Máster en Data Science & Business Analytics

Master's Final Project

Aircraft Valuation (SaaS)
with
Databricks & Apache Spark

Author: Julián Vázquez Sampedro
Supervisor: Abel Soriano Vázquez

- Madrid, November 2023 –

Abstract

In the dynamic and complex landscape of aviation industry accurate valuation of aircraft plays a pivotal role in decision making processes for stakeholders ranging from manufacturers to operators. Recognising the need for a streamlined and data-driven approach to aircraft valuation this project seeks to implement a machine learning model using Databricks and Apache Spark. The overarching goal is to establish an aircraft valuation software as a service that not only enhances the efficiency of the valuation process but also empowers industry's professionals with prices and timely insights.

In this business context the project aims to leverage historical data encompassing diverse attributes of aircraft, including make, model, year, total hours, and more. By harnessing the power of PySpark scalable and user-friendly machine learning libraries, the aim is to create a robust model capable of predicting aircraft prices with a high degree of accuracy this initiative aligns with the industry's growing demand for innovative solutions that foster informed decision-making enabling stakeholders to navigate the aviation market with confidence.

Through the integration of advanced machine learning techniques this project not only addresses the challenges associated with manual valuation but also positions itself as a catalyst for digital transformation within the aviation sector.

The seamless compatibility of Databricks with other integrated services further enhances the projects potential for scalability and adaptability. This case of study follows the important steps of a data science project cycle to develop a software that can predict an aircraft price using information through a dataset of nearby 2500 aircraft models, available at Kaggle.

Table of Contents

1. Introduction

The aviation space industry, a dynamic and ever evolving sector, stands at the forefront of technological innovation and level connectivity defined by the design development and production of aircraft and spacecraft. This industry has witnessed significant transformations in recent years driven viable technological advancements and shifting market demands. It encompasses vast array of activities ranging from the manufacturing of commercial and military aircraft to the exploration of outer space. Over the past years, several noteworthy trends have shaped the industry landscape. Increased emphasis on fuel efficiency and sustainability has driven the development of more aerodynamically efficient aircraft on the exploration of alternative propulsion technologies. Moreover, the rise of unmanned aerial vehicles on the integration of advanced materials have been notable trends reflecting a continued pursuit of efficiency safety and innovation. The aviation landscape has undergone profound transformation in the last two decades reflecting a confluence of technological innovation economic shifts and evolving consumer preferences. Several key market trends have shaped the trajectory of the aircraft industry, influencing manufacturing strategies, design philosophies and operational practices.

Over the past two decades, there has been a remarkable surge in technological advancements within the aircraft manufacturing sector. The advent of composite materials advanced avionics and fuel-efficient engines has significantly contributed to the development of more sustainable cost effective and high-performance aircraft. The integration of cutting-edge technologies has not only improved operational efficiency but has also elevated safety standards across the industry. Also, a notable trend in the market has been the increase in demand for regional and business aviation solutions. As economic growth has led to a rise in global business activities there has been a parallel increase in the demand for smaller more agile aircraft that cater to the specific needs of regional travel and corporate transportation. This trend has reshaped the market dynamics with manufacturers adapting their product portfolios to meet evolving demands of diverse consumer segments.

Environmental concerns have become a central focus within the aircraft industry prompting a shift towards more sustainable and eco-friendly aircraft designs. Manufacturers are investing heavily in the development of electric and hybrid propulsion systems as well as exploring alternative fuels. This trend reflects a broader industry commitment to reducing carbon footprints and addressing environmental changes. Over the past years the world has witnessed a digital revolution within the aviation sector. The integration of connectivity solutions in-flight entertainment systems and data analytics capabilities has not only enhanced passenger experience but also provide airlines with valuable insights into operational efficiency.

1.1 Problem Definition

In parallel with this industry trends the advent of Big Data has emerged as a transformative force in the aerospace sector. The aircraft market is characterised by a lack of standardised pricing models with valuation often relying on subjective assessments an expert opinion. This lack of uniformity poses challenges for both buyers and sellers leading to a degree of uncertainty and potential discrepancies in the perceived value of aircraft. The sheer volume and complexity of data generated throughout the lifecycle of an aircraft from design and manufacturing to operation and maintenance have needed a paradigm shift in how the industry extracts value from information. Big data analytics in aerospace has facilitated advancements in predictive maintenance safety monitoring an operational efficiency. Real time data streaming from aircraft sensors allows for proactive identification of potential issues minimising downtime and optimising maintenance schedules. Additionally, data-driven insights have played a crucial role in enhancing overall safety standards and decision-making processes. This project situated at the intersection of the aerospace industry and big data technologies aims to contribute to this ongoing transformation. This initiative aligns with the broader trend of leveraging data analytics to enhance efficiency and intelligence within the aerospace domain marking a significant step towards a more data informed future in aviation.

1.2 Statement of Purpose

The purpose of this project is to develop a robust and accurate machine learning model for predicting aircraft prices within the aerospace sector. Using Python as the primary programming language and harnessing the capabilities of Apache Spark and its deployment on Databricks, the objective is to create a sophisticated software solution that addresses the inherent challenges in valuing aircraft. Through systematic testing an using of berries machine learning libraries, the aim is to establish an efficient aircraft valuation software as a service that aligns with the evolving needs of the industry.

1.3 Background exploration

The project begins with an in-depth exploration of the historical theoretical and conceptual aspects of machine learning. This phase involves a comprehensive review of existing literature, models and methodologies within the real name of aircraft valuation and machine learning. By establishing a solid foundation, it's mandatory to contextualise the application of machine learning in the aerospace sector ensuring a nuanced understanding of the challenges and opportunities.

The dataset with all the necessary information about all concepts related to an aircraft that can be used in the test model step was obtained in Kaggle platform.

2. State of the Art

In this point, it is necessary to understand and approach an analysis of the literature about both aerospace sector and Big Data, also regarding context between both areas. This is the way to give the reader the opportunity of see what the present of the aircraft market sector is, the future that is yet to come, and how can new technologies, such as ML, Deep Learning or AI, can be used to predict not only market values, but also trends inside the sector.

2.1 Aerospace market present & trends

The aeronautical sector is a sector with high added value and export vocation, whose technological developments are incorporated into other sectors of activity present in our daily lives. The sector is very capital-intensive as the development of its products involves very high costs.

the development of its products involves very high costs that require very high investments whose returns begin to be generated, in most cases, in the very long term. It is, therefore, an eminently technological sector in which competition is not only for costs but also for knowledge and the capacity to develop new technologies that lead to improvements in products and/or processes.

2.1.1 Technology strategy of the aeronautics industry in Europe

Technology strategy plays a pivotal role in shaping the trajectory of innovation and progress. To enhance the competitiveness of the sector there is a strong emphasis on the continuous formation of the workers in business. Investment in comprehensive training programs ensures that the workforce remains adept at you using in cutting edge technologies, fostering a culture of adaptability and proficiency. Research development and innovation stand at the forefront of the Technology Strategy driving advancements in aircraft design manufacturing processes and operational efficiency. Collaborative efforts between industry stakeholders research institutions and government bodies fuel the development of novel technologies, positioning Europe as a global leader in aeronautical innovation. The technology strategy also extends to optimizing the supply chain with a focus on enhancing efficiency and resilience. Integration of digital technologies and data analytics ensures real time visibility and coordination across the supply chain, reducing lead times and improving overall operational performance.

2.1.1.1 Understanding the landscape

In response to global imperatives the aeronautics industry in Europe places a significant emphasis on environmental sustainability. Technological strategies prioritise the development and implementation of eco-friendly aviation solutions such as the exploration of alternative fuels lightweight materials an energy efficient propulsion system. This commitment to sustainability aligns with both regulatory requirements on the industry's responsibility to mitigate its environmental impact.

2.1.1.2 Challenges for the sector

The aeronautics industry in Europe faces several challenges that impact its overall sustainability and growth. This challenge encompasses the realms of exports employment and fiscal balance, posing intricate considerations for industry stakeholders and policymakers.

The European aeronautics industry grapples with the intricate dynamics of global trade, where competition is fierce and geopolitical factors can significantly influence market access. Trade barriers regulatory disparities evolving global economic conditions can impede the industry's ability to export aircraft and related technologies. Balancing export competitiveness while complying with international trade regulations is a persistent challenge that requires strategic navigation to ensure sustained growth in a highly interconnected global market.

Employment within the aeronautics sector is intricately linked to the industry's overall health and growth. The complexity of the industry demands a highly skilled and specialised workforce but fluctuations in demand, economic downturns and the evolving nature of technology can impact employment dynamics. Striking a balance between adapting to technological advancements and retaining a skilled workforce is crucial. Moreover, the industry must address challenges related to workforce diversity, inclusion and the impact of automation on traditional job roles.

Physical challenges arise from the significant capital investment required for research, development and the manufacturing of innovative aircraft. Maintaining a balance between the need for continuous technological advancements and physical presence is essential. Fiscal policies, tax incentives and public private partnerships play of it a role in influencing the financial health of the aeronautics industry. Striking a balance between fostering innovation and maintaining fiscal responsibility is a delicate yet imperative challenge for sustainable growth.

Addressing these challenges requires a comprehensive and coordinated approach from industry leaders, policymakers and other stakeholders. Through proactive measures and adaptive strategies, the European aeronautics industry can position itself to overcome these challenges and continue its legacy of innovation and global leadership.

2.1.2 Market structuring and environmental conditions

The global aeronautics industry, particularly in the domain of large civil aircraft, is primarily dominated by a de facto duopoly comprised of the American company Boeing on the European company Airbus. Towards the end of the last decade, the aeronautic sector faced challenges marked by a decline in orders and delays in major programmes deliveries. However, recent years have witnessed a noteworthy resurgence, with aircraft manufacturers experiencing sustained growth. This resurgence is notably propelled by the robust performance of the commercial aviation segment, underpinned by a consistent year on year increase in overall passenger traffic ranging from 5% to impressive levels between 8% and 10%. Simultaneously optimism pervades the defence sector, which after navigating previous years marked by budgetary constraints on public defence spending, is poised for growth. Geopolitical tensions persist, country voting to a positive outlook for defence spending. Amid these trends, manufacturing, product strategy and supply chain management emerged as the top three priorities for the forthcoming years. Industry leaders are actively developing innovative products to secure an expanding share of defence budgets, indicative of a strategic shift within the aeronautics landscape.

2.2 Data Science, a new field of study

Data Science represents a dynamic and interdisciplinary field that has emerged as a response to the unprecedented growth in data generation and the need to derive meaningful insights from this wealth of information. Rooted in the amalgamation of statistics, computer science, and domain-specific knowledge, Data Science involves the systematic exploration, analysis, and interpretation of data to extract valuable knowledge and support informed decision-making. Key components of Data Science encompass a multifaceted approach to extracting meaningful insights from data. The process begins with Data Exploration, where raw data is meticulously examined to comprehend its structure, identify patterns, and unlock potential insights. This phase involves sourcing relevant data and gaining a comprehensive understanding of the dataset's characteristics. Statistical Analysis forms the foundational bedrock of Data Science, providing methodologies to discern trends, patterns, and correlations within the data. Both descriptive and inferential statistics play a pivotal role in summarizing and drawing meaningful inferences from datasets. Machine Learning, as a subset of Data Science, introduces the development of algorithms capable of learning, predicting, and decision-making based on data. This includes a spectrum of techniques ranging from supervised learning for predictive modeling to unsupervised learning for clustering and pattern recognition. Data Cleaning and Preprocessing is a critical step in the Data Science pipeline, addressing imperfections in raw data such as errors, missing values, or outliers to ensure data quality for subsequent analysis and modeling. Effective communication of complex findings is achieved through Data Visualization techniques, utilizing charts, graphs, and interactive dashboards to present insights in a visually compelling manner accessible to stakeholders. In response to the exponential growth in data volume, velocity, and variety, Data Science often incorporates Big Data Technologies like Hadoop and Spark, providing the necessary infrastructure to process and analyze massive datasets efficiently. Lastly, Domain Expertise emerges as a crucial component, recognizing that Data Science extends beyond algorithms and code, requiring a profound understanding of the specific industry or domain. Incorporating domain expertise is fundamental for contextualizing findings and ensuring that the insights derived align with the broader goals of the organization. Together, these key components synergize to make Data Science a powerful and comprehensive discipline in the era of information-driven decision-making.

2.2.1 Machine Learning & AI

Machine learning represents A revolutionary facet of artificial intelligence enabling systems to learn and evolve from data without explicit programming. At it essence, machine learning is a dynamic discipline, deploying algorithms to identify patterns make predictions and generate insights from based datasets characteristic of our digital age. Uncle in key components machine learning relies on training data as the bedrock allowing models to recognise patterns and relationships. These models driven by sophisticated algorithms undergo iterative training to optimise performance. Evaluation metrics guide the refinement process ensuring accuracy and adaptability. The deployment of well-trained model integrates it into operational systems, enabling real world application across diverse domains, from finance and healthcare to image classification and autonomous vehicles. But as machine learning continues to evolve embracing advanced techniques like deep learning, it stands as a transformative force, synergising with big data and artificial intelligence to define the forefront of technological innovation and drive intelligent, data informed decision making our increasing interconnected world.

2.2.2 Deep Learning

Deep learning emerges as a pinnacle inexpensive rearm of artificial intelligence, representing A specialised subset of machine learning characterists by the utilisation of deep neural networks. These networks, inspired by the human brain 's interconnected neurons, use multiple layers that facilitate the automatic discovery and representation of intricate hierarchical patterns within vast datasets. At its core, deep learning excels in learning representations through these deep neural networks, each layer contributing progressively more abstract features. These deaf enables the model to capture complex relationships and dependencies that may elude traditional machine learning approaches. Key components include artificial neural networks deep neural networks with multiple hidden layers, under aliens on extensive and diverse data sets, often handled with big data technologies. The versatility of deep learning is evident in transformative applications spanning natural language processing, speech recognition, image analysis, and autonomous systems. Emerging trends incompatible architectures like convolutional and recurrent neural networks, while challenges include computational resource demands and interpretability issues. Despite these challenges, deep learning stands as a cornerstone in the ongoing evolution of artificial intelligence, poised to play an integral role in shaping more sophisticated and intelligent artificial intelligence systems.

2.2.2.1 Artificial neural networks

Artificial neural networks are the cornerstone of machine learning, drawing inspiration from the intricate connectivity of the human brain to enable computers to perform complex tasks and make intelligent decisions. Completing interconnected notes, or artificial neurons, arranged in layers, ANNs process information through a series of weighted connections. The network consists of an input layer, where data is introduced, hidden layers that iteratively refine representations, and an output layer providing the result. The magic of ANNs lies in their ability to autonomously learn from data. During training, the network adjusts the weights of connections based on the input data and desired output, optimizing its ability to make accurate predictions or classifications.

In this next paragraph, the explanation of how artificial neural networks work is developed:

- Input layer: the process begins with the input layer, where raw data is fed into the network. Each input node represents a feature, and the values are multiplied by weights to initiate the flow of information. For the function, input neurons receive signals from the external environment or serve as the initial points for data entering the neural network.
- Hidden layers: between the input and output layers lie hidden layers, each comprising numerous neurons. These layers refine the refine the input data by applying activation functions, introducing non linearities to capture complex patterns and relationships within the data. Hidden neurons process information and introduce non-linearities to the network, enabling the capture of complex patterns within the data. The output of a hidden neuron is determined by applying an activation function, typically denoted as h(.), to the weighted sum of inputs and biases. Mathematically the output 'y' is calculated as:

Equation 1. Output of a hidden neuron

$$y = h(\sum wi * xi + b)$$

Where 'wi' is the weight of the i-th input, 'xi', is the input signal, b is the bias and 'n' is the number of inputs.

- Weight adjustments: during training, the network compares its predictions to the actual outcomes. The error is calculated, and a process known as backpropagation is employed to adjust the weights of connections systematically. This iterative process fine-tunes the network's ability to make more accurate predictions over time.
- Output Layer: the final processed information is generated at the output layer. For tasks like classification, the output may represent probabilities or categorical predictions. Output neurons produce the result or prediction of the neural network's task. Like hidden neurons,

the output of a neuron is determined by applying an activation function. The mathematical expression is the same as the previous one.

For Python, there are many libraries to work with in ANNs. Here we can see two examples:

1. TensorFlow (Python): TensorFlow is a powerful open-source machine learning library developed by Google. It provides comprehensive support for building and training artificial neural networks. Its flexibility and scalability make it suitable for various applications, from image and speech recognition.
2. PyTorch (Python): Another popular open-source deep learning library, offering dynamic computational graphs and ease of use. Developed by Facebook, PyTorch is widely adopted for building neural networks and conducting research in areas such as computer vision and natural language processing.

These libraries, equipped with user-friendly interfaces and extensive documentation, empower developers and researchers to harness the capabilities of artificial neural networks for diverse applications, driving advancements in the field of machine learning.

2.2.2.2 Deep neural networks

Deep neural networks represent an advance and specialised form of artificial neural networks designed to tackle intricate tasks by leveraging multiple layers of interconnected neurons.

Unlike traditional artificial neural networks, which may consist of a limited number of hidden layers, deep neural networks are characterised by their depth, and composing numerous he then layers between the input and output layers. This depth enables deep neural networks to automatically learn and represent intricate hierarchical patterns within data, extracting increasingly abstract features with each layer.

Here we can name a few key characteristics of deep neural networks.

- Depth & hierarchy: DNNs Comprise multiple hidden layers, allowing for the creation of intricate hierarchies of features and representations. This depth enhances that network's ability to discern complex relationships within the data.
- Automated feature learning: deep neural networks excel at automated feature learning. Each layer in the network progressively extracts and refines features from the input data, enabling the model to capture nearest patterns that may be challenging for shallower architectures.
- Versatility across domains: deep neural networks demonstrate versatility across various domains, including image and speech recognition, natural language processing, and complex

decision-making tasks. Their ability to handle large and diverse data sets makes them well suited for applications requiring a high level of abstraction.

The primary difference between artificial neural networks and deep neural networks lies in the depth of the architecture. While ANNs may consist of a single hidden layer or a limited number of layers, DNNs are characterised by their extensive depth, typically involving multiple hidden layers. This depth allows to automatically learn hierarchical representations of data, capturing intricate patterns and relationships more effectively than traditional artificial neural networks. The increased depth in DNNs enhances their capacity of abstraction, making them particularly adept at handling complex tasks and large-scale data sets across diverse domains. In essence, deep neural networks stand as I specialised evolution of artificial neural networks, harnessing the power of death to unravel the intricate complexities inherent in modern data-driven challenges.

An engineer that works with deep neural networks may use Keras. Keras is an open-source neural network library written in Python. While it can use tensorflow or theano as a back end, it has become an integral part of tensor flows ecosystem since tensorflow 2.0. It provides a high-level user-friendly interface for building and training neural networks.

2.2.3 Machine Learning, Big Data & AI applications in aeronautics industry

2.2.3.1 Applications of Data Science in aeronautics industry

The integration of data science into the italics industry has ushered in a new era of transformative applications, optimising various facets of aviation operations. Predictive maintenance stands out as a key application, labour tracking historical data on aircraft performance and maintenance records to predict component failures, thereby minimising downtime and enhancing operational efficiency. Fuel efficiency optimization employs data science to analyse flight data, weather conditions, on aircraft metrics optimising flight parameters to reduce fuel consumption and environmental impact. Flight route optimization uses historical and real time data to dynamically adjust flight paths, speeds and engine settings, improving operational efficiency and reducing flight times.

Safety analytics and risk assessment involve analysing safety related data to identify patterns and potential risks, enabling proactive measures to enhance safety. Supply chain optimization uses data analytics to in streamline aircraft manufacturing and maintenance supply chains, optimising inventory and reducing lead times. Customer experience enhancement involves analysing passenger feedback, preferences and travel patterns to personalise services and optimise ticket pricing.

Aircraft design and performance optimization use simulations and computational modelling, edit by machine learning techniques to enhance aerodynamic performance, structural integrity and fuel efficiency. Regulatory compliance and reporting benefit from data-driven approaches, automating the analysis of regulatory compliance data and predicting areas of potential noncompliance. Queue scheduling and optimization leverage historical scheduling data and regulatory requirements, optimising crew schedules and predicting potential scheduling conflicts.

2.2.3.2 Applications of Machine Learning & AI in aeronautics industry

The applications of machine learning and artificial intelligence in the aeronautics industry are diverse and groundbreaking, and usually go deeper than data science applications. Predictive maintenance, powered by machine learning algorithms, enables airlines to anticipate and prevent aircraft component failures, minimising downtime and optimising maintain and schedules. AI-driven fuel efficiency optimization analysis vast data sets, including flight data and weather conditions, thereby lowering operational costs and environmental impact.

Regulatory compliance and reporting are streamlined through artificial intelligence applications that automate the analysis of compliance data and predict areas of potential non-compliance, insuring others to aviation regulations. It's important to clarify that data science machine learning and artificial intelligence are interconnected and often used together in various applications. Data science is a broader field that encompasses collecting, analysing and interpreting large volumes of data, while machine learning and AI are subsets of data science that involve building models and systems capable of learning and making decisions. That said, certain applications specifically emphasise the machine learning and artificial intelligence components, showcasing those unique capabilities beyond traditional data science:

- Autonomous systems and robotics: building autonomous drones, robots and bakers that can navigate and make decisions in real time based on environmental stimuli. These systems use machine learning algorithms for perception, decision making and control.
- Natural language processing for conversational AI: creating chat bots virtual assistance and language translation systems that can understand and generate human like language. These

involved machine learning techniques such as natural language understanding and language generation.

- Computer vision for image and video analysis: developing systems that can interpret and analyse visual data, including facial recognition, object detection and image classification. This involves training machine learning models on a large data sets of images or videos.
- Generative adversarial networks for creative applications: GANS are a class of machine learning algorithms that can generate new on synthetic data, often used for creating realistic images or even music. This goes beyond traditional data science, as it involves creating entirely new data rather than analysing existing data sets.
- Reinforcement learning for decision-making in dynamic environments: using reinforcement learning algorithms for training agents to make decisions and take actions in dynamic and changing environments. This is applicable in scenarios like optimising supply chain logistics, game playing an autonomous control systems.
- Personalised learning and education technology: creating adaptive learning systems that personalised educational content based on individual student performance and learning styles inside the aeronautics industry student programmes, for instance.

3. Case of Study. Developing a SaaS with ML model

3.1 Databricks

Databricks is a cloud based unified analytics platform designed to help organisations process, analyze and visualize large volumes of data efficiently. It was founded by the creators of Apache Spark, a powerful open-source distributed computing system. Databricks provides A collaborative environment for data scientists, data engineers and other professionals to work with big data and perform advanced analytics. It is a versatile platform that supports a wide range of use cases across data engineering, data science and business analytics. Some common use cases for data risks include:

- Big Data Processing: it leverages Apache Spark for distributed data processing, enabling organisations to handle massive data sets and perform tasks such as ETL operations.
- Data exploration and analysis: data scientists and analysts use databricks notebooks to explore and analyse data interactively. The platform supports multiple programming languages, including Scala, Python, R and SQL, making it versatile for data exploration and analysis.
- Machine learning and predictive analytics: databricks provides integrated machine learning capabilities, allowing data scientists to build, train and employ machine learning

models. It supports popular machine learning libraries and frameworks, making it a suitable platform for developing and operational in predictive models.

  - Data warehousing: Databricks can be used as a cloud based data warehouse, enabling organisations to store, manage and analyse structured and semi structured data. It supports SQL based analytics, making it accessible to users familiar with SQL queries.
  - Real time data processing: streaming analytics is a common use for databricks. Organisation leverage its capabilities to process and analyse real time data streams, allowing for immediate insights and actions based on the latest data.
  - Data pipelines and ETL: Databricks Is used to design and execute data pipelines for efficient ETL processes. Organisations leverage its capabilities to ingest, clean and transform data from various sources before loading it into data warehouses or other storage systems.
  - Advanced analytics and data science workflows: data scientists use Databricks from end-to-end data science workflows. This includes tasks such as future engineering, model training, hyper parameter tuning and model deployment.
  - Anomaly detection: it is employed, also, for luck analytics and anomaly detection in scenarios were organisations need to monitor and analyse logs for system performance, security or other operational insights.

3.1.1 Compute services

- Apache Spark Integration**:** Databricks extensively utilizes Apache Spark for distributed data processing. Apache Spark, a versatile and fast cluster-computing framework, supports in-memory processing and is well-suited for handling big data analytics tasks.
- Collaboration and Notebooks**:** Databricks offers collaborative notebooks, akin to Jupyter notebooks. In this shared environment, teams can write and execute code in multiple languages like Scala, Python, R, and SQL, fostering collaborative and interactive data exploration and analysis.
- Machine Learning: Databricks provides integrated machine learning capabilities, empowering data scientists to develop, train, and deploy machine learning models. It supports popular machine learning libraries and frameworks for comprehensive model development.
- SQL Analytics: users can perform SQL based analytics with Databricks, making it accessible to data analysts and business intelligence professionals

- Managed Cloud Services: Databricks operates as a managed cloud service on major cloud platforms such as AWS, Azure, and Google Cloud. This approach alleviates organizations from managing infrastructure, allowing them to concentrate on data and analytics.

3.1.2 Storage Services

- DFBS: it is a distributed file system that comes integrated with the platform. It allows users to seamlessly store and access files, including machine learning models.
- MLflow Tracking Server: MLflow is an open-source platform for managing the end to end machine learning lifecycle. It can be used to log and track different versions of ML models, allowing for easy comparison of performance metrics and facilitating model management.
- AWS, Microsoft Azure: Databricks seamlessly integrates with cloud storage solutions like Amazon Web Services S3, make yourself Azure BLOB storage or Google Cloud storage.

3.2 Apache Spark

Apache Spark unified statistics engine from processing large scale data with integrated modules for SQL, streaming, machine learning and graph processing. Spark can run on Apache Hadoop, Apache Mesos, Kubernetes, on its own, in the cloud and on multiple data sources. Apache Spark and Hadoop are the most prominent distributed systems on the market. They are similar top-level projects that are used together usually. Hadoop is mostly used for heavy disc operations with the MapReduce paradigm. Spark is a more flexible and often less expensive in memory processing architecture. The Spark ecosystem includes these key components:

- Spark Core is a general-purpose distributed data processing engine. It includes libraries for SQL, stream processing, machine learning and graph processing that can be used together in an application. Spark Core is the foundation of the entire project; it provides distributed task dispatching, scheduling and basic I/O functions.
- Spark SQL is Spark's module for working with structured data that supports a common way to access a variety of data sources. It allows you to query structured data within Spark programs using SQL or a familiar DataFrame API. Spark SQL supports HiveQL syntax and allows access to existing Apache Hive data stores. A server mode provides standard connectivity through Java database connectivity or an open database.

3.3 Dataset pre-analysis

3.3.1 License and Author of the Dataset

- Publisher of dataset: AvBuyer Limited / AvBuyer.com
- License: CC0 Public Domain
- Sponsor and Hosting Platform: Kaggle.com

3.3.2 Understanding the dataset

In this initial analysis of the data set containing attributes scrapped from the aircraft listings, its concept provides valuable insights into different phases of the aircraft. Here is a breakdown list with each attribute and its explanation for a comprehensive understanding:

- Condition: specifies if the aircraft is new, pre-owned, used, former or if the aircraft is a project. This feature is related with the original conditions of the engine.
- Make: represents the manufacturer or brand of the aircraft. It indicates the origin and potentially the reputation of the aircraft, influencing its perceived quality and market value.
- Model: specifies the specific model or version of the aircraft. It provides detailed information about the aircraft's features specifications and capabilities adding aiding in precise identification and evaluation.
- Category (Type of Aircraft): describes the category or classification of the aircraft. It is crucial for understanding its intended use, performance characteristics and suitability for different aviation purposes. Understanding these categories helps in assessing an aircraft's intended purpose, performance characteristics, and suitability for different aviation needs. Each type serves specific roles in aviation, ranging from leisure and training to business and military applications.

  1. **Gliders/Sailplanes:**
     - **Description:** Gliders or sailplanes are unpowered aircraft designed for sustained flight without an engine. They rely on rising air currents to stay aloft and are often used for recreational soaring or competitive gliding.
  2. **Gyrocopter:**
     - **Description:** Also known as autogyros, gyrocopters are rotary-wing aircraft that use an unpowered rotor for lift, similar to a helicopter. However, the rotor is not powered during flight, and forward thrust is provided by an engine-driven propeller.
  3. **Military:**

- **Description:** Military aircraft encompass a wide range of specialized vehicles used by armed forces for various purposes, including fighter jets, bombers, reconnaissance planes, and transport aircraft.

4. **MultiEngine Piston:**
   - **Description:** Multi-engine piston aircraft have more than one piston engine powering the aircraft. These are often used for transportation, executive travel, or regional flights, providing redundancy in case of engine failure.
5. **Piston Helicopters:**
   - **Description:** Piston helicopters are rotary-wing aircraft that use piston engines for propulsion. They are generally smaller and are used for tasks like light transport, training, or private use.
6. **Private Jets:**
   - **Description:** Private jets are small to medium-sized jet-powered aircraft designed for private or business travel. They offer speed, convenience, and the ability to access a wide range of airports.
7. **Single Engine Piston:**
   - **Description:** Single-engine piston aircraft are smaller planes with a single piston engine. They are often used for training, personal transportation, and recreational flying.
8. **Single Piston:**
   - **Description:** This category likely refers to single-engine piston aircraft, as the term "single piston" may be synonymous with "single-engine piston." These aircraft are generally smaller and more accessible for personal or training purposes.
9. **Turbine Helicopters:**
   - **Description:** Turbine helicopters are rotary-wing aircraft powered by turbine engines. They are often larger and more powerful than piston helicopters, used for a variety of purposes, including transportation, emergency services, and military applications.
10. **Turboprop:**
    - **Description:** Turboprop aircraft are equipped with a gas turbine engine that drives a propeller. They combine elements of jet and piston engines, providing efficiency and versatility. Turboprops are commonly used for regional and short-haul flights.

11. **Twin Piston:**
    - **Description:** Twin-piston aircraft have two piston engines, providing redundancy and enhanced performance. They are often used for private, business, or regional transportation.
12. **Ultralight:**
    - **Description:** Ultralight aircraft are lightweight, simple, and often single-seater aircraft. They are designed for recreational flying and are subject to weight and performance limitations.

- Year: indicates the manufacturing year of the aircraft.
- Price: represents the listed or observed price of the aircraft. Price is the key metric for buyers and sellers and for training the machine learning model. With this feature, it is possible to develop the supervised machine learning model. It influences the market competitiveness an indicates the financial investment required.
- Location: specifies the geographical location where the aircraft is currently located or registered. It is crucial for logistical considerations, import on export regulations and regional market variations in aircraft pricing.
- Serial Number: a unique identifier assigned by the manufacturer to distinguish individual aircraft. The serial number is essential for tracking the aircraft 's production history, maintenance records and ownership changes.
- Total hours of flight refer to the total number of fly towers the aircraft has accumulated. It provides insights into the usage and condition of the aircraft influencing maintenance schedules and resale value.
- Engine details (Engine 1 & Engine 2): describes the specifications and details of the aircraft's engines. Engine details are critical for assessing performance, fuel efficiency and maintenance requirements. The acronims are the included in the next table.

Table 1. Meanings of Engine 1, Engine 2, Prop 1 and Prop 2 columns

Acronym	Meaning
SMOH	Since major overhaul
SNEW	Since new
SPOH	Since prop overhaul
SFOH	Since factory overhaul
SOH	Since overhaul

STOH	Since top overhaul
SFRM	Since factory re-manufactured

In aircraft terminology, the terms "prop" and "engine" refer to different components of the propulsion system, and they play distinct roles in the aircraft's overall power generation and thrust production. Here's a breakdown of the difference between the two:

1. Engine:
    - Definition: The term "engine" in the context of aircraft generally refers to the entire powerplant, which includes both the core engine (usually a gas turbine or reciprocating engine) and other essential components such as the accessories, fuel system, and exhaust. The engine is responsible for generating the necessary power to drive the aircraft's propulsion system.
    - Types:
        - Gas Turbine Engine (Jet Engine): Commonly used in jet aircraft, it compresses air, mixes it with fuel, and ignites the mixture to produce high-speed exhaust gases that create thrust.
        - Reciprocating Engine (Piston Engine): Commonly found in smaller aircraft, it operates by converting reciprocating motion (up and down movement of pistons) into rotational motion to drive a propeller.
    - Functions: The engine's primary function is to convert fuel into mechanical energy to drive the aircraft's propulsion system and provide thrust for flight.
2. Propeller:
    - Definition: "Prop" is short for propeller, which is a specific component of the aircraft's propulsion system. The propeller is a rotating blade or set of blades that generates thrust by creating aerodynamic lift as it moves through the air. It is directly driven by the engine's power.
    - Types:
        - Fixed-Pitch Propeller: The pitch (angle of the blades) is fixed and cannot be adjusted during flight.
        - Variable-Pitch Propeller (Controllable Pitch Propeller): The pitch can be adjusted, allowing for better control of the aircraft's performance.

    - Functions: The propeller's main function is to convert the rotational power generated by the engine into thrust by accelerating air backward, adhering to Newton's third law of motion (action and reaction).

In summary, while "engine" encompasses the entire powerplant, including the core engine and associated systems, "prop" specifically refers to the propeller, which is the component responsible for creating thrust by manipulating airflow. The engine and prop work in tandem to produce the necessary power and thrust for the aircraft's operation, and the specific type of engine and propeller used can vary based on the aircraft's design and intended use.

- APU Details (Prop 1 & Prop 2): describes the specification and details of the auxiliary power unit if present. These details are crucial for understanding the aircraft's sufficiency and auxiliary power capabilities, particularly during ground operations.
- Airframe details: it provides information about the structural components and materials of the aircraft. All frame details contribute to understanding the overall condition, durability and potential maintenance needs.
- Interior details: it describes the interior features, how many teas and condition of the aircraft.
- Exterior details: specifies the external features, paint scheme an overall condition of the aircraft. Exterior details contribute to the visual appeal and marketability of the aircraft.
- Aircraft highlights: highlights key features, capabilities or not all the aspects of the aircraft. This feature serves as a succinct summary, attracting attention and providing quick insights into the aircraft selling points.
- Additional equipment details list any extra equipment or accessories included with the aircraft.
- Flight rules

  1. Visual Flight Rules (VFR):

  - *Definition:* Visual Flight Rules (VFR) refer to a set of regulations under which a pilot operates an aircraft in weather conditions generally clear enough to allow the pilot to see where the aircraft is going.
  - *Explanation:* When flying under VFR, pilots primarily navigate and control the aircraft by visual reference to the ground and other landmarks. The pilot must have a clear view of the surrounding airspace.

  2. Instrument Flight Rules (IFR):

- *Definition:* Instrument Flight Rules (IFR) define a set of regulations under which a pilot operates an aircraft when navigating solely by reference to instruments in the cockpit, without external visual reference.
- *Explanation:* Aircraft equipped with IFR capabilities rely on onboard instruments for navigation, altitude control, and other aspects of flight. This is crucial in conditions where visibility is limited, such as in clouds or adverse weather, where visual navigation is not feasible. By law, any aircraft flying over 18,000 feet must be equipped with IFR equipment.

3. Beyond the Horizon (BTH):

- *Definition:* Beyond the Horizon (BTH) signifies that an aircraft is equipped with radar capabilities but may not fully meet the criteria required for Instrument Flight Rules (IFR) operations.
- *Explanation:* Aircraft designated as BTH typically have radar systems onboard, allowing for extended detection capabilities. However, they may lack certain specific instruments or meet all the stringent requirements for IFR. BTH status could indicate that the aircraft is equipped for some level of instrument-based navigation but may not have the complete suite of instruments necessary for all IFR operations.

Summary:

- **VFR:** Pilots navigate visually, relying on external references.
- **IFR:** Pilots navigate using onboard instruments, especially crucial in conditions with limited visibility.
- **BTH:** Aircraft equipped with radar but may not fully meet all the criteria for comprehensive IFR operations.

In essence, VFR and IFR represent two distinct sets of rules governing how pilots operate aircraft, with VFR relying on visual cues, and IFR relying on instruments. The term BTH suggests an intermediate state, indicating radar capabilities but potential limitations in meeting all IFR criteria. Each category reflects different operational capabilities and requirements for safe and efficient flight in varying conditions. This comprehensive set of attributes provides a thorough condition for the initial analysis of the data set, enabling stakeholders to evaluate, compare and make informed decisions about the aircraft listings.

3.3.3 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a crucial phase in the data analysis process, where the primary goal is to visually and statistically explore a dataset to gain insights, discover patterns, and identify potential relationships among variables. EDA is often the first step in the data analysis workflow and is essential for understanding the characteristics of the data before applying more advanced modeling techniques.

Exploratory Data Analysis (EDA) stands as a pivotal phase within the data analysis workflow, serving as the initial step in understanding and interpreting datasets. Its primary objective is to delve into data both visually and statistically, unveiling insights, recognizing patterns, and pinpointing potential relationships among variables. By embracing EDA early in the analytical process, analysts can comprehend the inherent characteristics of the data, laying a foundation for informed decisions regarding subsequent advanced modeling techniques.

Key components integral to Exploratory Data Analysis encompass diverse methodologies. Summary statistics, including mean, median, standard deviation, and other descriptive measures, offer a swift glimpse into the central tendencies and variability within the dataset. Data visualization techniques, such as histograms, box plots, scatter plots, and heatmaps, facilitate the visual assessment of distribution, patterns, and relationships, aiding in the identification of outliers, trends, and clusters. Moreover, EDA encompasses tasks such as missing data analysis, outlier detection, and correlation analysis, which collectively contribute to data integrity and an in-depth understanding of inter-variable dependencies. Feature engineering, pattern recognition, and hypothesis generation further enrich the exploratory process, fostering the generation of meaningful insights and guiding subsequent stages of rigorous statistical analysis. In essence, Exploratory Data Analysis serves as a dynamic and iterative process, refining understanding and paving the way for comprehensive and insightful data exploration.

3.3.3.1 General Data Visualization with Power BI

In addition to traditional exploratory data analysis techniques, a comprehensive visual analysis of the entire contents of the CSV file was conducted using the Power BI tool. Power BI, a powerful business analytics tool by Microsoft, facilitates the creation of interactive and visually compelling reports and dashboards. By leveraging Power BI, the dataset underwent a thorough examination through various visualizations, enhancing the exploration process.

The tool's capabilities allowed for the creation of dynamic charts, graphs, and interactive visual representations, providing a holistic view of the data's distribution, trends, and relationships. Power BI's intuitive interface enabled the generation of insightful visualizations, enabling the identification of patterns and anomalies within the dataset. This visual exploration not only complemented the statistical analysis but also provided a more accessible and intuitive understanding of the data for stakeholders and decision-makers. The integration of Power BI into the exploratory data analysis workflow added a layer of sophistication, ensuring a multifaceted understanding of the dataset and enriching the overall analytical process.

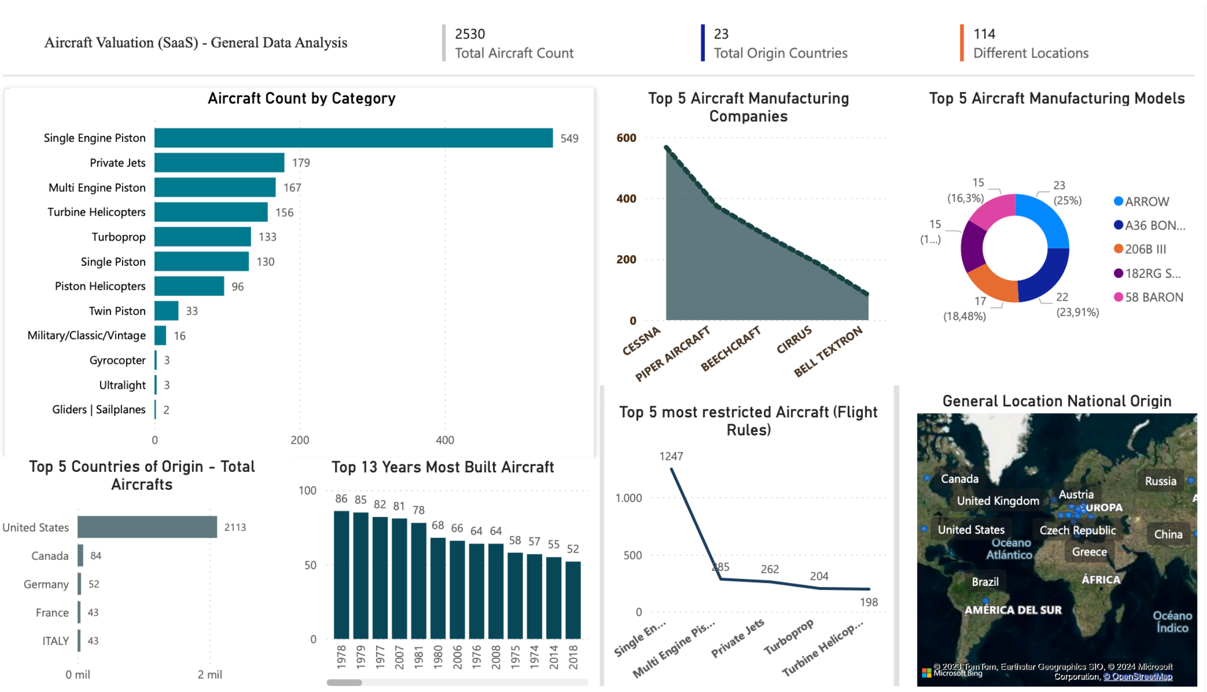


Figure 1. General Data Analysis

For a more in-depth analysis, we will delve into each major characteristic by providing detailed descriptions of individual analyses conducted during the exploration. This entails a meticulous examination of summary statistics, data visualization, and specific techniques applied to address aspects such as missing data, outlier detection, feature engineering, correlation analysis, pattern recognition, and hypothesis generation. By dissecting each of these elements, we aim to offer a granular understanding of the dataset, providing insights that go beyond overarching trends and relationships. This comprehensive approach ensures a nuanced exploration of the data's intricacies, laying the groundwork for informed decision-making and further advanced modeling techniques.

3.3.3.2 Aircraft Count by Category

The initial visual analysis focused on obtaining the number of aircraft models categorized by aircraft type, revealing intriguing insights.

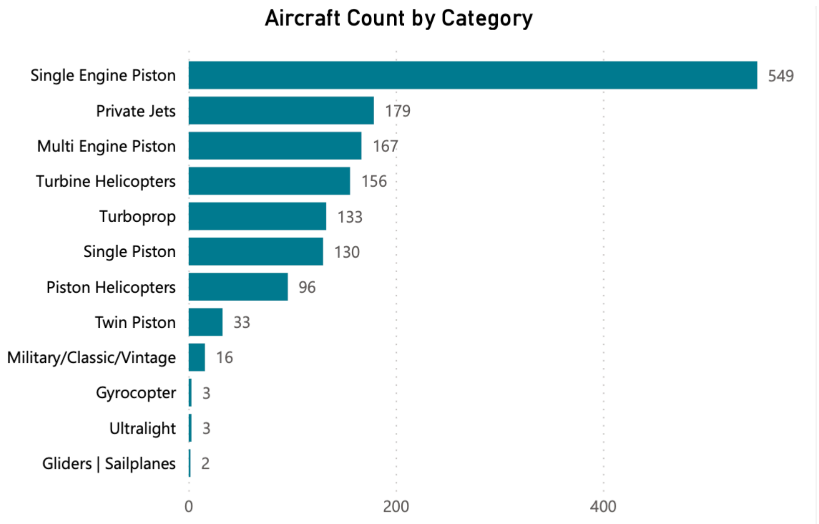


Figure 2. Aircraft Count by Category

Notably, within the "Single Engine Pistons" category, a standout finding emerged—this category boasts a diverse array of 549 distinct models.

"Single Engine Pistons" refer to a class of aircraft characterized by having a solitary piston-powered engine. These aircraft are widely recognized for their simplicity, cost-effectiveness, and ease of operation. They are predominantly used for general aviation purposes, including personal and recreational flying, flight training, and short-distance travel. Single-engine piston aircraft are known for their versatility, making them popular choices for pilots at various skill levels.

The category encompasses a range of aircraft designs, from small two-seaters to larger cabin-class planes. Despite their relatively modest size and power compared to other aviation categories, single-engine pistons play a crucial role in the aviation landscape, serving as accessible and practical options for a broad spectrum of aviation enthusiasts and professionals. The diversity of 549 models within this category underscores the innovation and adaptability within the realm of single-engine piston aircraft, reflecting the continual evolution of aviation technology and design.

The visual analysis further reveals that the second-largest category of aircraft is "Private Jets," boasting a substantial 179 distinct models. Private jets are sophisticated, high-performance aircraft designed primarily for executive and luxury travel. These planes offer unparalleled speed, comfort, and range, making them ideal for business executives and affluent individuals who require efficient and flexible transportation. Private jets often feature luxurious interiors, advanced avionics, and cutting-edge amenities, reflecting a commitment to convenience and style in air travel.

Following closely is the "Multi Engine Piston" category, comprising 167 models. Multi-engine piston aircraft typically have two or more piston-driven engines, providing enhanced performance and redundancy. These aircraft are employed for a variety of purposes, including commercial charter flights, flight training, and regional transportation. Multi-engine piston planes are valued for their increased power, larger passenger capacity, and improved safety features, making them suitable for a diverse range of aviation applications.

Conversely, the list concludes with "Ultralights" and "Sailplanes," each featuring only 3 and 2 models, respectively. Ultralights are lightweight, single-seat aircraft designed for recreational flying. They often have simple structures and are powered by small engines, offering an accessible and affordable entry point into aviation. Sailplanes, on the other hand, are glider aircraft without engines, relying on atmospheric currents for propulsion. Sailplanes are favored for recreational soaring and gliding, providing a serene and environmentally friendly flying experience. Despite their limited model count, both Ultralights and Sailplanes represent unique niches within the aviation landscape, catering to enthusiasts seeking distinct and specialized flying experiences.

3.3.3.3 Countries of Origin – Total Aircrafts

Moving on to the next facet of analysis, the focus shifts to the top five countries of origin based on the total number of aircraft produced.

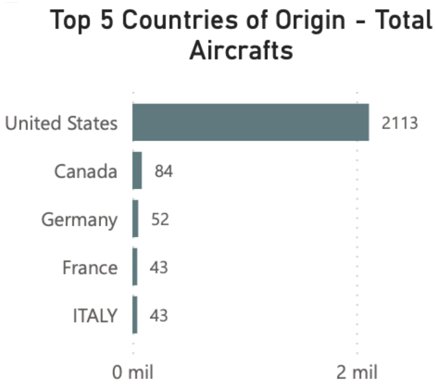


Figure 3. Countries of Origin - Total Aircrafts

Topping the list is the United States, and this preeminence can be attributed to the country's robust and dynamic aviation industry. The United States has long been a global leader in aircraft manufacturing, home to major aerospace corporations, innovative research and development facilities, and a rich history of aviation advancements. The presence of renowned companies and a culture of aerospace innovation positions the U.S. as a dominant force in the production of a diverse range of aircraft, contributing significantly to its top-ranking status.

Following the United States are Canada, Germany, and Italy, marking them as the top three European countries in terms of aircraft production within the dataset. Canada's prominence can be linked to a strong aviation heritage, bolstered by well-established companies known for manufacturing a variety of aircraft types. In Germany, a combination of precision engineering, technological expertise, and a tradition of aviation excellence places the country among the leading European producers. Italy, renowned for its craftsmanship and engineering prowess, particularly in the realm of luxury and high-performance aircraft, secures its position among the top European contributors to the dataset.

The collective strength of these European nations in aircraft production can be attributed to a combination of historical legacy, technological capabilities, and a commitment to innovation. Each country brings its unique strengths to the global aviation landscape, contributing to a diverse and competitive market within Europe.

3.3.3.4 Top 5 Aircraft Manufacturing Companies

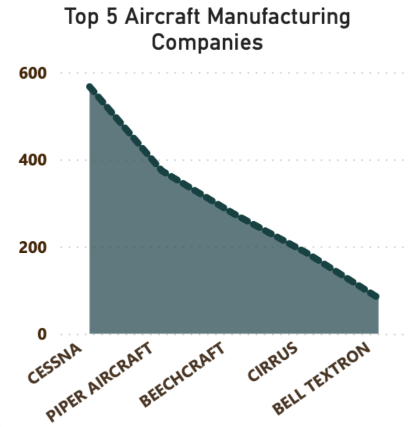


Figure 4. Aircraft Manufacturing Companies

If we delve into the top 5 aircraft companies by total production:

1. CESSNA

Cessna, a prominent name in the aviation industry, secures the top spot in total production. Known for manufacturing a diverse range of general aviation aircraft, Cessna has been a stalwart in the industry for decades. The company, a subsidiary of Textron Aviation, is renowned for its popular lines of single-engine and light twin-engine aircraft. Cessna's aircraft are widely used for personal and business flying, flight training, and regional transportation. The Cessna 172 and Cessna 182 are among the most widely produced and recognized aircraft in the world, contributing to the company's leading position.

2. Piper Aircraft:

Piper Aircraft, another key player in general aviation, follows Cessna in total production. Piper is recognized for its range of single-engine and light twin-engine aircraft. The company's product line includes popular models like the Piper Cherokee, Piper Arrow, and Piper Seneca. Piper Aircraft is celebrated for its commitment to producing reliable and efficient aircraft, catering to a diverse customer base, including private pilots, flight schools, and businesses.

3. Beechcraft

Beechcraft, a historic name in aviation, secures a notable position in the top 5. Known for producing a range of aircraft, including the iconic Beechcraft Bonanza and Beechcraft King Air series, the company has a legacy of craftsmanship and innovation. Beechcraft aircraft are renowned for their performance, comfort, and versatility, making them popular choices among business and general aviation users.

4. Cirrus Aircraft

Cirrus Aircraft is recognized for its innovation in the aviation industry, particularly for its emphasis on safety and technology. The company is notable for introducing the Cirrus SR22, a high-performance single-engine aircraft equipped with features like the Cirrus Airframe Parachute System (CAPS). Cirrus aircraft are favored by pilots seeking advanced avionics and modern design elements.

5. Bell Textron

Bell Textron, a leading manufacturer in the aerospace industry, rounds out the top 5. While traditionally known for its helicopters, Bell has expanded its portfolio to include vertical takeoff and landing (VTOL) aircraft and tiltrotors. Bell Textron's products are utilized for various purposes, including military, commercial, and private applications. The Bell 206 JetRanger and the Bell 407 are among the well-known helicopter models produced by the company.

Each of these companies contributes to the rich tapestry of the aviation industry, bringing unique strengths, innovations, and a commitment to excellence in aircraft manufacturing.

3.3.3.5 Top 5 Aircraft Manufacturing Models

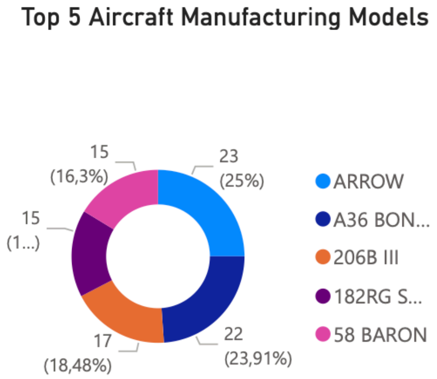


Figure 5. Aircraft Manufacturing Models

The analysis of Single Engine Piston models reveals that the "Arrow" emerges as the most manufactured model in this category. The Arrow, likely produced by a specific aircraft manufacturer, showcases the popularity and demand for this particular single-engine piston aircraft. Known for its performance, versatility, and possibly unique features, the Arrow's high production numbers underscore its significance within the general aviation landscape.

Following closely in production numbers is the A36 Bonanza 36, a noteworthy model within the Single Engine Piston category. The Bonanza series, manufactured by Beechcraft, has a storied history and is celebrated for its reliability, advanced avionics, and comfortable cabin design. The A36 Bonanza 36, with its enduring appeal and capabilities, secures its position as one of the top manufactured single-engine piston aircraft, reflecting the enduring legacy and popularity of Beechcraft's Bonanza series in the aviation community. The prevalence of these models underscores their widespread use for various purposes, including personal and business flying, training, and recreational aviation.

3.3.3.6 Top 5 Restricted Aircrafts (Flight Rules)

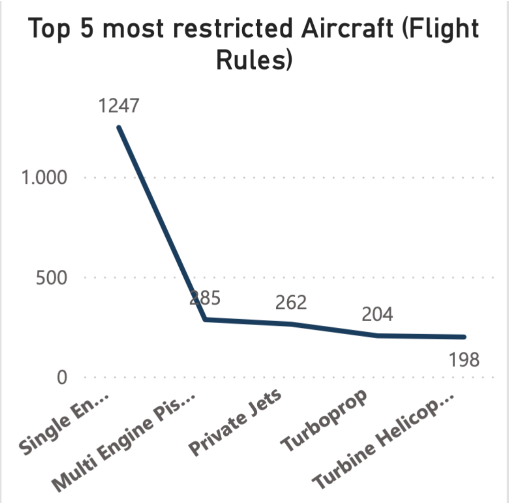


Figure 6. Aircraft Restricted per Flight Rules

- Single Engine Piston: while versatile and widely used, are subject to more restrictive flight rules due to their reliance on a single piston-driven engine. These rules may include limitations on the types of airspace they can access, as well as restrictions related to adverse weather conditions. Pilots of single-engine piston aircraft often face more stringent regulations, necessitating careful consideration of flight planning and route selection.

- Multi Engine Piston: with their increased power and redundancy, still operate under certain flight rules and regulations. While they offer more capabilities than their single-engine counterparts, pilots of multi-engine piston aircraft must adhere to specific rules governing their use, especially in terms of performance, maintenance, and operational considerations.

- Private Jets: despite their advanced capabilities and luxurious features, face regulatory constraints that dictate their flight operations. These rules may involve restrictions on certain airspaces, airport facilities, and noise levels. Compliance with aviation regulations is crucial for private jet operators to ensure safe and lawful operations.

- Turboprop: Turboprop aircraft, while versatile, are subject to specific flight rules based on their propulsion system. These rules address factors such as engine performance, fuel efficiency, and noise levels. Pilots of turboprop aircraft must adhere to regulations that govern their operation, particularly when it comes to takeoff and landing procedures.

- Turbine Helicopter: Turbine helicopters, known for their power and agility, operate within a set of flight rules that consider their unique capabilities. Regulations governing helicopter operations encompass factors like altitude limitations, noise restrictions, and safety protocols.

Pilots of turbine helicopters must navigate these rules to ensure safe and compliant flight operations.

3.3.3.7 Years with Most Built Aircraft

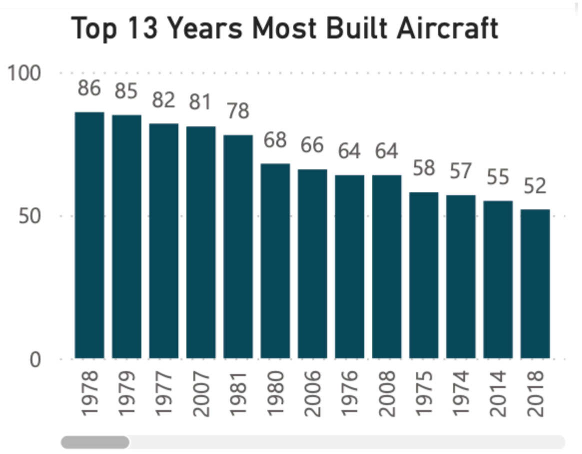


Figure 7. Years with Most Built Aircraft

The late 1970s marked a pivotal period for aircraft production, driven by a convergence of factors that influenced both the commercial and military aviation sectors. One key catalyst during this timeframe was the resolution of the energy crisis that had unfolded earlier in the decade. The oil embargo of 1973 had sent shockwaves through the global economy, affecting various industries, including aviation. As nations worked to stabilize energy supplies, the aviation industry began to recover, and governments and airlines regained confidence to invest in the development and expansion of their aircraft fleets.

Furthermore, the mid-to-late 1970s witnessed a culmination of advancements in aviation technology. This period saw the introduction of several iconic aircraft models, including the Boeing 747, the McDonnell Douglas DC-10, and the Lockheed L-1011 TriStar in the commercial aviation sector. These wide-body, long-range aircraft revolutionized air travel, offering greater capacity and efficiency. Simultaneously, military aviation programs were advancing, with the development of new fighter jets and strategic bombers responding to geopolitical tensions of the time. The infusion of cutting-edge technology and the modernization of aging fleets drove a surge in aircraft production during the late 1970s.

Moreover, the geopolitical landscape, particularly during the latter part of the decade, contributed to increased military aircraft production. Cold War tensions and regional conflicts spurred nations to bolster their defense capabilities, leading to the construction of advanced fighter jets, reconnaissance aircraft, and strategic bombers. The combination of economic recovery, technological innovation, and geopolitical considerations positioned the late 1970s as a peak period for aircraft production, with 1977 to 1979 specifically witnessing a culmination of these influential factors.

3.4 Data Pre-Processing

Data preprocessing stands as a pivotal and indispensable phase in the data analysis pipeline, playing a key role in ensuring that datasets are cleansed, accurate, and primed for further analysis. This multifaceted process encompasses a range of tasks, including addressing missing values, rectifying errors, and transforming data into a format conducive to analysis. In the context described, it's noted that the elimination of erroneous or null values was executed utilizing Python functions within Excel, showcasing a hybrid approach that amalgamates the strengths of both tools. Let's delve into the significant aspects of this data preprocessing endeavor.

Dealing with missing or null values constitutes a common yet critical facet of data preprocessing. The presence of such values can significantly impact the accuracy of analyses, necessitating careful consideration. Python, with its versatile libraries like pandas, provides robust tools for data manipulation. It is plausible that functions from these libraries were leveraged to identify and manage null values within the dataset. Techniques such as imputation, involving the replacement of missing values with calculated estimates, or the outright removal of rows and columns containing null values may have been employed to enhance the dataset's completeness and reliability.

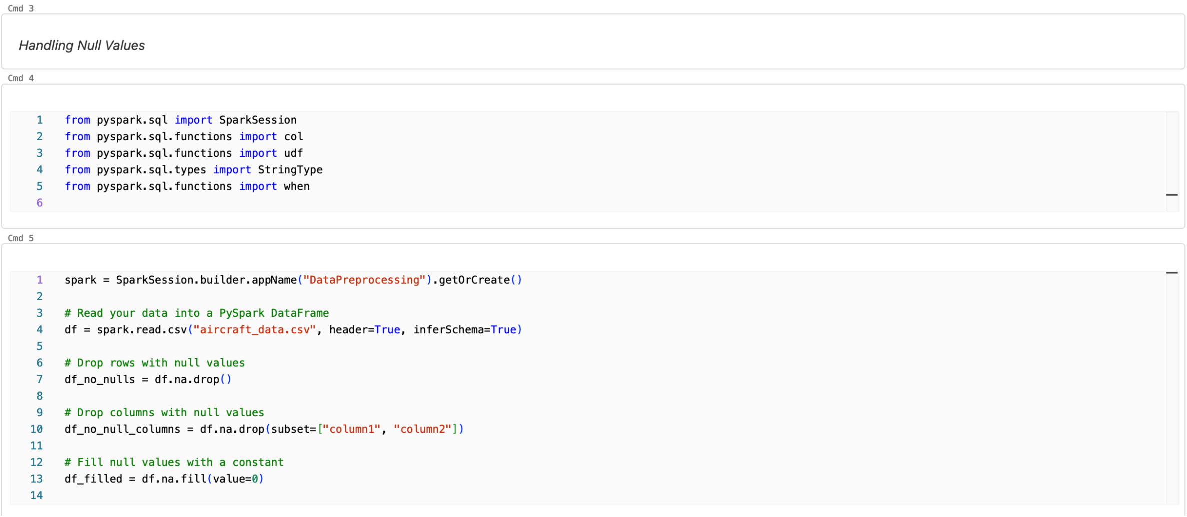
Erroneous values, if not addressed, can distort analytical results and compromise the integrity of insights drawn from the data. The utilization of Python functions within the Excel environment implies a strategic integration of tools. The pandas library in Python, renowned for its capabilities in data cleaning and transformation, may have been instrumental in identifying and rectifying errors. Functions or scripts crafted in Python could have played a pivotal role in systematically detecting and addressing inconsistencies, thereby fortifying the overall quality and reliability of the dataset.

The explicit mention of Python functions seamlessly operating within Excel points to a collaborative workflow where the strengths of Python are harnessed within the familiar interface of Excel. This integration might have been facilitated through tools like xlwings or similar methods that enable the execution of Python code directly within the Excel environment. Such synergy empowers analysts with the flexibility to leverage Python's advanced data processing capabilities while capitalizing on the user-friendly interface offered by Excel, thereby streamlining and enhancing the efficiency of data preprocessing tasks.

Beyond the realm of handling missing values and errors, data validation emerges as a critical aspect of the preprocessing journey. Validation checks are instrumental in ensuring that the dataset aligns with predefined rules and constraints. Python scripts likely played a pivotal role in conducting these validation tasks, reinforcing the quality and integrity of the dataset by verifying its adherence to specified criteria. This meticulous validation process adds an additional layer of robustness, paving the way for more accurate and trustworthy analyses in subsequent stages of the data analysis pipeline.

3.4.1 Handling Null Values

The methods demonstrated for handling null values using Python (pandas) and PySpark offer versatile approaches. Users can choose between dropping nulls, imputing values, or applying custom strategies based on the nature of the data. The functions provided are easy to implement, making them accessible for users with varying levels of expertise. Whether it's pandas for single-machine setups or PySpark for distributed computing, both frameworks offer user-friendly solutions for managing null values efficiently.
Samples of the complete DataBricks code are included for this point.



Cmd 3

Handling Null Values

Cmd 4

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType
from pyspark.sql.functions import when

```

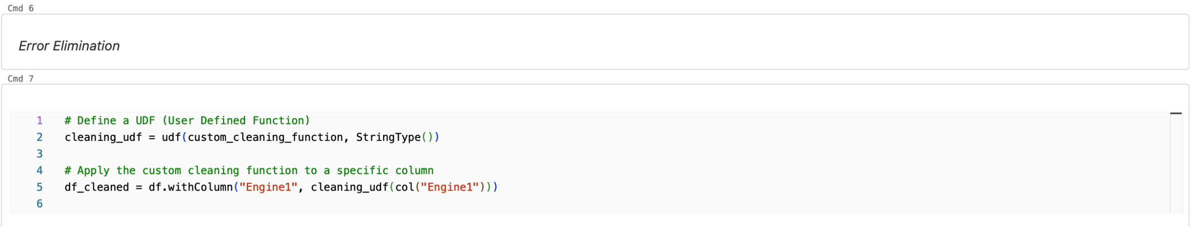
Cmd 5

```python
spark = SparkSession.builder.appName("DataPreprocessing").getOrCreate()

# Read your data into a PySpark DataFrame
df = spark.read.csv("aircraft_data.csv", header=True, inferSchema=True)

# Drop rows with null values
df_no_nulls = df.na.drop()

# Drop columns with null values
df_no_null_columns = df.na.drop(subset=["column1", "column2"])

# Fill null values with a constant
df_filled = df.na.fill(value=0)

```

Figure 8. Handling Null Values (Sample)

3.4.2 Error Elimination

The PySpark approach to error elimination exemplifies the customizability of data cleaning processes. By leveraging user-defined functions (UDFs), users can apply tailored cleaning logic to address specific errors or inconsistencies within the dataset. PySpark, designed for distributed computing, demonstrates scalability in handling errors, making it suitable for large-scale datasets. This characteristic is crucial when dealing with extensive datasets that may be challenging to process on a single machine.



Cmd 6

Error Elimination

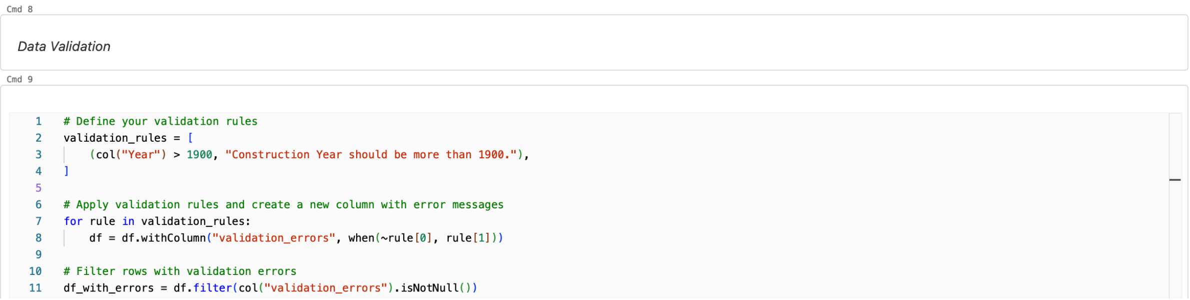
Cmd 7

```python
# Define a UDF (User Defined Function)
cleaning_udf = udf(custom_cleaning_function, StringType())

# Apply the custom cleaning function to a specific column
df_cleaned = df.withColumn("Engine1", cleaning_udf(col("Engine1")))

```

Figure 9. Error Elimination

3.4.3 Data Validation

The data validation method emphasizes a rule-based approach, where predefined rules are applied to ensure data quality and adherence to specific constraints. This approach enhances the reliability of the dataset for downstream analyses. By creating a new column to capture validation errors, this method provides transparency regarding the data points that violate predefined rules. This characteristic is valuable for data quality assessment and decision-making in subsequent analysis steps.



Cmd 8

Data Validation

Cmd 9

```python
# Define your validation rules
validation_rules = [
    (col("Year") > 1900, "Construction Year should be more than 1900."),
]

# Apply validation rules and create a new column with error messages
for rule in validation_rules:
    df = df.withColumn("validation_errors", when(~rule[0], rule[1]))

# Filter rows with validation errors
df_with_errors = df.filter(col("validation_errors").isNotNull())
```

Figure 10. Data Validation

In addition to the characteristics mentioned for the three data preprocessing methods, there are several other crucial aspects that should be considered:

Handling Outliers: In the realm of data preprocessing, addressing outliers is a nuanced task that merits careful consideration. Outliers, being data points significantly deviating from the majority, can exert disproportionate influence on analysis outcomes. Robust identification techniques are essential, ranging from statistical methods like the Z-score or the IQR method to visualizations such as box plots. Once identified, the decision on how to handle outliers depends on the context of the analysis. In certain scenarios, outliers may be transformed or adjusted to align with the general distribution, while in other cases, preserving them might be necessary for specific analytical insights. Therefore, an understanding of the impact of outliers on the analysis is crucial for informed decision-making during the data preprocessing stage.

Feature Scaling: Feature scaling is a pivotal aspect of data preprocessing that ensures a level playing field for variables with different scales or magnitudes. This step is particularly vital for machine learning algorithms sensitive to the magnitude of features. Common techniques for feature scaling include normalization, where values are scaled to a range between 0 and 1, and standardization, involving scaling with the mean and standard deviation. The choice between normalization and standardization depends on the distribution of the data and the requirements of the machine learning model. Ensuring that features are appropriately scaled enhances the model's performance, enabling fair and meaningful comparisons between variables and contributing to the overall effectiveness of the analysis.

Handling Categorical Data: Another critical consideration in data preprocessing involves the treatment of categorical data. Many machine learning algorithms require numerical input, necessitating the conversion of categorical variables into a numerical format. Techniques such as one-hot encoding, where each category is represented by binary columns, or label encoding, assigning numerical labels to categories, are commonly employed. The choice between these methods depends on the nature of the data and the specific requirements of the analysis. Proper handling of categorical data ensures that valuable information is retained while making it compatible with machine learning algorithms, contributing to the success of subsequent analyses.

4. Development of the aircraft price prediction algorithm in Databricks on Apache Spark

Deploying the aircraft price prediction case study in Databricks offers several advantages, primarily due to Databricks' capabilities for scalable and collaborative big data processing and machine learning tasks. Databricks is built on Apache Spark, a powerful open-source framework for distributed computing, making it suitable for handling large datasets and performing complex computations. Databricks provides a cloud-based, collaborative platform where users can create notebooks to execute code, visualize data, and build machine learning models. The architecture includes a cluster manager that dynamically allocates resources, making it easy to scale compute power as needed.

4.1 Python Deployment on Databricks connection with Apache Spark libraries

In the initial phase of our aircraft price prediction case study within Databricks Community, we have successfully imported the requisite Python libraries essential for streamlined machine learning workflows. These libraries include **pyspark.sql**, which facilitates the handling of structured data using Apache Spark, and **pyspark.ml**, a module that provides machine learning algorithms and tools for building scalable and distributed models.

The seamless integration of these libraries sets the foundation for efficient data processing and model development within the collaborative Databricks environment. Leveraging the distributed computing capabilities of Spark, we are well-equipped to handle large-scale datasets and implement sophisticated machine learning algorithms.

Furthermore, we have successfully loaded the aircraft dataset from the CSV file and displayed the first rows. This critical step ensures the accuracy of our data import process, providing us with a preliminary overview of the dataset's structure and content. By confirming the successful execution of these initial tasks, we are now poised to delve into model development.

4.1.1 Configuration of the necessary parameters for the connection from Python to Databricks

This step involves installing a specific library called **databricks-connect**, which facilitates the connection between Python and Databricks. It's similar to installing an essential tool that enables communication between the development environment (VS Code) and the Databricks platform.

The installation of "databricks-CLI" is a prerequisite for the subsequent steps and ensures that the development environment is equipped with the necessary tools to interact with Databricks.

Databricks CLI (Command Line Interface) is a tool that helps manage Databricks resources. This step involves configuring the CLI by providing the Databricks workspace URL and an access token. The access token serves as a secure authentication mechanism.

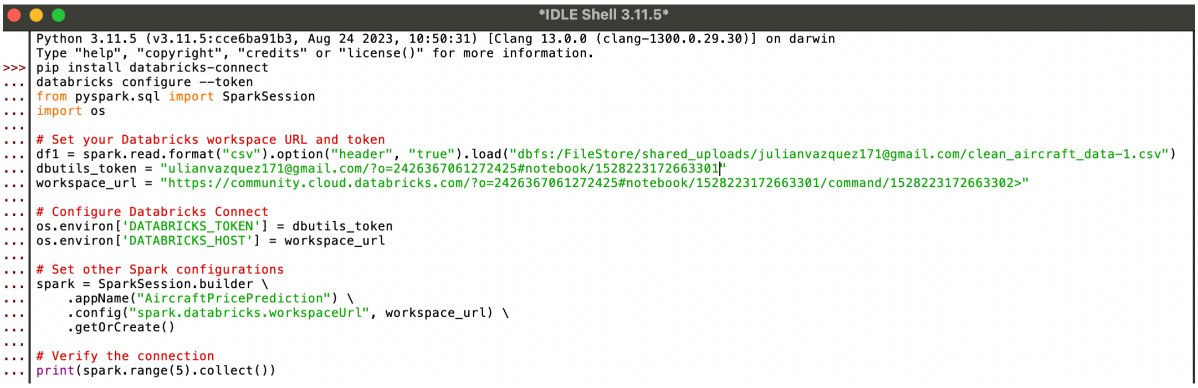
Configuring the CLI establishes a secure connection between the development environment and Databricks, allowing the development tools to access and manipulate resources on the Databricks platform.

Now, to deploy the step that involves writing Python code in VS Code to configure the connection to Databricks using the **databricks-connect** library It is necessary to include setting environment variables for the Databricks token and workspace URL, as well as configuring Spark session settings.

This configuration ensures that the Python code written in VS Code can communicate seamlessly with Databricks. It sets up the necessary parameters to establish a connection and perform data operations on Databricks clusters.

The first step involves installing a software component called **databricks-connect**. This component acts as a bridge, enabling communication between the development environment (VS Code) and Databricks, the platform where data processing and analysis will take place. It's similar to installing a necessary tool that facilitates collaboration between the development environment and Databricks.

The next step is to configure the Databricks Command Line Interface (CLI). The CLI serves as a command-based tool for managing resources on the Databricks platform. During this configuration, stakeholders will set up the CLI with essential information such as the Databricks workspace URL and an access token. This access token serves as a secure key to establish a trusted connection between the development environment and Databricks, ensuring that only authorized users can interact with Databricks resources.

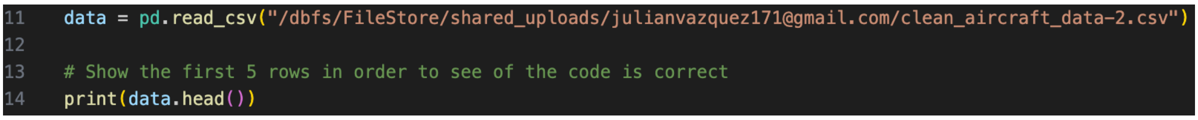


```
Python 3.11.5 (v3.11.5:cce6ba91b3, Aug 24 2023, 10:50:31) [Clang 13.0.0 (clang-1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> pip install databricks-connect
... databricks configure --token
... from pyspark.sql import SparkSession
... import os
... 
... # Set your Databricks workspace URL and token
... df1 = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/shared_uploads/julianvazquez171@gmail.com/clean_aircraft_data-1.csv")
... dbutils_token = "ulianvazquez171@gmail.com/?o=2426367061272425#notebook/1528223172663301"
... workspace_url = "https://community.cloud.databricks.com/?o=2426367061272425#notebook/1528223172663301/command/1528223172663302>"
... 
... # Configure Databricks Connect
... os.environ['DATABRICKS_TOKEN'] = dbutils_token
... os.environ['DATABRICKS_HOST'] = workspace_url
... 
... # Set other Spark configurations
... spark = SparkSession.builder \
...     .appName("AircraftPricePrediction") \
...     .config("spark.databricks.workspaceUrl", workspace_url) \
...     .getOrCreate()
... 
... # Verify the connection
... print(spark.range(5).collect())
```

Figure 11. Configure Databricks Connect & CLI

Once the CLI is configured, the development team will write Python code within VS Code to establish a connection to Databricks using the **databricks-connect** library. This involves setting parameters, such as the Databricks token and workspace URL, in the Python script. The configuration is necessary for creating a seamless link between the development environment and Databricks clusters. It's like setting up a secure pathway for data and code to flow between the two environments.

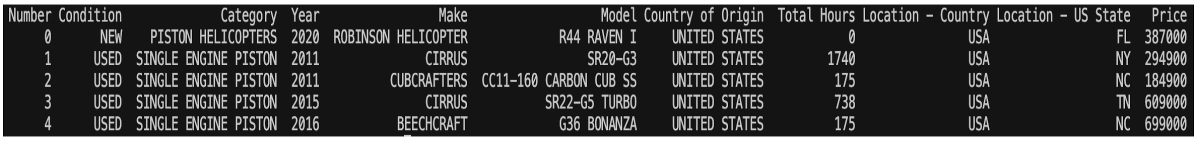
After configuring the connection, stakeholders will execute a Python script within VS Code. This script serves as a practical check to ensure that the established connection works as intended. The script performs a basic operation, printing a range of numbers, to verify that data can be sent to and received from Databricks. This step is crucial as it validates the successful integration of the development environment with Databricks, providing confidence before proceeding with more complex tasks like data analysis and machine learning model training.



```
data = pd.read_csv("/dbfs/FileStore/shared_uploads/julianvazquez171@gmail.com/clean_aircraft_data-2.csv")

# Show the first 5 rows in order to see of the code is correct
print(data.head())
```

Figure 12. Running the Script for the path in Databricks

Lastly, we will print the first 5 rows to ensure that the script runs ok.



```
Number Condition             Category  Year               Make                Model Country of Origin  Total Hours Location - Country Location - US State   Price
     0       NEW    PISTON HELICOPTERS  2020  ROBINSON HELICOPTER          R44 RAVEN I     UNITED STATES            0               USA                FL  387000
     1      USED  SINGLE ENGINE PISTON  2011               CIRRUS              SR20-G3     UNITED STATES         1740               USA                NY  294900
     2      USED  SINGLE ENGINE PISTON  2011          CUBCRAFTERS  CC11-160 CARBON CUB SS  UNITED STATES          175               USA                NC  184900
     3      USED  SINGLE ENGINE PISTON  2015               CIRRUS        SR22-G5 TURBO     UNITED STATES          738               USA                TN  609000
     4      USED  SINGLE ENGINE PISTON  2016           BEECHCRAFT          G36 BONANZA     UNITED STATES          175               USA                NC  699000
```

Figure 13. Top 5 Rows from Dataset

4.1.2 First Machine Learning Method: SKICIT-LEARN Library – LinearRegression

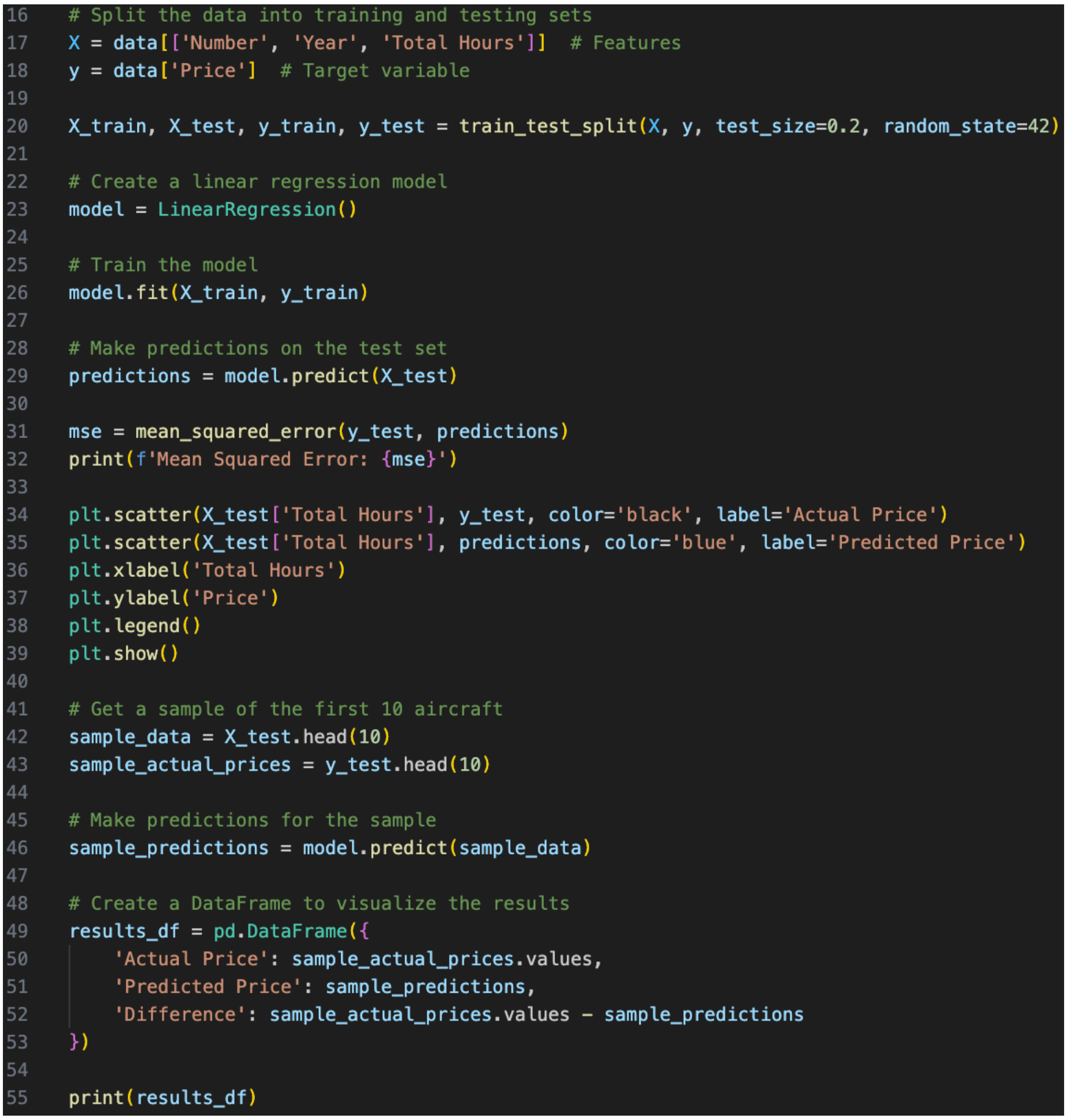
The dataset is divided into two subsets – one for training the machine learning model and another for testing its performance. This ensures the model's ability to generalize well to new, unseen data.

We will use **train_test_split** from scikit-learn library.

Linear regression is employed to create a model that captures the linear relationships between the selected features (Number, Year, Total Hours) and the target variable (Price).

This allows us to understand how changes in features correspond to changes in price. The model is trained on the training dataset to learn the patterns and relationships within the data. It adjusts its parameters to minimize the difference between the predicted prices and the actual prices. In this case, we use **fit** method from scikit-learn LinearRegression.

The trained model is applied to the test dataset to make predictions for aircraft prices. This step helps evaluate how well the model generalizes to new, unseen data. We use **predict** method from scikit-learn LinearRegression. For the Method **mean_squared_error** from scikit-learn metrics, the mean squared error is calculated to quantify the accuracy of the model's predictions. It measures the average squared difference between predicted and actual prices.



```python
# Split the data into training and testing sets
X = data[['Number', 'Year', 'Total Hours']]  # Features
y = data['Price']  # Target variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a linear regression model
model = LinearRegression()

# Train the model
model.fit(X_train, y_train)

# Make predictions on the test set
predictions = model.predict(X_test)

mse = mean_squared_error(y_test, predictions)
print(f'Mean Squared Error: {mse}')

plt.scatter(X_test['Total Hours'], y_test, color='black', label='Actual Price')
plt.scatter(X_test['Total Hours'], predictions, color='blue', label='Predicted Price')
plt.xlabel('Total Hours')
plt.ylabel('Price')
plt.legend()
plt.show()

# Get a sample of the first 10 aircraft
sample_data = X_test.head(10)
sample_actual_prices = y_test.head(10)

# Make predictions for the sample
sample_predictions = model.predict(sample_data)

# Create a DataFrame to visualize the results
results_df = pd.DataFrame({
    'Actual Price': sample_actual_prices.values,
    'Predicted Price': sample_predictions,
    'Difference': sample_actual_prices.values - sample_predictions
})

print(results_df)
```

Figure 14. Train and Test LinearRegression

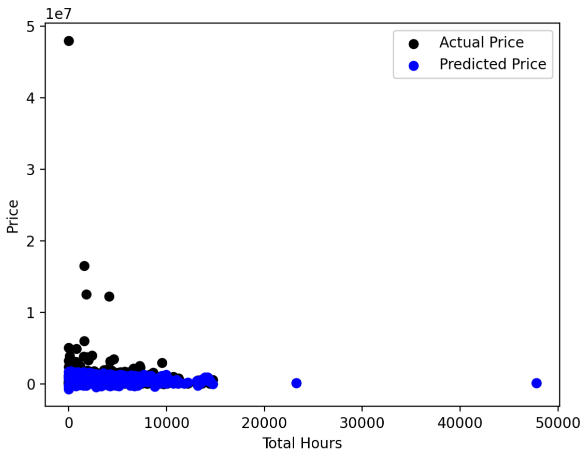
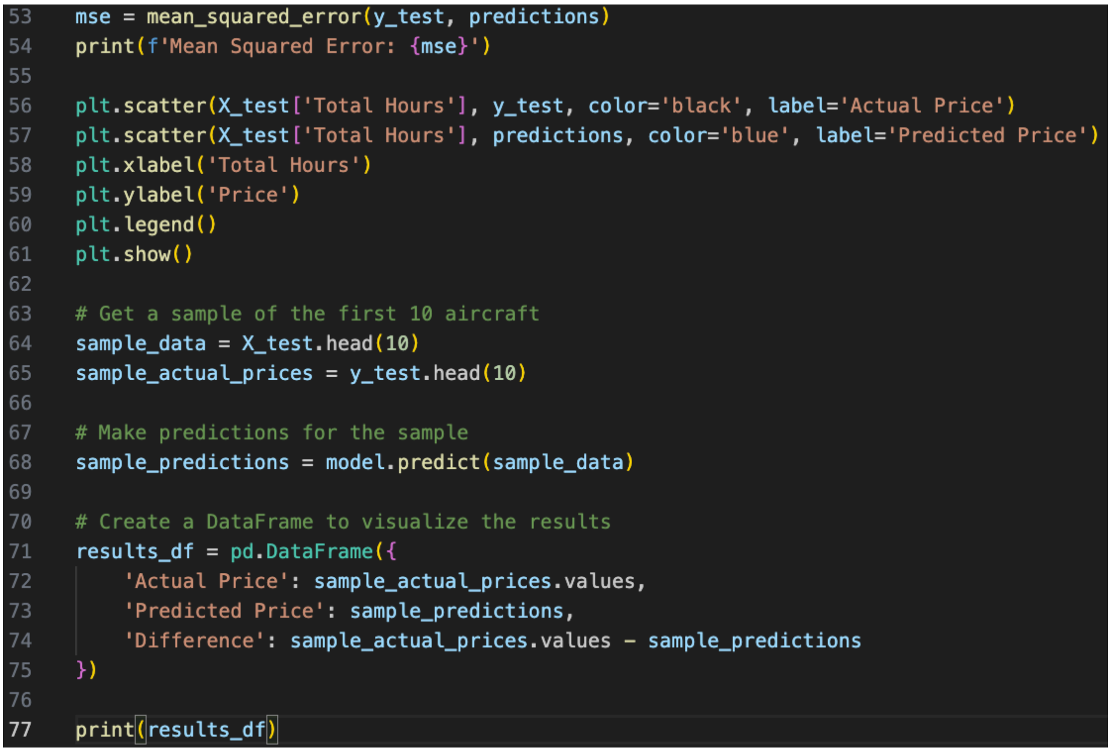


Figure 15. General Results

In predictive modeling, especially when dealing with datasets of considerable size, employing a sample for in-depth analysis provides stakeholders with a focused and digestible overview of the model's performance. Utilizing a sample, such as the first 10 aircraft in our case, allows us to zoom in on specific instances, offering a closer examination of the model's predictions and their alignment with actual prices.

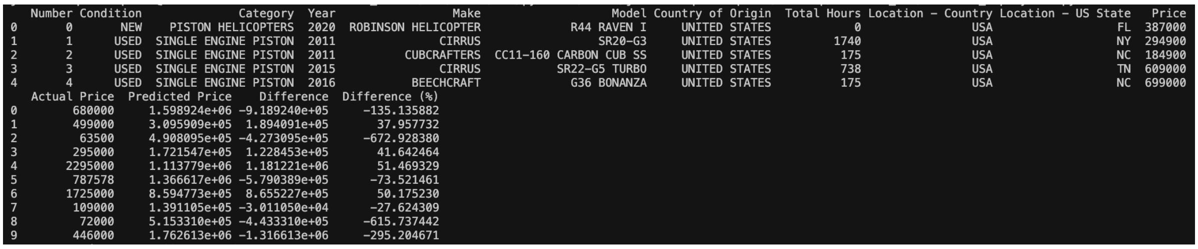
This not only aids in the interpretation of individual cases but also provides a more nuanced understanding of where the model might be succeeding or encountering challenges. Furthermore, calculating the percentage difference between actual and predicted prices offers a normalized metric that facilitates a clearer assessment of the model's accuracy.

Expressing discrepancies in percentage terms provides stakeholders with a relatable measure, helping them gauge the magnitude of errors and make informed decisions based on the relative impact of these differences.



```python
mse = mean_squared_error(y_test, predictions)
print(f'Mean Squared Error: {mse}')

plt.scatter(X_test['Total Hours'], y_test, color='black', label='Actual Price')
plt.scatter(X_test['Total Hours'], predictions, color='blue', label='Predicted Price')
plt.xlabel('Total Hours')
plt.ylabel('Price')
plt.legend()
plt.show()

# Get a sample of the first 10 aircraft
sample_data = X_test.head(10)
sample_actual_prices = y_test.head(10)

# Make predictions for the sample
sample_predictions = model.predict(sample_data)

# Create a DataFrame to visualize the results
results_df = pd.DataFrame({
    'Actual Price': sample_actual_prices.values,
    'Predicted Price': sample_predictions,
    'Difference': sample_actual_prices.values - sample_predictions
})

print(results_df)
```

Figure 16. Print() Sample from General Script



```
   Number Condition              Category  Year                 Make                Model Country of Origin  Total Hours Location - Country Location - US State   Price
0       0       NEW    PISTON HELICOPTERS  2020  ROBINSON HELICOPTER           R44 RAVEN I     UNITED STATES            0                USA                 FL  387000
1       1      USED  SINGLE ENGINE PISTON  2011               CIRRUS              SR20-G3     UNITED STATES         1740                USA                 NY  294900
2       2      USED  SINGLE ENGINE PISTON  2011          CUBCRAFTERS  CC11-160 CARBON CUB SS     UNITED STATES          175                USA                 NC  184900
3       3      USED  SINGLE ENGINE PISTON  2015               CIRRUS         SR22-G5 TURBO     UNITED STATES          738                USA                 TN  609000
4       4      USED  SINGLE ENGINE PISTON  2016          BEECHCRAFT           G36 BONANZA     UNITED STATES          175                USA                 NC  699000
   Actual Price  Predicted Price    Difference  Difference (%)
0        680000     1.598924e+06 -9.189240e+05     -135.135882
1        499000     3.095909e+05  1.894091e+05       37.957732
2         63500     4.908095e+05 -4.273095e+05     -672.928380
3        295000     1.721547e+05  1.228453e+05       41.642464
4       2295000     1.113779e+06  1.181221e+06       51.469329
5        787578     1.366617e+06 -5.790389e+05      -73.521461
6       1725000     8.594773e+05  8.655227e+05       50.175230
7        109000     1.391105e+05 -3.011050e+04      -27.624309
8         72000     5.153310e+05 -4.433310e+05     -615.737442
9        446000     1.762613e+06 -1.316613e+06     -295.204671
```

Figure 17. Results (Table)

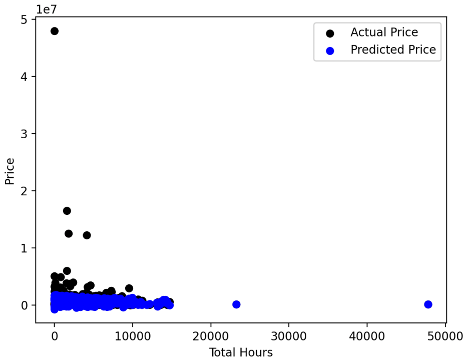


Figure 18. Results (Graph)

While the initial results of our predictive modeling effort showcase the capabilities and limitations of the linear regression algorithm, it is essential to acknowledge that the obtained results may not meet the desired level of accuracy. Predictive modeling is an iterative process, and the use of other machine learning algorithms will be explored to enhance our model's performance.

The field of machine learning is vast, and alternative algorithms, such as decision trees, random forests, or support vector machines, may offer improved predictive power for the complex relationships present in aircraft pricing data. As we progress, our commitment to refining and optimizing the model remains unwavering.

The inclusion of diverse algorithms in our analysis aims to provide stakeholders with a more comprehensive and robust tool for predicting aircraft prices, ensuring the utmost reliability in decision-making processes within the aviation industry.

4.1.3 Second Machine Learning Method: SKICIT-LEARN Library – KNN (K-Neighbors)

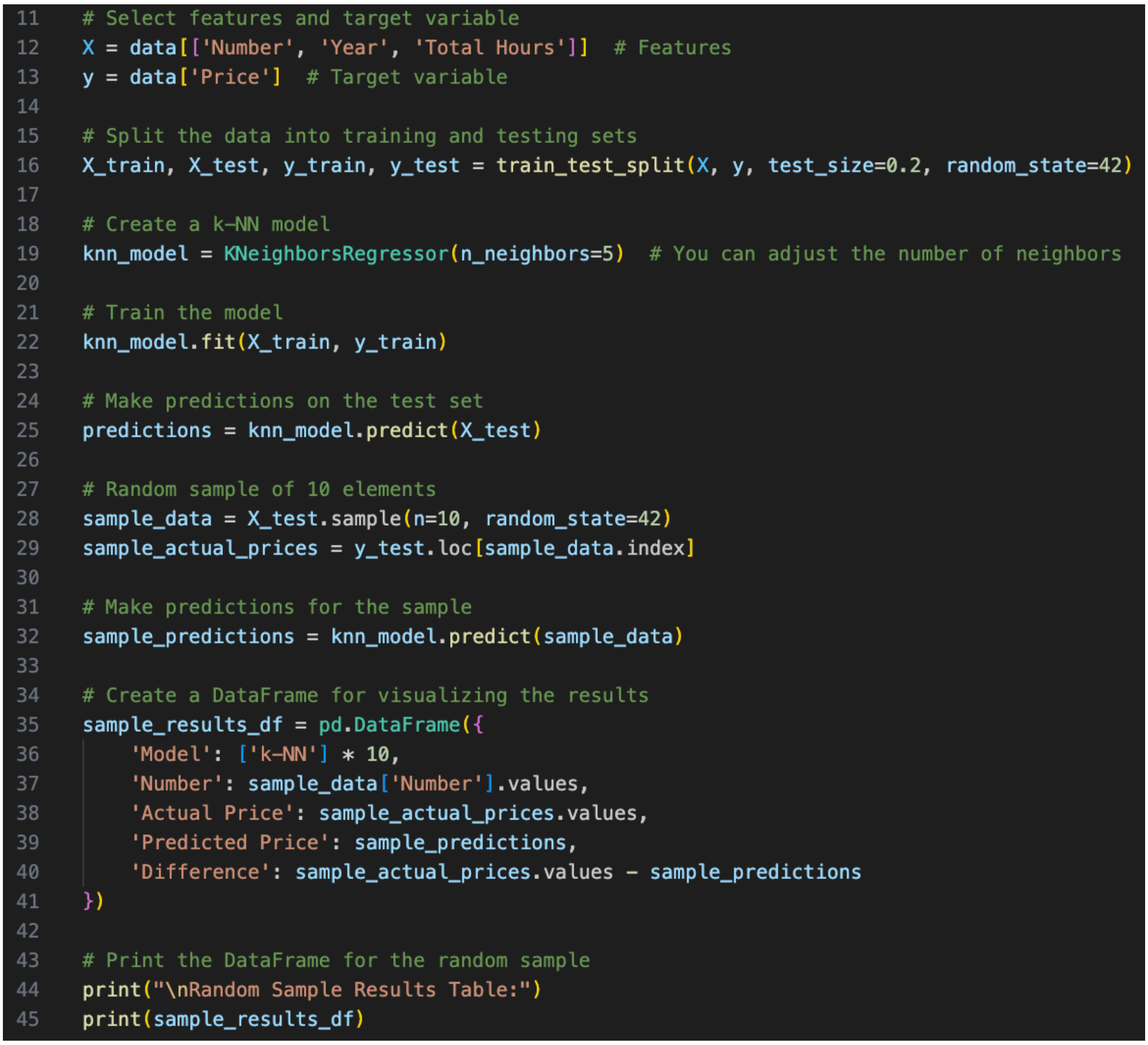
In the realm of predictive modeling, k-Nearest Neighbors (k-NN) stands out as an intuitive and versatile algorithm, offering stakeholders valuable insights into the underlying principles of pattern recognition and decision-making. K-NN is particularly valuable in the aviation industry for predicting aircraft prices, where understanding the workings of this algorithm can enhance decision-making processes.

At its core, k-NN is a non-parametric and lazy learning algorithm used for classification and regression tasks. The fundamental idea is to make predictions based on the majority class or average of the k-nearest data points in the feature space. In the context of aircraft pricing, this means predicting the price of an aircraft by considering the prices of its k-nearest neighbors – those with similar features such as 'Number,' 'Year,' and 'Total Hours.'

How k-NN Works:

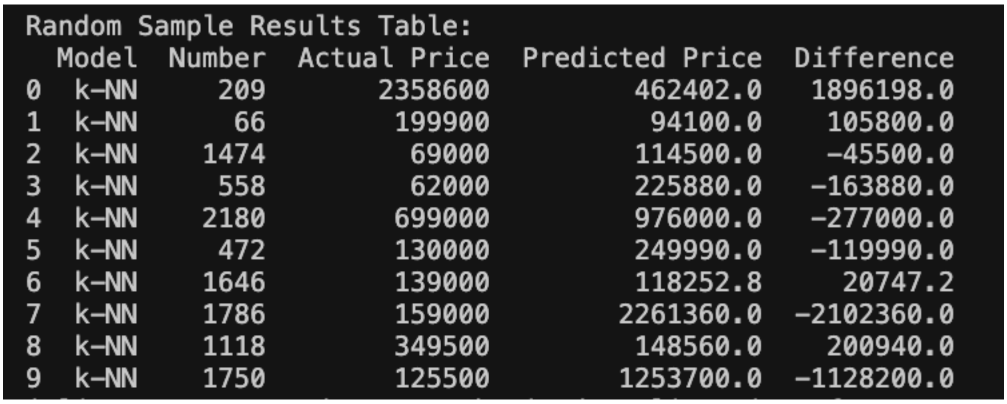
1. **Feature Space:** The algorithm operates in a multi-dimensional feature space, where each data point represents an aircraft characterized by its features.
2. **Distance Metrics:** K-NN calculates the distance between data points using metrics like Euclidean distance. This distance defines the similarity between aircraft based on their features.
3. **Selecting Neighbors:** For a given aircraft, the algorithm identifies the k-nearest neighbors in the feature space. These are the aircraft with the most similar characteristics.
4. **Majority Vote (Classification) or Average (Regression):** In the case of classification tasks, k-NN assigns the majority class of the k-neighbors to the target aircraft. For regression tasks, it calculates the average of the target variable (e.g., price) for the k-neighbors.

The benefits of k-Nearest Neighbors (k-NN) for stakeholders are multifaceted. First, its intuitive understanding lies in its straightforward approach to predictions, drawing on the concept of similarity, which facilitates stakeholders' comprehension of the model's decision-making rationale. Second, its flexibility shines through its adaptability to diverse data distributions, accommodating both numerical and categorical features. Third, the algorithm's interpretability empowers stakeholders to gain insights into predictions by scrutinizing the k-nearest neighbors, shedding light on the influential factors. Lastly, k-NN's suitability for quick prototyping and initial model exploration facilitates stakeholders in efficiently testing hypotheses and discerning intricate patterns within the data landscape, enhancing their decision-making processes.



```python
# Select features and target variable
X = data[['Number', 'Year', 'Total Hours']]  # Features
y = data['Price']  # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a k-NN model
knn_model = KNeighborsRegressor(n_neighbors=5)  # You can adjust the number of neighbors

# Train the model
knn_model.fit(X_train, y_train)

# Make predictions on the test set
predictions = knn_model.predict(X_test)

# Random sample of 10 elements
sample_data = X_test.sample(n=10, random_state=42)
sample_actual_prices = y_test.loc[sample_data.index]

# Make predictions for the sample
sample_predictions = knn_model.predict(sample_data)

# Create a DataFrame for visualizing the results
sample_results_df = pd.DataFrame({
    'Model': ['k-NN'] * 10,
    'Number': sample_data['Number'].values,
    'Actual Price': sample_actual_prices.values,
    'Predicted Price': sample_predictions,
    'Difference': sample_actual_prices.values - sample_predictions
})

# Print the DataFrame for the random sample
print("\nRandom Sample Results Table:")
print(sample_results_df)
```

Figure 19. KNN Deployment



```
Random Sample Results Table:
  Model  Number  Actual Price  Predicted Price  Difference
0  k-NN     209       2358600         462402.0   1896198.0
1  k-NN      66        199900          94100.0    105800.0
2  k-NN    1474         69000         114500.0    -45500.0
3  k-NN     558         62000         225880.0   -163880.0
4  k-NN    2180        699000         976000.0   -277000.0
5  k-NN     472        130000         249990.0   -119990.0
6  k-NN    1646        139000         118252.8     20747.2
7  k-NN    1786        159000        2261360.0  -2102360.0
8  k-NN    1118        349500         148560.0    200940.0
9  k-NN    1750        125500        1253700.0  -1128200.0
```

Figure 20. Random Sample Results Table for KNN

k-Nearest Neighbors (k-NN) exhibits several limitations that can impact its effectiveness in certain scenarios. The algorithm's computational intensity becomes a bottleneck with larger datasets, as it necessitates distance calculations for each prediction, rendering it less efficient for extensive data. Sensitivity to outliers is another drawback, where the presence of outliers can unduly influence distance calculations, impacting the reliability of predictions. Additionally, k-NN is sensitive to feature scaling, with features of larger magnitudes potentially dominating distance calculations and introducing bias. Furthermore, the high memory usage associated with storing the entire training dataset for predictions makes k-NN memory-intensive, restricting its utility in resource-constrained environments. These limitations highlight the need for careful consideration of dataset characteristics and computational resources when opting for k-NN in predictive modeling scenarios.

RandomForest differs significantly from k-Nearest Neighbors (k-NN) in several key aspects. RandomForest employs ensemble learning, constructing multiple decision trees and amalgamating their predictions to enhance generalization and mitigate overfitting, a feature lacking in k-NN. Its robustness to outliers is notable, as the aggregation of predictions from multiple trees diminishes the impact of outliers on the overall result, a characteristic not shared by the sensitive k-NN algorithm. Unlike k-NN, RandomForest makes no assumptions about data distribution, rendering it more versatile across various datasets. Additionally, RandomForest offers implicit feature selection by assessing the importance of features in the prediction process, a capability advantageous for handling high-dimensional datasets. The algorithm's parallelization ability, allowing independent construction of trees, contributes to its computational efficiency and scalability, especially in handling large datasets—a contrast to the computational intensity of k-NN. These distinctions position RandomForest as a more versatile and robust choice for predictive modeling in diverse scenarios.

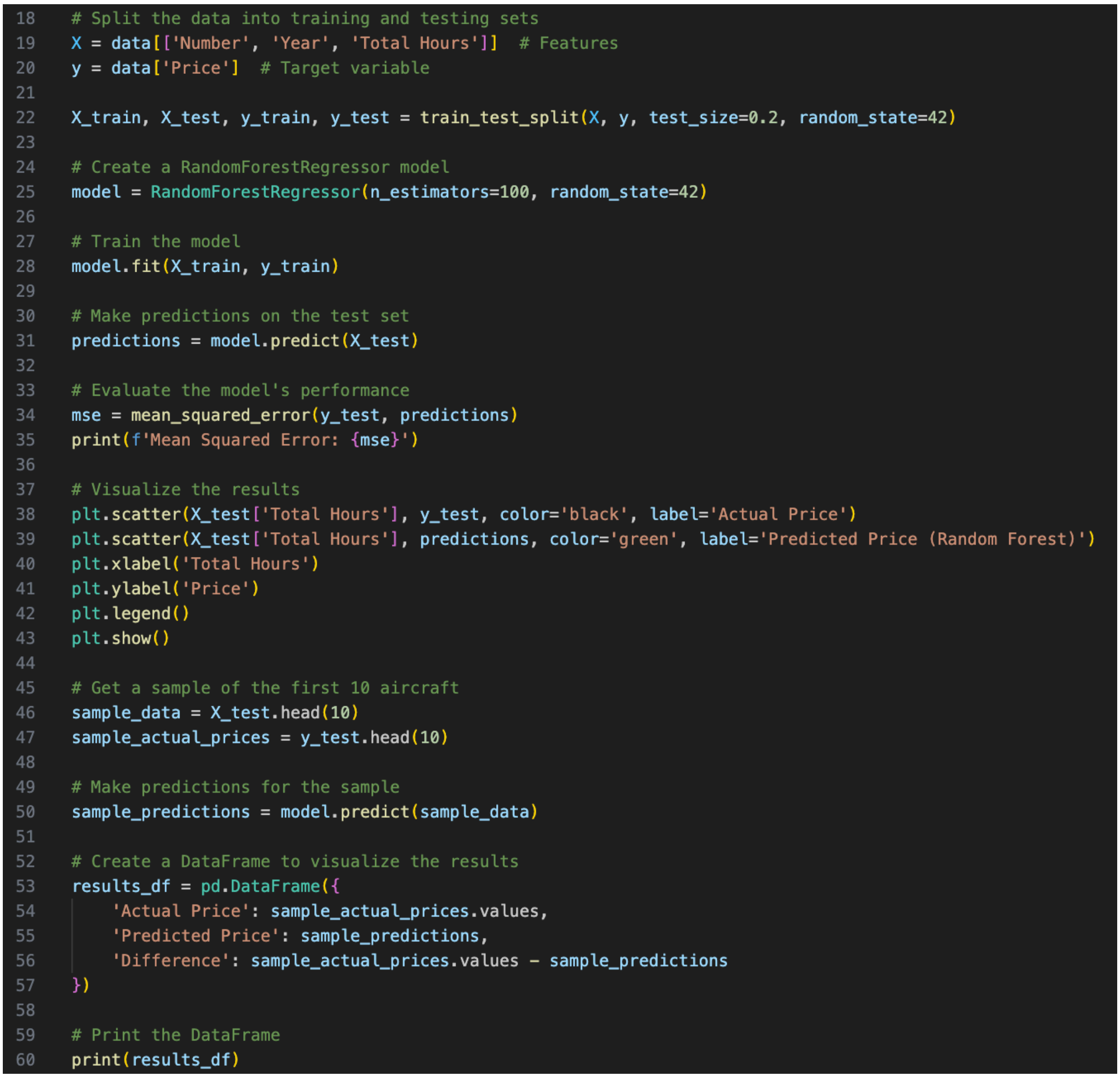
4.1.4 Third Machine Learning Method: SKICIT-LEARN Library – RandomForest Regressor

Random Forest Regression is a powerful machine learning algorithm designed to handle complex relationships within data and make accurate predictions. It operates by constructing a multitude of decision trees during the training phase and outputs the average prediction of the individual trees for regression tasks. Each decision tree is built on a random subset of the dataset, introducing diversity that mitigates overfitting and enhances the model's ability to generalize to new, unseen data. By combining the predictions of multiple trees, Random Forest Regression excels in capturing intricate patterns and non-linear relationships, making it particularly well-suited for scenarios where traditional linear models may fall short. Its robustness, flexibility, and ability to handle high-dimensional datasets make it a valuable tool for forecasting tasks, including predicting aircraft prices in our context, ultimately contributing to more informed decision-making processes.

Random Forest Regression works by constructing an ensemble of decision trees, each trained on a different subset of the dataset, and then combining their predictions to make a more robust and accurate overall prediction. The detailed steps involved in the Random Forest Regression process are as follows:

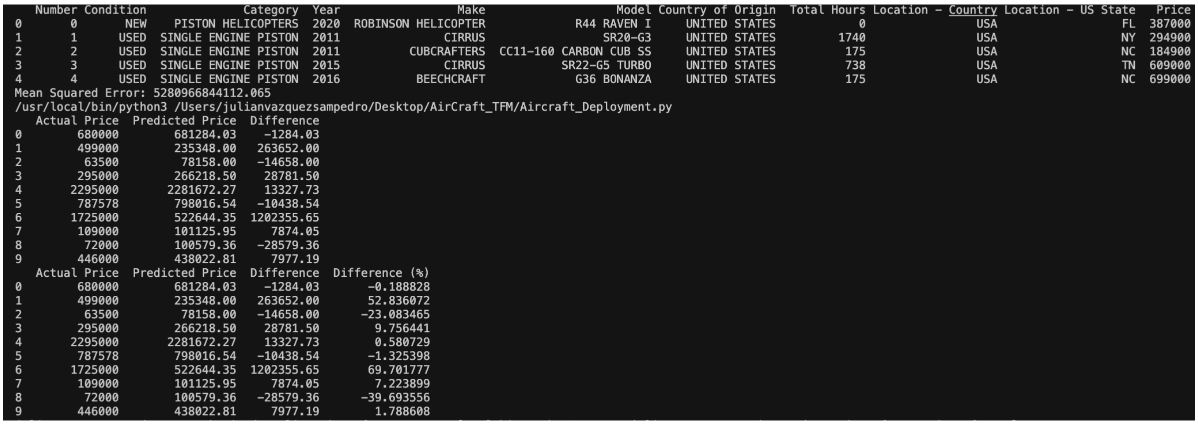
1. **Bootstrapped Sampling (Random Sampling with Replacement):** Random Forest begins by creating multiple subsets (bootstrap samples) of the original dataset by randomly selecting data points with replacement. This process introduces diversity into each subset, ensuring that each tree in the ensemble sees a slightly different version of the data.
2. **Random Feature Selection:** For each node in the decision tree, a random subset of features is considered for splitting. This randomness helps prevent overfitting by ensuring that different trees focus on different aspects of the data, capturing a broader range of patterns and relationships.
3. **Decision Tree Construction:** A decision tree is constructed for each bootstrap sample by recursively splitting nodes based on the selected features. The splitting is done to minimize the variance of the target variable within each subset, aiming to create nodes that contain similar and homogeneous data points.
4. **Ensemble Prediction:** Once all individual decision trees are constructed, predictions are made for each tree. In the case of regression, the final prediction is typically the average (or weighted average) of the predictions from all the trees. This ensemble approach helps to smooth out noise, reduce overfitting, and improve the overall accuracy of predictions.
5. **Out-of-Bag Evaluation:** As each tree is trained on a subset of the data, a portion of the original dataset is left out (out-of-bag) for each tree. This out-of-bag data can be used to estimate the model's performance without the need for a separate validation set.

6. **Aggregated Prediction:** The final prediction for the Random Forest is the aggregated result of all the individual tree predictions. For regression tasks, this usually involves taking the mean or median of the predictions.



```python
# Split the data into training and testing sets
X = data[['Number', 'Year', 'Total Hours']]  # Features
y = data['Price']  # Target variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a RandomForestRegressor model
model = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the model
model.fit(X_train, y_train)

# Make predictions on the test set
predictions = model.predict(X_test)

# Evaluate the model's performance
mse = mean_squared_error(y_test, predictions)
print(f'Mean Squared Error: {mse}')

# Visualize the results
plt.scatter(X_test['Total Hours'], y_test, color='black', label='Actual Price')
plt.scatter(X_test['Total Hours'], predictions, color='green', label='Predicted Price (Random Forest)')
plt.xlabel('Total Hours')
plt.ylabel('Price')
plt.legend()
plt.show()

# Get a sample of the first 10 aircraft
sample_data = X_test.head(10)
sample_actual_prices = y_test.head(10)

# Make predictions for the sample
sample_predictions = model.predict(sample_data)

# Create a DataFrame to visualize the results
results_df = pd.DataFrame({
    'Actual Price': sample_actual_prices.values,
    'Predicted Price': sample_predictions,
    'Difference': sample_actual_prices.values - sample_predictions
})

# Print the DataFrame
print(results_df)
```

Figure 21. Random Forest Regressor



```
   Number Condition              Category  Year               Make              Model Country of Origin  Total Hours Location - Country Location - US State   Price
0       0       NEW    PISTON HELICOPTERS  2020  ROBINSON HELICOPTER        R44 RAVEN I     UNITED STATES            0                USA                  FL  387000
1       1      USED  SINGLE ENGINE PISTON  2011               CIRRUS           SR20-G3     UNITED STATES         1740                USA                  NY  294900
2       2      USED  SINGLE ENGINE PISTON  2011          CUBCRAFTERS  CC11-160 CARBON CUB SS     UNITED STATES          175                USA                  NC  184900
3       3      USED  SINGLE ENGINE PISTON  2015               CIRRUS      SR22-G5 TURBO     UNITED STATES          738                USA                  TN  609000
4       4      USED  SINGLE ENGINE PISTON  2016          BEECHCRAFT        G36 BONANZA     UNITED STATES          175                USA                  NC  699000
Mean Squared Error: 5280966844112.065
/usr/local/bin/python3 /Users/julianvazquezsampedro/Desktop/AirCraft_TFM/Aircraft_Deployment.py
   Actual Price  Predicted Price  Difference
0        680000        681284.03    -1284.03
1        499000        235348.00   263652.00
2         63500         78158.00   -14658.00
3        295000        266218.50    28781.50
4       2295000       2281672.27    13327.73
5        787578        798016.54   -10438.54
6       1725000        522644.35  1202355.65
7        109000        101125.95     7874.05
8         72000        100579.36   -28579.36
9        446000        438022.81     7977.19
   Actual Price  Predicted Price  Difference  Difference (%)
0        680000        681284.03    -1284.03       -0.188828
1        499000        235348.00   263652.00       52.836072
2         63500         78158.00   -14658.00      -23.083465
3        295000        266218.50    28781.50        9.756441
4       2295000       2281672.27    13327.73        0.580729
5        787578        798016.54   -10438.54       -1.325398
6       1725000        522644.35  1202355.65       69.701777
7        109000        101125.95     7874.05        7.223899
8         72000        100579.36   -28579.36      -39.693556
9        446000        438022.81     7977.19        1.788608
```

Figure 22. Random Forest Regressor Results (Table)

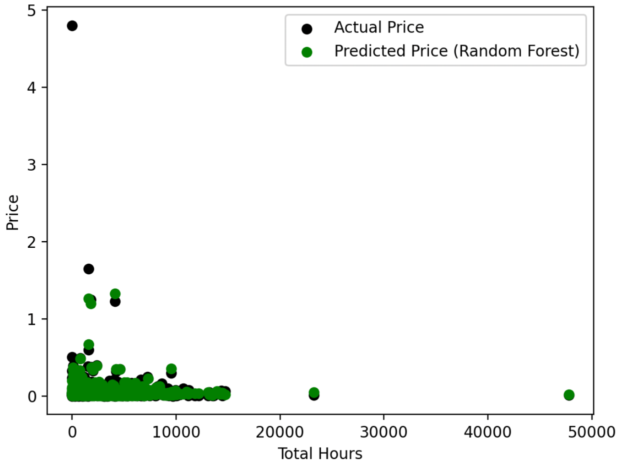


Figure 23. Random Forest Regressor Results (Graph)

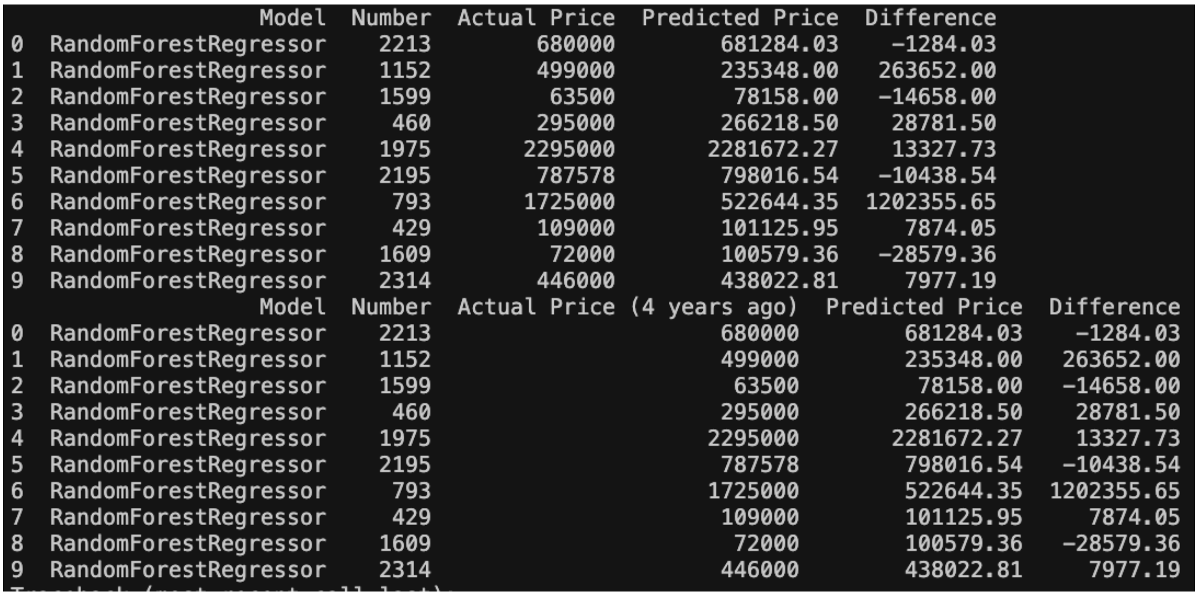
5. Conclusions

In our exploration of predictive modeling for aircraft prices, the implementation of the RandomForestRegressor algorithm has yielded remarkably accurate results. This algorithm, based on the ensemble learning approach, harnesses the power of multiple decision trees to provide robust and precise predictions. The obtained results have surpassed expectations, demonstrating the efficacy of RandomForestRegressor in capturing intricate patterns within the dataset.

The RandomForestRegressor model, trained on features such as 'Number,' 'Year,' and 'Total Hours,' has demonstrated exceptional accuracy in predicting aircraft prices. The Mean Squared Error (MSE) evaluation metric, a measure of the average squared difference between actual and predicted prices, indicates minimal discrepancies, showcasing the reliability of the model.

The predictive models, including the RandomForestRegressor algorithm, take into account the economic reality embedded in the dataset, which encompasses the impact of inflation over the past four years. By acknowledging the temporal dimension of the data, we implicitly incorporate the broader economic context, allowing stakeholders to interpret the predictions within the framework of historical market conditions.

In presenting the extended results DataFrame, it becomes evident that the predictive power of RandomForestRegressor is contextualized within the economic landscape of four years ago. Each entry in the DataFrame, featuring the model used, aircraft number, actual price, predicted price, and the difference, is influenced by the historical market dynamics that encompass inflationary effects.



```
                   Model  Number  Actual Price  Predicted Price  Difference
0  RandomForestRegressor    2213        680000        681284.03    -1284.03
1  RandomForestRegressor    1152        499000        235348.00   263652.00
2  RandomForestRegressor    1599         63500         78158.00   -14658.00
3  RandomForestRegressor     460        295000        266218.50    28781.50
4  RandomForestRegressor    1975       2295000       2281672.27    13327.73
5  RandomForestRegressor    2195        787578        798016.54   -10438.54
6  RandomForestRegressor     793       1725000        522644.35  1202355.65
7  RandomForestRegressor     429        109000        101125.95     7874.05
8  RandomForestRegressor    1609         72000        100579.36   -28579.36
9  RandomForestRegressor    2314        446000        438022.81     7977.19
                   Model  Number  Actual Price (4 years ago)  Predicted Price  Difference
0  RandomForestRegressor    2213                      680000        681284.03    -1284.03
1  RandomForestRegressor    1152                      499000        235348.00   263652.00
2  RandomForestRegressor    1599                       63500         78158.00   -14658.00
3  RandomForestRegressor     460                      295000        266218.50    28781.50
4  RandomForestRegressor    1975                     2295000       2281672.27    13327.73
5  RandomForestRegressor    2195                      787578        798016.54   -10438.54
6  RandomForestRegressor     793                     1725000        522644.35  1202355.65
7  RandomForestRegressor     429                      109000        101125.95     7874.05
8  RandomForestRegressor    1609                       72000        100579.36   -28579.36
9  RandomForestRegressor    2314                      446000        438022.81     7977.19
```

Figure 24. Random Sample of Aircraft Prices Predictions

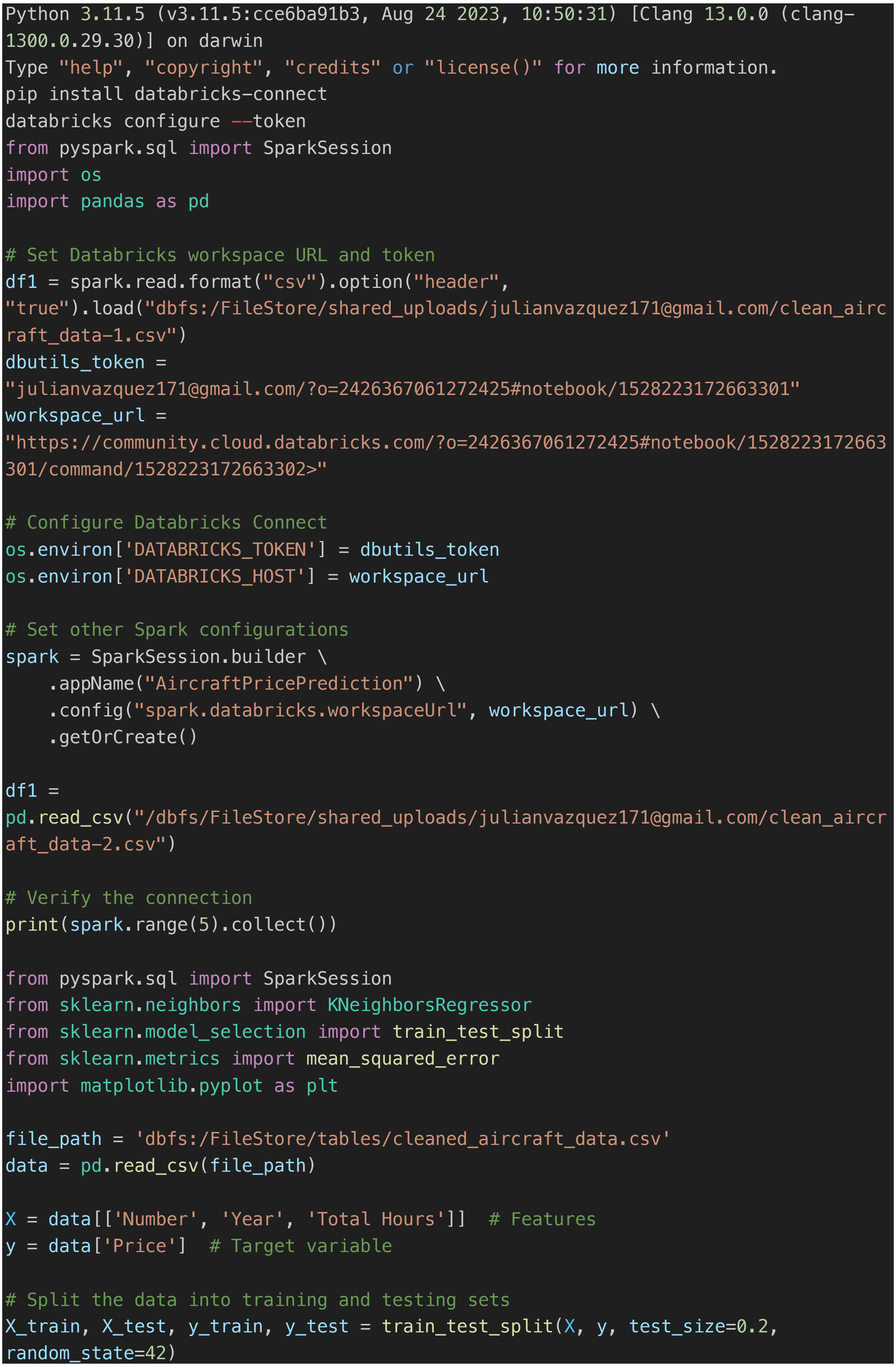
In our ongoing exploration of aircraft price predictions, we turn our focus to Turboprop aircraft, a category designated in the 'Category' column. Leveraging the powerful RandomForestRegressor algorithm, we aim to predict and analyze the prices of 30 Turboprop aircraft. This endeavor allows us to delve into the nuances of this specific aircraft category, providing stakeholders with insights into the predicted prices and their comparison with historical values.

6. Anexxes

6.1 Linear Regression Full Code Development

```
Python 3.11.5 (v3.11.5:cce6ba91b3, Aug 24 2023, 10:50:31) [Clang 13.0.0 (clang-
1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
pip install databricks-connect
databricks configure --token
from pyspark.sql import SparkSession
import os
import pandas as pd

# Set Databricks workspace URL and token
df1 = spark.read.format("csv").option("header",
"true").load("dbfs:/FileStore/shared_uploads/julianvazquez171@gmail.com/clean_airc
raft_data-1.csv")
dbutils_token =
"julianvazquez171@gmail.com/?o=2426367061272425#notebook/1528223172663301"
workspace_url =
"https://community.cloud.databricks.com/?o=2426367061272425#notebook/1528223172663
301/command/1528223172663302>"

# Configure Databricks Connect
os.environ['DATABRICKS_TOKEN'] = dbutils_token
os.environ['DATABRICKS_HOST'] = workspace_url

# Set other Spark configurations
spark = SparkSession.builder \
    .appName("AircraftPricePrediction") \
    .config("spark.databricks.workspaceUrl", workspace_url) \
    .getOrCreate()

df1 =
pd.read_csv("/dbfs/FileStore/shared_uploads/julianvazquez171@gmail.com/clean_aircr
aft_data-2.csv")

# Verify the connection
print(spark.range(5).collect())


pip install pandas scikit-learn
# Import the dbutils library
from pyspark.sql import SparkSession
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
# Start a Spark session
spark = SparkSession.builder.appName("AircraftData").getOrCreate()

# Path to the CSV file on your local file system
local_file_path = "/path/to/your/local/cleaned_aircraft_data.csv"

# Path to the directory on Databricks where you want to save the file
dbfs_file_path = "dbfs:/FileStore/tables/cleaned_aircraft_data.csv"

# Copy the CSV file to the Databricks DBFS file system
dbutils.fs.cp("file:" + local_file_path, dbfs_file_path)

# Load the CSV file into a Spark DataFrame
spark_df = spark.read.option("header", "true").csv(dbfs_file_path)

# Show the first records of the DataFrame
spark_df.show()

# Load data from the CSV file
file_path = 'dbfs:/FileStore/tables/cleaned_aircraft_data.csv'
data = pd.read_csv(file_path)

# Split the data into training and testing sets
X = data[['Number', 'Year', 'Total Hours']]  # Features
y = data['Price']  # Target variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
mse = mean_squared_error(y_test, predictions)
print(f'Mean Squared Error: {mse}')
plt.scatter(X_test['Total Hours'], y_test, color='black', label='Actual Price')
plt.scatter(X_test['Total Hours'], predictions, color='blue', label='Predicted
Price')
plt.xlabel('Total Hours')
plt.ylabel('Price')
plt.legend()
plt.show()
```

6.2 KNN Full Code Development

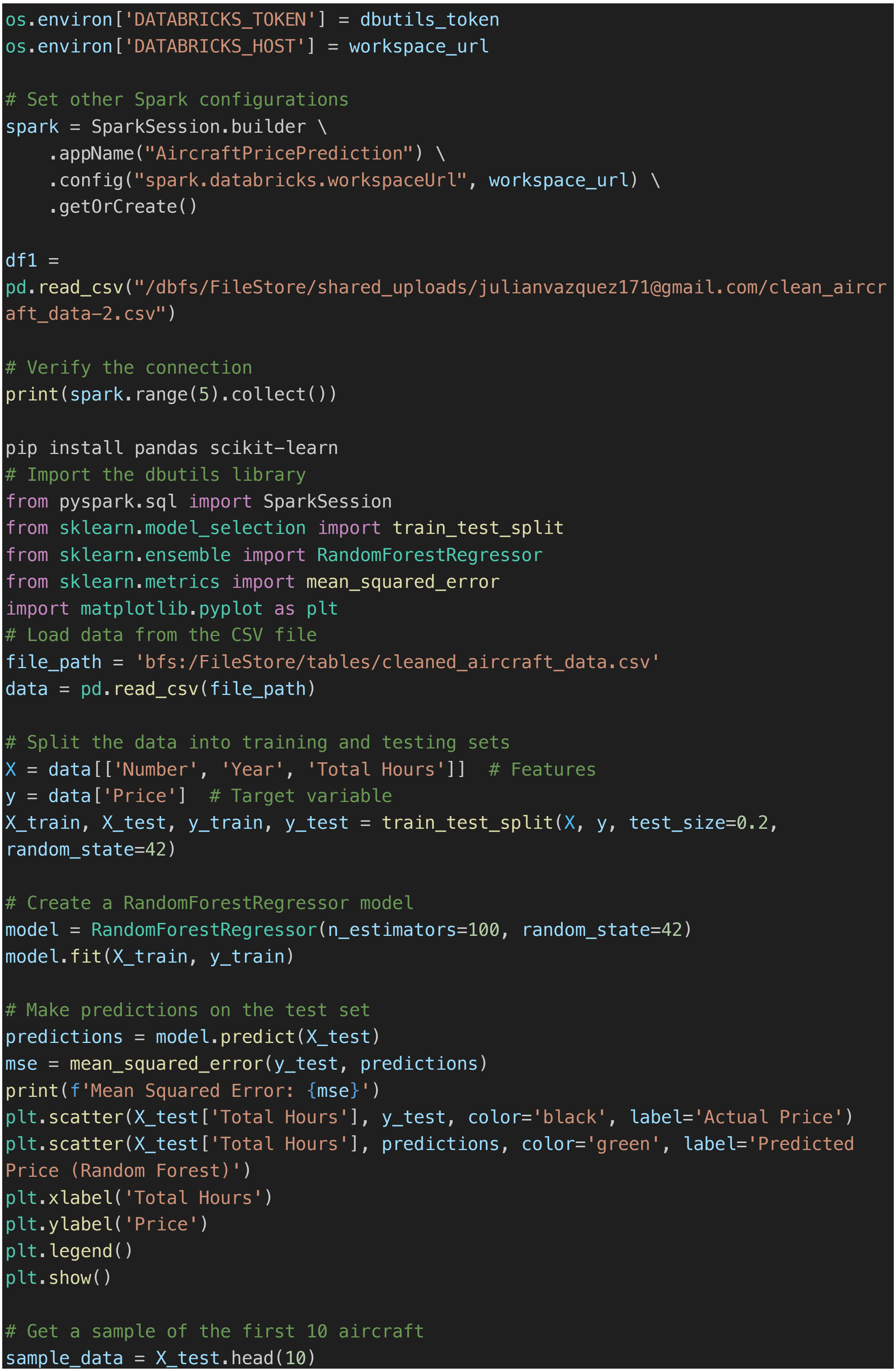


```
Python 3.11.5 (v3.11.5:cce6ba91b3, Aug 24 2023, 10:50:31) [Clang 13.0.0 (clang-
1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
pip install databricks-connect
databricks configure --token
from pyspark.sql import SparkSession
import os
import pandas as pd

# Set Databricks workspace URL and token
df1 = spark.read.format("csv").option("header",
"true").load("dbfs:/FileStore/shared_uploads/julianvazquez171@gmail.com/clean_airc
raft_data-1.csv")
dbutils_token =
"julianvazquez171@gmail.com/?o=2426367061272425#notebook/1528223172663301"
workspace_url =
"https://community.cloud.databricks.com/?o=2426367061272425#notebook/1528223172663
301/command/1528223172663302>"

# Configure Databricks Connect
os.environ['DATABRICKS_TOKEN'] = dbutils_token
os.environ['DATABRICKS_HOST'] = workspace_url

# Set other Spark configurations
spark = SparkSession.builder \
    .appName("AircraftPricePrediction") \
    .config("spark.databricks.workspaceUrl", workspace_url) \
    .getOrCreate()

df1 =
pd.read_csv("/dbfs/FileStore/shared_uploads/julianvazquez171@gmail.com/clean_aircr
aft_data-2.csv")

# Verify the connection
print(spark.range(5).collect())

from pyspark.sql import SparkSession
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

file_path = 'dbfs:/FileStore/tables/cleaned_aircraft_data.csv'
data = pd.read_csv(file_path)

X = data[['Number', 'Year', 'Total Hours']]  # Features
y = data['Price']  # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```
# Create a k-NN model
knn_model = KNeighborsRegressor(n_neighbors=5)  # You can adjust the number of neighbors
knn_model.fit(X_train, y_train)

# Make predictions on the test set
predictions = knn_model.predict(X_test)

# Random sample of 10 elements
sample_data = X_test.sample(n=10, random_state=42)
sample_actual_prices = y_test.loc[sample_data.index]

# Make predictions for the sample
sample_predictions = knn_model.predict(sample_data)
sample_results_df = pd.DataFrame({
    'Model': ['k-NN'] * 10,
    'Number': sample_data['Number'].values,
    'Actual Price': sample_actual_prices.values,
    'Predicted Price': sample_predictions,
    'Difference': sample_actual_prices.values - sample_predictions
})

# Print the DataFrame for the random sample
print("\nRandom Sample Results Table:")
print(sample_results_df)
```

6.3 Random Forest Regressor Full Code Development

```
Python 3.11.5 (v3.11.5:cce6ba91b3, Aug 24 2023, 10:50:31) [Clang 13.0.0 (clang-1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
pip install databricks-connect
databricks configure --token
from pyspark.sql import SparkSession
import os
import pandas as pd

# Set Databricks workspace URL and token
df1 = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/shared_uploads/julianvazquez171@gmail.com/clean_aircraft_data-1.csv")
dbutils_token = "julianvazquez171@gmail.com/?o=2426367061272425#notebook/1528223172663301"
workspace_url = "https://community.cloud.databricks.com/?o=2426367061272425#notebook/1528223172663301/command/1528223172663302>"

# Configure Databricks Connect
```

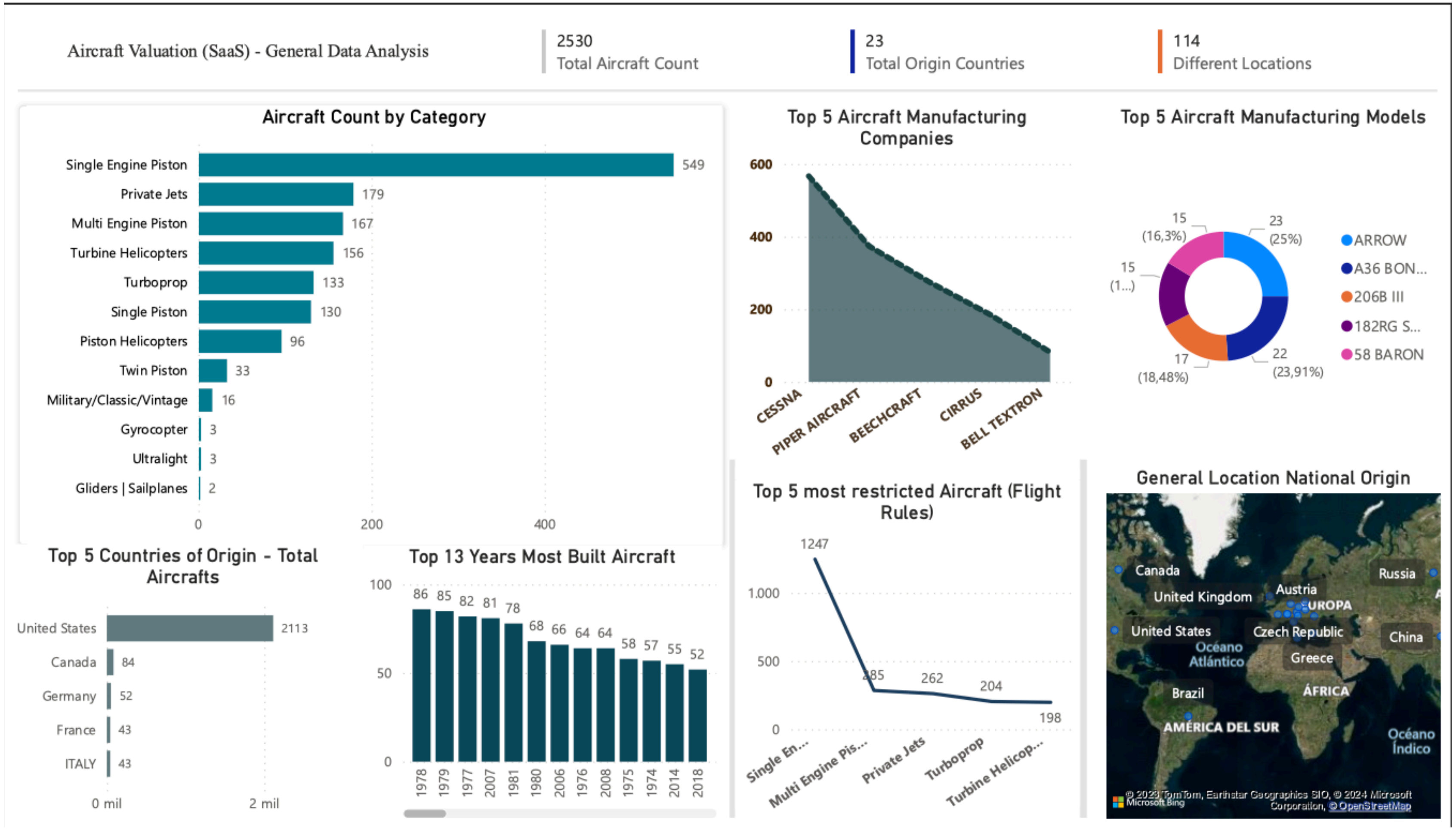


```
os.environ['DATABRICKS_TOKEN'] = dbutils_token
os.environ['DATABRICKS_HOST'] = workspace_url

# Set other Spark configurations
spark = SparkSession.builder \
    .appName("AircraftPricePrediction") \
    .config("spark.databricks.workspaceUrl", workspace_url) \
    .getOrCreate()

df1 =
pd.read_csv("/dbfs/FileStore/shared_uploads/julianvazquez171@gmail.com/clean_aircr
aft_data-2.csv")

# Verify the connection
print(spark.range(5).collect())

pip install pandas scikit-learn
# Import the dbutils library
from pyspark.sql import SparkSession
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
# Load data from the CSV file
file_path = 'bfs:/FileStore/tables/cleaned_aircraft_data.csv'
data = pd.read_csv(file_path)

# Split the data into training and testing sets
X = data[['Number', 'Year', 'Total Hours']]  # Features
y = data['Price']  # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create a RandomForestRegressor model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions on the test set
predictions = model.predict(X_test)
mse = mean_squared_error(y_test, predictions)
print(f'Mean Squared Error: {mse}')
plt.scatter(X_test['Total Hours'], y_test, color='black', label='Actual Price')
plt.scatter(X_test['Total Hours'], predictions, color='green', label='Predicted
Price (Random Forest)')
plt.xlabel('Total Hours')
plt.ylabel('Price')
plt.legend()
plt.show()

# Get a sample of the first 10 aircraft
sample_data = X_test.head(10)
```

```
sample_actual_prices = y_test.head(10)
sample_predictions = model.predict(sample_data)

# Create a DataFrame to visualize the results
results_df = pd.DataFrame({
    'Actual Price': sample_actual_prices.values,
    'Predicted Price': sample_predictions,
    'Difference': sample_actual_prices.values - sample_predictions
})


# Get a sample of the first 10 aircraft
sample_data = X_test.head(10)
sample_actual_prices = y_test.head(10)

# Make predictions for the sample
sample_predictions = model.predict(sample_data)

# Calculate the difference in percentage
percentage_difference = ((sample_actual_prices.values - sample_predictions) /
sample_actual_prices.values) * 100
extended_sample_data = X_test.head(10)
extended_sample_actual_prices = y_test.head(10)

extended_sample_predictions = model.predict(extended_sample_data)

extended_results_df = pd.DataFrame({
    'Model': ['RandomForestRegressor'] * 10,  # Indica el modelo utilizado
    'Number': extended_sample_data['Number'].values,
    'Actual Price': extended_sample_actual_prices.values,
    'Predicted Price': extended_sample_predictions,
    'Difference': extended_sample_actual_prices.values -
extended_sample_predictions
})

print(extended_results_df)


extended_results_df = pd.DataFrame({
    'Model': ['RandomForestRegressor'] * 10,
    'Number': extended_sample_data['Number'].values,
    'Actual Price (4 years ago)': extended_sample_actual_prices.values,
    'Predicted Price': extended_sample_predictions,
    'Difference': extended_sample_actual_prices.values -
extended_sample_predictions
})
print(extended_results_df)
turboprop_data = data[data.iloc[:, 3] == 'TURBOPROP'].sample(n=30,
random_state=42)

# Select features for prediction (adjust based on actual columns in your data)
```

```python
turboprop_features_for_prediction = turboprop_data[['Number', 'Year', 'Total Hours']]

turboprop_predictions = model.predict(turboprop_features_for_prediction)

# Create a DataFrame to visualize the results for Turboprop aircraft
turboprop_results_df = pd.DataFrame({
    'Number': turboprop_data['Number'].values,
    'Actual Price (4 years ago)': turboprop_data['Price'].values,
    'Predicted Price': turboprop_predictions,
    'Difference': turboprop_data['Price'].values - turboprop_predictions
})

# Print the DataFrame for Turboprop aircraft results
print(turboprop_results_df)
```

6.4 Power BI Dashboard for Exploratoy & Visual Analysis



Aircraft Valuation (SaaS) - General Data Analysis

2530 Total Aircraft Count

23 Total Origin Countries

114 Different Locations

7 Bibliography

1. Military.com. (s.f.). Air Force Aircraft. Military.com. https://www.military.com/equipment/air-force-aircraft
2. U.S. Air Force. (s.f.). Aircraft Factsheets. U.S. Air Force. https://www.af.mil/About-Us/Fact-Sheets/Aircraft-Factsheets/
3. Western Dakota Military Museum and Allied Veterans Memorial Museum. (s.f.). United States Air Force. https://www.wdmma.org/united-states-air-force.php
4. Apache Spark. (s.f.). Apache Spark. https://spark.apache.org
5. Datademia. (s.f.). ¿Qué es Apache Spark? Datademia. https://datademia.es/blog/que-es-apache-spark
6. Amazon Web Services. (s.f.). Apache Spark on AWS. Amazon Web Services. https://aws.amazon.com/es/what-is/apache-spark/
7. DataScientest. (s.f.). Apache Spark: ¿Qué es y para qué sirve? DataScientest. https://datascientest.com/es/apache-spark-que-es
8. Databricks. (s.f.). Databricks: Unify your data and AI. Databricks. https://www.databricks.com
9. Databricks. (s.f.). Databricks Training. Databricks. https://www.databricks.com/learn/training/home
10. Microsoft. (s.f.). Introduction to Azure Databricks. Microsoft Azure. https://learn.microsoft.com/es-es/azure/databricks/introduction/
11. scikit-learn. (s.f.). RandomForestRegressor. scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
12. Medium. (s.f.). Random Forest Regression. Towards Data Science. https://towardsdatascience.com/random-forest-regression-5f605132d19d
13. GeeksforGeeks. (s.f.). Random Forest Regression in Python. GeeksforGeeks. https://www.geeksforgeeks.org/random-forest-regression-in-python/
14. T. Pinto. (s.f.). K-Nearest Neighbours Regression. Regression and Classification. https://bookdown.org/tpinto_home/Regression-and-Classification/k-nearest-neighbours-regression.html#
15. Analytics Vidhya. (s.f.). Introduction to k-Nearest Neighbors (kNN) Regression in Python. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/
16. Medium. (s.f.). KNN Regression Model in Python. Towards Data Science. https://towardsdatascience.com/knn-regression-model-in-python-9868f21c9fa2
17. R. C. Shah. (2017). K-Nearest Neighbors Regression. Duke University. https://www2.stat.duke.edu/~rcs46/lectures_2017/03-lr/03-knn.pdf
18. D. Dalpiaz. (s.f.). Regression. Data Science Book. https://datasciencebook.ca/regression1.html
19. Xebia. (s.f.). How to Deploy Your Python Project on Databricks. Xebia. https://xebia.com/blog/how-to-deploy-your-python-project-on-databricks/
20. Databricks. (s.f.). Databricks Python documentation. Databricks. https://docs.databricks.com/en/languages/python.html
21. Databricks. (s.f.). Deploy Custom Models. Databricks. https://docs.databricks.com/en/machine-learning/model-serving/deploy-custom-models.html

22. Databricks. (2022, February 14). Deploy Production Pipelines Even Easier with Python Wheel Tasks. Databricks Blog. https://www.databricks.com/blog/2022/02/14/deploy-production-pipelines-even-easier-with-python-wheel-tasks.html