothesis testing in domains formerly reserved for data scientists.
- Operational Usability: Evidence for data preprocessing, model running, and code refinements on iterations suggests that some chatbots might already be in the low-code co-pilot territory, albeit with caveats. This fortifies the general observation touched upon in the literature review regarding the normalization of AI within science workflows (Rao et al., 2023; Shaikh, 2023).
- Model Accountability: The disparities and hallucinations have brought attention to the need for human intervention and a delineated specific evaluation framework to these types of models (Samuylova, 2025).

## 5.4 Limitations of Study

There were some limitations to the study that should be acknowledged:

- Short-Term Evaluation: The study was conducted in a very short time, which failed to determine the long-term reliability of workflows guided by chatbots.
- Limited Data Complexity: Though simulated with real-time features, the models did not integrate live data streaming pipelines beyond weather information, as described by Rodriguez (2024) or Durgut (2025).
- Platform-Specific Limitations: Performance for Meta AI and Copilot was curtailed by token limitations, memory loss, and truncated responses, which may not have even scratched the surface of their capabilities in other contexts.

However, it should also be noted that the evaluation was contingent upon the versioning and accessing level of the AI chatbots. For example, the user tested ChatGPT using the paid version of the chatbot (GPT-4), which has higher capabilities such as better retention of memory and enhanced reasoning. In contrast, Google Gemini and Microsoft Copilot were assessed using their free versions, while DeepSeek and Meta AI currently offer free access only. Such discrepancies in access levels could be one of the reasons for different performances and should, therefore, be given consideration when interpreting the comparative results.

Nonetheless, the study remains valid in showing that AI chatbots would meaningfully support some aspects of the modelling process, particularly under constricted and semi-automated conditions.

## 5.5 Recommendations to Practice and Future Research

The following are the recommendations based on the findings:

- For Practitioners: Firstly, aviation companies and logistics organizations could consider the use of chatbot interfaces to support their data systems in exploratory modelling and early-stage analytics with human oversight in final decision-making.

- For Developers: Contextual memory, interpretability, and incorporation of real-time data input should be given priority for AI chatbot platforms to support professional workflows. Additionally, embedding online debugging tools while improving the handling of class imbalance scenarios would be beneficial.
- For Future Research:
    - Conduct longitudinal studies considering chatbot-guided modelling over time with dynamic datasets.
    - Investigate the use of chatbots among various expert domains, for example, aviation versus healthcare.
    - Create thorough benchmarking frameworks for chatbot-assisted model development in high-stakes fields.

# 6. Conclusion

This research aimed to determine whether mainstream AI chatbots, specifically ChatGPT, DeepSeek, Google Gemini, Microsoft Copilot, and Meta AI, could adequately assist non-specialist in developing air traffic delay predictive models by using real-time data inputs and Python code. Air traffic delays have long been a bottleneck in the aviation sector, affecting efficiency, cost, and passenger satisfaction. Given the intricate processes involved in machine-learning models' development, this study was designed to find out whether AI chatbots may help bridge the gap in expertise and, therefore, democratize access to advanced predictive analytics.

## 6.1 Summary of Findings

This study found significant divergences among the capabilities of various AI chatbots, yet they present a real possibility of helping with some machine learning workflow stages. It is the credibly innovative interactive assistance of ChatGPT, that newly provides the user quite comfortably with data preprocessing, feature selection, model training, and performance evaluation. In contrast, Gemini and Copilot demonstrated their own strengths in certain areas (regression metrics, for example, or integration with an API) but were found lacking in coherence, memory storage, or rectification. DeepSeek and Meta AI, despite being compared and evaluated, exhibited massive verbosity, poor workflow management, and token limitations.

The implications from these findings serve to solidify the position that AI chatbots are not substitutes for human dominance yet act as strong assistants, especially in early modelling, debugging, and decision support.

## 6.2 Contributions to Research and Practice

This thesis contributes to the emerging research line into the usability and trustworthiness of AI chatbots for data science work. Unlike current substantial studies involving AI on coding or customer services, this research looks at the exercise of chatbots in high-impact, domain-specific settings such as aviation analytics. It illustrates that:

- AI chatbots can reduce the barrier to entry for non-experts, allowing for exploratory modelling and hypothesis testing.
- Chat-based AI models such as ChatGPT enable feature selection using real-time data, run API via code, and check for correctness iteratively.
- The performance of chatbots has considerable variability, indicating the necessity of human oversight when working with such assisted workflows.

The implication for practitioners is that the integration of chatbot interfaces into real-world aviation systems may be possible, notably for exploratory analysis, early modelling, or teaching. For developers, it highlights the importance of context memory and enhancements to error recovery.

## 6.3 Limitations

This study has a number of limitations. The study was conducted in a very short period of time and did not analyse collaborative workflows or longitudinal performance. The study mainly involved structured aviation datasets and did not take into account anything related to highly unstructured or multimodal datasets. Furthermore, disparities in the platforms through which these chatbots are accessed, such as GPT-4o versus the free-tier alternatives, could also affect the performance comparisons.

Live data integration has been implemented only partially due to the constraints of APIs and system limitations, and the full potential of live modelling could not be realized under AI guidance.

## 6.4 Recommendations for Future Research

Longitudinal studies could be designed in the future to ensure the sustained reliability of chatbot-guided modelling over time and with dynamic datasets. Comparisons among other domains, from health to finance, will further enhance our understanding of between-domain effectiveness of chatbots, and will include yet another variable to examine in future research. There is certainly a need for standardized benchmarking frameworks measuring AI chatbots in terms of correctness, usability, transparency, and flexibility to adaptation in complex environments. Hybrid models where AI chatbots and human experts act cooperatively in safety-critical industries such as aviation need to be explored as well.

## 6.5 Last Reflections

These chatbots, while still much in evolution, hold out great prospects for data-driven companionship. This work shows that with proper guardrails and human oversight, they can be transformative in democratizing machine learning by bringing advanced tools into the hands of users who might otherwise find themselves excluded by technical barriers of advanced tools. Closing the gap between the promise of data science and its real-world implementation, especially in high-stakes sectors demanding accuracy, speed, and flexibility.

# References

Alfarhood, M., Alotaibi, R., Abdulrahim, B., Einieh, A., Almousa, M., & Alkhanifer, A. (2024). Predicting Flight Delays with Machine Learning: A Case Study from Saudi Arabian Airlines. *International Journal of Aerospace Engineering*, *2024*(1), e3385463. https://doi.org/10.1155/2024/3385463

Alphaidze, T., & Renart, P. (2024, October 15). *From Chaos to Control: Real-Time Data Analytics for Airlines*. MongoDB. https://www.mongodb.com/blog/post/from-chaos-to-control-real-time-data-analytics-for-airlines

Atemkeng, M., Hamlomo, S., Welman, B., Oyentunji, N., Ataei, P., & Louis, J. (2024). Ethics of Software Programming with Generative AI: Is Programming without Generative AI always radical? *ArXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2408.10554

Banerjee, D., Singh, P., Avadhanam, A., & Srivastava, S. (2023, August 8). *Benchmarking LLM powered Chatbots: Methods and Metrics*. ArXiv.org. https://doi.org/10.48550/arXiv.2308.04624

Bani. (2025). *AI Chatbot for Data Analytics: Improving Efficiency and Accuracy | Tars Blog*. Tars. https://hellotars.com/blog/ai-chatbot-for-data-analytics-improving-efficiency-and-accuracy?utm_source=chatgpt.com

Brooker, A. (2024, June 3). *Predicting Flight Delays: How Airlines Are Harnessing AI to Minimize Disruptions*. Cirium. https://www.cirium.com/thoughtcloud/predicting-flight-delays-how-airlines-are-harnessing-ai-to-minimize-disruptions/

ChatBees. (2024, July 13). *5 Chatbot Evaluation Metrics To Improve Your Chatbot Performance*. ChatBees. https://www.chatbees.ai/blog/chatbot-evaluation-metrics?utm_source=chatgpt.com

Chen, D., Huang, R. S., Jomy, J., Wong, P., Yan, M., Croke, J., Tong, D., Hope, A., Eng, L., & Raman, S. (2024). Performance of Multimodal Artificial Intelligence Chatbots Evaluated on Clinical Oncology Cases. *JAMA Network Open*, *7*(10), e2437711. https://doi.org/10.1001/jamanetworkopen.2024.37711

Chia, A. (2024, August 29). *6 Unique Ways to Use AI in Data Analytics*. Datacamp.com; DataCamp. https://www.datacamp.com/blog/unique-ways-to-use-ai-in-data-analytics?utm_source=chatgpt.com

Dai, M. (2024). A hybrid machine learning-based model for predicting flight delay through aviation big data. *Scientific Reports*, *14*(1), 4603. https://doi.org/10.1038/s41598-024-55217-z

Dataquest Labs. (2024). *Capabilities and Limitations of AI Chatbots: Takeaways*. Dataquest Labs, Inc.

Daython3. (2024, January 23). *AI Chatbots: Transforming Data Science Productivity and Insights*. Medium. https://medium.com/@daython3/ai-chatbots-transforming-data-science-productivity-and-insights-f839abe52d31

Dsouza, L. F. (2023, February 8). *Ethical considerations of AI in the Aviation Industry*. Medium. https://medium.com/%40lynnfdsouza/ethical-considerations-of-ai-in-the-aviation-industry-be12bf7677fe

Durgut, M. (2025, February 4). *Challenges Associated with Big Data in Aviation and Solutions from a Velocity Perspective*. Aviationfile-Gateway to Aviation World | Aviation Pioneers, Accidents. https://www.aviationfile.com/challenges-associated-with-big-data-in-aviation-and-solutions-from-a-velocity-perspective/

Eurocontrol. (2024, October 15). *Digitalisation and AI in Air Traffic control: Balancing Innovation with the Human Element*. Eurocontrol.int. https://www.eurocontrol.int/article/digitalisation-and-ai-air-traffic-control-balancing-innovation-human-element

FAA. (2022). *NextGen Annual Report*. U.S. Department of Transportation.

Freshworks. (2024, April 12). *Essential chatbot analytics 2024- ultimate guide*. Freshworks. https://www.freshworks.com/chatbots/analytics/

Ghosh, D., Zhu, H., & Bayley, I. (2024). Benchmarks and Metrics for Evaluations of Code Generation: A Critical Review. *ArXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2406.12655

Gopalakrishnan, K. K., & Balakrishnan, H. (2017). *A comparative analysis of models for predicting delays in air traffic networks*.

Gupta, A. (2020, October 10). *Feature Selection Techniques in Machine Learning*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/

Hultberg, P. T., David Santandreu Calonge, Firuz Kamalov, & Smail, L. (2024). Comparing and assessing four AI chatbots' competence in economics. *PloS One*, *19*(5), e0297804–e0297804. https://doi.org/10.1371/journal.pone.0297804

Khoury, R., Avila, A. R., Brunelle, J., & Camara, B. M. (2023, April 19). *How Secure is Code Generated by ChatGPT?* ArXiv.org. https://doi.org/10.48550/arXiv.2304.09655

Laoudai, O. (2024, September 10). *AI-Powered Analytics vs. Traditional Data Analysis: Which Is Better for Consultancy Firms*. Infomineo. https://infomineo.com/blog/ai-powered-analytics-vs-traditional-data-analysis-which-is-better-for-consultancy-firms/

Law, M. (2023, May 24). *Top 10 no-code AI tools*. Aimagazine.com; Bizclik Media Ltd. https://aimagazine.com/top10/top-10-no-code-ai-tools

Lawton, G. (2024, July 23). *Generative AI Ethics: 8 Biggest Concerns*. TechTarget. https://www.techtarget.com/searchenterpriseai/tip/Generative-AI-ethics-8-biggest-concerns

Li, M. (2018). Air Traffic Delay Prediction Based on Machine Learning and Delay Propagation. *Open Access Theses*, *1415*.

Lung, L. (2024, September 26). *Training Data and AI: How an AI Chatbot is Trained*. Qualimero. https://www.qualimero.com/en/blog/training-ai-chatbots-guide

Ma, W., Song, Y., Xue, M., Wen, S., & Xiang, Y. (2023). The "code" of Ethics: A Holistic Audit of AI Code Generators. *ArXiv.org*, *abs/2305.12747*. https://doi.org/10.48550/arXiv.2305.12747

Malec, M. (2024, December 2). *Large Language Models: What You Need to Know in 2025 | HatchWorks AI*. HatchWorks AI. https://hatchworks.com/blog/gen-ai/large-language-models-guide/?utm_source=chatgpt.com

Mamdouh, M., Ezzat, M., & Hefny, H. A. (2023). A novel intelligent approach for flight delay prediction. *Journal of Big Data*, *10*(1). https://doi.org/10.1186/s40537-023-00854-w

Merigan, N. (2023, August 9). *Feature Selection Importance in Machine Learning Algorithms*. Pss Blog. https://www.pranathiss.com/blog/machine-learning-algorithms/

Mishra, M., Braham, A., Marsom, C., Chung, B., Griffin, G., Sidnerlikar, D., Sarin, C., & Rajaram, A. (2024). DataAgent: Evaluating Large Language Models' Ability to Answer Zero-Shot, Natural Language Queries. *2024 IEEE 3rd International Conference on AI in Cybersecurity (ICAIC)*, 1–5. https://doi.org/10.1109/icaic60265.2024.10433803

Nguyen, M. H., João Sedoc, & Taylor, C. O. (2024). Usability, Engagement, and Report Usefulness of Chatbot-Based Family Health History Data Collection: Mixed Methods Analysis. *Journal of Medical Internet Research*, *26*, e55164–e55164. https://doi.org/10.2196/55164

O'gorman, K. D., & Macintosh, R. (2014). *Research methods for business & management : a guide to writing your dissertation*. Goodfellow Publishers Limited.

Paharia, P. T. (2024, March 6). *News-Medical*. News-Medical. https://www.news-medical.net/news/20240306/Exploring-researcher-attitudes-The-scientific-potential-of-artificial-intelligence-chatbots.aspx?utm_source=chatgpt.com

Pividori, M. (2024). Chatbots in science: What can ChatGPT do for you? *Nature*. https://doi.org/10.1038/d41586-024-02630-z

Quan, S., Yang, J., Yu, B., Zheng, B., Liu, D., Yang, A., Ren, X., Gao, B., Miao, Y., Feng, Y., Wang, Z., Yang, J., Cui, Z., Fan, Y., Zhang, Y., Hui, B., & Lin, J. (2025). CodeElo: Benchmarking Competition-level Code Generation of LLMs with Human-comparable Elo Ratings. *ArXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2501.01257

Rao, N., Tsay, J., Kate, K., Hellendoorn, V. J., & Hirzel, M. (2023). AI for Low-Code for AI. *Cornell University*. https://doi.org/10.48550/arXiv.2305.20015

Reda, M. A. (2024). Intelligent Assistant Agents: Comparative Analysis of Chatbots through Diverse Methodologies. *Global Scientific Journals*, *12*(10). https://doi.org/10.13140/RG.2.2.23344.57602

Reitmann, S., & Schultz, M. (2022). An Adaptive Framework for Optimization and Prediction of Air Traffic Management (Sub-)Systems with Machine Learning. *Aerospace*, *9*(2), 77. https://doi.org/10.3390/aerospace9020077

Rodriguez, M. (2024, December 16). *Real-Time Data and Machine Learning for Air Traffic Management*. Mosaic ATM. https://mosaicatm.com/2024/12/16/machine-learning-for-air-traffic-management/

Samuylova, E. (2025, February 20). *20 LLM evaluation benchmarks and how they work*. Evidentlyai.com. https://www.evidentlyai.com/llm-guide/llm-benchmarks

Saunders, M., Lewis, P., & Thornhill, A. (2019). *Research Methods for Business Students* (8th ed.). Pearson.

Shaikh, R. (2023, September 24). *How to train an Chatbot with Custom Datasets - Rayyan Shaikh - Medium*. Medium. https://medium.com/%40shaikhrayyan123/how-to-train-an-chatbot-with-custom-datasets-107ce09f4326

Smutny, P., & Bojko, M. (2024). Comparative Analysis of Chatbots Using Large Language Models for Web Development Tasks. *Applied Sciences*, *14*(21), 10048. https://doi.org/10.3390/app142110048

Synoptek. (2023, November 17). *Synoptek*. Synoptek. https://synoptek.com/insights/it-blogs/data-insights/ai-chatbots-understanding-the-benefits-and-limitations/?utm_source=chatgpt.com

Timbó, R. (2024, December 31). *Understanding the Risks & Benefits of AI Code*. Www.revelo.com. https://www.revelo.com/blog/ai-generated-code

Tufano, M., Agarwal, A., Jang, J., Moghaddam, R. Z., & Sundaresan, N. (2024). AutoDev: Automated AI-Driven Development. *ArXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2403.08299

Uramis, P. (2023, November 23). *How to Test Your AI Chatbot? Techniques, Tools, and Metrics - LiveChatAI.com*. Livechatai.com. https://livechatai.com/blog/how-to-test-your-ai-chatbot

Veras, M., Dyer, J.-O., Rooney, M., Silva, P. G. B., Rutherford, D., & Kairy, D. (2023). Usability and Efficacy of Artificial Intelligence Chatbots (ChatGPT) for Health Sciences Students: Protocol for a Crossover Randomized Controlled Trial. *JMIR Research Protocols*, *12*(1), e51873. https://doi.org/10.2196/51873

Weka. (2023, June 26). *AI: A Complete Guide in Simple Terms*. WEKA. https://www.weka.io/learn/guide/ai-ml/what-is-ai/

Yetiştiren, B., Özsoy, I., Ayerdem, M., & Tüzün, E. (2023). Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT. *ArXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2304.10778

Zhang, H., Dong, Y., Xiao, C., & Masafumi Oyamada. (2024). Large Language Models as Data Preprocessors. *ArXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2308.16361

## Appendix 1: Detailed ChatGPT Conversation Transcript

**User:** Wants help developing a Python machine learning model in Visual Studio Code to predict flight delays or cancellations using real-time weather data. They have two datasets: historical hourly weather data at JFK (2010–2018) and flight departure/delay data for JFK, EWR, LGA in 2013. Both datasets need cleaning and will be used to train the model, which should then predict delays based on current weather.

**ChatGPT:** Provides a plan to tackle the project. It suggests:

1. **Load & Clean Data:** Read the historical weather and flight datasets, handle missing values and inconsistencies.
2. **Feature Engineering:** Extract relevant weather features (temperature, wind, visibility, precipitation, etc.), merge the datasets on date/hour and airport, and categorize flight delays (e.g., on-time vs. delayed vs. cancelled).
3. **Real-Time Weather Data:** Use an API (like OpenWeather) to fetch current weather for JFK in the same format as the historical data.
4. **Model Training:** Train a machine learning model (e.g., Random Forest or XGBoost, possibly even an LSTM for time series) on the cleaned data to predict delays. Evaluate it with appropriate metrics (accuracy, F1-score, etc.).
5. **Deployment:** Optionally, deploy the model via a web service (Flask/FastAPI) to provide real-time delay predictions.

ChatGPT asks if the user has a preferred weather API or wants any visualization dashboard.

**User:** Provides the JFK weather dataset (an hourly historical weather dataset for JFK) for the model. (The user's interface attempted to show a preview, but it wasn't displayed.)

**ChatGPT:** Proceeds to inspect and clean the JFK weather data. It notes the dataset is large (~114,545 rows, 90 columns) and identifies several issues: many missing values, some numeric columns stored as strings, and some irrelevant columns. Key relevant columns are selected (Date/Time, temperature, dew point, wind speed/direction, visibility, precipitation). ChatGPT converts the date column to datetime, fixes data types for numeric columns, and drops unusable columns. It shows a snippet of the cleaned JFK weather data with those fields and a few sample rows [Cleaned JFK weather data table]. The cleaning steps are summarized (e.g. converting types, filling or removing missing data). ChatGPT then asks the user to review the cleaned data and suggests uploading the flight data next for cleaning and merging.

**User:** Points out that the "HOURLYPRSENTWEATHERTYPE" column (present weather type) should be excluded from the analysis.

**ChatGPT:** Acknowledges the request and removes the "HOURLYPRSENTWEATHERTYPE" column from the weather dataset. It presents an updated weather data sample without that column [Updated JFK weather dataset snippet]. The dataset is now ready. ChatGPT confirms the column was dropped and invites the user to provide the flight dataset so it can proceed to clean and merge it with the weather data.

**User:** Shares the flight departures and delays dataset for the New York airports (JFK, EWR, LGA for 2013). (Again, a preview was attempted but not displayed.)

**ChatGPT:** Loads and examines the flight data. It reports the dataset's size (~336,776 rows, 19 columns) and points out problems: there are missing values in fields like departure time and delay, the time_hour column (flight departure timestamp) is in string format, and there are redundant columns (year, month, day, etc. which are already part of time_hour). It also notes that some delays are negative (indicating flights that departed early).

ChatGPT outlines the cleaning strategy: keep only relevant columns and remove or consolidate others. It retains time_hour (to be converted to datetime), origin airport, dest airport, carrier (airline code), dep_delay (departure delay in minutes), and distance. Columns like year, day, scheduled times, etc., are dropped since time_hour contains that info. Missing delay values are set to 0 (assuming on-time for missing data). It then converts time_hour to a proper datetime. The cleaned flight data is shown with key columns and a few entries (e.g., time, origin, dest, carrier, dep_delay, distance) [Cleaned flight data sample]. ChatGPT suggests next merging this flight data with the JFK weather data on matching timestamps and airport (using origin to match the weather station, in this case presumably JFK's weather for JFK flights, etc.).

**User:** Agrees and asks ChatGPT to merge the weather and flight datasets.

**ChatGPT:** Merges the two datasets. It joins each flight record with the corresponding weather conditions based on the date/hour (time_hour) and the airport origin. It then presents a snippet of the **merged dataset** [Merged flight+weather data sample]. Each entry now includes flight information (origin, destination, carrier, delay, distance) along with the weather features (temperature, dew point, wind speed/direction, visibility, precipitation) at the time of departure. ChatGPT confirms the merge was successful and suggests next steps. It asks if the user wants to handle any remaining missing values in this merged data, move on to feature selection/model training, or perform some exploratory data analysis (EDA) to uncover patterns.

**User:** Requests that ChatGPT handle any missing values in the merged dataset and explain how it's done.

**ChatGPT:** Cleans the merged dataset further by addressing remaining missing values. Specifically, it:

- Fills missing precipitation entries with 0 (assuming no precipitation when not recorded),
- Fills missing visibility with the median visibility value,
- Fills missing wind speed and wind direction with their median values,
- Drops any rows that still have missing critical fields (like if a timestamp or origin was missing, though that is unlikely after the merge).

After these steps, ChatGPT shows a brief "Final Cleaned Merged Data" output (with no missing values) to indicate the dataset is ready. It explains each step (e.g., replacing missing precip with 0, etc.) for transparency. The data is now fully cleaned for modeling. ChatGPT then asks how the user would like to proceed: it offers to perform Exploratory Data Analysis (EDA) to find patterns in delays, to move on to feature engineering, or to begin training a prediction model.

**User:** Asks ChatGPT to perform exploratory data analysis on the merged data.

**ChatGPT:** Conducts an Exploratory Data Analysis and summarizes several findings:

- **Distribution of Flight Delays:** Most flights are only slightly delayed or depart on time (some even early). Only a smaller fraction experience very long delays. This indicates a right-skewed delay distribution (many small delays, few large ones).
- **Average Delay by Origin Airport:** There are differences between airports. In general, JFK flights have higher average delays compared to Newark (EWR) and LaGuardia (LGA), which suggests JFK might experience more congestion or weather impacts.
- **Effect of Precipitation on Delays:** When precipitation increases, delays tend to become more likely or longer. However, the relationship isn't absolute—flights can still be on time in light rain, but heavy rain or snow is often associated with significant delays.
- **Flight Delay Trends by Hour of Day:** Flights leaving in the early morning (around 5–8 AM) tend to have minimal delays. Delays grow worse later in the day, peaking in the late

afternoon to early evening (approximately 5–7 PM), likely due to cumulative air traffic and weather disruptions as the day progresses.

(ChatGPT had attempted to include charts for these, but only textual summaries are provided here.) After discussing these patterns, ChatGPT asks the user if they'd like to move on to feature engineering (creating modeling features) or go straight into training a machine learning model for prediction.

**User:** Instructs ChatGPT to proceed with feature engineering for the model.

**ChatGPT:** Performs feature engineering on the dataset to prepare for modeling. It makes several modifications:

- **Delay Categorization:** Converts the numerical departure delay (dep_delay) into categorical labels (for classification). For example, it creates bins such as: on-time or early (delay ≤ 0 minutes), minor delay (1–15 minutes), moderate delay (16–60 minutes), and severe delay (> 60 minutes). Each flight is assigned a category based on its delay. (This will be useful if predicting categories instead of exact minutes.)

- **Time Features:** Extracts new features from the timestamp, such as the month of the year and the day of the week for each flight. These can capture seasonal effects or weekly patterns in delays.

- **Encode Categorical Variables:** Applies one-hot encoding to categorical features like origin airport and carrier (airline code). This creates new binary columns (e.g., a column for each carrier indicating 1 if the flight is by that carrier, else 0) so that the model can use carrier and airport information.

- **Normalize Continuous Variables:** Scales numeric features (especially the weather-related features and flight distance) to a common range (for instance, using Min-Max normalization to scale values between 0 and 1). This step ensures that features like temperature and distance, which have different units and scales, won't unduly dominate the model training just because of their magnitude.

After these steps, the dataset is transformed and ready for modeling with all features numeric and scaled. ChatGPT confirms that the data is now suitable for training a machine learning model. It then asks what the user would like to do next: train a model to predict flight delays, perform feature selection to see which features are most important, etc.

**User:** Asks for the Python code to train the machine learning model (in Visual Studio Code) and guidance on how to implement it.

**ChatGPT:** Provides a detailed Python script and instructions for training a flight delay prediction model. It explains each part of the implementation:

- **Library Installation:** It notes that the user should have certain libraries installed (pandas for data, scikit-learn for modeling, matplotlib/seaborn for any plotting). For example: pip install pandas scikit-learn matplotlib seaborn.

- **Data Preparation:** The script expects the cleaned and merged dataset to be saved as something like merged_data.csv in the working directory. It will load this data into a DataFrame.

- **Feature and Target Definition:** In code, it drops the columns that won't be used as features (e.g., the time stamp, destination, raw delay minutes, and any categorical target column if created). It uses the engineered features as X and uses the delay category as the target y (since the approach at this stage is classification of delay category).

- **Train-Test Split:** Splits the data into a training set and testing set (for example, 80% training, 20% testing) to evaluate model performance on unseen data.
- **Model Training:** Initializes a machine learning model; ChatGPT chooses a Random Forest classifier for this task (as it's a robust model for classification). It fits the model on the training data (X_train, y_train).
- **Model Evaluation:** After training, the script makes predictions on the test set. It then computes the accuracy and also prints a detailed classification report (precision, recall, F1-score for each delay category). It also mentions generating a confusion matrix to visualize how the predictions fall into each category vs. actual outcomes (though in practice it might print or plot it).
- **Usage Guidance:** ChatGPT describes how to run this script in Visual Studio Code's terminal (e.g., by running python flight_delay_prediction.py). It also mentions that the script can be used to predict new flights: for example, by creating a new_flight dictionary with current weather and flight info and calling the predict_flight_delay(new_flight) function (which was defined in the script to output a predicted category). The assistant emphasizes inserting the real-time weather data (from the API) into that new_flight dictionary for live predictions.

*(The actual code is lengthy, so it is summarized here rather than shown in full. It essentially loads the data, trains a RandomForestClassifier, and prints out performance metrics along with instructions to run and use the model.)*

**User:** Runs the provided training script and reports an error. They share a traceback indicating a ValueError about NaN values: "Input X contains NaN. RandomForestClassifier does not accept missing values encoded as NaN…". This error occurred when trying to fit the model.

**ChatGPT:** Explains that the error means there are still missing values in the feature matrix X, which the Random Forest model cannot handle. This likely happened because not all NaNs were eliminated during preprocessing. To fix this, ChatGPT suggests ensuring all missing values are handled before training. It recommends adding code to check for NaNs and then fill or drop them:

- **Identify NaNs:** Print out X.isnull().sum() to see which features have missing values.
- **Fill or Drop:** If only a few values are missing per column, fill them (impute) with a reasonable statistic like the median of that column. ChatGPT provides a snippet: X.fillna(X.median(), inplace=True) to replace any NaNs with median values for each feature. It also suggests an alternative: drop any rows with missing values (X.dropna(inplace=True)) and make sure to drop the corresponding entries in y as well, then re-split the data.

ChatGPT updates the training instructions to incorporate this fix. For example, it shows code printing the count of missing values before and after filling, then proceeding with model training once X has no NaNs. It explains that after this change, the Random Forest should be able to train without issues. The assistant encourages the user to re-run the script after adding these steps to ensure the error is resolved. [The user will implement the fix to fill missing values with median or drop them.]

**User:** After fixing missing values, the user encounters another issue. They report a KeyError when trying to use the predict_flight_delay() function. The error message lists columns like 'Unnamed: 0', 'hour', 'origin_LGA', 'carrier_AS', ... not in index'. This implies that the prediction function tried to access features that are not present in the input.

**ChatGPT:** Diagnoses the KeyError as a mismatch between the features the model was trained on and the keys in the new_flight input dictionary. Specifically, the error mentions Unnamed: 0 (which is likely an index column) and several one-hot encoded carrier columns that the model expects but were not provided in new_flight.

To fix this, ChatGPT suggests ensuring two things:

1. The training feature set excludes any index or unnecessary columns like Unnamed: 0 so the model doesn't expect them. (Probably the user's data still had an index column when read from CSV.)
2. The predict_flight_delay function should add any missing feature columns to the input data frame with default values (0) before running the prediction, which the user had partially implemented.

ChatGPT provides a revised version of the predict_flight_delay function for clarity, highlighting that it already does: create a DataFrame from the input dict, add any missing columns with df[col] = 0 for each expected col not in the input, then reorder columns to match X's columns. It ensures that 'Unnamed: 0' and any absent carrier dummy columns will be filled with 0 if not present in new_flight.

The assistant also notes the importance of dropping Unnamed: 0 during training. It shows how the code defining X could explicitly drop "Unnamed: 0" so that it isn't considered a feature at all.

Finally, ChatGPT gives an example of how to call predict_flight_delay with a complete feature set. For instance, if origin and carrier were one-hot encoded, the new_flight dictionary should include at least one of those (or they'll be filled as 0). It demonstrates an example new_flight including fields like "origin_JFK": 1 and "carrier_AA": 1 (with other carriers implicitly 0) to represent, say, a JFK flight on airline AA.

With these adjustments, the KeyError should be resolved. The predict function will no longer look for columns that aren't there because Unnamed: 0 will be dropped from X and any missing one-hot columns will be added automatically in the input. The assistant instructs the user to replace their predict function with this robust version and to ensure the model was trained without the unwanted columns. This will allow them to get a prediction without errors.

**User:** Asks for a suggestion of an API to get real-time weather data for JFK and how to integrate it into their code.

**ChatGPT:** Suggests using the OpenWeather API for real-time weather information at JFK Airport. It provides a step-by-step guide to integrate it:

- **Get an API Key:** The user should sign up on OpenWeather and obtain an API key (free tier should suffice for current weather data).
- **Use the Requests Library:** In Python, import requests and construct a query URL for current weather by specifying JFK's latitude and longitude (40.6413 N, -73.7781 W) along with the API key and desired units (e.g., imperial for Fahrenheit). The endpoint will look like: http://api.openweathermap.org/data/2.5/weather?lat=40.6413&lon=-73.7781&appid=YOUR_API_KEY&units=imperial.
- **Parse the Response:** Upon calling requests.get(url), check if the response status is 200 (OK). Then parse the JSON to extract relevant fields. ChatGPT outlines the fields to extract:
    - Temperature (which it maps to the model's HOURLYDRYBULBTEMPF feature) from data["main"]["temp"].
    - Dew point or "feels like" temperature as a proxy (HOURLYDewPointTempF) from data["main"].get("feels_like", 0).

- Wind speed (HOURLYWindSpeed) from data["wind"]["speed"] (OpenWeather gives this in m/s or mph depending on units; with units=imperial it's in miles/hour, but ChatGPT might convert to knots if needed — it mentioned dividing by 1.151 to convert mph to knots in code).
- Wind direction (HOURLYWindDirection) from data["wind"]["deg"].
- Visibility (HOURLYVISIBILITY) from data.get("visibility"), converting meters to miles (OpenWeather provides visibility in meters; dividing by 1609 gives miles).
- Precipitation (HOURLYPrecip) from data.get("rain", {}).get("1h", 0) which would give rainfall in the last hour in mm (0 if not raining).

- **Integrate Data:** Package these extracted values into a dictionary named weather_data with keys matching the model's features. Return this dictionary.

ChatGPT then shows how to integrate this function into the user's script. For example, after training the model, they can call real_time_weather = get_real_time_weather(), and if it returns data successfully, combine it with time features (month, day_of_week, hour) to form a new_flight dict. Then call predict_flight_delay(new_flight) to get a prediction. All of this is demonstrated in the code snippet it provided.

*(The answer includes the structure of the code for the API call and how to merge its output into the model's input. The actual API key is left as a placeholder in the example.)*

**User:** Encounters a SyntaxError when attempting to run the Python script in the terminal. The error message suggests an issue with a command, showing something like an & in the command usage.

**ChatGPT:** Explains that the SyntaxError likely resulted from trying to run the script in the wrong context. It looks like the user might have pasted a command (with an & character, possibly from copying a PowerShell command) directly into the Python interactive shell, which is not correct.

ChatGPT clarifies how to properly run the script:

- In Windows Command Prompt or PowerShell: Navigate to the directory containing the script, then run python "c:/path/to/Delay_Prediction_ML.py". The & is not needed in this context; it might have been mistakenly included if the user ran it from within a Python REPL or used a wrong command format.
- If using PowerShell, the & operator is used to run commands but if you already call python, you don't need a leading &. So just execute the command as above without leading &.
- If using a Bash terminal (like Git Bash on Windows or a Unix system), the command would be python3 "path/to/Delay_Prediction_ML.py" if Python 3 is invoked via python3. But in any case, the entire path should be in quotes because of spaces in the path ("OneDrive - Coventry University/…/Thesis").
- It also suggests if the script name or path has spaces or special characters, ensure they are properly quoted. Alternatively, rename the script to a simpler name (e.g., delay_prediction.py) to avoid confusion.
- As a general check, ChatGPT says to verify Python is installed and accessible (for instance, running python --version in the terminal to ensure the command works).

Basically, the assistant is guiding the user to run the script as a system command, not inside the Python interpreter, and to make sure the syntax is correct for their environment. It concludes with next steps: try running the corrected command, and if there are still issues, provide details about the environment (cmd vs PowerShell vs Bash) and any error output for further help.

**User:** After running the model, finds that no matter what input they use for new_flight, the model always predicts delay category 2. They ask how they can change the input values to get a different prediction (for example, a category other than 2).

**ChatGPT:** Analyzes why the model might be predicting the same category each time and how to influence it. Since category "2" corresponds to a moderate delay in the scheme, ChatGPT explains that to change the output, one must significantly alter the input features that affect the prediction. It provides guidance based on feature influence:

- **Factors Increasing Delay:** High precipitation, low visibility, strong or variable winds, and perhaps winter conditions likely push the model towards predicting longer delays (higher category).
- **Factors Decreasing Delay:** Clear weather (no precipitation, good visibility), calm winds, and maybe times with less congestion should push towards on-time (category 0) predictions.

It then gives concrete examples of new_flight inputs:

1. **For a Severe Delay (Category 3)**: Use extreme adverse conditions. e.g., "distance": 0.9 (assuming scaled distance, a long flight), very low temperature and dew point (representing potentially extreme cold or storms), "HOURLYWindSpeed": 1.0 (very high winds), "HOURLYWindDirection": 1.0 (turbulent wind direction changes), "HOURLYVISIBILITY": 0.0001 (near-zero visibility, e.g., heavy fog or storm), "HOURLYPrecip": 1.0 (heavy precipitation), and choose a winter month and a busy day/time. This combination, ChatGPT notes, should make the model predict a severe delay (category 3) because it represents a worst-case scenario for weather and possibly traffic.
2. **For an On-time/Early Flight (Category 0)**: Use ideal conditions. e.g., "distance": 0.1 (short flight), "HOURLYDRYBULBTEMPF": 0.7 and "HOURLYDewPointTempF": 0.7 (mild, pleasant weather), "HOURLYWindSpeed": 0.1 (little wind), "HOURLYWindDirection": 0.5 (no significant wind change issues), "HOURLYVISIBILITY": 1.0 (clear skies), "HOURLYPrecip": 0.0 (no precipitation), and perhaps a summer month on a Monday (less busy than weekend). These inputs mimic perfect conditions, which should lead the model to predict no delay or only a minimal delay (category 0 or 1).

ChatGPT states that under extremely bad weather, the model is more likely to predict a severe delay, and under ideal weather, it's more likely to predict on-time. If inputs are moderate (like the original examples), the model was predicting category 2, so the key is to push the inputs to more extreme ends. It encourages the user to try these adjusted values to see different outputs. (It essentially provided two example dictionaries for new_flight: one with extreme bad weather and one with extremely good weather, instead of the moderate example that always yielded 2.)

**User:** Tried both the extreme scenarios (the severe-delay scenario and the on-time scenario), but reports that the model still predicts category 2 for both. They are frustrated that the output isn't changing.

**ChatGPT:** This result indicates a deeper issue with the model. ChatGPT outlines a few possible reasons why the model always outputs category 2 regardless of input:

- **Model Bias (Class Imbalance):** It suspects that category 2 might be overwhelmingly common in the training data, so the model learned to predict 2 for almost everything (it's basically biased towards the majority class). If the dataset wasn't balanced and most flights fell into delay category 2, the model would default to 2 to achieve higher accuracy, ignoring other classes.
- **Feature Scaling or Input Range Issues:** The user's new_flight examples used scaled values between 0 and 1 (as per normalization). If by chance the model was not using

scaled features (or if the input scaling is inconsistent), the model might not be interpreting those extreme 0-1 values as truly extreme real-world values. For example, if scaling was off, the model might still "see" those inputs as moderate. It suggests verifying that the inputs are scaled the same way the training data was.

- **Low Sensitivity to Weather Features:** From the earlier feature importance analysis, it's possible the model relies mostly on features like flight distance or time of day (and maybe those haven't changed much in the examples) rather than the weather inputs. If "distance" or other factors dominate the splits in the Random Forest, then even huge changes in precipitation might not flip the prediction. Essentially, the model might not be very complex or sensitive in the way we expect.

ChatGPT then recommends a few fixes and debugging steps:

1. **Check Class Distribution:** It suggests printing the distribution of the delay categories in the training data (for example, the percentage of flights in each category) to see if category 2 indeed dominates. It provides a quick code snippet using value_counts(normalize=True) on the delay category. If category 2 comprises a large portion of the data (say > 50-60%), that confirms an imbalance issue. In that case, the solution is to balance the dataset. It reiterates the idea of using RandomOverSampler (from imblearn) to oversample the minority classes so that each category has equal representation in the training set. It shows code to apply RandomOverSampler to X and y, then re-split and retrain the RandomForestClassifier on this balanced data. This would force the model to pay equal attention to categories 0, 1, 2, 3 instead of always predicting 2.

2. **Verify Feature Scaling:** It reminds the user that if they normalized features (with MinMaxScaler earlier), they must also normalize any manual input in the same way. The extreme examples given were already in 0-1 scale assuming that's what the model saw. But ChatGPT wants to double-check that in the code, the model was indeed trained on a scaled X. If not, perhaps the model was trained on raw values (which might be large numbers), and here the user is feeding 0-1 thinking they are extreme, but the model interprets 0-1 as very low values (since it expected actual measurements). Conversely, if the model was scaled and the user forgot to scale the new input, that could also cause a problem. So the assistant explicitly instructs: ensure the same scaler used on training data is applied to new_flight input. It even provides example code: recreate or use the fitted scaler to transform the new_flight DataFrame before calling model.predict. This will align the input's scale with what the model expects.

3. **Increase Model Complexity or Sensitivity:** The assistant revisits model parameters. The Random Forest might not be picking up subtle differences. It suggests making the Random Forest more sensitive by increasing the number of trees (n_estimators), allowing deeper trees (max_depth) and setting class_weight="balanced" within the classifier to automatically give more weight to underrepresented classes. It provides an example of initializing the RandomForestClassifier with n_estimators=200, max_depth=12, min_samples_split=5, and class_weight='balanced'. This combination means the model will try harder to not always predict the majority class and might find more intricate patterns. The class_weight='balanced' especially will penalize misclassifications of minority classes more, hopefully encouraging the model to sometimes predict 0, 1, or 3 when appropriate.

ChatGPT combines these into a set of next steps: balance the classes and retrain, use proper scaling for any new input, and tweak the model parameters for better performance. It encourages the user to implement these adjustments and test again, hopefully to finally see varied predictions for different scenarios.

**User:** Decides to change strategy: instead of classifying delays into categories, they ask to build a model that predicts the exact number of minutes of delay.

**ChatGPT:** Agrees and describes how to modify the approach for a regression task (predicting a continuous value rather than a category). It outlines the steps to convert the classifier into a regressor:

- **Target Change:** Use the actual delay minutes (dep_delay) as the target y for training, instead of the categorized delay label. That means the model will output a number (delay minutes).
- **Model Choice:** Use a regression model. ChatGPT proposes using RandomForestRegressor (from scikit-learn) which is analogous to the classifier but for regression.
- **Feature Set:** Many of the features remain useful (weather, time features, etc.). However, any columns specifically related to the category classification (like the delay_category itself) should be removed. Also, one-hot encoded features can still be used. If the user had dropped some features earlier (like carriers or Unnamed: 0), those should stay dropped if they were deemed not useful.
- **Scaling:** It's often still beneficial to scale features for regression, especially if using distance and weather features which have different units. ChatGPT suggests applying MinMaxScaler to the feature matrix so that all inputs are between 0 and 1 for training, which can help the model converge a bit better (though Random Forest is not sensitive to scaling, but scaling might help with interpretation or other algorithms).
- **Train/Test Split:** Same as before, split the data into training and testing subsets.
- **Training:** Train the RandomForestRegressor on the training data.
- **Evaluation:** Calculate regression metrics on the test set. ChatGPT specifically points out:
  - **Mean Absolute Error (MAE):** average absolute difference between predicted and actual delays.
  - **Mean Squared Error (MSE):** average squared difference (penalizes large errors more).
  - **$R^2$ Score:** how much variance in the delay is explained by the model (with 1.0 being perfect and 0 meaning it does no better than predicting the mean).

It provides code illustrating these steps. For example:

- Drops unnecessary columns (it even shows dropping delay_category and also the index Unnamed: 0 if present, and possibly the one-hot origin and carrier columns if they decide to drop them to reduce dimensionality).
- Sets X and y = data["dep_delay"].
- Scales X with MinMaxScaler, then splits into train/test.
- Initializes RandomForestRegressor (it even uses some tuned parameters like n_estimators=200, max_depth=12 for better performance) and fits it.
- Predicts on the test set to get y_pred.
- Calculates MAE, MSE, $R^2$ and prints them out.
- Additionally, it produces a scatter plot of actual vs predicted delay minutes to visually assess how well the predictions line up (points along the diagonal line would indicate accurate predictions).

- **Updated predict_flight_delay function:** ChatGPT updates this utility to output a numeric prediction. The function will take a new_flight input dict, create a DataFrame, ensure all the required columns are present (adding any missing with 0, similar to before), then scale this input using the same scaler fitted on training data (df_scaled = scaler.transform(df)). It then does model.predict(df_scaled) which will return an array with one value (predicted delay in minutes), and it returns that single value.

- **Example usage:** It shows how to call this new predict_flight_delay function. For instance, after obtaining real-time weather via the API, and computing current month, day_of_week, hour, the user can do:

  new_flight = { **real_time_weather, "month": current_month, "day_of_week": current_day_of_week, "hour": current_hour }

  predicted_delay = predict_flight_delay(new_flight)

  print(f"Predicted Flight Delay: {predicted_delay:.2f} minutes")

- This will print the predicted delay in minutes (formatted to 2 decimal places).

ChatGPT emphasizes the key improvements in this new approach: instead of classifying into broad categories, the model will give a specific number of minutes for the delay, using RandomForestRegressor to handle continuous output. It includes the regression metrics to help the user evaluate the model's accuracy in minutes. The assistant then says to update the script accordingly and test with real inputs, and that they can further tune the model if needed.

**User:** Obtains the results of the regression model and asks for an explanation of the output metrics. They specifically mention the model performance values: **MAE = 20.3765**, **MSE = 1491.9499**, **$R^2$ = 0.1389**.

**ChatGPT:** Interprets these regression metrics in plain language:

- **Mean Absolute Error (~20.38 minutes):** On average, the model's predictions are about 20.38 minutes off from the actual delay. For example, if a flight was actually delayed 50 minutes, the model might predict around 30 or 70 minutes (20-minute error either way on average).

- **Mean Squared Error (~1491.95):** This is the average of squared errors. Because errors are squared, larger errors weigh more heavily. A high MSE like 1491 suggests that there are some instances where the model's error is quite large (since 1491's square root is about 38.6, that gives a sense of typical error magnitude in minutes). In other words, while many predictions might be off by ~20 minutes, some might be off by more (e.g., 40-50+ minutes), which drives the MSE up.

- **$R^2$ Score (~0.139):** This means the model only explains about 13.89% of the variance in the delay times. An $R^2$ of 1 would mean perfect prediction, and 0 would mean the model is no better than always guessing the average delay. An $R^2$ of 0.139 is quite low, indicating the model isn't capturing most factors that determine delays. Roughly 86% of the variability in delays is due to factors not included or not learned by the model.

ChatGPT then explains what these mean for the user's scenario: The model's predictions have a lot of uncertainty (±20 minutes error on average is quite high when you consider many flights might not be delayed at all, for example). The low $R^2$ implies the model has limited predictive power — many other variables (like airline operations issues, air traffic control, etc., which were not in the data) likely affect delays.

It clarifies that $R^2$ is not a probability of delay but rather a measure of fit: an $R^2$ of ~0.14 does *not* mean "14% chance of the flight being delayed by the predicted amount," but simply that the model accounts for 14% of the variability in delay duration.

ChatGPT suggests that to improve these metrics, the user should consider:

- Adding more features (e.g., indicators of airport congestion, whether the flight's inbound aircraft arrived late, etc.). Weather alone might not be enough to predict delays accurately.
- Tuning the model parameters or trying more advanced models (it already hinted at XGBoost or similar which can sometimes capture nonlinear patterns better).
- Ensuring that any important known factors (like distance, time of day) are properly utilized, and perhaps performing feature selection to see if some features are just adding noise.

It concludes by summarizing that currently the model isn't very accurate (with examples: "only ~14% of delay variation explained"), and that more data or better modeling is needed to get more precise predictions.

**User:** Asks a question to clarify $R^2$: They say, if the model predicts a 28-minute delay and $R^2$ is ~0.1447, does that mean there's a 14% chance their flight will be delayed by 28 minutes?

**ChatGPT:** Clears up the confusion: *No, that is not the correct interpretation.* It explains:

- The $R^2$ score of 0.1447 (~14.47%) indicates model fit, not probability. It means the model explains about 14.47% of the variation in delays in the dataset. It does not translate to a probability of a specific outcome.
- The predicted 28-minute delay is the model's best guess given the inputs, but it doesn't come with a probability. $R^2$ definitely isn't the probability of that exact delay happening.

ChatGPT elaborates by breaking it down:

- **What $R^2$ = 0.1447 means:** The model isn't capturing about 85% of the factors that affect delay. So it's not very reliable. Many things that cause delays aren't in the data (e.g., air traffic, mechanical issues, crew availability).
- **What a "28-minute prediction" means:** The model, considering current weather and time, output 28 minutes. This is an estimate, not a guarantee.
- **Does 14% chance of 28 minutes delay make sense?** No. Instead, one should say something like: "Based on the weather and time inputs, the model predicts a 28-minute delay. However, given the model's low $R^2$ (~14%), this prediction is very uncertain, and actual delays could differ widely."

ChatGPT might even provide a correct interpretation example: *"There is a predicted delay of 28 minutes, but since our model only explains ~14% of delay variability, there's a lot of uncertainty. In reality, the flight could end up on time or have a much longer delay, because the model doesn't account for many other factors."*

It then suggests how to improve prediction accuracy if we want more confidence: incorporate more features such as real-time flight traffic, historical delays of that flight route, etc., which it mentioned earlier. The main takeaway it gives: $R^2$ is about model explanatory power, not a direct probability of an event, and currently the model is not very powerful.

**User:** Shares a new error from their script. The error shows a DtypeWarning about certain columns having mixed types and then a KeyError when trying to drop some columns (specifically 'time_hour', 'dest', 'dep_delay', 'delay_category', 'Unnamed: 0', 'origin_LGA', 'origin_JFK', 'distance' not found in axis). They have a long traceback indicating an issue around dropping columns in their code.

**ChatGPT:** Addresses both parts of this issue:

- **DtypeWarning (mixed types):** This warning is from pandas when reading a CSV that has, for example, some columns with both numbers and strings. It suggests that some columns (like perhaps flight numbers or others) weren't consistently typed. ChatGPT advises how to handle it:
    - When reading the CSV, one can specify dtype=str for all columns or use low_memory=False to suppress the warning and let pandas read in chunks. It even shows pd.read_csv("merged_data.csv", dtype=str, low_memory=False) to force all data to be read as strings initially.
    - After reading as strings, convert the relevant columns to numeric types explicitly. It lists the columns that should be numeric, such as "dep_delay" and the various "HOURLY..." weather columns. It then loops through these and uses pd.to_numeric(data[col], errors='coerce') to convert them; errors='coerce' will turn any non-numeric values or problematic entries into NaN, which can then be handled (e.g., by filling with 0 or median as done before).
    - This approach ensures that the data is loaded without type conflicts and that any unexpected values are handled in a controlled way.
- **KeyError during drop:** The error message suggests the code attempted to drop columns that didn't exist in the DataFrame. Specifically, after reading the data, maybe some column names changed or were already dropped. For instance, time_hour or dep_delay might not be present if the user already removed them or if the CSV didn't include them (maybe the CSV was saved after dropping these columns). Also, one-hot columns like 'origin_LGA' might not exist if only one origin airport's data was saved.

ChatGPT's solution: check which columns actually exist before dropping. It suggests printing data.columns.tolist() to see the list of column names. Then adjust the drop list accordingly or use a dynamic approach:

- For example, build the list of columns to drop, then filter it: columns_to_drop = [col for col in columns_to_drop if col in data.columns]. This way, only columns present will be dropped.
- Alternatively, use data.drop(..., errors='ignore') to simply ignore any columns that aren't found instead of raising an error.
- The assistant reiterates the need to drop Unnamed: 0 if present (since it's just an index from the CSV), and any other unwanted columns, but to do so carefully.

ChatGPT then walks through an updated sequence:

1. Read CSV with safe dtypes.
2. Convert numeric columns properly.
3. Use code to only drop columns that exist (and maybe print the remaining columns to verify).

It likely provides code snippets for each step. For example:

```
# After reading data

print("Available Columns:", data.columns.tolist())


# Drop unwanted columns if they exist
```

```
columns_to_drop = ["time_hour", "dest", "dep_delay", "delay_category",
"Unnamed: 0", "origin_LGA", "origin_JFK", "distance"]

columns_to_drop = [col for col in columns_to_drop if col in data.columns]

X = data.drop(columns=columns_to_drop, errors='ignore')
```

This would handle the KeyError by ensuring only valid columns are dropped.

It also covers the rationale: maybe 'dep_delay' or certain origin dummies weren't in the data because of prior processing or because the user saved a modified CSV. The user needs to align the code with the actual data content.

By following these suggestions, the DtypeWarning can be resolved (by uniform typing) and the KeyError can be avoided (by checking columns). ChatGPT tells the user to update their script accordingly and try again.

**User:** Implements the above but then says, when they try to convert columns to numeric, it fails because 'dep_delay' is not found (KeyError). They mention it "does not want to print" presumably because the code crashed at data[col] = pd.to_numeric(...) for 'dep_delay'.

**ChatGPT:** Explains why 'dep_delay' might be missing. It hypothesizes that maybe 'dep_delay' was dropped earlier in the code or wasn't included in the CSV file the user is loading (for example, if the user's merged CSV only contained the categorized delay, not the raw delay, or if they removed it as part of cleaning).

It advises to verify the columns in the data again: maybe the user saved a version of the data where dep_delay was not present (since they moved to using delay categories or something).

To avoid the KeyError in conversion, ChatGPT tells them to only attempt to convert 'dep_delay' if it's actually in data.columns. It modifies the numeric conversion loop to check for the column's existence:

```
for col in num_cols:

if col in data.columns:

data[col] = pd.to_numeric(data[col], errors='coerce')
```

This way, if 'dep_delay' isn't in the DataFrame, it simply skips it and no error is thrown.

It also suggests double-checking the source of truth: open the CSV in a text editor or use print(data.head()) to see if 'dep_delay' is a column. Possibly the column got renamed or dropped. For instance, sometimes merging or saving could result in 'dep_delay' being omitted especially if only delay_category was saved.

Additionally, ChatGPT says if 'dep_delay' was unintentionally removed in preprocessing, the user might need to re-add it or ensure the CSV was generated correctly. But assuming the goal is now regression, 'dep_delay' should be present. So it stresses verifying the contents of merged_data.csv.

Finally, ChatGPT summarizes: the user should update the code to be robust to missing columns (like skipping conversion for missing ones), and ensure that if dep_delay is needed as target, it is indeed in the data. If it's not, the user might need to recreate the merged CSV including dep_delay.

It then instructs to run the script again after these checks.

**User:** Shares their current code and asks for an analysis of what is wrong. The code includes data loading, creation of carrier_columns to drop all carrier dummies, dropping several columns (time_hour, dest, dep_delay, delay_category, Unnamed: 0, origin_LGA, origin_JFK, distance, plus all carrier columns), defining X and y (y = data["dep_delay"]), scaling, train-test split, training an XGBRegressor, evaluating performance, plotting feature importance, and a predict_flight_delay function (similar to what was developed).

**ChatGPT:** Reviews the provided code and points out potential issues and logical problems:

1. **Dropping dep_delay then using it as y:** The code shows X = data.drop(columns=[... "dep_delay", ...] + carrier_columns) and then y = data["dep_delay"]. If this code executes, it might not error out immediately (because data["dep_delay"] might still work if drop was applied to a copy and not inplace), but logically it's confusing. Likely, X is a new DataFrame with dep_delay removed, but y is taken from the original data. If the data was read fresh, this is okay. But if the user did data.drop(...) inplace earlier, dep_delay might be gone. ChatGPT points this out: make sure dep_delay exists at the time of defining y. If the user had already dropped it, that's a problem. The fix is to drop dep_delay from X (features) but not from the original DataFrame until after y is extracted. In pandas, doing X = data.drop(columns=[...]) does not remove those columns from data itself (unless inplace=True was used), it returns a new object X. So it might be fine as is. But ChatGPT wants the user to double-check that logic to avoid KeyError and conceptual inconsistency. It suggests ensuring that the column is present when accessing it. (It may even wrap it in a check or raise an error with a clear message if not found, to avoid silent issues.)
2. **Data Type Conversion:** It asks if the code converted columns to numeric after reading. In the snippet the user gave, there was no explicit conversion or filling of NaNs (maybe the user integrated that off-screen or earlier). If not, then the model might be training on strings or objects which XGBoost might handle internally or throw an error. ChatGPT re-emphasizes ensuring that numeric columns (dep_delay, weather features, etc.) are actually numeric types. If not, the model's results would be meaningless. So it likely reminds the user to incorporate the earlier conversion step.
3. **predict_flight_delay completeness:** The function as shown creates a DataFrame from input and adds missing columns with 0. ChatGPT checks if anything might be off. It sees that after dropping all carriers from training (the user wanted to drop all carrier variables), the carrier_columns list was used in drop, so presumably X no longer has any "carrier_*" columns. However, in the predict_flight_delay example in the code snippet, it still shows an example new_flight with "carrier_AA":1, "carrier_DL":0, "carrier_B6":0. If all carriers were dropped from X, then the presence of these in new_flight would actually cause them to be considered extra features not used by the model. Conversely, if the user hasn't updated the example, it might be fine but unnecessary. ChatGPT might flag this: if carriers are dropped, ensure the predict function and any example input doesn't include them (or if it does, it will just ignore them since they won't be in X.columns after the drop). It basically wants consistency: dropping all carriers means the model doesn't use airline info at all, focusing only on weather/time.
4. **Feature Importance for XGBoost:** The code tries model.feature_importances_. ChatGPT knows that XGBoost's sklearn API's feature_importances_ property might work (it usually does, representing gain or weight importance), but it can sometimes be 0 or not as expected if the model wasn't fitted or if using certain booster parameters. If feature_importances_ is not populated, it could confuse the user. ChatGPT suggests a more robust way: use model.get_booster().get_score() to get feature importance from XGBoost's perspective. It even provided earlier a snippet for that. Here it reiterates: since the code includes both RandomForest and XGBRegressor usage, one can write a check like:

```
if hasattr(model, "get_booster"):
    # XGBoost importance
    importance_dict =
model.get_booster().get_score(importance_type="weight")
```

```
    importance_df = pd.DataFrame(list(importance_dict.items()),
columns=["Feature", "Importance"])

else:

    # RandomForest importance

    importance_df = pd.DataFrame({"Feature": X.columns, "Importance":
model.feature_importances_})
```

This ensures that no matter which model is used, we get a meaningful feature importance. ChatGPT provides this as an improvement so that the feature importance analysis part of the code works for XGBoost.

5. **Scaling of New Input:** The code snippet shows they scaled X with MinMaxScaler and stored X_scaled. However, in the predict_flight_delay function, they use df_scaled = scaler.transform(df), meaning they kept the scaler object around. ChatGPT likely double-checks that the scaler was fitted on the original X (the code shows scaler.fit_transform(X) earlier), so that's good. It warns to ensure that this scaler is the same one (maybe it's defined in an outer scope, likely yes in the snippet). So that part looks fine as long as the script is structured accordingly. If anything, ChatGPT might remind the user to use the same scaling on input. But it sees that it is done.

After listing these, ChatGPT provides specific fixes or confirmations for each:

- For dropping variables: "Ensure you drop unwanted columns when forming X, but still have dep_delay in your DataFrame for y. Dropping "dep_delay" from X is fine (we don't want to train on the target), just don't remove it from the DataFrame before assigning y."
- For data types: "Make sure to convert all numerical columns to numeric types (we discussed this earlier). If not done, do it before training."
- For predict function missing features: It reaffirms the approach of adding missing columns with 0, which is already implemented. It just says to align it with the current feature set. For example, since all carrier_ columns were dropped, the model no longer expects them, which simplifies prediction (no need to provide carrier info at all). The function as written will add any column that was in X.columns but not in input; if carriers are gone from X, it won't look for them. So likely fine.
- For feature importances with XGBoost: It explicitly suggests modifying that part of the code. It notes that XGBRegressor might not populate feature_importances_ in the same way, or if it does, it might use a default importance type that's not immediately clear. So using the booster's get_score method is more direct. It provides an updated code snippet for feature importance plotting that accounts for XGBoost.
- It also mentions scaling new data (point already covered by the function) and possibly the presence of the 'hour' feature: In the user's example usage snippet, they had "hour": 0.9 in new_flight, but earlier error indicated 'hour' wasn't found. Possibly they didn't include "hour" when training X (they extracted month and day_of_week but maybe not hour of day). If they did drop 'hour', then including it in new_flight would do nothing unless predict_flight_delay adds it as missing (which it might not if it's not in X.columns). ChatGPT might note: if the model doesn't use 'hour' as a feature, then adding "hour" in new_flight won't affect the prediction. Conversely, if they want to use departure hour as a feature, they should include it in X when training. So ensure consistency: either include 'hour' as a feature in training and prediction, or omit it entirely. The error 'hour' not in index earlier suggests they didn't include 'hour'. ChatGPT might suggest if they want to include hour of day, to add it as a feature similarly to month and day_of_week.

After this thorough analysis, ChatGPT summarizes the action items:

- Verify and possibly correct the dropping of columns (especially make sure you're not inadvertently removing needed ones like dep_delay before use).
- Convert data types to numeric and handle missing values (the user did some of this, just double-check).
- Use the dynamic drop of carrier columns (which they did with list comprehension).
- Fix the feature importance code for XGBoost vs RandomForest (so it works regardless of model).
- Ensure any new flight input is scaled and matches the feature set used by the model.

It ends by encouraging the user to implement these fixes and then test again to see if the issues are resolved, and to ask if further adjustments are needed.

## Appendix 2: Microsoft Copilot Conversation Transcript

**User:** Hello, I need assistance developing a machine learning model for predicting flight delays using historical flight and weather datasets. Can you help me with the full pipeline?

**Copilot:** Absolutely! Here's the recommended structure:

- Load and explore data
- Clean datasets
- Feature engineer key variables
- Merge datasets
- Build and evaluate the machine learning model
- (Optional) Connect real-time data sources for future predictions.

**User:** Great.
I will upload two datasets: jfk_weather.csv and flight_data.csv.
Please help me clean them and prepare for modeling.

**Copilot:** (Acknowledged datasets.)
Outlined that first step is loading the datasets using pandas.

(Provided initial Python code to load and preview the data.)

**User:** (Encountered DtypeWarning about mixed types.)

**Copilot:** Explained that this warning occurs when columns have mixed types (e.g., numbers and text).
Suggested:

- Adding low_memory=False to pd.read_csv()
- Optionally using dtype=str to read all columns as strings.

**User:** (Implemented suggestions but asked for more compact outputs to avoid flooding the terminal.)

**Copilot:** (Provided modified code using df.info() and df.head() to preview small parts of the datasets neatly.)

**User:** (Shared dataset previews.)
How should I clean these datasets?

**Copilot:** Recommended:

- Dropping irrelevant columns (e.g., "tailnum", "flight", "origin", "dest").
- Parsing datetime fields properly.
- Handling missing values:
    - Filling weather data gaps with medians.

- Dropping flights with missing delay/cancellation labels.

(Provided cleaning code for both datasets.)

**User:** (Ran into ValueError when parsing time_hour column.)

**Copilot:** Diagnosed likely mixed datetime formats.
Suggested using errors='coerce' when parsing to force invalid formats to NaT and then dropping bad rows.

(Provided updated datetime parsing code.)

**User:** (Fixed datetime issues.)
Please guide merging datasets.

**Copilot:** Recommended:

- Sorting both datasets by datetime field
- Using merge_asof() to perform nearest-time matching.

(Provided correct merge code.)

**User:** After merging, what next?

**Copilot:** Proceed to Feature Engineering:

- Extract features like hour, day_of_week, month
- Create binary delay flags
- Normalize continuous features
- Encode categorical variables using One-Hot Encoding.

(Provided code examples for feature engineering.)

**User:** Now let's train the classification model.

**Copilot:** Suggested Random Forest Classifier as the starting point.

(Provided Python code for:

- Data splitting
- Model training
- Prediction
- Evaluation using confusion matrix and classification report.)

**User:** (Encountered error about NaNs during model training.)

**Copilot:** Recommended:

- Filling missing values in training set with SimpleImputer
- Verifying no NaNs remain before fitting.

(Provided imputation code.)

**User:** After training, I want to connect real-time weather data.

**Copilot:** Proposed integrating an API like OpenWeatherMap.
Outlined steps:

- Pull real-time weather data
- Apply same feature engineering transformations
- Feed into trained model for live predictions.

**User:** Thank you. However, Copilot sometimes suggested wrong columns or wrong assumptions.

**Copilot:** (When prompted for clarification)
Apologized and attempted to regenerate corrected code.
Acknowledged that some assumptions were made based on limited sample previews.

**User:** (Noted that some suggestions led to repetitive loops and required manual intervention.)

**Copilot:** (Did not recognize mistakes without explicit user correction.)
Required manual restarting of the cleaning and modeling pipeline multiple times.

**User:** Nevertheless, thanks for the guidance.

**Copilot:** You're welcome! Feel free to reach out for help deploying models or optimizing hyperparameters.