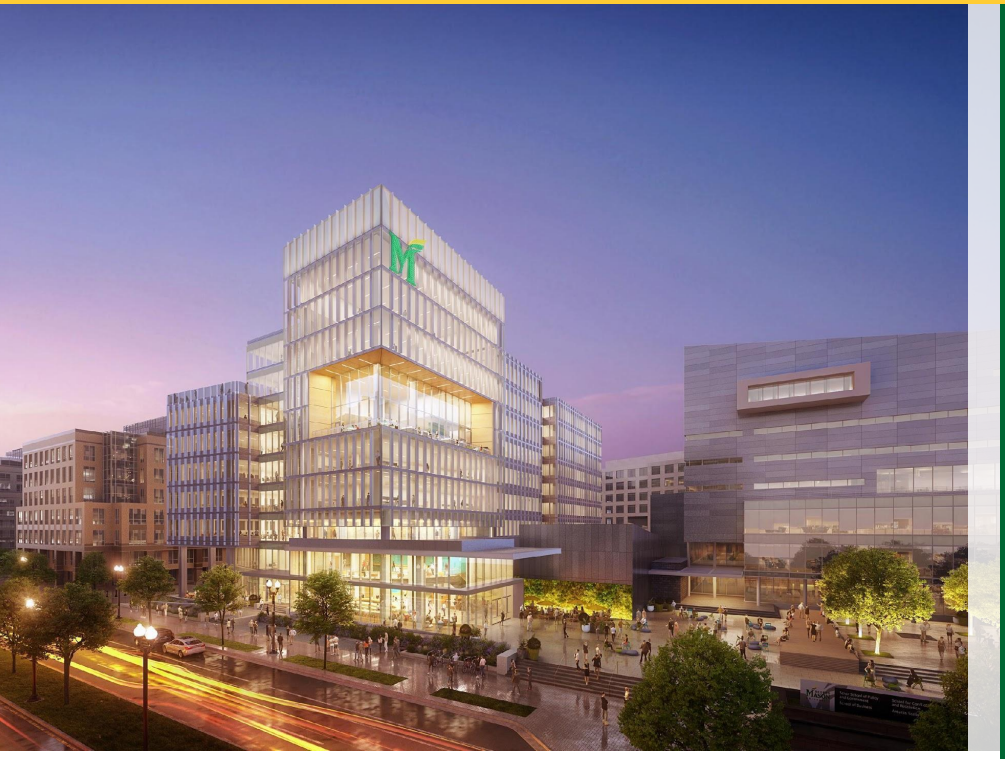
GEORGE MASON UNIVERSITY

Data Analytics Engineering

Fall 2022

EXTRACT CONTENT FROM UNSTRUCTURED TYPE CERTIFICATE DATA SHEET



DAEN 690 Project Report

Patcharaporn Adchariyavivit
Kai chun chan
Rohit Varma Chekuri
Bach Xuan Le
Saroj Siril Palaparthi

About the Cover

George Mason University's Arlington Campus has become Mason Square, a dynamic, collaborative hub uniting scholars, students, researchers, policymakers, and business developers that will accelerate the pace of change and solve grand challenges.

Situated within the Rosslyn-Ballston corridor the campus, minutes from the Virginia Square-GMU Metrorail station, consists of Vernon Smith Hall, Van Metre Hall, and Hazel Hall. In addition to classroom space, the campus site hosts the Antonin Scalia Law School, the Schar School of Policy and Government, the Jimmy and Rosalynn Carter School for Peace and Conflict Resolution, programs in Arts Management and the School of Business, and Continuing and Professional Education.

The heart of Mason Square will be FUSE (see cover rendering). With an estimated construction completion date of December 31, 2025, FUSE is a technology-forward building that will integrate classrooms, labs and workspaces, start-ups, and public programming. FUSE will promote the exchange of new ideas and the transfer of intellectual property between Mason faculty and students, university partners, the tech industry, and the community.

With nearly 350,000 square feet, FUSE at Mason Square will be a net-zero carbon emissions-ready building with:

- LEED Platinum, FitWel 2.1, and RELi 2.0 certifications
- Energy-efficient green roof for heating and cooling,
- An agile floor design to enable responsive team or project growth,
- Specialized labs for robotics, VR, simulation, security, and data visualization.
- A 50-seat theater-style multipurpose room,
- Double height atrium and outdoor plaza for retail and community events

Table of Contents

Table of Figures

Table of Tables

Abstract

A Type Certificate Data Sheet (TCDS) is a document that consists of information about a flight model, where the data acts as a standard for flight manufacturers. Every aircraft manufactured by any organization must adhere to all the requirements set by the government and will need a TCDS issued by the Federal Aviation Administration (FAA) to fly in the USA. This study aims to extract the prioritized information for each model from the TCDS. The FAA has a dynamic dataset, and each TCDS is populated whenever a new model gets added. Formats have evolved over decades, starting from the early 1920s. Therefore, we have numerous variations in the TCDS formats. Extracting critical information into an easily accessible datasheet is vital to save time and effort for the end user of the TCDS. The TCDS has been used and modified for a long time, however, automating and extracting prioritized data remains to be accomplished. Our task was to determine what methods could reach that goal and produce a format conducive to getting the required information. The dataset is given by the FAA, which consists of more than 2000 pdf files; we have performed our pre-processing, data analysis, and cleaning to extract high-priority information. Using the geometry of the TCDS, its structure, and with the support of existing Python libraries, we can differentiate between key terms, model header, and information. The result is a two-column format output file as a data frame and CSV file containing all the prioritized information for each model in the TCDS. They are reader-friendly and make it easy to pull up desired information. The result consists of a python code, producing an output of structured format with over 75% coverage of the critical information.

1 Introduction

1.1 Background

The Federal Aviation Administration (FAA) is the largest modern transportation agency and a governmental body of the United States with powers to regulate all aspects of civil aviation in that nation as well as over its surrounding international waters. [1] Its powers include air traffic management, certification of personnel and aircraft, setting standards for airports, and protection of U.S. assets during the launch or re-entry of commercial space vehicles.

An aircraft's primary attribute is flight. Hence, ensuring its airworthiness is of paramount importance. FAA stores aircraft design information in the form of type certificate documents, called Type Certificate Data Sheet (TCDS). TCDS is a design approval issued by Federal Aviation Administration (FAA), which describes the compliance of a product with applicable regulations. The TCDS is a formal description of the aircraft, engine, or propeller. This document lists limitations and information required for type certification including thrust limitations, weight limits, airspeed limits etc.

II - Model 737-200 (Approved December 21, 1967) Transport Aircraft

Engines: 2 Pratt and Whitney Turbofan Engines JT8D-7, JT8D-7A, JT8D-7B, JT8D-9, JT8D-9A, JT8D-15, JT8D-15A, JT8D-17, and JT8D-17A; Refer to the FAA Approved Airplane Flight Manual for aircraft engine and engine intermix eligibility. (Engine Type Certificate No. E2EA)

Fuel: See NOTE 4 for authorized types of fuel.

Engine Ratings:

	Takeoff static thrust, standard day, sea level conditions (5 min) lb.	Maximum continuous static thrust, standard day, sea level conditions lbs.
JT8D-7, -7A, -7B	14,000	12,600
JT8D-9, -9A	14,500	12,600
JT8D-15, -15A	15,500	13,750
JT8D-17, -17A	16,000	15,200

Engine and Weight Limits

Thrust Settings: The appropriate thrust setting curve (EPR or Pt7), in the FAA Approved Airplane Flight Manual or AFM Appendices must be used for control of engine thrust.

Airspeed Limits: See the appropriate FAA Approved Airplane Flight Manual listed in Note 2.

C.G. Range: See the appropriate FAA Approved Airplane Flight Manual listed in Note 2.

Maximum Weights: See the appropriate FAA Approved Airplane Flight Manual listed in Note 2.

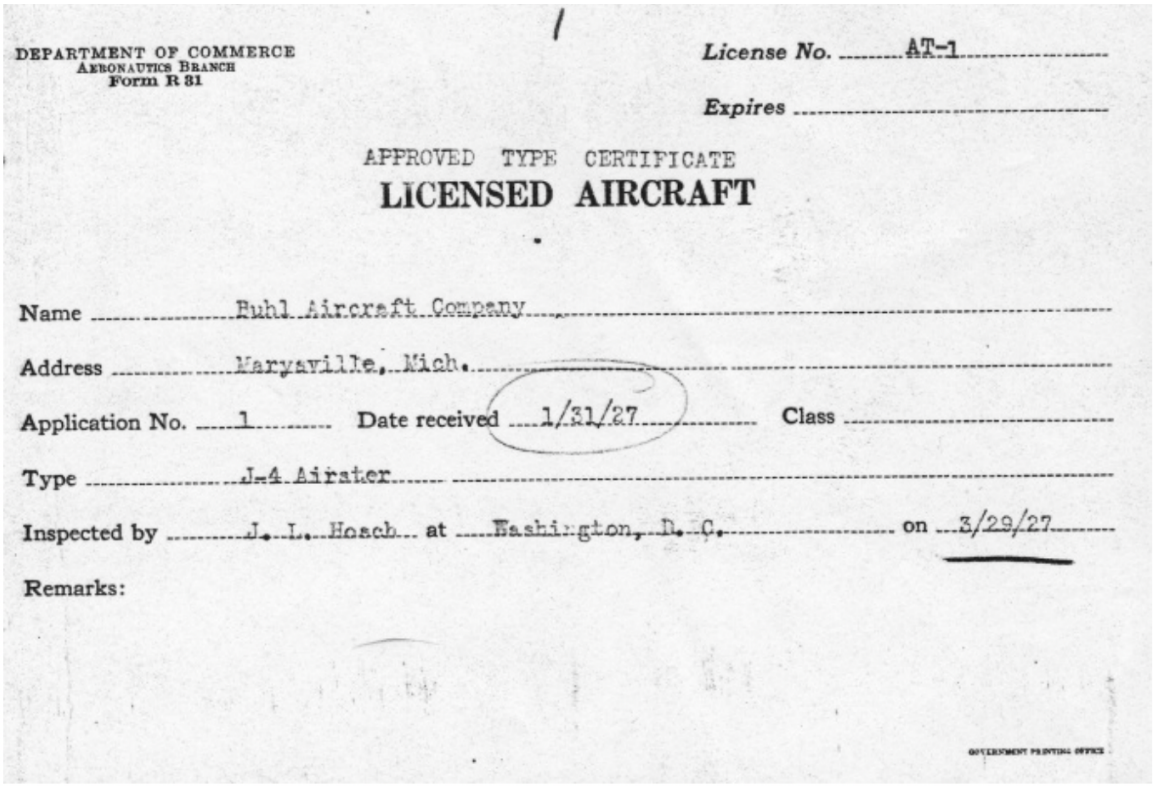
Eligible Serial Numbers:

Model:	
737-201	19418-19423, 20211-20216, 21665-21667, 21815-21818, 22018, 22273-22275, 22352-22355, 22443-22445, 22751-22758, 22795-22799, 22806, 22866-22869, 22961, 22962
737-204	19707-19712, 20236, 20417, 20632, 20633, 20806-20808, 21335, 21336, 21693, 21694, 22057-22059, 22364, 22365, 22638-22640, 22966, 22967
737-205	19408, 19409, 20412, 20711, 21184, 21219, 21445, 21729, 21765, 22022, 23464-23469

Figure 1 Example of TCDS [Boeing-737]

History

On March 29, 1927, the federal government of United States issued its first aircraft TC or a Type Certificate, when the Department of Commerce Aeronautics Branch issued Aircraft Type Certificate No. 1 to the Buhl Airster CA-3 (also known as the J4 Airster), a three-place open biplane. The company had applied for the certificate on January 31, 1927 [2].



DEPARTMENT OF COMMERCE
AERONAUTICS BRANCH
Form R 31

License No. AT-1

Expires

APPROVED TYPE CERTIFICATE

LICENSED AIRCRAFT

Name Buhl Aircraft Company

Address Marysville, Mich.

Application No. 1 Date received 1/31/27 Class

Type J-4 Airster

Inspected by J. L. Hoach at Washington, D. C. on 3/29/27

Remarks:

Figure 2 The first TCDS which is issued on 01/31/1927

Following this, in October 1927, the Branch required aircraft manufacturers to meet minimum engineering standards set forth in detail in a new handbook in order simplify the certification process. This handbook consisted of the standards that the flight of that particular type needed to adhere to in order to be certified. If the data from the application adhered to the standards, a Branch inspector went to the applicant's factory to determine whether the manufacturer adhered to the standard design and specification. In case the inspector issued an approval, one aircraft of the particular type underwent flight tests – first by a company pilot and then by a federal inspector. If the aircraft passed all the required tests, the Department of Commerce issued a type certificate.

Importance

An important function of a TCDS is its role in airworthiness certification. An airworthiness certificate is an FAA document which grants authorization to operate an aircraft in flight. An aircraft must conform to the type certificate of a its type in order to receive an airworthiness certificate [3]. In addition to this, the TCDS contains information about the design and specifications of every type of aircraft approved by the FAA.

Challenges and Goal

Since the inception of Type certificate datasheets, there have been over 2000 type certificates created. These TCDSs are further revised regularly, especially when a new model is introduced. The format and structure of these certificates has developed and changed over time. This has led to few of these certificates becoming unstructured and hard to extract information.

The goal of the project is to create a repeatable process with an aim to extract important information from multiple unstructured FAA Type Certificate Data Sheet (TCDS) documents and store the information in a structured format for FAA to analyze.

1.2 Problem Space

There are currently more than 2,200 TCDS administered by the FAA, most of which are stored in PDF format. The PDF format has the advantage of maintaining the file format and not easily modified, however, TCDS needs to be modified frequently as the aircraft model is updated. In addition, TCDS lacks normalization after decades of manual revision, and the same field may appear in many different ways of recording. These unstructured data cause a huge burden on management, and most of the management needs to be done manually. Not only does this take a lot of time, but there is also a greater chance of errors in content recording. FAA project partners told us that the current TCDS search system's description below the PDF download point is not guaranteed to be completely accurate. It is impractical to manually proofread the contents of such a large number of files. Therefore, what the FAA needs is an algorithmic model that can automate the transformation of existing TCDS into a structured data format.

Figure 3 Example of current key terms from dynamic regulatory system

Unstructured data format

A TCDS is the official document confirming that an aircraft can legally fly in FAA jurisdiction, and any aircraft model in service should have a valid TCDS. Due to the lack of standardization and unification of the production process of TCDS, there have been many differences in the format of documents over the years. The same field may appear differently between different TCDSs, such as tables, figures, or narrative sentences. In addition, the content of this TCDS needs to be updated when the aircraft model hardware settings are changed, or extended models are released. Sometimes these format differences even exist between previous and subsequent updates of the same TCDS. The format differences between these files brought great challenges to the project team in data processing and model building. For example, the project team needs to worry about using parts Whether the model established by TCDS can cope with the unknown data format type. This makes the complexity and time requirements of the project more difficult to predict.

Punctuation and data understanding

The punctuation marks used in TCDS cause confusion in understanding and processing difficulties. During the data exploration process, the task force found that the punctuation used between the TCDS files was very inconsistent. Since TCDS was previously edited manually, editors in different periods have different usage habits of punctuation. This resulted in confusing content in some fields. For example, the figure below shows the serial number field of two different TCDS. The upper content uses "through" to indicate the continuity of the data, while the lower content uses "-". However, even the FAA's project partners could not immediately explain whether the "-" punctuation in the figure below represents continuous data or a hyphen for a single data.

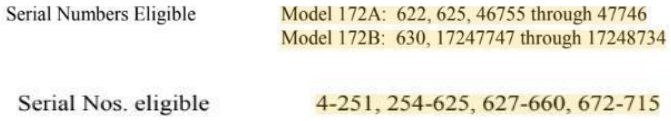


Figure 4 Examples of serial number

Furthermore, in addition to using different TCDS punctuation between different TCDSs, some punctuation differences can even appear in different models within the same TCDS file. These TCDS original issue days are probably decades away now, with many different editors. Due to the lack of knowledge in the aerospace field, judging whether these punctuation differences are meaningful or not brings many difficulties to the project team. We expect significant communication costs to confirm the meaning of these punctuation marks with FAA partners.

1.3 Research

The research has been performed after the first meeting with the FAA team. The FAA has loads of unstructured data which needs to be cleaned and sorted in a clear format for the association to easily extract the important details from the reports. The first meeting with the FAA answered some of our research team's questions. They focus mostly on analyzing the TCDS, then building a database or an API for the FAA to make the most information from different writing formats of TCDS.

The research process starts with looking at different projects that have existed on the Internet [4]. With that being said, the Internet has a lot of research that has been done on unstructured data. From what we have found, most of the research is focusing on one specific task, such as doing language translation, or extracting information from pictures only (Because most of the projects include extraction from a pdf file which contains text data and images containing metadata [5]. They have trained models to identify what has to be processed by the algorithm and which information has to be scrapped). We found that there are so many possible ways for text mining and it's one of the most important tools companies today are looking for. Some of the techniques include Classification Analysis, Association Rule learning, Anomaly or outlier detection, Cluster Analysis, and Regression Analysis but the adjustment will be made after we look closer at the datasets. All these data mining techniques can help analyze different data from different perspectives. The flowchart data mining technique follows is Unstructured data - Identity data - Analysis of Data - Extraction.

PDF Extraction is a major data breakthrough for any organization to make strides in their respective fields. To reorganize data to meet your needs, data extraction from PDFs is essential. Extracting a little amount of data is rather simple in other document formats like DOC, XLS, or CSV. But from a dataset containing loads of pdf files, extraction can result in errors, delays, and cost overruns when performing in mass, which could have a negative effect on your bottom line. There are a lot of tools and packages like 'camelot', 'PyMuPDF', 'Textract',

‘PDFminer’, ’Tabula’, ‘PyPDF2’ and more used for extraction of the data. Camelot gives you control over the table extraction process with tweakable settings. You can discard bad tables based on metrics like accuracy, Whitespace, without having to check manually at each table. Each table is converted into a pandas DataFrame, which fits easily into processes for ETL and data analysis. Additionally, you may export tables in a variety of file types, including CSV, JSON, Excel, HTML, Markdown, and Sqlite.

NLP (Natural Language Processing) helps data engineers to extract insights from natural languages such as English and other scripts [6]. These insights support the decision-making processes of businesses, helping them to make data-based decisions which can be a game changer. NLP enables computers to interpret and understand human spoken languages.

Further from Natural Language Processing, there are several models that exist to perform the task. One of the biggest and well-known models that have existed is BERT, which stands for Bidirectional Encoder Representation from Transformers [7]. BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. From the research team point of view, we agreed that BERT can be used for processing text that will be extracted from TCDS files.

Besides from BERT, PyText is a model that is easy to modify and change. PyText is written by an author in Pytorch, an easy-learned and modified deep learning language. PyText addresses the often-conflicting requirements of enabling rapid experimentation and of serving models at scale. It supports deep learning and is easy to tackle, is one of the things that our team was looking for.

1.4 Solution Space

We expect to provide the FAA's project partners with a method to automatically transform unstructured TCDS into a structured data format. The final product may be a ready-to-use program or a fully functional algorithm model. This final product will consist of two main parts, PDF transformation and text analysis. For PDF transformation, the project team will use python's PDF processing packages such as pdfplumber and Pypdf2 to convert the data on TCDS into text mode that can be edited. Then, use the technology of text mining to analyze the transformed text file, extract the key information in the narrative sentence and export it into a structured format. Ultimately, these steps are automated and repeated using python syntax.

Our final product is expected to provide several benefits to the FAA. First, the FAA will be able to more easily manage the TCDS held by them. TCDS needs to be modified as aircraft manufacturers make changes to the product content. In the past, the manual modification and uploading method is not only time-consuming, but also prone to errors. Using a structured data format allows administrators to quickly find and modify content that needs to be changed.

Second, reduce confusing data. The punctuation used in the original TCDS is not uniform, and some even affect the reader's understanding of the data. Our algorithmic model will solve this problem, and we expect to save users the time cost of double-checking data. Third, FAA is expected to create more value from these processed data in the future. FAA has many professional and powerful engineers who will be able to use structured data to build database systems and fully upgrade TCDS services. For example, the metadata currently provided by the FAA's query page does not exactly match the TCDS content. This problem can be easily corrected through a database system built using structured data. In conclusion, the final product we provide can not only deal with the problems that have occurred so far, but also has great potential to upgrade the TCDS service provided by the FAA in the future.

1.5 Project Objectives

Text mining is the key to this project. This project's difficulty is developing the working prototype and creating the proper model to extract desired information from the text. TCDS is unstructured, which affects when those TCDSs are searched for analysis by engineers or FAA employees. Moreover, TCDSs are regularly revised, especially when there are newly added aircraft. Thus, a repeatable automatic process is desired to handle this issue. The final working prototype is a package of Python script, which includes multiple PDF conversions and text extraction together. The successful measure is 75% coverage accuracy of text extracted TCDS compared with original TCDS. More than the accuracy of Text extraction, the time spent inputting new TCDS into the system is shorter than the current process. Currently, all TCDS on the dynamic regulatory system were input manually. It means all revised TCDS need to be input one by one into the system, so it takes time when there are many TCDSs. The final output of this project points to the standardization of the structure of TCDS to FAA for further new TCDS written format, which makes it easier for text extraction.

1.6 Primary User Stories:

As an FAA employee, I want to automatically get my desired key term from multiple TCDS files in PDF.

As an Airline company in the United States, when I want to purchase new aircraft, I want to get faster approval from FAA after FAA confirms the TCDS of the aircraft. That means I can arrange more flight schedules and get excellent customer satisfaction because of the new airplane, which causes a higher income for an airline.

As an Airline customer, I want more flight availability and great satisfaction. If the airline uses the new aircraft, that means I can have more choice of flight schedules and travel with the new airplane.

1.7 Product Vision

“For FAA engineers and employees who have to analyze aircraft information that does not exist anywhere other than in the TCDS files, the automatic text extraction working prototype is a Python script package that saves time extracting desired key terms from TCDSs. Unlike previous manually input text extract procedures in dynamic regulatory systems, Our product extracts information from PDF files automatically with satisfactory accuracy.”

“For others, Aviation organizations who have TCDSs in their own system, such as European Union Aviation Safety Agency (EASA), the automatic text extraction working prototype is a Python script package that saves time extracting desired key terms from TCDSs. Unlike the current TCDS search engine system that doesn’t have many options for key terms, our product extracts information from PDF files automatically with satisfactory accuracy and covers all the most used and essential key terms.”

1.8 Definition of Terms

N-Number	Identification number assigned to aircraft.
Aircraft Mfr Model Code	A code assigned to the aircraft manufacturer, model, and series.
Engine Mfr Mode Code	A code assigned to the engine manufacturer and model.
Year Mfr	Year manufactured.
Type Registrant	1 - Individual 2 - Partnership 3 - Corporation 4 - Co-Owned 5 – Government 7 - LLC 8 - Non Citizen Corporation 9 - Non Citizen Co-Owned

Registrant's Name	The first registrant's name which appears on the Application for Registration
County Mail	A code representing the county which appears on the Application for Registration.
Last Activity Date	Format YYYY/MM/DD
Certificate Issue Date	Format YYYY/MM/DD

Certification requested and uses:

- A - Airworthiness

Classification Code	The airworthiness certificate class which is reported on the Application for Airworthiness, FAA Form 8130-6. 1 - Standard 2 - Limited 3 - Restricted 4 - Experimental 5 - Provisional 6 – Multiple 7 - Primary 8 - Special Flight Permit 9 – Light Sport

- B - Approved

Operation Codes	The approved operations (up to a maximum of six) which are reported on the Application for Airworthiness, FAA Form 8130-6.

-Standard	N - Normal
	U - Utility
	A - Acrobatic
	T - Transport
	G - Glider
	B - Balloon
	C - Commuter
	O - Other
-Restricted	May contain a code of 0-7. 0 - Other
	1 - Agriculture and Pest Control
	2 - Aerial Surveying
	3 - Aerial Advertising
	4 - Forest
	5 - Patrolling
	6 - Weather Control
	7 - Carriage of Cargo
-Experimental	May contain a code of 0-9. 0 - To show compliance with FAR
	1 - Research and Development
	2 - Amateur Built
	3 - Exhibition
	4 - Racing
	5 - Crew Training
	6 - Market Survey
	7 - Operating Kit Built Aircraft 8A - Reg. Prior to 01/31/08 8B - Operating Light-Sport Kit-Built

	8C - Operating Light-Sport Previously issued cert under 21.190
	9A - Unmanned Aircraft - Research and Development
	9B - Unmanned Aircraft - Market Survey
	9C - Unmanned Aircraft - Crew Training
	9D - Unmanned Aircraft – Exhibition
	9E - Unmanned Aircraft – Compliance With CFR
-Provisional	1 - Class I
	2 - Class II
-Type Aircraft	1 - Glider
	2 - Balloon
	3 - Blimp/Dirigible
	4 - Fixed wing single engine
	5 - Fixed wing multi engine
	6 - Rotorcraft
	7 - Weight-shift-control
	8 - Powered Parachute
	9 - Gyroplane
	H - Hybrid Lift
	O - Other
-Type Engine	0 - None
	1 - Reciprocating
	2 - Turbo-prop
	3 - Turbo-shaft

	4 - Turbo-jet
	5 - Turbo-fan
	6 - Ramjet
	7 - 2 Cycle
	8 - 4 Cycle
	9 – Unknown
	10 – Electric
	11 - Rotary
-Mode S Code	Aircraft Transponder Code

2. Data Acquisition

2.1 Overview:

The Type Certificate Data Sheet (TCDS) is publicly available data managed by the FAA, and the current effective number of TCDS documents in the dataset given is 2,200 and keeps increasing on the official website. Everyone has access to the open-source website and can retrieve information about any type of aircraft through specific TCDSs on the official FAA page, most of which are stored in PDF or DOC file formats. There are also some old TCDS stored in scanned PDF format. The data format varies from one TCDS to the next, eg the format of aircrafts from the early 1950s is totally different from the newer ones added. The information is quite vague in the earlier TCDS, and the newer ones have all the required information in it.

The content of each TCDS is basically divided into holder information and model data with its specifications and limitations. The holder information is at the top of the TCDS, which records the address of the type certificate holder and the list of models that will appear in the file and a revision number of the certificate. Next, the model data records various parameters of the aircraft, such as engine style, fuel tank capacity, take-off weight, etc. At the bottom of each model, the note sections provide the additional content that each model needs to be supplemented with its recorded data.

The TCDS may increase the content of records due to the addition of new models, so the content length will become longer with time. Most TCDS documents are between 30 and 60 pages. Due to the unique design of each aircraft, there are many differences in the fields of model data and the presentation of data. These differences exist between different models of the same TCDS or between different TCDSs. These differences in content make the content of TCDS highly complex and unstructured and will bring difficulties and complications to the project.

2.2 Field Descriptions:

- URL (Type: string) – https://drs.faa.gov/browse/doctypeDetails.

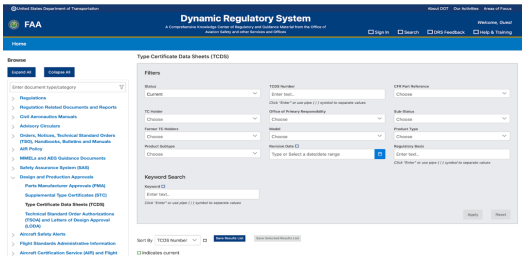


Figure 5 Dynamic Regulatory System

- Title (Type: string) – Type Certificate Data Sheets (TCDS)
- Authors (Type: string) –Type Certificate Data Sheets (TCDS) is issued by the National Aviation Authority (NAA) of the State of the Operator. For this project, we are focusing on TCDS which is issued by the Federal Aviation Administration (FAA).
- Publication Date (Type: datetime) – TCDS documents have been issued from January 31, 1927, until now. Each TCDS doesn't have an exact published date, it depends on when there are new added aircraft and aircraft modification occurs. The issued date in each TCDS may be written in a different format. From data exploration, there are 3 formats including MM/DD/YYYY, DD Full month YYYY and Full month DD YYYY.
- Text (Type: string) – The contents in TCDS include the aircraft model, specification, manufacturer, serial number, and notes. The format of TCDS depends on each TCDS's issuer.
- Tags (Type: string) – TCDS, Type Certificate Data Sheets, Aircraft Worthiness, Type Certificate

The above-mentioned fields are examples of the information fields available on a given certificate. Each type certificate contains too many fields to be added to the report. The Key fields are elaborated on and mentioned in the other sections of the report.

2.3 Data Context:

The data available is in the form of TDCS or Type Certificate Data Sheet documents. A type certificate data sheet is a document that consists of information about a flight model, where the data acts as a standard for flight manufacturers. The datasheet is issued by the Federal aviation administration (FAA). The data within a type certificate is created by a combination of both the manufacturer and the FAA. The manufacturer builds the aircraft based on the standards issued by the FAA. Information includes engine details, fuel capacity, airspeed limits, etc.

This TCDS data has been increasing in volume since its inception in 1927. Every time a new model of aircraft is created, the FAA will make a new TCDS or revise an already existing one which will increment the revision number of the certificate. If the new model has many similarities with other existing models, the FAA will add it to the old TCDS. Despite its role as a guidebook for aircraft the main purpose behind the creation of a TCDS document is its role in the airworthiness certification of aircrafts. When a manufacturer wants to acquire an airworthiness certificate for their aircraft, they first need to acquire a TCDS for that model and make sure the aircraft adheres to the standards within the datasheet. Each TCDS has its own serial number and the revision number through which we get the information on how many times it has been revised and its number.

To write a TCDS, Aircraft Certification Offices (ACOs), directorate staff, and oversight offices prepare TCDS notes. According to an FAA report from 2015, Several current TCDS notes could have been written incorrectly [8]. For example, a common misinterpretation is that a TCDS specifies regulatory requirements only in the main body and that the notes are intended to contain reminders and information for convenience. Some TCDS notes are poorly written and contradict the regulations. For example, notes have specified that all repairs must be approved by the TC holder, which contradicts 14 CFR parts 43 and 21. Depending on the notes provided, the FAA issues a Type Certificate, and they are assumed to be accurate.

In order to test-fly the aircraft, the N-Number is necessary. To make it clear to anybody entering the cabin that the aircraft is an experimental one, if it is a brand-new design seeking an Airworthiness Certificate, the word Experimental must be prominently displayed. The removal of the Experimental sign will be governed by FAA regulations depending on whether the aircraft is intended for general retail sale or was built by a private manufacturer. After receiving an Airworthiness Certificate, an aircraft manufacturer is given a Type Certificate so that, in the future, all aircraft built in accordance with that Type Certificate can be issued an Airworthiness Certificate by the Manufacturer after a test flight to ensure the aircraft operates as intended. Everything that was fitted in the aircraft will be listed on the type certificate, which will be given following the last test flight. This will guarantee uniform performance across all planes in the future. One issue in the 1980s was that some of the older were given Type Certificates with cigar/cigarette lighters. Pilots wanted to remove the cigarette lighter as smoking declined and portable intercoms gained popularity. Pilots sought to remove the cigarette lighter so they could plug in the intercom or some other portable electronic device as smoking became less common and portable intercoms gained popularity. The issue was that the lighter was on the original equipment list, so removing it from the aircraft without a modification STC was against the AC. Since the FAA modified that rule, that has altered in the majority of circumstances.

Moreover, there is not a single airplane without TCDS. Additionally, every aircraft has TCDS however, that does not imply that all aircraft will receive airworthiness certification. TCDS functions as each aircraft's birth certificate. But if you want that aircraft to be permitted to fly over the United States, you need an airworthiness certificate in each aircraft for safe flight.

The primary audience is all Aviation Safety Engineers (ASEs) in Aircraft Certification Service (AIR), and the secondary audience is all aviation safety inspectors (ASIs) in Flight Standards Service (AFS), flight test pilots, and FAA Academy instructors. When changes are needed to an airframe or on-board equipment, there are two options. One is to initiate a modification by the type design holder (manufacturer), and the other is to request a third-party Supplemental Type Certificate (STC). The choice is determined by considering whether or not

the change constitutes a new design (i.e. introduces risk not considered in the first type of design). If so, then the type design holder must develop and approve a modification to the type design.

2.4 Data Conditioning

Data preprocessing, which is a crucial phase in the data mining process, can be defined as modifying the data by altering or dropping data before usage to ensure or optimize the performance of the model. We first collected all the data required for the project by contacting the FAA officials. At first, we were provided only samples of pdf format TCDS which gave us an idea of what TCDS is and what it looks like. According to FAA Order 8110.4, a TCDS is a document created by the FAA that lists the type certification information of a product, including control surface movement limits, operating restrictions, placards, and weight and balance. Each aircraft has a unique TCDS with all its specifications and limitations listed in it.

As a team we first analyzed all the TCDS and decided to prioritize the key information like serial number, metadata listed, revision number, engine details, manufacturer information, previous owner details, and more are only required for the project. We then received our final dataset from the FAA which contained all the Type Certificates (containing pdf) needed for our project.

Firstly, we decided to use pdf extraction techniques to get the required data. After implementing trial and error on PDF files using extraction packages like PDFMiner, PyMuPdf, Pdftotext, PyPDF2, and PdfPlumber we have decided to use PyMuPDF to get the desired output. PyMuPDF is a unique package that helps us find the location of the required object using the coordinates [9].

Secondly, we are splitting each line and converting each word on the line into a block and using the block as a reference to find the coordinates. Each block has (X0, Y0, X1, Y1) as coordinates. We then extract the coordinates of each word as a block. Later we search for blocks whose coordinates are required and extract the Information. The text between the starting coordinate and the ending coordinate is extracted and shown as the output.

Finally, we store the output in a structured format and store it in csv for further reference.

2.5 Data Quality Assessment:

- **Completeness:** FAA partners have provided us with all currently valid 2200 TCDS archives and metadata. We have access to their entire dataset as our partner has added them to the group's GitHub. All the documents received are in PDF format.
- **Uniqueness:** There is no duplication amongst records in the dataset. Every file is different and has a unique serial number. Each TCDS has its own unique content and

format. As new aircraft technology emerges, TCDS may record new fields that did not exist before. In addition, since similar aircraft models may be recorded on the same TCDS, there are large differences in length between TCDSs. The format of older TCDSs is quite different from the newer ones added.

- **Accuracy:** The TCDS data received has no known inaccuracies since it was released by the Federal Aviation Administration itself. Although no TCDS errors have been found during the current operation, we still assume that there may be unknown errors in the data content. The metadata given by the FAA is only 90% accurate.
- **Timeliness:** The timeliness of each TCDS depends on if it's getting altered or not. If the TCDS is changed or altered, it becomes a new document with an added revision number. Once the document is uploaded on the website it's staying there forever until it's revised or changed.
- **Validity:** The Dataset is valid and in the form of PDFs. The comparison of the data within the PDF and the metadata of the file gives us a 90% match, which means there are 10% errors in the metadata file.
- **Consistency:** TCDS are highly inconsistent data. The same attributes will have different representation methods, which may be tables, pictures, or text. In addition, as the aircraft design has evolved, the attributes of different TCDSs are also not consistent. The placing of some fields on the documents is different from the others.

3. Analytics and Algorithms

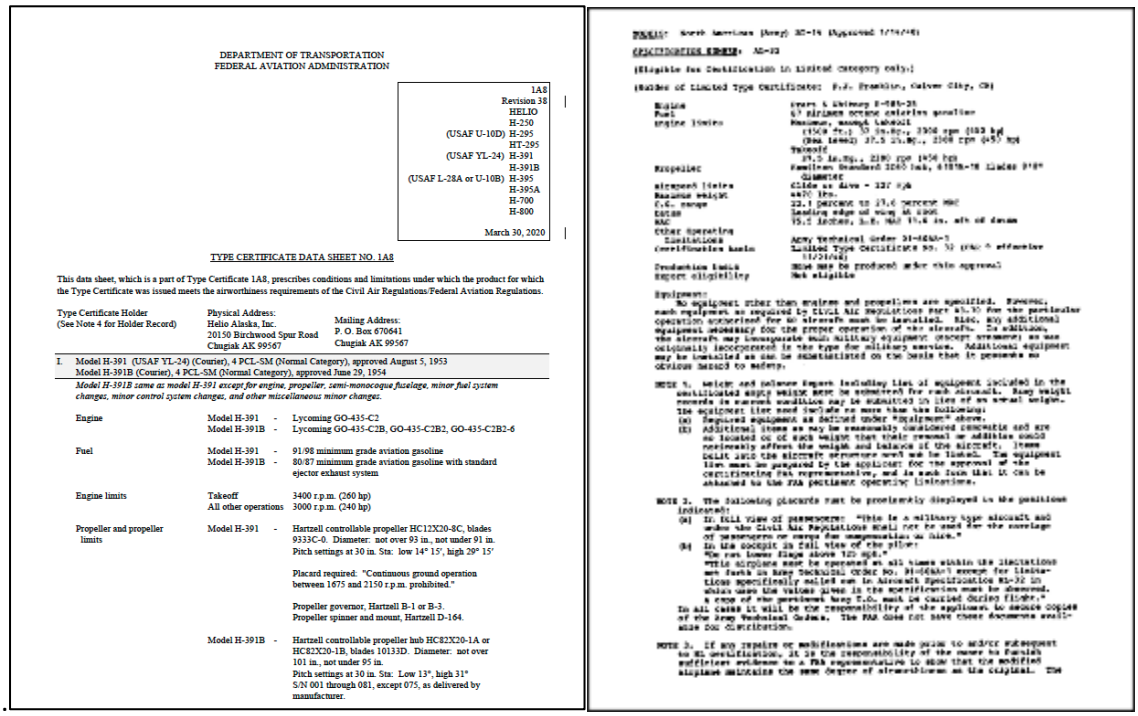
3.1 Exploratory Data Analysis (EDA)

There are 2 outputs of exploratory data analysis(EDA), including the type of PDF and key term writing format.

- **Type of PDF**

  There are three types of PDF in TCDS: text PDF, scanned PDF, and old scanned PDF. After using PyMuPDF to clarify the text and scan PDF type. The result shows that there are 67% of Text PDF and 33% scan PDF. Most of the recent TCDS are in text PDF form, which is able to extract directly with the PyMuPDF library. A scanned PDF is a PDF file with high-quality images. Some of them are also in recent use, so our team needs to extract them. For scanned PDFs, we need to use Pytesseract to convert images to text before information extraction by using PyMuPDF. The last type is sold scanned PDF. Most of this type is old TCDSs. We need to have pre-processing before converting images to text,

such as deblurring, etc. However, the FAA partner gave a high priority to text PDF files and some scanned PDFs based on the recent and frequency uses.



DEPARTMENT OF TRANSPORTATION
FEDERAL AVIATION ADMINISTRATION

1A8
Revision 38
HELIO
H-250
(USAF U-10D) H-295
HT-295
(USAF YL-24) H-391
H-391B
(USAF L-28A or U-10B) H-395
H-395A
H-700
H-800

March 30, 2020

TYPE CERTIFICATE DATA SHEET NO. 1A8

This data sheet, which is a part of Type Certificate 1A8, prescribes conditions and limitations under which the product for which the Type Certificate was issued meets the airworthiness requirements of the Civil Air Regulations/Federal Aviation Regulations.

Type Certificate Holder (See Note 4 for Holder Record)	Physical Address:	Mailing Address:
	Helio Alaska, Inc. 20150 Birchwood Spur Road Chugiak AK 99567	P. O. Box 670641 Chugiak AK 99567

I. Model H-391 (USAF YL-24) (Courier), 4 PCL-SM (Normal Category), approved August 5, 1953
Model H-391B (Courier), 4 PCL-SM (Normal Category), approved June 29, 1954

Model H-391B same as model H-391 except for engine, propeller, semi-monocoque fuselage, minor fuel system changes, minor control system changes, and other miscellaneous minor changes.

Engine	Model H-391 -	Lycoming GO-435-C2
	Model H-391B -	Lycoming GO-435-C2B, GO-435-C2B2, GO-435-C2B2-6
Fuel	Model H-391 -	91/98 minimum grade aviation gasoline
	Model H-391B -	80/87 minimum grade aviation gasoline with standard ejector exhaust system
Engine limits	Takeoff	3400 r.p.m. (260 hp)
	All other operations	3000 r.p.m. (240 hp)
Propeller and propeller limits	Model H-391 -	Hartzell controllable propeller HC12X20-8C, blades 9333C-0. Diameter: not over 93 in., not under 91 in. Pitch settings at 30 in. Sta: low 14° 15', high 29° 15' Placard required: "Continuous ground operation between 1675 and 2150 r.p.m. prohibited." Propeller governor, Hartzell B-1 or B-3. Propeller spinner and mount, Hartzell D-164.
	Model H-391B -	Hartzell controllable propeller hub HC82X20-1A or HC82X20-1B, blades 10133D. Diameter: not over 101 in., not under 95 in. Pitch settings at 30 in. Sta: Low 13°, high 31° S/N 001 through 081, except 075, as delivered by manufacturer.

Figure 6 Type of TCDS PDF files

Text PDF (left) – Old scanned PDF

- Key terms writing format

Extracting information from each key term is the objective of this project, so exploring variance of key terms is important to understand varieties and format. After collecting TCDSs, “Tabula” was used to extract information to be structured in tables. Tabular is a wrapper of tabula-java that allows users to extract the table and convert the PDF file directly into Data frames using Python Programming language. Even though the accuracy of extracting information is low, for this state focusing on only key terms is desired. The below table is the result of using Tabula to extract key tern to be collected in column A and information is in the next column. Then we filtered column A to see all key terms and list-up them.

	A	B	C	D	E	F	G	H	I
1130	Maximum Weights		lb. lb.	lb.	lb.	lb.	lb.		
1131		Ramp	47700 51250	51250	53250	53250	53250		
1132		Takeoff	47450 51000	51000	53000	53000	53000		
1133		Landing	44700 46750	47000	46750	47000	47000		
1134		Zero Fuel	42200 44000	44000	44000	44000	39500		
1135		Minimum fli	30000 30000	30000	30000	30000	30000		
1136		NOTE: The maximum take-off weight and/or maximum landing weight may be further limited due to							
1137		performance considerations (refer to Airplane Flight Manual).							
1138									
1139	Minimum Crew	Two (Pilot and Co-pilot)							
1140	Maximum Occupants	Series 100 - Fifty-five (55) (including 50 passengers, 4 crew, and 1 flight observer)							
1141		Series 440 - Forty-Nine (49) (including 44 passengers, 4 crew, and 1 flight observer)							
1142									
1143		CL-600-2B19 Green Aircraft Configuration							
1144		Refer to NOTE 4 & 5.							
1145									
1146	Fuel Capacity (usable)		Load*		Weight*				
1147			U.S. Gal.	Imp. Gal.	Kg.	lb.			
1148		2 main tanks	700	582.8	2159	4760			
1149		Center tank	735	612	2267	4998			
1150		Total	2135	1669.6	6585	14518			
1151		* Pressure refueling (based on 0.8028 kg/L)							
1152									
1153	Oil Capacity		Load		Weight				
1154			U.S. Gal.	Imp. Gal.	Kg.	lb.			
1155		2 Engines (e	1.7	1.42	5.94	13.09			
1156		Total	3.4	2.84	11.88	26.18			

Table 1 The result of text extraction for key terms by using Tabula

The result of data exploration for key terms is that each key term has different wording and writing variances. The FAA clarified that it is because there is no official format to keep control of the structure or format of TCDSs which one of project expectations is being able to suggest the format of TCDS at the end of this project. All list of key terms are listed in Table2 with suggested key terms.

Key terms	Variances
Engine	Engine, Engine:, Engines, Engine.
Engine Limit	Engine Limit, Engine limit
Holder	Type Certificate Holder , Type Certificate Holder:
Holder Record	Type Certificate Holder Record, Type Certificate Holder Record:
Fuel	Fuel, Fuel:, *Fuel, Fuels, Fuel*
Fuel Capacity	Fuel Capacity, Fuel capacity, FUEL CAPACITY
Serial numbers	Eligible Serial Numbers, Eligible Serial Numbers:, Serial Nos. Eligible, Manufacturer's Serial Nos., SERIAL NUMBERS

Table 2 Key terms variation

3.2 Product implementation

The final product of this project is a python script that includes PDF conversion, text-to-information extraction, and data frame creation.

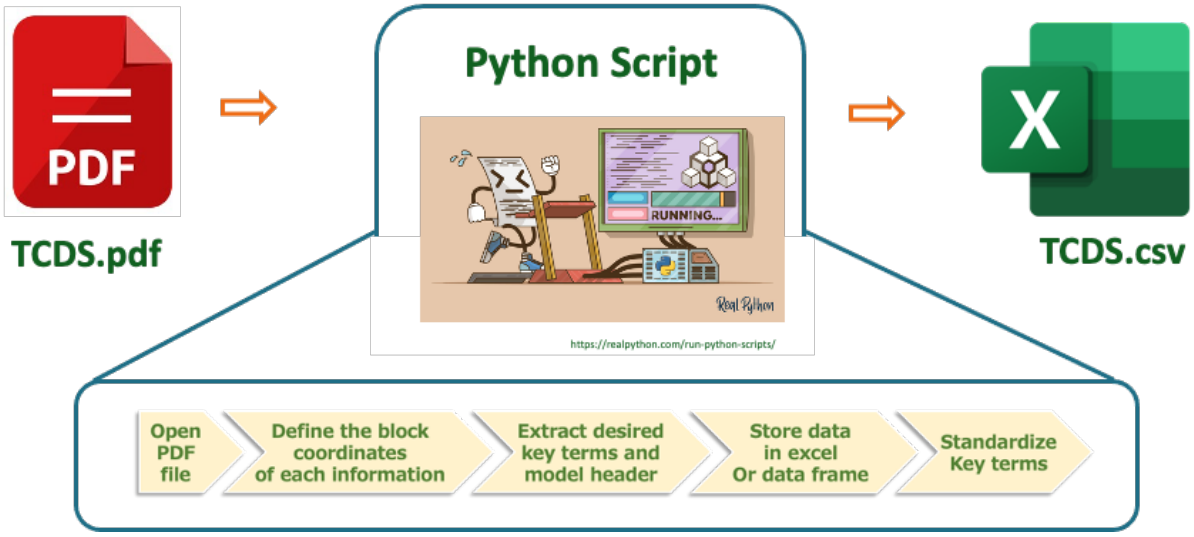


Figure 7 The pipeline

3.2.1 libraries

There are many open-source library tools to extract PDF to text such as PyPDF, PyMuPDF, Tabula, PDFplumber, PDFminor, etc. After trialing them, our team decided to use PyMuPDF as the base algorithm because it has the best output of text extraction. The result of each library is as below

PDFplumber: There are strange characters that occur when extracting text from PDF

```
\nDEPARTMENT OF TRANSPORTATION
FEDERAL AVIATION ADMINISTRATION
EW3A
Revision 19
GNIEOB
727 Series
727-100 Series
727C Series
727-100C Series
727-200 Series
727-200F Series
February 20, 1991
TYPE CERTIFICATE DATA SHEET A3WE
This data sheet, which is part of Type Certificate No. A3WE, prescribes conditions and limitations under which the pr
oduct for
.snoital   usl r gtifieenvoAhReitmCeri usqseernihtro  wse rthdietea e uemsstasawicifi  t ehrepcehyictthw
:redl oeHtacifitr eeCpyT ynap mgonCi eeohBT
P.O. Box 3707
Seattle, Washington  98124
```

Figure 8 PDFplumber output

PyPDF2: The extraction accuracy is impressive and also is able to keep the format. The performance of PyPDF2 is similar to PuMuPDF.

```
 DEPARTMENT OF TRANSPORTATION FEDERAL AVIATION ADMINISTRATION A16WE Revision  71
BOEING
737-100 Series
737-200 Series
737-200C Series
737-300 Series
737-400 Series
737-500 Series
737-700 Series
737-800 Series
737-600 Series
737-700C Series
737-900 Series
737-900ER Series
737-8
737-9
737-8200
Date  July 12, 2022
 TYPE CERTIFICATE DATA SHEET A16WE

This data sheet, which is part of Type Certificate No. A16WE, prescribes conditions and limitations under which the p
roduct for  which the type certificate was issued meets the airwor thiness requirements of the Federal Aviation Regul
ations.  Type Certificate Holder:   The Boeing Company  737 Logan Ave N  Renton, WA  98057-0000

I - Model 737-100 (Approved December 15, 1967) Transport Aircraft

Engines: 2 Pratt and Whitney Turbofan Engines JT8D-7, JT 8D-7A, JT8D-7B, JT8D-9, JT8D-9A, and JT8D-15; refer to the F
AA Approved Airplane Flight Manual for aircraft engine and engine intermix eligibility.  (Engine Type Certificate No.
E2EA)  Fuel: See NOTE 4 for authorized types of fuel.  Engine Ratings:  Takeoff static thrust Maximum continuous stat
ic   standard day, sea leve l thrust, standard day,   conditions (5 min.) lb. sea level conditions lb.
 JT8D-7, -7A, -7B 14,000 12,600  JT8D-9, -9A 14,500 12,600  JT8D-15 15,500 13,700

Engine and Weight Limits:   For engine operating limits see engine Type Certificate Data Sheet No. E2EA or the FAA Ap
proved Airplane Flight Manual.  Thrust Setting: The appropriate EPR thrust setting curve (EPR  or PT 7), in the FAA A
pproved Airplane Flight Manual of AFM Appendices must be used for control of engine thrust.  Airspeed Limits: See the
appropriate FAA Approved Airplane Flight Manual listed in NOTE 2.  C.G. Range: See the appropriate FAA Approved Airpl
ane Flight Manual listed in NOTE 2.
```

Figure 9 PyPDF2 output

PyMuPDF: The performance of PyMuPDF is impressive, and PyMuPDF is also able to extract by using coordinate, which is an advanced function more than PyPDF2, so our team decided to use PyMuPDF as the main algorithm for this project.

```
  727 Series
  727-100 Series
  727C Series
  727-100C Series
  727-200 Series
  727-200F Series
  February 20, 1991
  TYPE CERTIFICATE DATA SHEET A3WE
  This data sheet, which is part of Type Certificate No. A3WE, prescribes conditions and limitations under which the pr
  oduct for
  which the type certificate was issued meets the airworthiness requirements of the Civil Air Regulations.
  Type Certificate Holder:
  The Boeing Company
  P.O. Box 3707
  Seattle, Washington  98124
  I.      Model 727, 727-100 (approved December 24, 1963) and 727C and 727-100C (approved January 13, 1966) Transport
  Aircraft
  Engines
  3 Pratt and Whitney JT8D-1, JT8D-1A, JT8D-1B, JT8D-7, JT8D-7A, JT8D-7B,
  JT8D-9, JT8D-9A, JT8D-11, JT8D-15, or JT8D-17 Turbofans (see -11, -15, and
  -17 ratings under 727-200).  (For engine intermix, see Note 9).
  Fuel
  See Note 5.
  Engine Ratings:
  JT8D-1, -1A, -1B,
  -7, -7A, -7B
  JT8D-9, -9A
  Takeoff static thrust, standard day, sea
    level conditions (5 min) lb.
  14,000
  14,500
  Maximum continuous static thrust,
    standard day, sea level conditions lb.
```

Figure 10 PyMuPDF output

3.2.2 Text extraction to information

3.2.2.1 Pure Text Method

The first method is using data structure. All TCDSs have similar sequence of information, this concept helps to get information between each key terms. The procedure is searching for key terms which are in next to line of each other, then extract the information between them.

```
start1 = "Type Certificate Holder"
end1 = "Type Certificate Holder Record"
A1=(text.split(start1))[1].split(end1)[0]
print(A1)
```

```
Piper Aircraft, Inc.
2926 Piper Drive
Vero Beach, Florida 32960
```

```
start2 = "Type Certificate Holder Record"
end2 = "I"
A2=(text.split(start2))[1].split(end2)[0]
print(A2)
```

```
The New Piper Aircraft,
```

```
start3 = "Engine"
end3 = "Fuel"
A3=(text.split(start3))[1].split(end3)[0]
print(A3)
```

```
Lycoming O-320-B2B or O-320-D2A with carburetor setting 10-3678-32
```

Figure 11 Example of coding

The final outputs are collected in term of data frame and excel file as below

	A	
1	**key terms**	**Information**
2	TCDS number	5A5
3	Type Certificate Holder	Regal Air, Inc.2040 Airway Ave.Fort Collins, Colorado 80524I - Model 305A (USAF 0-1A), 2 PCLM (Utility Category), Ap
4	Type Certificate Holder Record	: Beech Aircraft Corporation transferred to Raytheon Aircraft Company on April 15, 1996 Raytheon Aircraft Compan
5	Engine	Continental O-470-11 or -11B*
6		
7		

Figure 12 Example of output from data structure (TCDS no.5A5)

3.2.2.2 Defining coordinate by using PyMuPDF method

PyMuPDF has the unique function of displaying coordinates for each word on the page. Because of this trait the main concept of this project's text extraction is using coordinates. From Data exploration, most of the TCDS format is the same. The key terms are on the left and information is on the right. Both are written on the same vertical line for the entire page on each TCDS, but it doesn't have the same vertical coordinate in all TCDSs. It has slightly different locations.

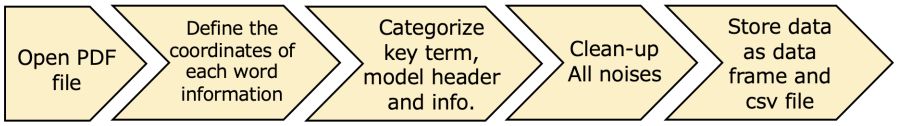


Figure 13 shows the process of text extraction into information

***Open PDF file*:** import Fitz library and use function 'open' to open PDF file. This function opens the pdf and displays it for the viewer.

***Define the coordinates of each word information*:** Using Content.GetText('Words') to get coordinates for each word. Because Key terms coordinate is desired, so the words that are related to information vertical coordinate are selected such as "Holder". Since there are a lot of key terms to be extracted, this method is a more reliable one.

```
(293.14501953125, 362.4730224609375, 340.04403686523344, 374.47900390625, 'requirements', 32, 2, 9)
(342.2940368652344, 362.4730224609375, 349.8720397949219, 374.47900390625, 'of', 32, 2, 10)
(352.0230407714844, 362.4730224609375, 363.0930480957031, 374.47900390625, 'the', 32, 2, 11)
(365.3250427246094, 362.4730224609375, 392.2890319824219, 374.47900390625, 'Federal', 32, 2, 12)
(394.5750427246094, 362.4730224609375, 425.9490661621094, 374.47900390625, 'Aviation', 32, 2, 13)
(428.27105712890625, 362.4730224609375, 473.60406494140625, 374.47900390625, 'Regulations.', 32, 2, 14)
(72.02400207519531, 383.593017578125, 90.29400634765625, 395.5989990234375, 'Type', 32, 4, 0)
(92.5260009765625, 383.593017578125, 130.3350067138672, 395.5989990234375, 'Certificate', 32, 4, 1)
(132.5850067138672, 383.593017578125, 157.65901184082203, 395.5989990234375, 'Holder', 32, 4, 2)
(216.0500030517578, 383.593017578125, 244.41799926757812, 395.5989990234375, 'Textron', 32, 5, 0)
(246.74000549316406, 383.593017578125, 278.11407470703125, 395.5989990234375, 'Aviation', 32, 5, 1)
(280.4360656738281, 383.593017578125, 294.1970520019531, 395.5989990234375, 'Inc.', 32, 5, 2)
(216.0500030517578, 394.15301513671875, 231.0800018310547, 406.15899658203125, 'One', 33, 2, 0)
(233.3300018310547, 394.15301513671875, 258.8000183105469, 406.15899658203125, 'Cessna', 33, 2, 1)
(261.0320129394531, 394.15301513671875, 298.5080261230469, 406.15899658203125, 'Boulevard', 33, 2, 2)
(216.0500030517578, 404.7130126953125, 246.7220001220703, 416.718994140625, 'Wichita,', 34, 2, 0)
(249.0080108642578, 404.7130126953125, 274.946044921875, 416.718994140625, 'Kansas', 34, 2, 1)
(279.4730529785156, 404.7130126953125, 302.2699890136719, 416.718994140625, '67215', 34, 2, 2)
(72.02400207519531, 425.8330078125, 90.29400634765625, 437.8389892578125, 'Type', 35, 1, 0)
(92.5260009765625, 425.8330078125, 130.3350067138672, 437.8389892578125, 'Certificate', 35, 1, 1)
```

Figure 14 The coordinate of key terms

The 4 numeric values are (X, Y) coordinates which represent [(X0, Y0) (X1, Y1)] of each block. Figures show the coordinates for each word on the page. Each word is treated individually and has ((X0, Y0) , (X1,Y1)) coordinates. Hence, we get the coordinates for all the blocks for the entire pdf if required. From this we get the coordinates of the desired key terms. For example: See the picture below.

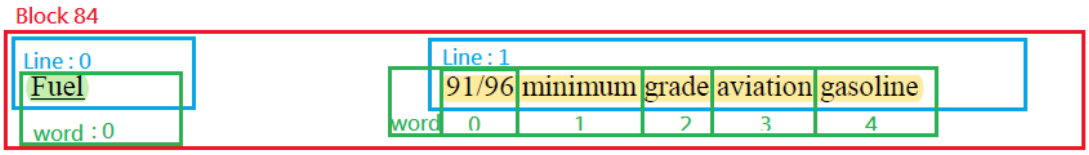
```
(90.02400207519531, 110.54302215576172, 115.60199737548828, 122.54901885986328, 'Engine', 63, 0, 0)
(216.0500030517578, 110.54302215576172, 252.69801330566406, 122.54901885986328, 'Lycoming', 63, 1, 0)
(254.90301513671875, 110.54302215576172, 300.6409912109375, 122.54901885986328, 'IO-360-C1C', 63, 1, 1)
(307.4449768066406, 110.54302215576172, 323.4289855957031, 122.54901885986328, '(See', 63, 1, 2)
(325.6609802246094, 110.54302215576172, 349.5199279785156, 122.54901885986328, 'NOTE', 63, 1, 3)
(351.81494140625, 110.54302215576172, 363.919921875, 122.54901885986328, '22)', 63, 1, 4)
(216.0500030517578, 121.10301971435547, 252.69801330566406, 133.10902404785156, 'Lycoming', 63, 2, 0)
(254.90301513671875, 121.10301971435547, 305.1409912109375, 133.10902404785156, 'IO-360-C1C6', 63, 2, 1)
(309.72198486328125, 121.10301971435547, 325.70599365234375, 133.10902404785156, '(See', 63, 2, 2)
(327.93798828125, 121.10301971435547, 351.79693603515625, 133.10902404785156, 'NOTE', 63, 2, 3)
(354.0919494628906, 121.10301971435547, 366.1969299316406, 133.10902404785156, '23)', 63, 2, 4)
```

Figure 15 The coordinates of the block 63

The last three numeric values represent block_no, line_no, and word_no in order. Block_no is the number given by Pymupdf to the blocks on the PDF file. These numbers are consecutive numbers from top to bottom. Lines are sub elements of blocks, which are used to distinguish the text content divided by format design (ex. tab) in the block. Each block can contain one or more lines, and each line will have its own line_no. Words are the sub element of lines, used to distinguish text separated by spaces in each line. Through these three values, each block, line, and word have a corresponding unique encoding. This will bring great convenience to specifying text and other operations later

Here the block_nois 63 and the words ‘Engine’, ‘Lycoming’, ‘IO-360-C1C’, ‘See’, ‘NOTE, and ‘22’ are on that line. The words ‘Engine’ belongs to word group 0 and ‘Lycoming IO-360-C1C See NOTE 22’ belongs to the word group 1. Since Engine is the only word for its word group it has the order number (third number) as 0.

```
['Fuel', 84, 0, 0]
['91/96', 'minimum', 'grade', 'aviation', 'gasoline', 84, 1, 4]
```




```
(90.02400207519531, 442.7530212402344, 106.07099914550781, 454.7590026855469, 'Fuel', 84, 0, 0)
(216.0500030517578, 442.7530212402344, 236.7320098876953, 454.7590026855469, '91/96', 84, 1, 0)
(239.05401611328125, 442.7530212402344, 273.8840026855469, 454.7590026855469, 'minimum', 84, 1, 1)
(275.989990234375, 442.7530212402344, 295.9339904785156, 454.7590026855469, 'grade', 84, 1, 2)
(298.1659851074219, 442.7530212402344, 327.13702392578125, 454.7590026855469, 'aviation', 84, 1, 3)
(329.4590148925781, 442.7530212402344, 359.4650573730469, 454.7590026855469, 'gasoline', 84, 1, 4)
```

Figure 16 The block_no , line_no and the word_no

The picture above is an example of the content in block84. The red box is the extent of the entire block84 on the PDF file, it contains two lines (line0 & line1), which are marked with blue boxes. The green box marks the text element in each line, line 0 contains one text and line 1 contains five.

```
[214.00999450683594, 56.68177795410156, 397.8815612792969, 70.47637939453125, 'DEPARTMENT OF TRANSPORTATION', 63, 0, 2]
[208.85000610351562, 68.34175109863281, 403.09002685546875, 82.1363525390625, 'FEDERAL AVIATION ADMINISTRATION', 64, 0, 2]
[504.2200012207031, 91.10301971435547, 524.1909790039062, 103.10901641845703, '2A13', 65, 0, 0]
[480.7900085449219, 101.66301727294922, 524.2000122070312, 113.66901397705078, 'Revision 63', 66, 0, 1]
[458.4700012207031, 112.22301483154297, 524.2150268554688, 124.22901153564453, 'Piper Aircraft, Inc', 67, 0, 2]
[390.19000244140625, 133.34300231933594, 430.4505920410156, 145.34901428222656, 'PA-28-140', 68, 0, 0]
[454.6300048828125, 133.34300231933594, 494.8905944824219, 145.34901428222656, 'PA-28-151', 68, 1, 0]
[390.19000244140625, 143.9029998779297, 430.4505920410156, 155.9090118408203, 'PA-28-150', 69, 0, 0]
[454.6300048828125, 143.9029998779297, 494.8905944824219, 155.9090118408203, 'PA-28-161', 69, 1, 0]
[390.19000244140625, 154.46299743652344, 430.4505920410156, 166.46900939941406, 'PA-28-160', 70, 0, 0]
[454.6300048828125, 154.46299743652344, 494.8905944824219, 166.46900939941406, 'PA-28-181', 70, 1, 0]
[390.19000244140625, 165.0229949951172, 430.4505920410156, 177.0290069580078, 'PA-28-180', 71, 0, 0]
[454.6300048828125, 165.0229949951172, 500.92059326171875, 177.0290069580078, 'PA-28R-201', 71, 1, 0]
[390.19000244140625, 175.58299255371094, 430.4505920410156, 187.58900451660156, 'PA-28-235', 72, 0, 0]
```

Figure 17 The output

The above picture is an accurate representation of the output obtained from this step. Each line on the PDF is treated individually by PyMuPDF and all the coordinates are extracted for each word. So we have the (X0,Y0) , (X1,Y1) , block_no , line_no and word_no in the end for the entire pdf.

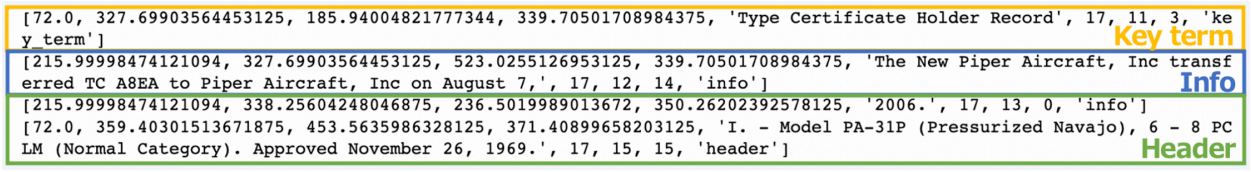


Figure 18 The example of coordinate feature output from PyMuPDF

Categorize key term, model header and info: In this step we will classify each content in the data into, header, key terms, and information. Each model header contains their model number and approval date. According to the TCDS format, the starting coordinates (X0) of all model headers, key terms and descriptive content are on the left side of the file, and the starting coordinates of all key term information are in the center of the file. Therefore, we will use the coordinate (X1) to the right of the key term 'Fuel' as a dividing line to distinguish information from other content. There must be a key term 'Fuel' in each TCDS, and this word does not appear in earlier content. We have tried using fixed coordinates to split the file, but we can't be sure that the fixed coordinates will work for all highly unstructured TCDS. Therefore, rather than using fixed coordinates to split the file, it would be a better choice to use the first 'Fuel' that appears in TCDS to define the split line.

Key terms:

We use two conditions to define key terms in TCDS.

- if any words have their X0 less than X1 of Fuel, they are considered to be the keyterm.
- The word number of each keyword must less than five words. From data exploration, the keyterm have maximum word number at 5. Since the number of characters in a key term is no more than five words, this condition can distinguish key terms from header or descriptive content that meets the first two conditions.

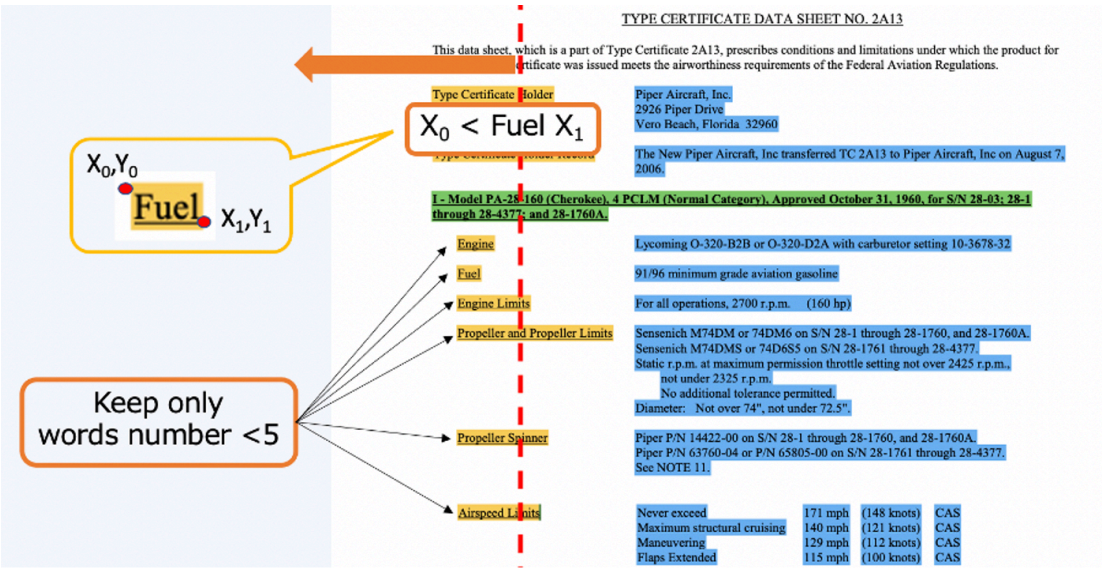


Figure 19 The split line to extract key terms

Model header:

For the model header, we will also use three conditions to define it. The first and second conditions are the same as defining key terms. Since all model headers contain the model number and approved date, the third condition is that the text in each line must contain the words 'Model' and 'Approved'. Since most model headers have several lines of content, in fact only the first line of these content will meet the three defined conditions. Therefore, the last step in defining model headers is to also mark as model header the line that has the same block_no as the result of the definition.

In the above figure, it can be observed that the second line of the model header is marked in yellow, which means the line_no of this line is 1. The reason for this result is because of the principle of defining block_no, line_no, and word_no according to pymupdf. Generally speaking, to create a new block_no it requires the editor to perform a line break action. The line break here refers to the use of the "Enter" key on the keyboard rather than a visual line break. For content that automatically wraps due to insufficient space, the PyMuPdf algorithm will define the content as another line in the same block. In addition, in the text of the same height, the "Tab" key or other formatting design will also cause the generation of a new line in the same block.

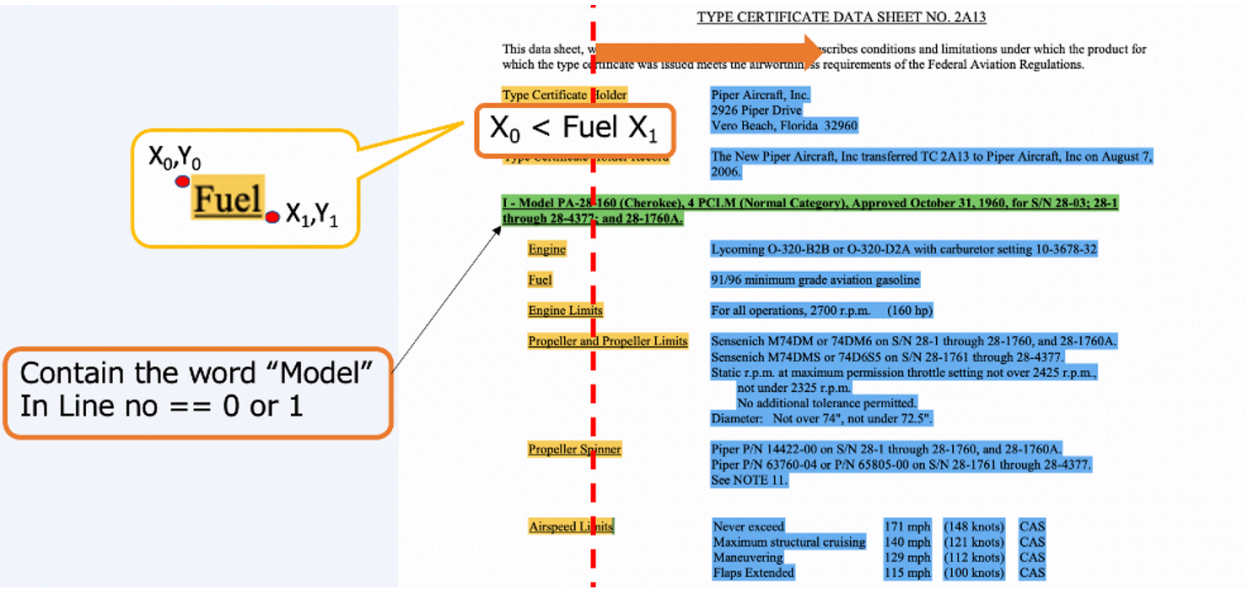


Figure 20 The split line to extract Model header

Other descriptive content:

Content that meets the first two conditions for defining key terms and is not key terms or model headers will be defined as descriptive content.

Information:

We will define content between two key terms or a key term and a header as information.

After defining what category, the content of all lines belongs to, we will add the classification tags corresponding to their categories in the data frame. And these marked data frames will be the output of this step.

Clean-up all noises:

- The disorder by Y-coordinate arrangement: When the PyMuPDF extract the text from PDF, it defines the coordinate of each words including X0,Y0,X1 and Y1. The logic of PyMuPDF when extracting is when Y0 in each words in same line_no is different, PyMuPDF will order the sequence words following the high to low Y0 position and make it in the different line.

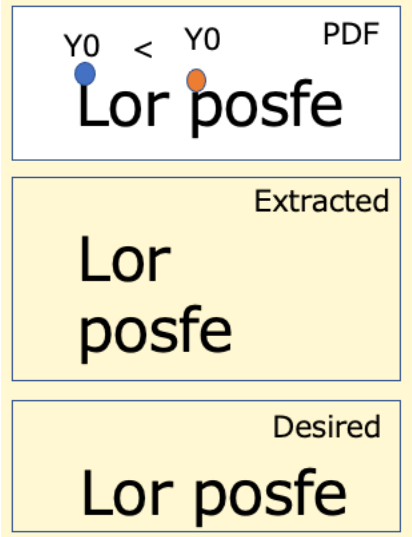


Figure 21 The example of disorder extraction from Y coordinate

The issue is solved by setting the condition to make all Y0 are the same, if those words are in the same line_no.

- Removing unwanted info: Unwanted info are considered as noises. To keep only necessary data, we need to drop or delete those noises out.
  - The page number and some information in header when there are multiple pages: Every TCDSs we have, they contain the page number and TCDS number at the top of the page, but they don't' have consistent location. To remove these noises, First, defining the lowest Y1 in each page excludes the first page. Secondly set the lowest Y1 as a variable and then giving a condition that if there are any words have Y1 equal to the lowest Y1, those words will be dropped out.

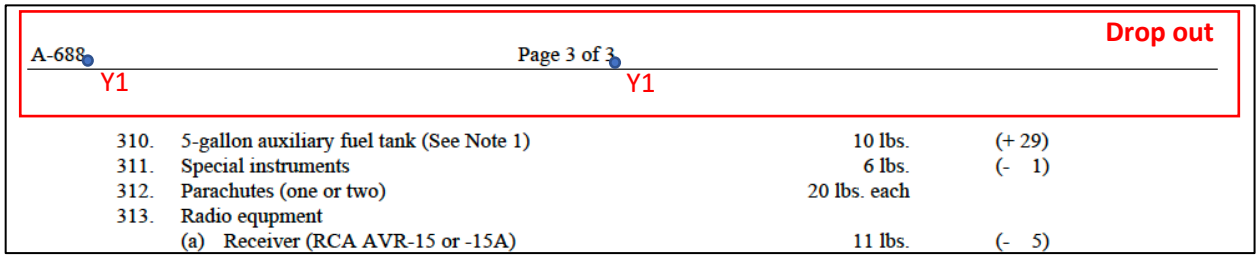
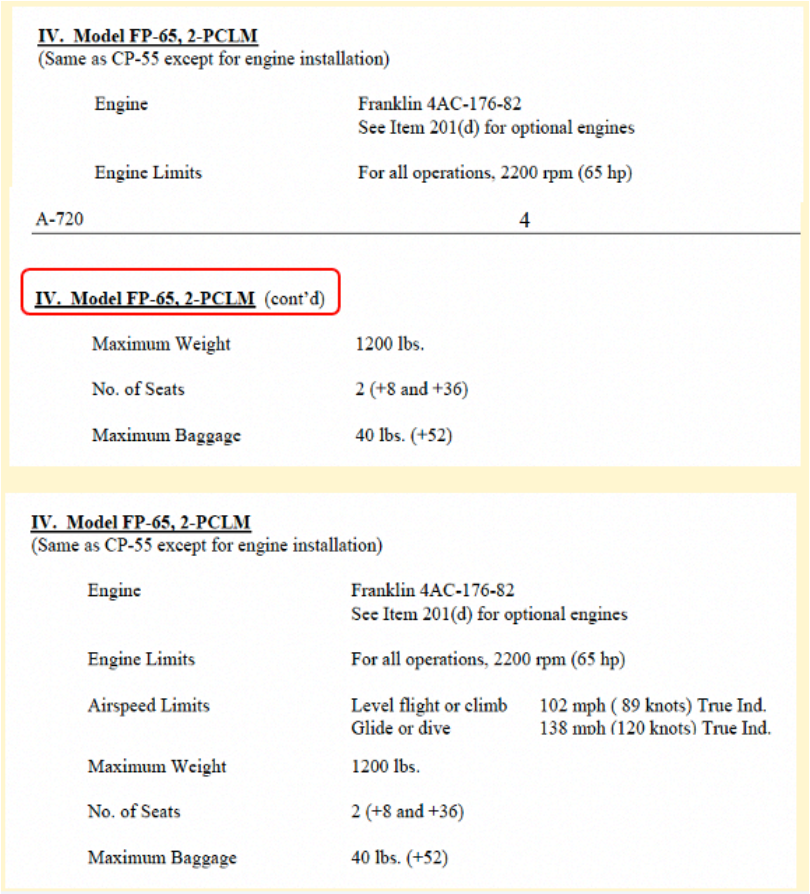


Figure 22 Example of the page number's noise

  - "Continue" in model header when there are multiple pages:
    When there are new continue information, in the next pages the continue keywords will show. We consider this is one of noises because when this model header is stored in dataframe and csv, it will not be used. To remove this noise, we only set the condition that, if there are words of "cond't", it needs to be remove for all words and line no.

From our data exploration, we found contd, (cont'd), *(continued)* ,cont'd and cont. these are variation of continue keywords.



IV. Model FP-65, 2-PCLM
(Same as CP-55 except for engine installation)

Engine	Franklin 4AC-176-82 See Item 201(d) for optional engines
Engine Limits	For all operations, 2200 rpm (65 hp)

A-720 4

IV. Model FP-65, 2-PCLM (cont'd)

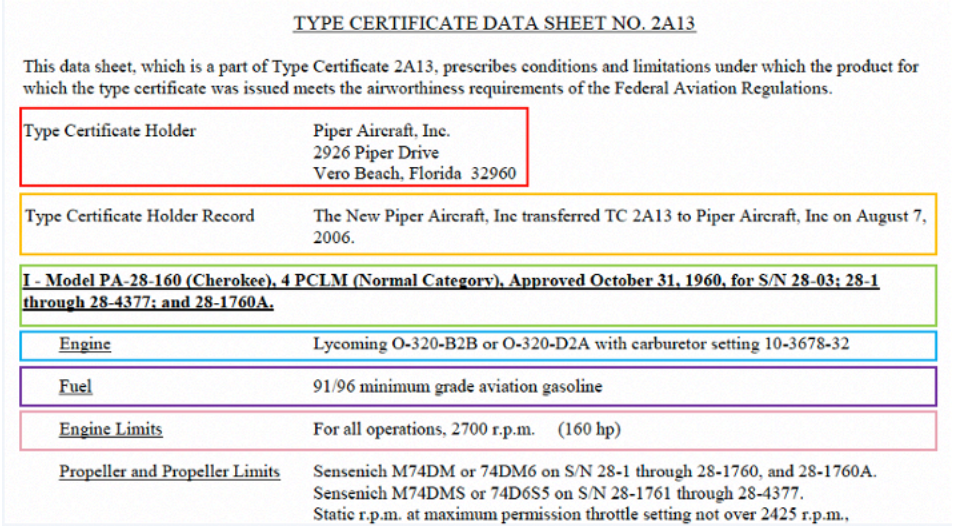
Maximum Weight	1200 lbs.
No. of Seats	2 (+8 and +36)
Maximum Baggage	40 lbs. (+52)

IV. Model FP-65, 2-PCLM
(Same as CP-55 except for engine installation)

Engine	Franklin 4AC-176-82 See Item 201(d) for optional engines	
Engine Limits	For all operations, 2200 rpm (65 hp)	
Airspeed Limits	Level flight or climb Glide or dive	102 mph (89 knots) True Ind. 138 mph (120 knots) True Ind.
Maximum Weight	1200 lbs.	
No. of Seats	2 (+8 and +36)	
Maximum Baggage	40 lbs. (+52)	

Figure 23 Example of (cont'd) noise

Stored in Excel file or data frame: The output of extraction will be stored in csv file because it is a structured format and easy to be a working file for further use.



TYPE CERTIFICATE DATA SHEET NO. 2A13

This data sheet, which is a part of Type Certificate 2A13, prescribes conditions and limitations under which the product for which the type certificate was issued meets the airworthiness requirements of the Federal Aviation Regulations.

Type Certificate Holder	Piper Aircraft, Inc. 2926 Piper Drive Vero Beach, Florida 32960
Type Certificate Holder Record	The New Piper Aircraft, Inc transferred TC 2A13 to Piper Aircraft, Inc on August 7, 2006.
I - Model PA-28-160 (Cherokee), 4 PCLM (Normal Category), Approved October 31, 1960, for S/N 28-03; 28-1 through 28-4377; and 28-1760A.	
Engine	Lycoming O-320-B2B or O-320-D2A with carburetor setting 10-3678-32
Fuel	91/96 minimum grade aviation gasoline
Engine Limits	For all operations, 2700 r.p.m. (160 hp)
Propeller and Propeller Limits	Sensenich M74DM or 74DM6 on S/N 28-1 through 28-1760, and 28-1760A. Sensenich M74DMS or 74D6S5 on S/N 28-1761 through 28-4377. Static r.p.m. at maximum permission throttle setting not over 2425 r.p.m.,

Figure 24 TCDS no.2A132

Key term	Info
2A13 Revision 63 Piper Aircraft, Inc PA-28-140 PA-28-151 PA-28-150 PA-28-161 PA-28-160 PA-28-181 PA-28-180 PA-28R-201 PA-28-23	
Type Certificate Holder	Piper Aircraft, Inc. 2926 Piper Drive Vero Beach, Florida 32960
Type Certificate Holder Record	The New Piper Aircraft, Inc transferred TC 2A13 to Piper Aircraft, Inc on August 7, 2006.
I - Model PA-28-160 (Cherokee), 4 PCLM (Normal Category), Approved October 31, 1960, for S/N 28-03; 28-1 through 284377 and 281760A	
Engine	Lycoming O-320-B2B or O-320-D2A with carburetor setting 10-3678-32
Fuel	91/96 minimum grade aviation gasoline
Engine Limits	For all operations, 2700 r.p.m. (160 hp)
Propeller and Propeller Limits	Sensenich M74DM or 74DM6 on S/N 28-1 through 28-1760, and 28-1760A. Sensenich M74DMS or 74D6S

Figure 25 Extracted information in csv file of TCDS no.2A13

Our team provides the automatic information extraction to FAA. They are able to extract multiple TCDSs at once and receive all csv files separately in the same folder.

Figure 26 Loop function to make automatic extraction

THE CODING EXPLANATION

```
import string
import pandas as pd
import fitz
import os
```

"def add_page" is for adding pages to all words

```
def add_page(in_put, value):
    lst = []
    out = []
    index = 0

    for i in range(0, len(in_put)):
        if(i < len(in_put)-1):
            out += [[]]
            lst += in_put[i]
            lst.insert( 8, value+1)
            out[index] = lst
            index += 1
            lst = []
        else:
            out +=[[]]
            lst += in_put[i]
            lst.insert( 8, value+1)
            out[index] = lst

    return out
```

"def get_all_text" is getting all text from given pdf file.

- WC stands for word with coordinate.
- out is the return value of each words characteristic by PyMuPDF feature. The order of the output shows like [x0, y0, x1, y1, word, block_no, line_no, word_no, page].

```
def get_all_text(data):
    page_count = data.page_count
    out = []
    for i in range(0, page_count):
        load_page = data.load_page(i)
        #WC stand for word with coordinate
        WC = load_page.get_text('words')
        agw = add_page(WC, i)
        out += agw

    return out
```

"def word_line" is merging the words by block_no and line_no

```
def word_line(in_put):
    #in_put += [in_put[-1]]
    index = 0
    lst = []
    out = []
    for i in range(0, len(in_put)-1):
        # in_put[i][7] == word_no
        if(in_put[i][7]+1 == in_put[i+1][7]):
            lst += [in_put[i][4]]
        else:
            if(in_put[i+1][7] == 0):
                lst += [in_put[i][4]]
                lst = [' '.join(lst)]
                lst += [in_put[i][5], in_put[i][6], in_put[i][7],in_put[i][8]]
                lst.insert( 0, in_put[i-in_put[i][7]][0])
                lst.insert( 1, in_put[i-in_put[i][7]][1])
                lst.insert( 2, in_put[i][2])
                lst.insert( 3, in_put[i][3])
            out += [[]]
            out[index] += lst
            index += 1
            lst = []
    #print(out)
    return out
```

"def Same_y" uses for rearrange the order of words after extraction. PyMuPDF extracts text by using the coordinate of each word. Sometimes the y coordinates in the same line will be a little different. If one of the words in the same line has the smaller y position, It means that word will be extracted before the higher Y position words.

example: A & B in the same line y0 of A ==1, y0 of B == 0.9 the order will be changed to B A since the y is more sensitive condition of the coming reorder work. This function is to make the texts in the same line have the same y.

```
def Same_y(df):
    fill = [0,9999,0,9999,0,0,0,0,0]
    df += [fill]
    num = 0
    big_y = 99999
    highery = big_y

    for i in range(0, len(df)-1):
        # Smaller Y  == higher position in the file
        # df[i][1] == Y0, df[i][3] == Y1
        # if the average Y of text i in the between of the Ys of text i-1, we say they are the same line
        if((df[i][1]+df[i][3])/2 > df[i+1][1]) & ((df[i][1]+df[i][3])/2 < df[i+1][3]) & (df[i][1] < highery):
            highery = df[i][1]
            num += 1
            #print(num)
        elif((df[i][1]+df[i][3])/2 > df[i+1][1]) & ((df[i][1]+df[i][3])/2 < df[i+1][3]):
            num += 1
            #print(num)
        # if the average Y of the text i is smaller than the text i+1, we know the i+1 is next line
        #elif((df[i][1]+df[i][3])/2 < df[i+1][1]) or (i == len(df)-1):
        else:
            if(num == 0):
                pass
            else:
                for j in range(0, num+1):
                    df[i-j][1] = highery
                num = 0
                highery = big_y
    df.pop(-1)


    return df
```

"def get_fuel_x1" is getting the X1 coordinate by using **"Fuel"** keyterm to be the reference of left and right column. There are various words of "fuel" across all TCDSs. These are all keywords of "Fuel" keyterm that our team found based on TCDSs we have *Fuel* fuel: **fuel* fuel. *fuel*

If users have new TCDSs with other variations of "Fuel" more than list above, you can add more to get the coding to recognize the new variation by adding or (df[i][4].lower() == '---your new variation of "Fuel"---')

```
def get_fuel_x1(df):
    for i in range(0,len(df)):
        if(df[i][4].lower() == 'fuel') or (df[i][4].lower() == 'fuel:') or (df[i][4].lower() == '*fuel') or (df[i][4].lower() == 'fuel.')
            #print(df[i][2])
            return df[i][2]
        else:
            if(df[i][4].lower() == 'engine') or (df[i][4].lower() == 'engines') or (df[i][4].lower() == 'engine:') or (df[i][4].lower() ==
                return df[i][2]
```

"def detect_khi" is defining the word lines' categories of (key_term, header, info)

```
def detect_khi(df):
    index = 0
    lst = []
    out = []
    for i in range(0,len(df)):
        ## old one add & (in_put[i][6] == 0) in both if & elif
        if(df[i][0] <= fuel_x1) & (df[i][7] < 5 ):
            lst += df[i]
            lst.insert(8, 'key_term')
            out += [[]]
            out[index] = lst
            lst = []
            index += 1
        elif(df[i][0] <= fuel_x1) & (df[i][4].lower().find('model') >= 0):
            lst += df[i]
            lst.insert(8, 'header')
            out += [[]]
            out[index] = lst
            lst = []
            index += 1
        else:
            lst += df[i]
            lst.insert(8, 'info')
            out += [[]]
            out[index] = lst
            lst = []
            index += 1

    #print(out_put)
    return out
```

def Drop_blw is dropping some columns (block_no, line_no, word_no) that we don't use it anymore after definding categories.

```
def Drop_blw(df):
    # df == [x0, y0, x1, y1, word, block_no, line_no, word_no, category, page]
    lst = []
    index = 0
    out = []
    for i in range(0, len(df)):
        lst += [df[i][0],df[i][1],df[i][2],df[i][3],df[i][4],df[i][8],df[i][9]]
        out += [[]]
        out[index] = lst
        index += 1
        lst = []
    # out == [x0, y0, x1, y1, word, category, page]
    return out
```

def hder is used for ensure all model header will be collected in dataframe as the header. Sometimes the header's number will be detected as key term, this function is used to solve it. example: real header == I. Model xyz.... and the df might be ['I.', 'key_term'], ['Model xyz...', 'header'] hder function is to catch 'I.' form keyterm to header.

```
def hder(df):
    for i in range(0, len(df)-1):
        if(df[i+1][5] == "header") & (int(df[i][1]) == int(df[i+1][1])):
            df[i][5] = "header"
    return df
```

def firsty1 is finding first line of following pages. It is important to let the coding know where to starting to extract. Most of TCDSs, when they start the next page, on the top left-middle-right will have the page no, TCDS no and revision. We consider them as the noises.

```
def firsty1(df):
    lst = []
    page = 1

    for i in range(0, len(df)-1):
        if(df[i][6] == page+1):
            lst += [df[i][3]]
            page += 1
    #print(lst)
    return lst
```

def drop_fline is dropping first line of following pages.

```
def drop_fline(df, fy):
    lst = []
    if len(fy) == 0:
        pass
    else:
        ylmax = max(fy)
        for i in range(0, len(df)):
            #df[i][6] == page_no
            if(df[i][6] > 1) & (df[i][1] < ylmax):
                lst += [i]
                #print(df[i][4])
#    print(lst)
    lst.reverse()
#    print(lst)
    for j in range(0, len(lst)):
        df.pop(lst[j])

    return df
```

def clean_cont is dropping the lines which contain of Cond't stuff for model header. We consider the cond't stuff as the noise that will be better to remove. There are various words of "cond't" across all TCDSs. These are all keywords of "cond't" keyterm that our team found based on TCDSs we have *contd* (cont'd) *(continued)* cont d *cont

```
def clean_cont(df):

    lst = []

    for i in range(0, len(df)):
        if(df[i][0] <= fuel_xl):
            if(df[i][4].lower().find('contd') > -1) or (df[i][4].lower().find('(cont'd)') > -1) or (df[i][4].lower().find('(continued)') :
                lst += [i]

    #print(lst)
    lst.reverse()
    for j in range(0, len(lst)):
        df.pop(lst[j])

    return df
```

def all_header is used to make sure the left side has all header or keyterm. Some TCDSs have model headers with multiple lines so this function lets all the lines after model header to the first keyterm(which always be Engine") to be defined as model header too.

```
def all_header(df, am):
    if(am == -1):
        am = len(df)
    else:
        pass

    for i in range(1 , am):
        if(df[i-1][5] == 'header'):
            if(df[i][4].lower() == 'engine') or (df[i][4].lower() == 'engines'):
                pass
            else:
                df[i][5] = 'header'
    return df
```

def allmodel_is_header is defining all model section as the model header.

```
def allmodel_is_header(df):

    for i in range(0, len(df)):
        if(df[i][5] == 'key_term') & (df[i][4].lower().find('all') > 0):
            if(df[i][4].lower().find('model') > 0) or (df[i][4].lower().find('models') > 0 ):
                df[i][5] = 'header'


    return df
```

def df_rev is extracting the top-right block at the first page of TCDSs and collect it in the 1st row in csv file.

```
def df_rev(df):
    fkt = -1
    for i in range(0, len(df)):
        if(df[i][5] == 'key_term') & (fkt < 0):
            fkt = i
#    print(fkt)
    for i in range(0, fkt):
        if(df[i][0] > 300):
            df[i][5] = 'rev'

    return df
```

def merge_all is merging everything according all above conditions together in each categories.

```
def merge_all(df):
    df += [[0,0,0,0,0,0,0]]
    lst = []
    out = []
    index = 0
    num = 0
    for i in range(0, len(df)-1):
        if(df[i][5] == 'header'):
            if(df[i+1][5] == 'header'):
                lst += [df[i][4]]
                num += 1
            else:
                out += [[]]
                lst += [df[i][4]]
                lst = [' \n '.join(lst)]
                lst += [df[i][5]]
                lst.insert( 0, df[i-num][6])
                lst.insert( 1, df[i][1])
                lst.insert( 2, df[i][6])
                lst.insert( 3, df[i][3])
                # out put will be [start page, y start, end page, y end, text, class]
                out[index] = lst
                num = 0
                index += 1
                lst = []

        elif(df[i][5] == 'key_term'):
            out += [[]]
            lst += [df[i][6], df[i][1], df[i][6], df[i][3], df[i][4], df[i][5]]
            out[index] = lst
            index += 1
            lst = []

        elif(df[i][5] == 'info'):
            if(df[i+1][5] == 'info'):
                lst += [df[i][4]]
                num += 1
            else:
                out += [[]]
                lst += [df[i][4]]
                lst = [' \n '.join(lst)]
                lst += [df[i][5]]
                lst.insert( 0, df[i-num][6])
                lst.insert( 1, df[i][1])
                lst.insert( 2, df[i][6])
                lst.insert( 3, df[i][3])
                # out put will be [start page, y start, end page, y end, text, class]
                out[index] = lst
                num = 0
                index += 1
                lst = []
```

```
        elif(df[i][5] == 'rev'):
            if(df[i+1][5] == 'rev'):
                lst += [df[i][4]]
                num += 1
            else:
                out += [[]]
                lst += [df[i][4]]
                lst = [' \n '.join(lst)]
                lst += [df[i][5]]
                lst.insert( 0, df[i-num][6])
                lst.insert( 1, df[i][1])
                lst.insert( 2, df[i][6])
                lst.insert( 3, df[i][3])
                # out put will be [start page, y start, end page, y end, text, class]
                out[index] = lst
                num = 0
                index += 1
                lst = []
    return out
```

def prep_final_output is adding TCDS revision and model header in the first column. Then if the keyterm is found, store it in the first column, follow by its info. In case, there is keyterm without info, that keyterm will be dropped.

```
def prep_final_output(df):
    df += [[0,0,0,0,0,0]]
    index = 0
    lst = []
    out = []

    for i in range(0, len(df)-1):
        if(df[i][5] == 'rev') or (df[i][5] == 'header'):
            out += [[]]
            lst += [df[i][4], '']
            out[index] = lst

            lst = []
            index += 1

        elif(df[i][5] == 'key_term') & (df[i+1][5] == 'info'):
            lst += [df[i][4]]
        elif(df[i][5] == 'info') & (len(lst) > 0):
            out += [[]]
            lst += [df[i][4]]
            out[index] = lst

            lst =[]
            index += 1

    return out
```

Loop function

```
path = "/Users/bbteela/Desktop/Work/TCDS-Extraction-Aviators/dataset_tcds"
list_files = os.walk(path)
dfinal = pd.DataFrame()
a = []
nof = []
scan = []
for root, dirs, files in os.walk(path):
    for file in files:
        if not file.startswith("."):
            p = os.path.join(root,file)
            print(p)
            pdf = fitz.open(p)
            if pdf == None:
                scan.append(file)
                pass
            else:
                all_text = get_all_text(pdf)
                merged_words = word_line(all_text)
#-----------------------------------
# Sometimes the y coordinates in the same line will be a little different,
# the smaller y, the higher position
# example: A & B in the same line
# y0 of A ==1, y0 of B == 0.9
# the order will be changed to B A since the y is more sensitive condition of the coming reorder work
# This function is to make the texts in the same line have the same y
                Sy = Same_y(merged_words)

#--------------------------------------
#sorted key + lambda
# ref: https://blog.csdn.net/w417950004/article/details/86253721
# reorder the word lines by coordinates (page > y0 > x0)
                St = sorted(Sy, key = lambda x:(int(x[8]), float(x[1]), float(x[0])))

                fuel_xl = get_fuel_xl(St)
                if fuel_xl == None:
                    nof.append(file)
                    pdf.close()
                else:
                    khi = detect_khi(St)
#for i in range(0, len(khi)):
#    if(khi[i][8] == 'header'):
#        print(khi[i][4])
#Cleaned khi
                    Ckhi = Drop_blw(khi)
```

Replace your directory in here where contains of desired TCDS files

```
#-------------------------------------------------------------------------
# Sometimes the header's number will be detect as key term, this function is used to solve it.
#example: real header == I. Model xyz....  and the df might be ['I.', 'key_term'], ['Model xyz...', 'header']
# hder function is to catch 'I.' form keyterm to header.
                    Ckhi = hder(Ckhi)


#-----------------------
#clean the page stuff
#fyl function will retun the yl of the first lines
                    fyl = firsty1(Ckhi)
#use the max value to clean the tc titles which has been detect as KT
                    drop_fline(Ckhi, fyl)
#-------------------------------------------------------------------
                    C_cont = clean_cont(Ckhi)
#-----------------------------------------------------------------------------------------------
#drop punctuations of KT
#https://datagy.io/python-remove-punctuation-from-string/
                    for i in range(0, len(C_cont)-1):
                        if(C_cont[i][5] == 'key_term'):
                            C_cont[i][4] = C_cont[i][4].translate(str.maketrans('', '', string.punctuation))
#-----------------------------------------------------------------------------------------------
                    amh = allmodel_is_header(C_cont)
#-----------------------------------------------------------------------------------------------
# Find out where is the header 'all model',
                    i_am = -1
                    for i in range(0, len(amh)):
                        if(amh[i][5] == 'header'):
                            if(amh[i][4].lower().find('all model') > -1) or (amh[i][4].lower().find('all models') > -1):
                                #print(amh[i][4], i)
                                i_am = i
#-------------------------------------------
                    ah = all_header(amh, i_am)
#-------------------------------------------
                    df_rev(ah)

                    mall = merge_all(ah)
#for i in range(0, len(mall)):
#    print(mall[i])
                    fout = prep_final_output(mall)

                    b = pd.DataFrame(fout, columns=['Keyterm', 'Info'])
                    b.to_csv(rf'/Users/bbteela/Desktop/Work/TCDS-Extraction-Aviators/csv/{file}.csv', index=False)
                    a.append(b)

                #dfinal.loc[len(dfinal)] = fout
                    pdf.close()
                #for i in range(0, len(fout)):
                #    print(fout[i])

                #dfinal = pd.DataFrame(fout, columns=['L', 'R'])
                #dfinal

dfinal = pd.concat(a)
no_fuel = pd.DataFrame(nof)
pdf_scan =  pd.DataFrame(scan)
```

Replace your directory in here before {file}.csv

4. Visualization

The dataset is a collection of Type Certificate Data Sheet (TCDS) documents in PDF format. Extracting information from each key term is the objective of this project. This characteristic led to the primary idea of this project's text extraction—using coordinates. Initial tests performed on the data set are that of PDF conversion to text. There are many open-source library tools to extract PDF to text, and the team has tested packages such as PyPDF2, PyMuPDF, Tabula, PDFplumber, PDFminer, and Pdftotext to see if we get the desired output. All the above packages are great conversion tools, but they didn't pass our validation as we didn't get the expected output (all outputs mentioned in section 3). We focus on extracting by using PyMuPDF with coordinates for each word on the page. The coordinate function provides unique features such as block_no, line_no, and word_no for each word in each block. This result is accurate and can be proved by manually checking the pdf file. Using the coordinates of an imaginary vertical x-axis, we split the pdf file into two halves and extracted information on the left and right of the line. The tests done using this method is mainly trial and error to extract the desired information by modifying the conditions specified. Extraction using the line_no, block_no was also tested on some of the series of TCDS. We are using regular expressions for the model and serial numbers to get all the data in the serial number format. The desired output of this project is users can extract tables from PDF files and turn them straight into data frames using the Python programming language with more than 75% coverage accuracy.

4.1 Pure Text Method

The first method is known as Pure Text or text layer. All TCDSs have a similar sequence of information and this concept helps to extract information between each key term. The procedure is searching for key terms which are one below the other and then extract the information between them. This is the simplest method to extract information. However, the accuracy of this method is low. The accuracy of getting each key term correctly is provided in the below graph.

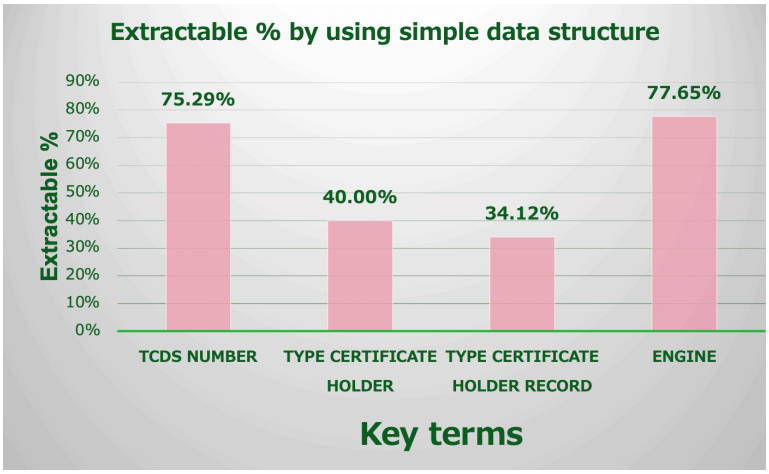


Figure 27 The accuracy of extracting each key term

We measured how each key term can get its information perfectly. The accuracy of extracting TCDS number and Engine are 75.29% and 77.65% respectively which is satisfying. However, the accuracy of extracting Type certificate holder and Type certificate holder record are low. The reasons are

1) Some TCDSs doesn't have Type certificate holder record, so the search function will not be able to detect the correct information
2) Various key word variations of key terms

From the pure text result, we can conclude that pure text method isn't suitable to this dataset due to low performance on some of the TCDS fields.

4.2 Coordinate Method

Based on Pure Text method output, we need to focus on the various key words of key terms. Extracting without using fixed key word for key terms would be the best solution in this case. PyMuPDF has a function to define the unique words feature including coordinate, line number, word group, and word length. Combining this function, we can define the condition of key terms, model headers, and information by ignoring the key words of key terms.

We measured accuracy of this method from how many TCDSs can extract all key terms and model header perfectly. The output accuracy is lower than what we expected from both key terms and model header variable. The key term extraction accuracy is only 34.88% and for model header is 44.19%.

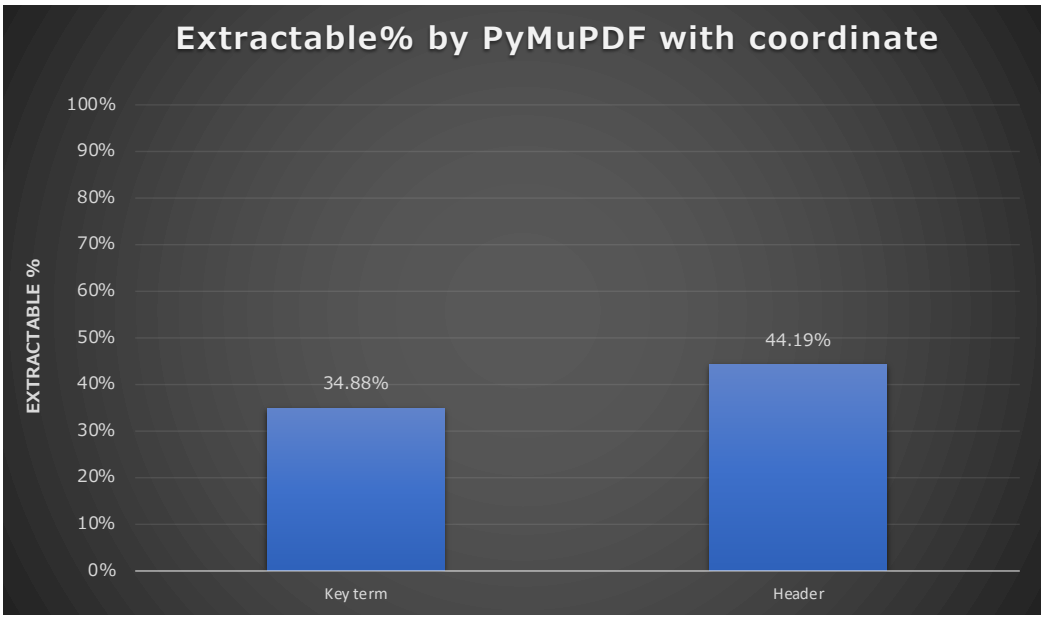


Figure 28 The accuracy of extracting key terms and model headers

After getting the output, we tried to find the reasons why the accuracy is lower than expected. Some of the TCDSs which have the same format also give an output which has very low accuracy. After looking through the unique feature of word coordinate, we found out that PyMuPDF might assign a different block_no or line_no for the information of the same key term.

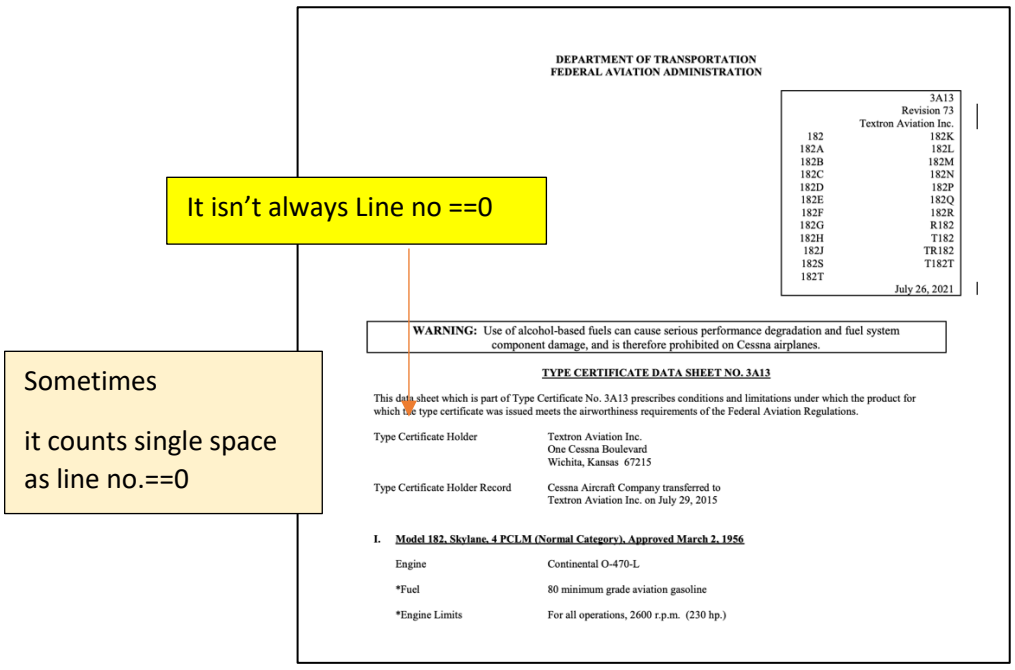


Figure 29 The line no. condition

Key terms	
Normal extraction	**W/O line no ==0**
Page No.	Page No.
Rev No.	Rev No.
Type Certificate Holder	**Type Certificate Holder**
Type Certificate Holder Record	**Type Certificate Holder Record**
Engine	**Engine**
Empty weight C.G. range	**Fuel**
1A3	Engine limits
I - Model 14-19 (con'd)	Airspeed limits
No. of seats	C.G. range (landing)
Serial Nos. eligible	gear extended)
Engine	Empty weight C.G. range
Empty weight C.G. range	Maximum weight
Maximum weight	1A3
II - Model 14-19-2 (cont'd)	I - Model 14-19 (con'd)
Fuel capacity	No. of seats
Serial Nos. eligible	Maximum baggage
Engine	Fuel capacity
Empty weight C.G. range	Oil capacity
Maximum weight	Control surface movements
1A3	(within + 1°)
Engine	**Serial Nos. eligible**
C.G. range (landing)	Required equipment
Empty weight C.G. range	**Engine**
Maximum weight	**Fuel**
Engine	Engine limits

Figure 30 Key term output before and after removing line no. == 0 out

Header	
Normal extraction	**W/O line no ==0**
I - Model 14-19, 4 PCLM (Normal Category), 2 PCL	I - Model 14-19, 4 PCLM (Normal Category), 2
II - Model 14-19-2, 4 PCLM (Normal Category), Ap	II - Model 14-19-2, 4 PCLM (Normal Category),
III - Model 14-19-3, 4 PCLM (Normal Category), A	III - Model 14-19-3, 4 PCLM (Normal Category),
IV - Model 14-19-3A, 4 PCLM (Normal Category), A	IV - Model 14-19-3A, 4 PCLM (Normal Category),
V - Model 17-30, 4 PCLM (Normal Category), Appro	V - Model 17-30, 4 PCLM (Normal Category), Ap
VI - Model 17-31TC, 4 PCLM (Normal Category), Ap	VI - Model 17-31TC, 4 PCLM (Normal Category),
VII - Model 17-31, 4 PCLM (Normal Category), App	VII - Model 17-31, 4 PCLM (Normal Category),

Figure 31 Header output before and after removing line no. == 0 out

The output shows that removing line no. == 0 condition gives the better extraction version of key terms. Bold represents the desired key terms. Without line no. ==0 can extract every key term in order and cover all desired key terms.

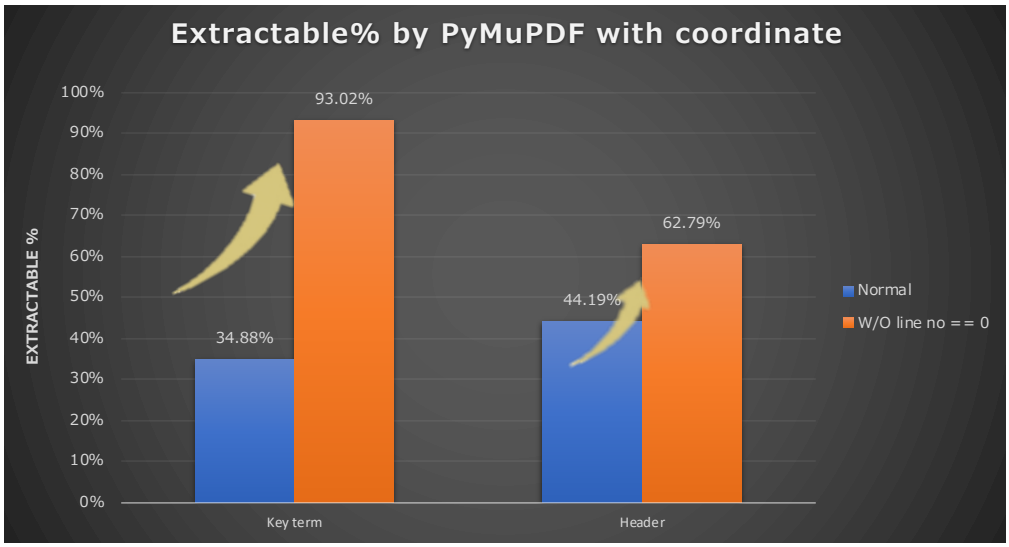


Figure 32 The accuracy of extracting key terms and model headers

After removing line no ==0 condition, the extraction accuracy increases for both key term and header. For key terms, they can be extracted to 93.02% which has increased drastically from previous condition. The above graph shows the visualization for the output.

4.3 Analysis

The text extraction methods had a lot of unexpected characters and disordered lines in the output. Results of the tests are not consistent with what we expected. As discussed in the section 3 the output of PdfPlumber, Pdfminer and Pdftotext are good but have strange characters or line disorientation in the output. But PyPDF2 and PyMuPDF almost have the similar output and are good tools for extraction as we got the perfect output as the document file. After successful extraction of text on the pdf file, the unique ability of PyMuPDF of getting the coordinates of each block was tested on the files. With this we get the coordinates for each block on the page. PyMuPDF assigns block numbers for information on lines present on the page. Some of the TCDS gave us the expected output while extraction using the line_no and block_no while the others failed.

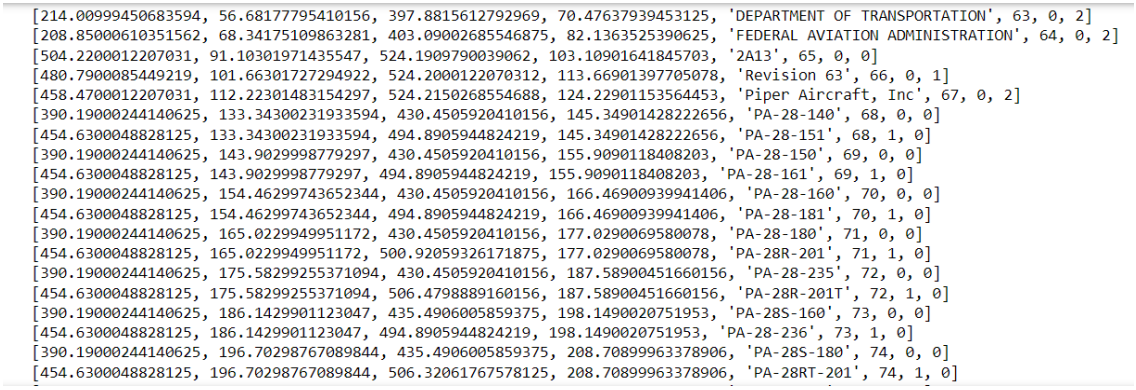


Figure 33 The line_no and block_no from PyMuPDF

From our analysis we have observed that most of the A series TCDS works well with the extraction method using the line_no and block_no. Below is the figure with highlighted parts which have been extracted perfectly. Yellow color highlighted information are the key terms of each model, blue color highlighted text is the information related to the Key term, and the green color highlighted text is the model header.

TYPE CERTIFICATE DATA SHEET NO. 2A13

This data sheet, which is a part of Type Certificate 2A13, prescribes conditions and limitations under which the product for which the type certificate was issued meets the airworthiness requirements of the Federal Aviation Regulations.

Type Certificate Holder	Piper Aircraft, Inc. 2926 Piper Drive Vero Beach, Florida 32960
Type Certificate Holder Record	The New Piper Aircraft, Inc transferred TC 2A13 to Piper Aircraft, Inc on August 7, 2006.

I - Model PA-28-160 (Cherokee), 4 PCLM (Normal Category), Approved October 31, 1960, for S/N 28-03; 28-1 through 28-4377; and 28-1760A.

Engine	Lycoming O-320-B2B or O-320-D2A with carburetor setting 10-3678-32
Fuel	91/96 minimum grade aviation gasoline
Engine Limits	For all operations, 2700 r.p.m. (160 hp)
Propeller and Propeller Limits	Sensenich M74DM or 74DM6 on S/N 28-1 through 28-1760, and 28-1760A. Sensenich M74DMS or 74D6S5 on S/N 28-1761 through 28-4377. Static r.p.m. at maximum permission throttle setting not over 2425 r.p.m., not under 2325 r.p.m. No additional tolerance permitted. Diameter: Not over 74", not under 72.5".
Propeller Spinner	Piper P/N 14422-00 on S/N 28-1 through 28-1760, and 28-1760A. Piper P/N 63760-04 or P/N 65805-00 on S/N 28-1761 through 28-4377. See NOTE 11.

Airspeed Limits	Never exceed	171 mph	(148 knots)	CAS
	Maximum structural cruising	140 mph	(121 knots)	CAS
	Maneuvering	129 mph	(112 knots)	CAS
	Flaps Extended	115 mph	(100 knots)	CAS

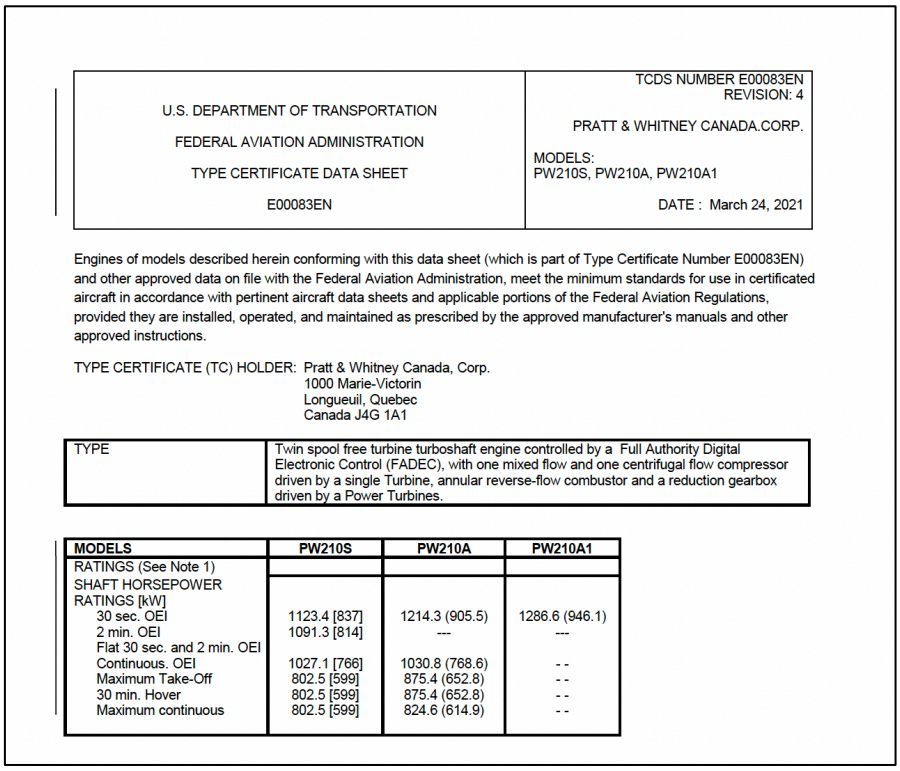
Figure 34 The output from coordinate system approach

Key_term	information
TCDS no.	2A13
Type Certificate Holder	Piper Aircraft, Inc. 2926 Piper Drive Vero Beach, Florida 32960
Type Certificate Holder Record	The New Piper Aircraft, Inc transferred TC 2A13 to Piper Aircraft, Inc on August 7, 2006.
I - Model 14-19, 4 PCLM (Normal Category), 2 PCLM (Utility Category), Approved September 26, 1949	
Engine	Lycoming O-320-B2B or O-320-D2A with carburetor setting 10-3678-32
Fuel	91/96 minimum grade aviation gasoline
Engine Limits	For all operations, 2700 r.p.m. (160 hp)
Propeller and Propeller Limits	Sensenich M74DM or 74DM6 on S/N 28-1 through 28-1760, and 28-1760A. Sensenich M74DMS or 74D6S5 on S/N 28-1761 through 28-4377. Static r.p.m. at maximum permission throttle setting not over 2425 r.p.m., no
Propeller Spinner	Piper P/N 14422-00 on S/N 28-1 through 28-1760, and 28-1760A. Piper P/N 63760-04 or P/N 65805-00 on S/N 28-1761 through 28-4377. See NOTE 11.
Empty Weight C. G. Range	None
Maximum Weight	2200 lb.
No. of Seats	4 (2 at +85.5, 2 at +118.1)
Maximum Baggage	125 lb. at (+142.8) on S/N 28-1 through 28-1760, and 28-1760A. See NOTE 8. 200 lb. at (+142.8) on S/N 28-1761 through 28-4377.
Fuel Capacity	50 gallons at (+95) (2 wing tanks) See NOTE 1 for data on system fuel.
Oil Capacity	8 quarts at (+32.5), 6 quarts usable See NOTE 1 for data on system oil.
Manufacturer's Serial Nos.	28-03; 28-1 through 28-4377; and 28-1760A.
II - Model 14-19-2, 4 PCLM (Normal Category), Approved January 7, 1957	
Engine	Lycoming O-320-A2B or O-320-E2A with carburetor setting 10-3678-32
Fuel	80/87 minimum grade aviation gasoline
Engine Limits	For all operations, 2700 r.p.m. (150 hp)
Propeller and Propeller Limits	Sensenich M74DM or 74DM6 on S/N 28-1 through 28-1760, and 28-1760A. Sensenich M74DMS or 74DM6S5 on S/N 28-1761 through 28-4377. Static r.p.m. at maximum permissible throttle setting not over 2375 r.p.m.,
Propeller Spinner	Piper P/N 14422-00 on S/N 28-1 through 28-1760, and 28-1760A. Piper P/N 63760-04 or 65805-00 on S/N 28-1761 through 28-4377. See NOTE 11.
Empty Wt. C. G. Range	None
Maximum Weight	2150 lb.
No. of Seats	4 (2 at +85.5, 2 at +118.1)
Maximum Baggage	125 lb. at (+142.8) on S/N 28-1 through 28-1760, and 28-1760A. See NOTE 8. 200 lb. at (+142.8) on S/N 28-1761 through 28-4377.
Fuel Capacity	50 gallons at (+95) (2 wing tanks) See NOTE 1 for data on system fuel.
Oil Capacity	8 quarts at (+32.5) (6 quarts usable) See NOTE 1 for data on system oil.
Nose Wheel Travel	Left 30° Right 30° (± 2□) (Effective on S/N 28-03; 28-1 through 28-3377; and 28-1760A) Left 22° Right 22° (□ 2□) (Effective on S/N 28-3378 through 28-4377)
Manufacturer's Serial Nos.	28-03; 28-1 through 28-4377; and 28-1760A.
III - Model PA-28-180 (Cherokee), 4 PCLM (Normal Category), Approved August 3, 1962; 2 PCLM (Utility Category),	
Engine	Lycoming O-360-A3A or 0-360-A4A with carburetor setting 10-3878 or 10-4164-1
Fuel	91/96 minimum grade aviation gasoline
Engine Limits	S/N 28-671 through 28-1760, and 28-1760A (except S/N 28-1571 and S/N 28-1573) (See NOTE 4): Maximum permissible takeoff, 2475 r.p.m. For all other operations, 2700 r.p.m. (180 hp) S/N 28-1571; 28-1573; 28-1761 t
Propeller and Propeller Limits	Sensenich M76EMM or 76EM8 on S/N 28-671 through 28-1760, and 28-1760A (except S/N 28-1571 and S/N 28-1573). Sensenich M76EMMS or 76EM8S5 on S/N 28-1571, 28-1573; 28-1761 through 28-5859; and 28-7105
Propeller Spinner	Piper P/N 14422-00 on S/N 28-671 through 28-1760, and 28-1760A. Piper P/N 63760-04 or 65805-00 on S/N 28-1761 through 28-5859; and 28-7105001 through 28-7205318. See NOTE 11.
Center of Gravity Range	Utility Category (See NOTE 9) (+84.0) to (+86.5) at 1650 lb. or less (+85.8) to (+86.5) at 1950 lb. Normal Category (See NOTE 15) (S/N 28-671 through 28-5859) (+84.0) to (+95.9) at 1650 lb. or less (+85.9) to (+95.9) at
Empty Weight C. G. Range	None Page 5 of 47 2A13
Maximum Weight	Normal Category: 2400 lb. Utility Category: 1950 lb.
No. of Seats	Normal Category: 4 (2 at +85.5, 2 at +118.1) Utility Category: 2 (2 at +85.5)
Maximum Baggage	Eligible Normal Category Only: 125 lb. at (+142.8) on S/N 28-671 through 28-1760, and 28-1760A. See NOTE 8. 200 lb. at (+142.8) on S/N 28-1761 through 28-5859, and 28-7105001 through 28-7205318.
Fuel Capacity	50 gallons at (+95) (2 wing tanks) See NOTE 1 for data on system fuel.
Oil Capacity	8 quarts at (+32.5) (6 quarts usable) See NOTE 1 for data on system oil.

Figure 35 shows the output from coordinate system approach in Excel Workbook

But this is not the case with all the series. This method fails for E and other TC series TCDS. The output is not as expected, random lines and characters are being extracted as the output. The reasons are:

1) TC series TCDSs are old scanned PDF files, so PyMuPDF can't read and extract text correctly. It may need PyTesseract or OCR to help to read image and extract to text before setting condition.
2) E series is Engine TCDS which doesn't be written in two-column format, and most of them written in the table. Based on coordinate approach, more than two-column format cannot be extracted. It is possible, if add more conditions to detect each column header. However, FAA has set a very low priority for E series and extracting them is not mandatory for the team.



U.S. DEPARTMENT OF TRANSPORTATION FEDERAL AVIATION ADMINISTRATION TYPE CERTIFICATE DATA SHEET E00083EN	TCDS NUMBER E00083EN REVISION: 4 PRATT & WHITNEY CANADA.CORP. MODELS: PW210S, PW210A, PW210A1 DATE : March 24, 2021

Engines of models described herein conforming with this data sheet (which is part of Type Certificate Number E00083EN) and other approved data on file with the Federal Aviation Administration, meet the minimum standards for use in certificated aircraft in accordance with pertinent aircraft data sheets and applicable portions of the Federal Aviation Regulations, provided they are installed, operated, and maintained as prescribed by the approved manufacturer's manuals and other approved instructions.

TYPE CERTIFICATE (TC) HOLDER: Pratt & Whitney Canada, Corp.
1000 Marie-Victorin
Longueuil, Quebec
Canada J4G 1A1

TYPE	Twin spool free turbine turboshaft engine controlled by a Full Authority Digital Electronic Control (FADEC), with one mixed flow and one centrifugal flow compressor driven by a single Turbine, annular reverse-flow combustor and a reduction gearbox driven by a Power Turbines.

MODELS	PW210S	PW210A	PW210A1
RATINGS (See Note 1)			
SHAFT HORSEPOWER RATINGS [kW]			
30 sec. OEI	1123.4 [837]	1214.3 (905.5)	1286.6 (946.1)
2 min. OEI	1091.3 [814]	---	---
Flat 30 sec. and 2 min. OEI			
Continuous. OEI	1027.1 [766]	1030.8 (768.6)	- -
Maximum Take-Off	802.5 [599]	875.4 (652.8)	- -
30 min. Hover	802.5 [599]	875.4 (652.8)	- -
Maximum continuous	802.5 [599]	824.6 (614.9)	- -

Figure 36 Example of E series TCDS (E00083EN)

L	R
	TCDS NUMBER E00083EN
	REVISION: 4
	U.S. DEPARTMENT OF TRANSPORTATION
	PRATT & WHITNEY CANADA.CORP.
	FEDERAL AVIATION ADMINISTRATION
	MODELS:
	TYPE CERTIFICATE DATA SHEET PW210S, PW210A, PW210A1
	E00083EN DATE : March 24, 2021
Engines of models described herein conforming with this data sheet (which is part of Type Cer	
and other approved data on file with the Federal Aviation Administration, meet the minimun	
aircraft in accordance with pertinent aircraft data sheets and applicable portions of the Feder	
provided they are installed, operated, and maintained as prescribed by the approved manufa	
approved instructions.	
TYPE CERTIFICATE (TC) HOLDER: Pratt & Whitney Canada, Corp.	
	1000 Marie-Victorin
	Longueuil, Quebec
	Canada J4G 1A1
TYPE	
	Twin spool free turbine turboshaft engine controlled by a Full Authority Digital
	Electronic Control (FADEC), with one mixed flow and one centrifugal flow compressor
	driven by a single Turbine, annular reverse-flow combustor and a reduction gearbox
	driven by a Power Turbines.
MODELS	
	PW210S PW210A PW210A1
RATINGS (See Note 1)	
SHAFT HORSEPOWER	
RATINGS [kW]	
30 sec. OEI	
	1123.4 [837] 1214.3 (905.5) 1286.6 (946.1)
2 min. OEI	
	1091.3 [814] --- ---
Flat 30 sec. and 2 min. OEI	
Continuous. OEI	
	1027.1 [766] 1030.8 (768.6) --

Figure 37 the extraction output from E00083EN

Our solution does not generalize well to out-of-sample data, it only works well with Type Certificate Data Sheets. This is because the goal of the project is mainly inclined towards creating a pipeline where we transform unstructured data to structured data. Unstructured data is in the form of TCDS in the dataset and our duty is to convert them into a structured format and store it in the form of a dataframe or excel workbook where it is easier for the FAA to navigate and pull up required information when needed. Hence this procedure only works well with 2 column format TCDS's. Incase this is applied to other real-world data; the program fails completely and gives you an error.

This solution is just a path opener for this kind of issue faced. There are many machine learning techniques and NLP approaches to tackle the issue. Advanced machine learning tools can be implemented to get the desired output. NLP can also be implemented by teaching the algorithm what should be read and what the machine should understand when it reads specific data. Regular expressions can also be very useful to extract the model and serial numbers. As the serial number format changes in some TCDS, Regex should be accurate to extract the data.

5. Findings

From our analysis, we see that using the PureText method can be very helpful when you need to extract information from two specific key terms from a TCDS. In this way, you need not worry about the variations in the key terms as you only need the specific information between two desired key terms. But when you try to automate this method, it fails because of the variations in the key terms. Hence, when one is sure of the information he needs it can be successfully extracted manually. In the below figure, the yellow color highlighted texts are the key terms, and the blue highlighted text is the information. Here the information between the Type Certificate Holder and Type Certificate Holder Record is extracted and the extracted output belongs to the Type Certificate Holder Key term.

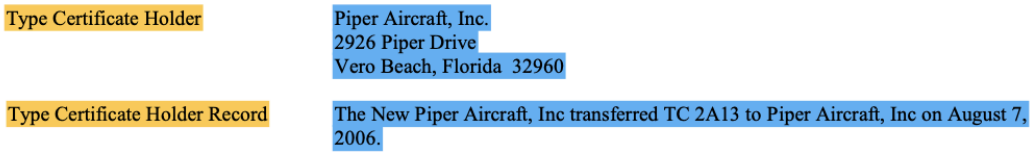


Figure 38 shows the output for pure text method

Next, the second method i.e the coordinate method is the usage of unique features of PyMuPdf as described in section 4. From our analysis, we see that this is a successful approach to getting our desired output. This method works perfectly well for two-column TCDSs. At first, the outputs from this method were not as we expected because of line no. == 0 condition. After examination and removal of the line number == 0 condition, the key phrases can be extracted more effectively. The algorithm may extract each keyword in sequence and cover all needed key terms without the line condition. This method can also be used to extract information from TCDS which do not follow the two-column format by adding appropriate conditions.

6. Summary

Transforming unstructured data into a structured format where the partners will be able to easily extract information is the goal of the project. We were given pdf documents of over 2000 files belonging to different TC formats from which we needed to extract information from over 1000 pdfs as per the weightage given to us by the FAA. By using the methods explained in the earlier sections we can successfully say that we have solved the problem that was given to us. We have extracted the information of all the key terms, headers, and metadata into an excel workbook where it is easier for the user to extract the data when needed.

Our method only functions properly with Type Certificate Data Sheets; it does not generalize well to out-of-sample data. This process is only effective with TCDSs in the 2-column format. If this is used with different types of real-world data, the application will fully crash and give you an error. This is because the conditions used for the extraction are implemented keeping in mind the 2-column format. This algorithm will work well and be successful for other formats if the conditions are updated according to the input data. We can firmly say that this method will give you the required results for TCDS which are published 10 years in the future if they still follow the 2-column format. Extraction results can be improved by adding regular expressions for extracting the serial and model number formats.

7. Future Work

This solution just serves as a starting point for problems of this nature. Machine learning methods and NLP strategies are the future of our rapidly advancing tech world. NLP algorithms to teach what to read and what to interpret when it scans data will be very useful when you are tackling projects like this. Advanced machine learning where the algorithm itself decides what format of the file is sent and does the extraction is the future vision for this project. The model and serial numbers can also be extracted quite well using regular expressions. The regex should be precise when extracting the data because the serial number format varies in some TCDS. More work is needed dealing with scanned and old TCDSs, which have been abandoned due to not being the priority for the project. OCR techniques will be of huge help to extract text from scanned documents. This project is accomplished mainly using python libraries in jupyter notebook but in the future, more advanced tools like java programming can also be executed. The team would be more than happy to be involved if any future work is done on the project.

8. Appendix A: Code References

https://github.com/KaiChan20/Aviators/blob/main/README.md

9. Appendix B: Risk Section

9.1 Sprint 1 Risks

Risk Name	Description	Probability	Impact	Mitigation
Dataset	Too messy, too much time to clean it up	Medium	High	Understanding the data
Text extraction	Data might be lost, or unnecessary data might be extracted during conversion process	Medium	Medium	Identify unnecessary text before hand
Low accuracy	The accuracy of model could be low	Medium	High	Training and evaluation

Table 3 shows risks for sprint 1

The first sprint was all about getting to know the dataset and understand the partner requirements. We identified the above 3 risks after our initial analysis on the dataset given by the FAA. The first risk would be the dataset itself. It contained more than 2000 pdf files with different kind variations of the TCDS like the A-series, E-series, ATC-series and so on. The second risk is text extraction which is the primary goal of the project. The team was skeptical at first if we can perfectly extract all the information needed, but we managed to successfully get the required output in the end. The team anticipated the extracted text's accuracy being low which is the final risk. Although we could manage to get the output accuracy more than 75%.

9.2 Sprint 2 Risks

Risk Name	Description	Probability	Impact	Mitigation
TCDS structure	The format changes for each manufacturer	Medium	High	Manual extraction

Text extraction	Different format date for each writer during conversion process	Medium	Low	After extraction, set the standard date format
Data Loss	Loss of data while extracting it through coordinates	Medium	Medium	Trail and improve
Time	It will take lots of time to define coordinates if we choose non-NLP approach	High	Low	Work hard, and more communication

Table 4 shows risks for sprint 2

Sprint 2 was Data Acquisition. Team understood in detail about the dataset and preprocessing of data was performed. We identified the above 4 risks during our data exploration. First being the TCDS structure which means the format of every TCDS changes for each manufacturer. This was a big concern for us, and manual extraction was the mitigation we chose to solve this issue. Extraction of text is the second risk as the data format is not consistent in all the TCDS's. Data loss using the coordinate system was the third risk. This had a significant impact but trail and improving the conditions for code was the solution. Final risk was time concern. We had 2 approaches in mind going forward i.e., using NLP and the non-NLP approach. Both approaches were heavily time-consuming but hard work was key.

9.3 Sprint 3 Risks

Risk Name	**Description**	**Probability**	**Impact**	**Mitigation**
Unable extracted data	Some information cannot be extracted due to different conditions	Medium	Medium	1. Trial and improve script by adding other conditions 2. Also studying regular regression for 2nd plan
Unreadable data	Some old PDFs were scanned. Since the quality of file may be low, the extraction accuracy is low	Low	Low	Add other libraries for scanned PDF such as Pytessaract (FAA set this type of TCDS series as low priority)

Table 5 shows risks for sprint 3

'Analytics and Algorithms' was the third sprint where we applied the algorithm using the techniques introduced in the earlier section. Two major issues we found were the unreadable data and data not being extracted because of different conditions. The unreadable data was because the PDF files were old-scanned documents which was solved using Pytesseract /OCR techniques. Team decided that we can overcome the information not being extracted concern by using the Regular Expression approach especially for serial and model numbers and improving our condition to extract key terms.

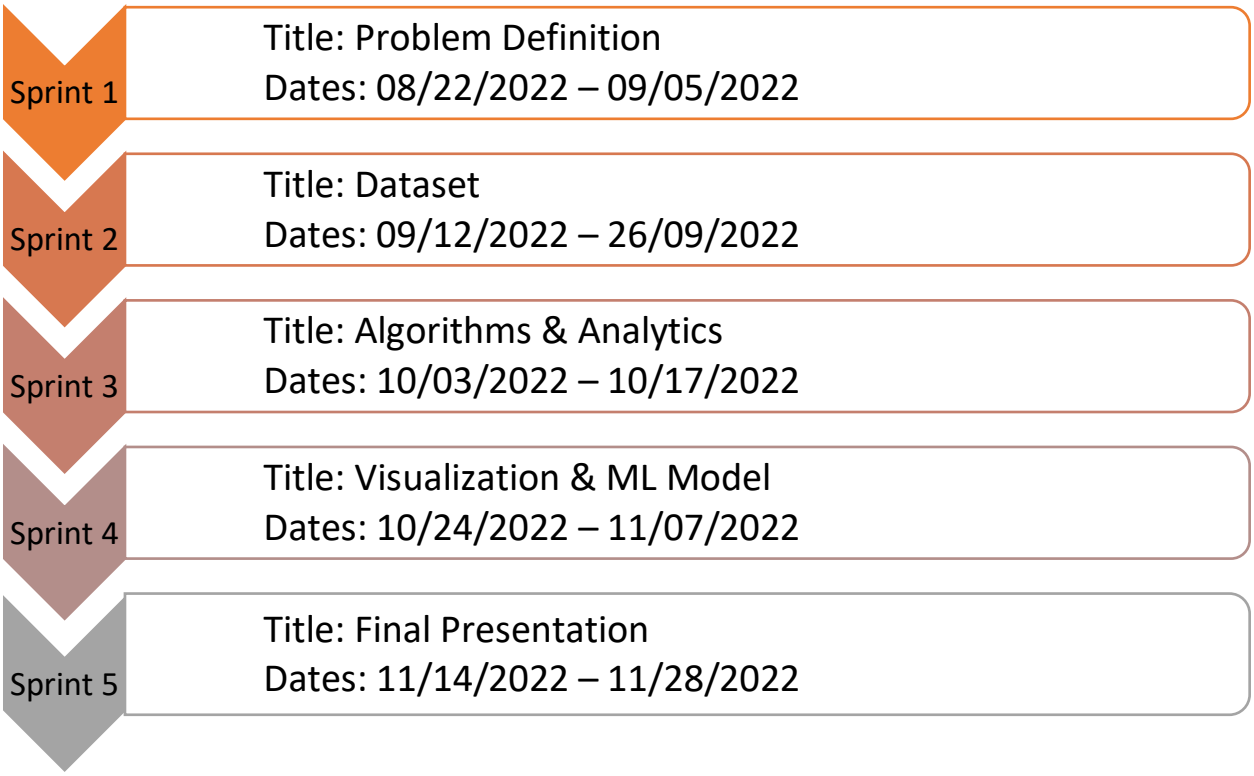
9.4 Sprint 4 Risks

Risk Name	**Description**	**Probability**	**Impact**	**Mitigation**
Accuracy check	No automatic way to check accuracy for all variations of TCDS.	High	Low	We need to manually check accuracy for each variation of TCDS.
Noise after extraction	Unwanted info.	High	Low	Apply conditions to clean the output
The unclear excel format	Same information stores in different excel cell	High	Low	Group the information to be in the same block no and collect them in the same cell

Table 6 shows risks for sprint4

Next sprint is the Visualizations and checking for the accuracy of the output from the previous sprint. We identified 3 main risks affecting our project. First one being the accuracy check because we couldn't automate the process of comparing the pdf and the output result. Team manually checked for the accuracy percentage. Second risk would be the unwanted information being extracted with the result and we classified it as Noise. This is a major issue, and we overcame it by using specific conditions to limit the noise. Third risk was the information that is supposed to be together, was split by PyMuPDF's block assign system and scattered into different cells when extracted. Team needed to group all the information into the same block and collect them in a single cell.

10. Sprint 5 Risks



Communication and effective collaboration are key components required for achieving team goals. This is exactly what the Scrum methodology has been designed for. The team conducted daily scrum, segregating tasks based on individual expertise. Though the process seemed new and complicated at first, the team quickly learned to integrate YouTrack into daily project objectives. We were also successful in enabling the team to be more in sync, identify problems and find effective solutions for impediments.

10.1 Sprint 1 Analysis

To solve a problem, it was important to first understand the problem given by the Federal Aviation Administration (FAA) and the need for it to be resolved. Our first task was to understand the current working system used by FAA and why it exhibits shortcomings. Since the objective was to convert unstructured data into a structured format, the team started by performing research on various related areas of study. The team effectively segregated tasks and helped each other. Managing activities during this Sprint was not very easy as it was our first time working with a project management framework. Working with YouTrack requires effective planning and communication, and we hoped to have managed that in a more efficient way.

10.2 Sprint 2 Analysis

Studying and understanding the dataset was a priority due to the different variations we noticed during initial exploration. The dataset was quite large (2200 PDFs), some dating back to as early as early 1900s. Hence the team decided to divide the dataset based on their type and individually study and understand the key information and the variation they exhibit from one another. We then created a priority list from the dataset and set them up to be used for analysis.

10.3 Sprint 3 Analysis

Once we studied and divided the dataset, we needed to find algorithms and techniques that can be used to extract information. We identified certain Python libraries that can be used to convert PDF to text, which can then be used to extract information from. Since we identified multiple libraries, the team members decided to work on one library each and compare the accuracy of extracted text to choose the optimal library for the project.

10.4 Sprint 4 Analysis

Since the dataset we are primarily a set of PDFs, and the task on hand was to perform extraction of insights, It was challenging to come up with visualizations. Hence the team created visualizations based on extractability of the TCDSs and the accuracy of extracted text.

10.5 Sprint 5 Analysis

The final Sprint was a task for the entire team as we had to put together everything we've worked on and created across all the previous Sprints efficiently. The team divided responsibilities for creating final presentation slides, as well as the final report, and later swapped roles to provide feedback and make improvements on the work.

11 References

[1] Wikipedia, "Federal Aviation Administration," 2022. [Online]. Available: https://en.wikipedia.org/wiki/Federal_Aviation_Administration.

[2] "FAA History," [Online]. Available: https://www.faa.gov/about/history/milestones/media/First_Type_Certificate.pdf.

[3] FAA, June 2022. [Online]. Available: https://www.faa.gov/aircraft/air_cert/airworthiness_certification/aw_overview.

[4] Ashish, "Data Extraction from Unstructured PDFs," 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/data-extraction-from-unstructured-pdfs/.

[5] "Geeks for Geeks," 2020. [Online]. Available: https://www.geeksforgeeks.org/how-to-extract-text-from-images-with-python/.

[6] "Text Mining in Python: Steps and Examples," 2022. [Online]. Available: https://www.kdnuggets.com/2020/05/text-mining-python-steps-examples.html.

[7] "Demystifying BERT: A Comprehensive Guide to the Groundbreaking NLP Framework," 2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/.

[8] FAA, 2015. [Online]. Available: https://www.faa.gov/documentLibrary/media/Order/FAA_Order_8110.121.pdf.

[9] "PyMuPDF," 2022. [Online]. Available: https://pymupdf.readthedocs.io/en/latest/app1.html.