KUN BAO

Predicting the status of flights using data analysis and machine learning

DEGREE PROGRAMME IN DATA ENGINEERING
2023

ABSTRACT

Kun, Bao: Predicting the status of flights using data analysis and machine learning

Bachelor’s thesis
Degree programme [Data engineering]
Month Year [October 2023]
Number of pages: 34

Abstract:
As the travel industry grows, so do the demanding aspects of air transport. The accuracy of flight status is critical to the travelling experience of passengers, the scheduling of airlines and the cost of airport operations. Most of the airports nowadays are forecasting flight routes to make airport scheduling more accurate, save time and cost, and improve travellers' experience. The use of data analytics and machine learning to predict multiple flight states will significantly improve the accuracy of the results and satisfy the needs of travellers, airlines and airport operations to a greater extent. This thesis will show how to use data analytics combined with machine learning models for flight status analysis and explain the conclusions.

Keywords: flight status, data analysis, machine learning, prediction,

CONTENTS

1 INTRODUCTION 4
2 LITERATURE REVIEW 5
  2.1 Overview of aviation industry trends 5
  2.2 Previous research on flights status prediction 7
  2.3 Data source, datasets and tools use in flight prediction 7
3 DATA PROCESSING 9
  3.1 Sources of flights data 9
  3.2 Data preparation 12
  3.3 Data analysis 13
4 MACHINE LEARNING AND PREDICTION 21
  4.1 Prediction model selection 21
  4.2 Type of algorithms 22
  4.3 Feature selections 23
    4.3.1Feature columns selections 23
    4.3.2Data type processing 24
    4.3.3Data scala 25
  4.4 Model training and validation 27
    4.4.1Selection of feature matrix (x) and target vector (y) 27
    4.4.2Training and testing datasets divided 28
    4.4.3Feature normalisation 29
    4.4.4Creating and training models 29
    4.4.5Model validation 30
5 CONCLUSION 31
REFERENCES 33

1 INTRODUCTION

Today's world has entered the digital age of transport interoperability, and data analytics has become an important tool in every field, and the aviation industry is no exception. Aviation operations involve a large amount of data information, including weather, flight data, etc., the analysis of which is critical to predicting flight status, improving operational efficiency and enhancing the passenger experience. Therefore, airlines as well as airport managers, including research experts, are committed to improving the accuracy of flight forecasts to meet the rapidly evolving needs of the air transport industry.

This study is dedicated to an in-depth exploration of the application of data analytics in flight status prediction, aiming to provide a data analytics-based approach to improve the existing prediction models and make the analyses more accurate. Changes in flight status not only affect passengers' travelling experience, but also increase airport operating costs and airlines' flight expenditures. Passengers who waste their time and are dissatisfied with the airline and airport arrangements may become disgusted and affect the airline's revenue. Flight status prediction is not only to meet the needs of airlines in predicting flight status and scheduling flights, but also for the sake of passengers' travelling experience, to save unnecessary operating expenses, to reduce airport operating costs, and to increase airline revenues.

There are a variety of data analysis tools and methods available to help us better understand the root causes of flight delays and cancellations (e.g. weather, airport traffic, traffic control, etc.). By analysing historical data from previous years, we can highlight the main factors that lead to flight delays or cancellations, and use machine learning and various data analytics tools to predict and model the flight status in advance, and take measures to improve the passenger experience, increase airline revenues and reduce airport operating costs.

This study will explore the application of data analytics methods in flight status prediction, including data collection, processing, feature engineering, model building and evaluation. The goal of this research is to construct predictive models after performing data analytics feature processing to make the data features more intuitive and salient, and to provide data to the airline industry to help them reduce the number of flight delays and cancellations, improve efficiency, and enhance the passenger experience.

The aim of this thesis is to analyse and investigate better ways of analysing data with machine learning prediction models to predict flight status more accurately.

Chapter 2 of this thesis describes in detail the data sources for the thesis and the research methods that will be used. Chapter 3 discusses the data analysis methodology in detail, and Chapter 4 will identify the machine learning model used, as well as how the model was constructed and trained. Finally, Chapter 5 will reflect on these steps and look for possible improvements.

2 LITERATURE REVIEW

2.1 Overview of aviation industry trends

Aviation is one of the fastest growing industries in the world today, with great prospects and a wide range of development areas. Whether it is business work or the development of tourism that makes the majority of passengers, workers and officers choose the comfort and convenience of air travel, or the booming development of the transport industry that makes air transport more and more in demand, the aviation industry has been constantly evolving to adapt to the different needs of different areas of today's society. The accuracy of flight status is crucial in today's society as it is related to cost and experience.

Global tourism is expanding rapidly and has become an important contributor to the national economies of many countries over the past decades. Over the past 25 years, worldwide international tourist arrivals have more than doubled, from 1.08 billion in 1995 to 2.4 billion in 2019 (WDG 2021). The rise in tourism has increased the demand for air transport and changes in flight status are a concern for travellers, and the increase in passenger traffic is a challenge for the air transport industry as well as for airports. As the impact of Covid wanes, not only the tourism industry, but also business travel is shifting from online to offline, and there are more exchanges and cooperation between countries and regions. Whether it's a conference or a reunion, the demand for punctuality in the aviation industry is very high, and in order to try to avoid the delay or cancellation of meetings due to delays or cancellations of flights, the importance of predicting the status of flights is even more prominent.

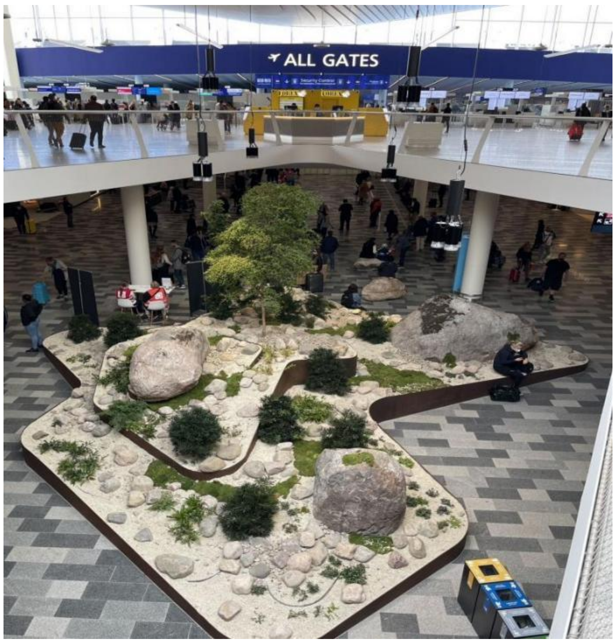


Figure1. Photo taken at Helsinki Vantaa Airport(2022)

2.2 Previous research on flights status prediction

In previous studies, scholars have focused on prediction through partitioning algorithms, density-based clustering algorithms and hybrid clustering algorithms. Flight trajectories and statuses are predicted and identified using the K-means algorithm, and the results obtained are compared with the existing anomalous flight statuses.

For the flight itself, it is difficult to quantify personal experience to assess the efficiency and availability of decision-making, which affects the status of the flight, as it can only rely on the pilot's empirical judgement when encountering different objective factors in the course of the flight. For the airlines themselves, it is the condition of the aircraft, whether it needs maintenance or whether there are any alerts that affect the flight status. On the ground handling side of the airport, flights may also be delayed and cancelled due to unforeseen circumstances.

Scholars have used various algorithms, including but not limited to ant colony algorithm, annealing algorithm, genetic algorithm and other intelligent algorithms, but intelligent algorithms are computationally intensive, and although intelligent algorithms can incorporate a lot of constraints in order to obtain the optimal solution, they tend to stick to the final optimal solution rather than achieving the global optimal, and they require more human and financial resources.

2.3 Data source, datasets and tools use in flight prediction

The source datasets used for this thesis is a public dataset from Kaggle website: Flight Status Predictions Kaggle's open datasets provides more options for researchers or people who are in the process of learning. kaggle datasets has greater transparency and usability because the kaggle community is an open and diverse community where people can rate the dataset, and people who is looking for a dataset can see more intuitively whether the dataset is usable, the mining and learning value that the dataset contains.

This dataset contains all the cancelled and delayed flight information from January 2018 onwards for each airline. The data is clear and intuitive, and is of high analytical value.

This dataset is a data-prepared dataset, i.e., it has been filtered for columns in the raw dataset that are mostly empty. The dataset contains csv files and parquet files for use. This allows for a variety of data analyses method and the all sorts selection of machine leaning models to be used for research of flight status prediction.

The data contained in the dataset is extracted from the Marketing Carrier "On-Time" Performance (from January 2018 to July 2022) datasheet of the "On-Time" database in the "TranStats" database. Real and valid, the dataset is highly analysable.

The various types of tools used to conduct the study are listed below:

Use Jupyter notebook and Google colab to create the code document.

Libraries for statistical analysis: Pandas(use for data cleaning, preprocessing and exploratory data analysis), NumPy(use for working with arrays and metrics).

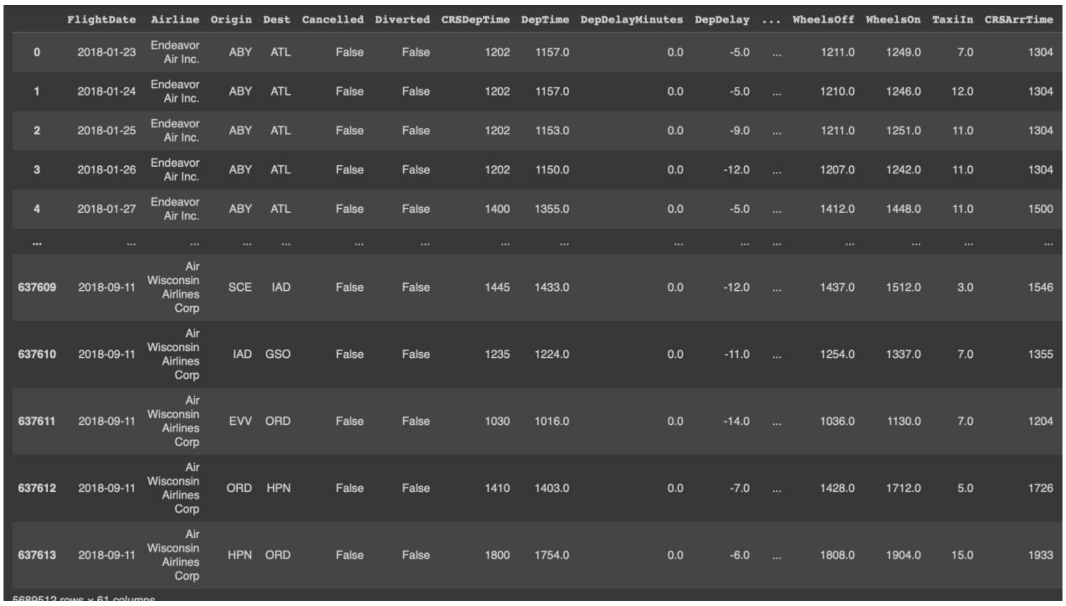
The image visualisation tools: Matplotlib and Seaborn, use for data and model visualization.

Machine learning methods libraries: All methods was based on library sklearn. Linear regression(use for establish linear relationships in data, e.g. for predicting numerical output variables), Decision Trees and Random Forests(use for classification and regression problems and can deal with non-linear relationships).

3 DATA PROCESSING

3.1 Sources of flight data

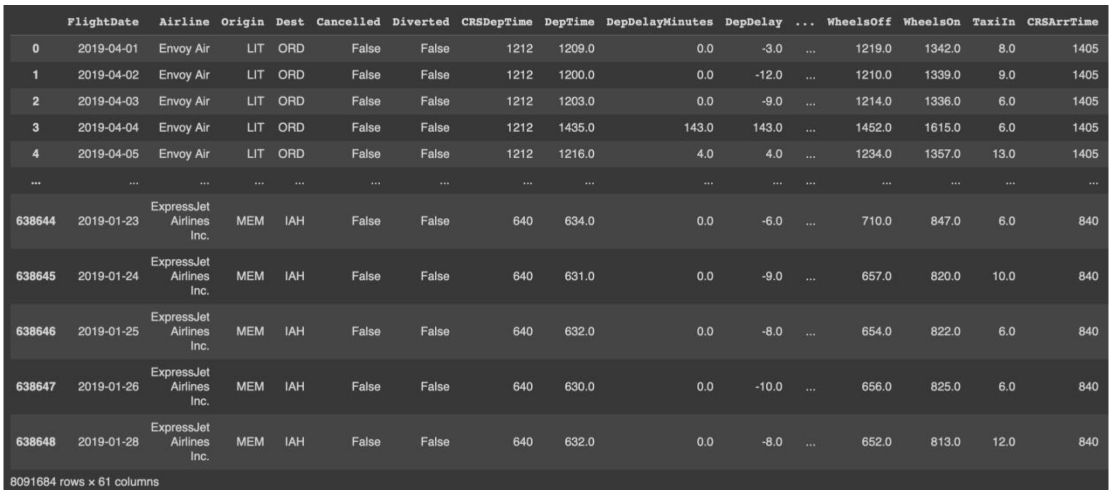
Data for each years:



	FlightDate	Airline	Origin	Dest	Cancelled	Diverted	CRSDepTime	DepTime	DepDelayMinutes	DepDelay	...	WheelsOff	WheelsOn	TaxiIn	CRSArrTime
0	2018-01-23	Endeavor Air Inc.	ABY	ATL	False	False	1202	1157.0	0.0	-5.0	...	1211.0	1249.0	7.0	1304
1	2018-01-24	Endeavor Air Inc.	ABY	ATL	False	False	1202	1157.0	0.0	-5.0	...	1210.0	1246.0	12.0	1304
2	2018-01-25	Endeavor Air Inc.	ABY	ATL	False	False	1202	1153.0	0.0	-9.0	...	1211.0	1251.0	11.0	1304
3	2018-01-26	Endeavor Air Inc.	ABY	ATL	False	False	1202	1150.0	0.0	-12.0	...	1207.0	1242.0	11.0	1304
4	2018-01-27	Endeavor Air Inc.	ABY	ATL	False	False	1400	1355.0	0.0	-5.0	...	1412.0	1448.0	11.0	1500
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
637609	2018-09-11	Air Wisconsin Airlines Corp	SCE	IAD	False	False	1445	1433.0	0.0	-12.0	...	1437.0	1512.0	3.0	1546
637610	2018-09-11	Air Wisconsin Airlines Corp	IAD	GSO	False	False	1235	1224.0	0.0	-11.0	...	1254.0	1337.0	7.0	1355
637611	2018-09-11	Air Wisconsin Airlines Corp	EVV	ORD	False	False	1030	1016.0	0.0	-14.0	...	1036.0	1130.0	7.0	1204
637612	2018-09-11	Air Wisconsin Airlines Corp	ORD	HPN	False	False	1410	1403.0	0.0	-7.0	...	1428.0	1712.0	5.0	1726
637613	2018-09-11	Air Wisconsin Airlines Corp	HPN	ORD	False	False	1800	1754.0	0.0	-6.0	...	1808.0	1904.0	15.0	1933

5689512 rows × 61 columns

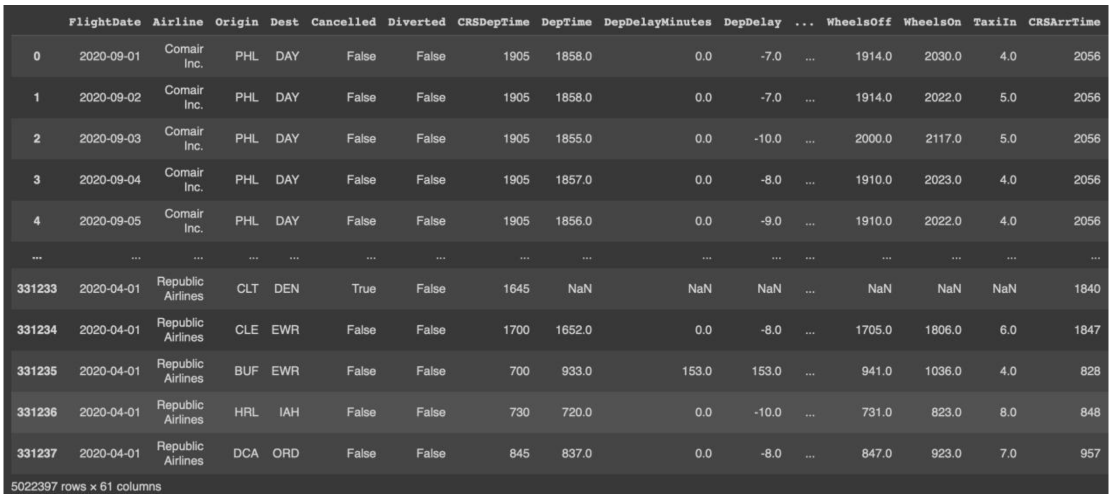
(Figure2: data for year 2018)



	FlightDate	Airline	Origin	Dest	Cancelled	Diverted	CRSDepTime	DepTime	DepDelayMinutes	DepDelay	...	WheelsOff	WheelsOn	TaxiIn	CRSArrTime
0	2019-04-01	Envoy Air	LIT	ORD	False	False	1212	1209.0	0.0	-3.0	...	1219.0	1342.0	8.0	1405
1	2019-04-02	Envoy Air	LIT	ORD	False	False	1212	1200.0	0.0	-12.0	...	1210.0	1339.0	9.0	1405
2	2019-04-03	Envoy Air	LIT	ORD	False	False	1212	1203.0	0.0	-9.0	...	1214.0	1336.0	6.0	1405
3	2019-04-04	Envoy Air	LIT	ORD	False	False	1212	1435.0	143.0	143.0	...	1452.0	1615.0	6.0	1405
4	2019-04-05	Envoy Air	LIT	ORD	False	False	1212	1216.0	4.0	4.0	...	1234.0	1357.0	13.0	1405
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
638644	2019-01-23	ExpressJet Airlines Inc.	MEM	IAH	False	False	640	634.0	0.0	-6.0	...	710.0	847.0	6.0	840
638645	2019-01-24	ExpressJet Airlines Inc.	MEM	IAH	False	False	640	631.0	0.0	-9.0	...	657.0	820.0	10.0	840
638646	2019-01-25	ExpressJet Airlines Inc.	MEM	IAH	False	False	640	632.0	0.0	-8.0	...	654.0	822.0	6.0	840
638647	2019-01-26	ExpressJet Airlines Inc.	MEM	IAH	False	False	640	630.0	0.0	-10.0	...	656.0	825.0	6.0	840
638648	2019-01-28	ExpressJet Airlines Inc.	MEM	IAH	False	False	640	632.0	0.0	-8.0	...	652.0	813.0	12.0	840

8091684 rows × 61 columns

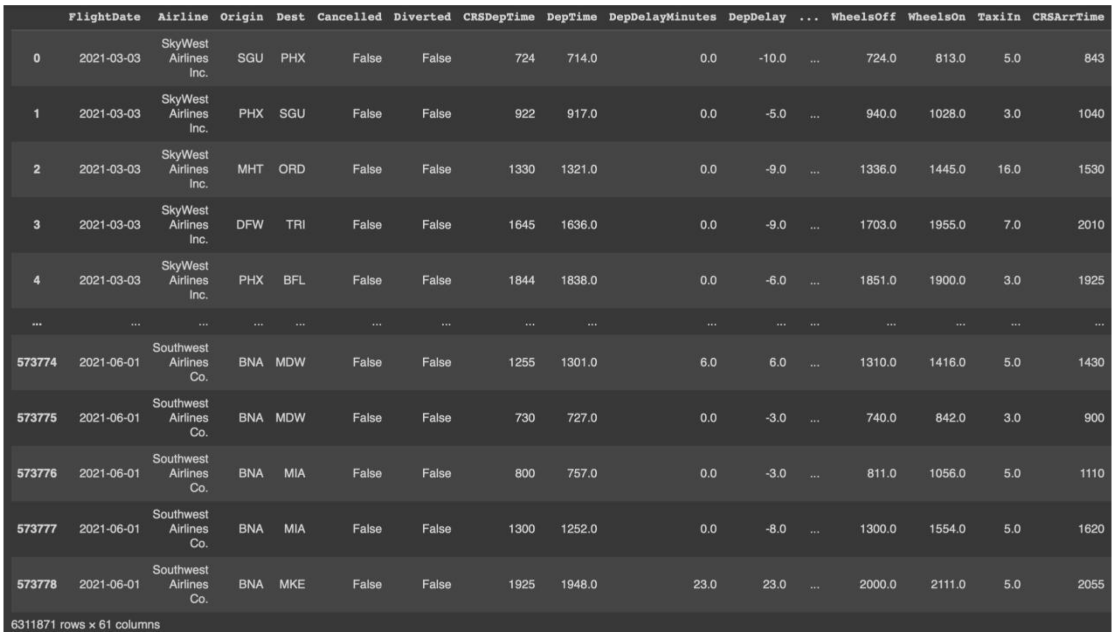
(Figure3: data for year 2019)



	FlightDate	Airline	Origin	Dest	Cancelled	Diverted	CRSDepTime	DepTime	DepDelayMinutes	DepDelay	...	WheelsOff	WheelsOn	TaxiIn	CRSArrTime
0	2020-09-01	Comair Inc.	PHL	DAY	False	False	1905	1858.0	0.0	-7.0	...	1914.0	2030.0	4.0	2056
1	2020-09-02	Comair Inc.	PHL	DAY	False	False	1905	1858.0	0.0	-7.0	...	1914.0	2022.0	5.0	2056
2	2020-09-03	Comair Inc.	PHL	DAY	False	False	1905	1855.0	0.0	-10.0	...	2000.0	2117.0	5.0	2056
3	2020-09-04	Comair Inc.	PHL	DAY	False	False	1905	1857.0	0.0	-8.0	...	1910.0	2023.0	4.0	2056
4	2020-09-05	Comair Inc.	PHL	DAY	False	False	1905	1856.0	0.0	-9.0	...	1910.0	2022.0	4.0	2056
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
331233	2020-04-01	Republic Airlines	CLT	DEN	True	False	1645	NaN	NaN	NaN	...	NaN	NaN	NaN	1840
331234	2020-04-01	Republic Airlines	CLE	EWR	False	False	1700	1652.0	0.0	-8.0	...	1705.0	1806.0	6.0	1847
331235	2020-04-01	Republic Airlines	BUF	EWR	False	False	700	933.0	153.0	153.0	...	941.0	1036.0	4.0	828
331236	2020-04-01	Republic Airlines	HRL	IAH	False	False	730	720.0	0.0	-10.0	...	731.0	823.0	8.0	848
331237	2020-04-01	Republic Airlines	DCA	ORD	False	False	845	837.0	0.0	-8.0	...	847.0	923.0	7.0	957

5022397 rows × 61 columns

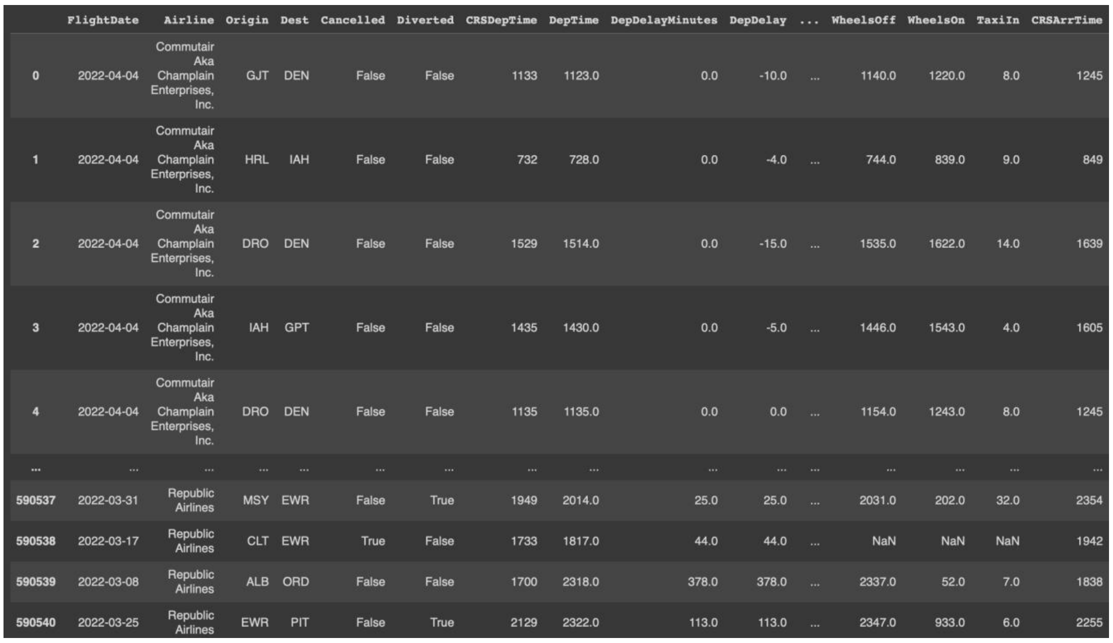
(Figure4: data for year 2020)



	FlightDate	Airline	Origin	Dest	Cancelled	Diverted	CRSDepTime	DepTime	DepDelayMinutes	DepDelay	...	WheelsOff	WheelsOn	TaxiIn	CRSArrTime
0	2021-03-03	SkyWest Airlines Inc.	SGU	PHX	False	False	724	714.0	0.0	-10.0	...	724.0	813.0	5.0	843
1	2021-03-03	SkyWest Airlines Inc.	PHX	SGU	False	False	922	917.0	0.0	-5.0	...	940.0	1028.0	3.0	1040
2	2021-03-03	SkyWest Airlines Inc.	MHT	ORD	False	False	1330	1321.0	0.0	-9.0	...	1336.0	1445.0	16.0	1530
3	2021-03-03	SkyWest Airlines Inc.	DFW	TRI	False	False	1645	1636.0	0.0	-9.0	...	1703.0	1955.0	7.0	2010
4	2021-03-03	SkyWest Airlines Inc.	PHX	BFL	False	False	1844	1838.0	0.0	-6.0	...	1851.0	1900.0	3.0	1925
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
573774	2021-06-01	Southwest Airlines Co.	BNA	MDW	False	False	1255	1301.0	6.0	6.0	...	1310.0	1416.0	5.0	1430
573775	2021-06-01	Southwest Airlines Co.	BNA	MDW	False	False	730	727.0	0.0	-3.0	...	740.0	842.0	3.0	900
573776	2021-06-01	Southwest Airlines Co.	BNA	MIA	False	False	800	757.0	0.0	-3.0	...	811.0	1056.0	5.0	1110
573777	2021-06-01	Southwest Airlines Co.	BNA	MIA	False	False	1300	1252.0	0.0	-8.0	...	1300.0	1554.0	5.0	1620
573778	2021-06-01	Southwest Airlines Co.	BNA	MKE	False	False	1925	1948.0	23.0	23.0	...	2000.0	2111.0	5.0	2055

6311871 rows × 61 columns

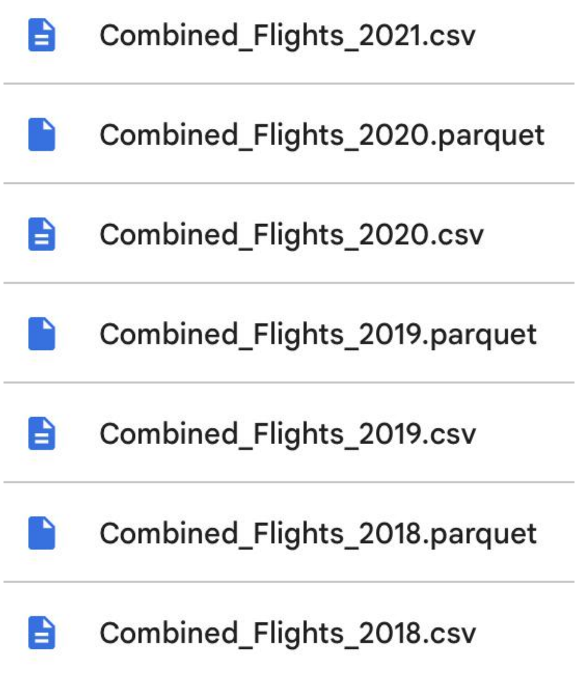
(Figure5: data for year 2021)



	FlightDate	Airline	Origin	Dest	Cancelled	Diverted	CRSDepTime	DepTime	DepDelayMinutes	DepDelay	...	WheelsOff	WheelsOn	TaxiIn	CRSArrTime
0	2022-04-04	Commutair Aka Champlain Enterprises, Inc.	GJT	DEN	False	False	1133	1123.0	0.0	-10.0	...	1140.0	1220.0	8.0	1245
1	2022-04-04	Commutair Aka Champlain Enterprises, Inc.	HRL	IAH	False	False	732	728.0	0.0	-4.0	...	744.0	839.0	9.0	849
2	2022-04-04	Commutair Aka Champlain Enterprises, Inc.	DRO	DEN	False	False	1529	1514.0	0.0	-15.0	...	1535.0	1622.0	14.0	1639
3	2022-04-04	Commutair Aka Champlain Enterprises, Inc.	IAH	GPT	False	False	1435	1430.0	0.0	-5.0	...	1446.0	1543.0	4.0	1605
4	2022-04-04	Commutair Aka Champlain Enterprises, Inc.	DRO	DEN	False	False	1135	1135.0	0.0	0.0	...	1154.0	1243.0	8.0	1245
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
590537	2022-03-31	Republic Airlines	MSY	EWR	False	True	1949	2014.0	25.0	25.0	...	2031.0	202.0	32.0	2354
590538	2022-03-17	Republic Airlines	CLT	EWR	True	False	1733	1817.0	44.0	44.0	...	NaN	NaN	NaN	1942
590539	2022-03-08	Republic Airlines	ALB	ORD	False	False	1700	2318.0	378.0	378.0	...	2337.0	52.0	7.0	1838
590540	2022-03-25	Republic Airlines	EWR	PIT	False	True	2129	2322.0	113.0	113.0	...	2347.0	933.0	6.0	2255

(Figure6: data for year 2022)

Total size of data is 4,5GB.



Combined_Flights_2021.csv

Combined_Flights_2020.parquet

Combined_Flights_2020.csv

Combined_Flights_2019.parquet

Combined_Flights_2019.csv

Combined_Flights_2018.parquet

Combined_Flights_2018.csv

(Figure7: total datasets review)

3.2 Data Preparation

Data preparation is a crucial step in data analysis, which involves transforming raw data into a state that is analytically valuable for the study and consistent with subsequent machine learning modelling.

After getting the data the first thing that should be done is to traverse the data and then perform data cleansing. This step involves dealing with missing values, duplicates, outliers and erroneous data in the data.

As this dataset is a clean dataset, the data processing step can be eliminated. And this dataset already contains parquet files, so you can use parquet files directly.

The next step is data transformation. Because the research conducted in this thesis requires the use of machine learning, the data needs to be transformed to include, but not limited to, normalisation, standardisation, and coding classification. This operation ensures that the required data meets the machine learning modelling standards and avoids any unnecessary hassle later on.

Following this process, we will perform feature engineering. This part performs feature extraction, integration and other operations for subsequent machine modelling.

The data is processed through these steps to better perform the following Explorative Data Analytics(EDA) part and machine learning modelling for flight state prediction.

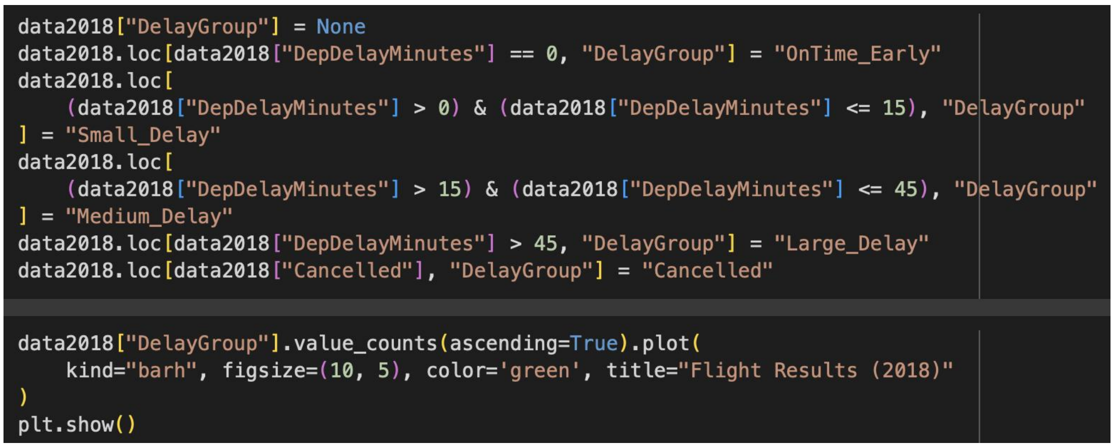
3.3 Data analysis

To the following Exploratory Data Analysis(EDA) section. After comparing the advantages and disadvantages of CSV files and Parquet files, I chose to use Parquet files for this part of the data analysis.

Parquet files are faster and easier to read, recall and modify than CSV files, I will explain the advantages and features in detail in the subsequent section on predicting flight status through machine learning.

First, use the basics pandas command line to access and examine the data information.

Here I give an example of year 2018:



```
data2018["DelayGroup"] = None
data2018.loc[data2018["DepDelayMinutes"] == 0, "DelayGroup"] = "OnTime_Early"
data2018.loc[
    (data2018["DepDelayMinutes"] > 0) & (data2018["DepDelayMinutes"] <= 15), "DelayGroup"
] = "Small_Delay"
data2018.loc[
    (data2018["DepDelayMinutes"] > 15) & (data2018["DepDelayMinutes"] <= 45), "DelayGroup"
] = "Medium_Delay"
data2018.loc[data2018["DepDelayMinutes"] > 45, "DelayGroup"] = "Large_Delay"
data2018.loc[data2018["Cancelled"], "DelayGroup"] = "Cancelled"
```

```
data2018["DelayGroup"].value_counts(ascending=True).plot(
    kind="barh", figsize=(10, 5), color='green', title="Flight Results (2018)"
)
plt.show()
```

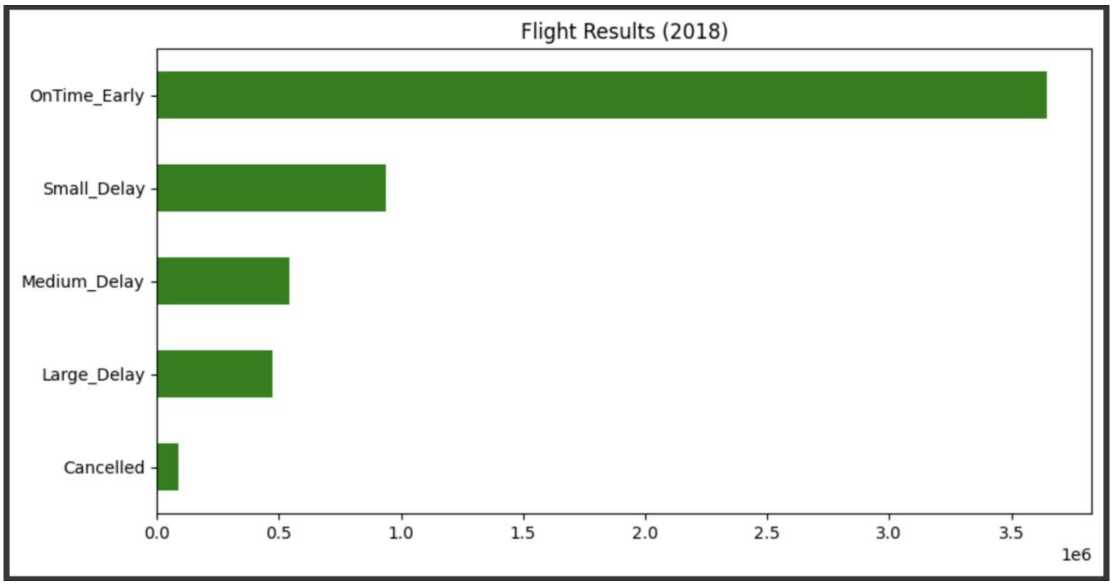
(Figure8: example code about data analysis)

To facilitate the analysis, it is necessary to further synthesise the status of the aircraft. The first step is to create and initialise a new column called "DelayGroup" and set all values to None, then group flights according to their delays or cancellations and record them in this column.

Assign "OnTime_Early" to flights where the value of "DepDelayMinutes" is equal to 0, which means that the flight is on time or early. Then assign "Small_Delay" to flights where the value of "DepDelayMinutes" is greater than 0 and less than or equal to 15, which means that the flight is slightly delayed and the delay is greater than 0 minutes and less than 15 minutes.

Assigning "Medium_Delay" to flights where the value of "DepDelayMinutes" is greater than 15 and less than or equal to 45 means that the flight is moderately delayed, and the delay time is greater than 15 minutes and less than 45 minutes. For the remaining flights where the value of "DepDelayMinutes" is greater than 45, "Large_Delay" is assigned to these flights, which indicates that the flight is severely delayed and the delay is longer than 45 minutes. Finally, for rows where the value of the "Cancelled" column is True, indicating that the flight has been cancelled, it is assigned to the Cancelled class in the DelayGroup.

After that, a new row is created in the "data2018" dataset based on the delay time and whether the flight is cancelled or not: "DelayGroup". This column will classify flights into OnTime_Early group, Small_Delay group, Medium_Delay group, Large_Delay group and Cancelled group based on their delay time. This will make the subsequent presentation of aircraft status data in the charts clearerand intuitive.

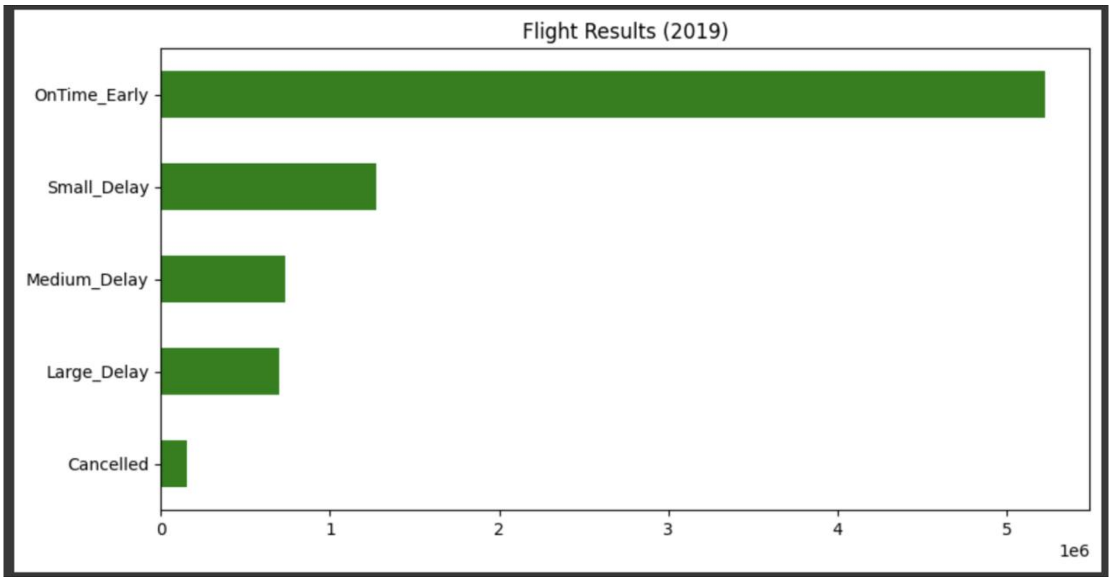


(Figure9: output of the code)

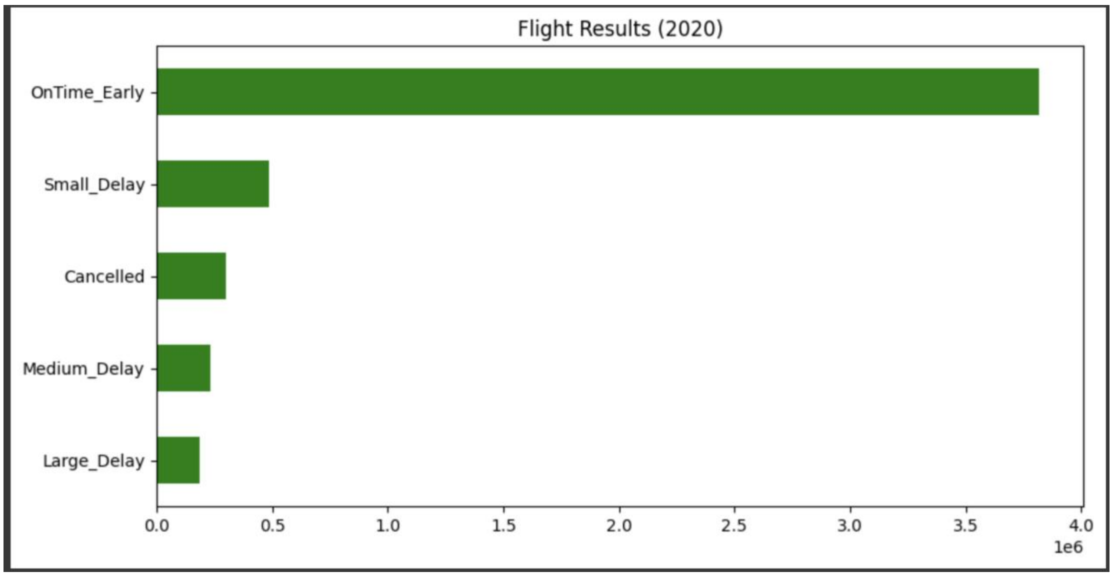
This image clearly shows the status of the data in each flight delay groups , allowing for a more intuitive view of the flight delay groups data.

The unique values in the "DelayGroup" column are then counted (in ascending order) and the result should contain all counted objects. Use the matplotlib tool to output an image that specifies the basic settings of the graph being plotted (e.g. graph shape, colour and image size). For this graph, I have chosen a horizontal bar graph (kind="barh") with an image size of (figsize=(10, 5)), a graph colour and a green bar shape (colour='green') as well as setting the graph title to "Flight Results (2018)".

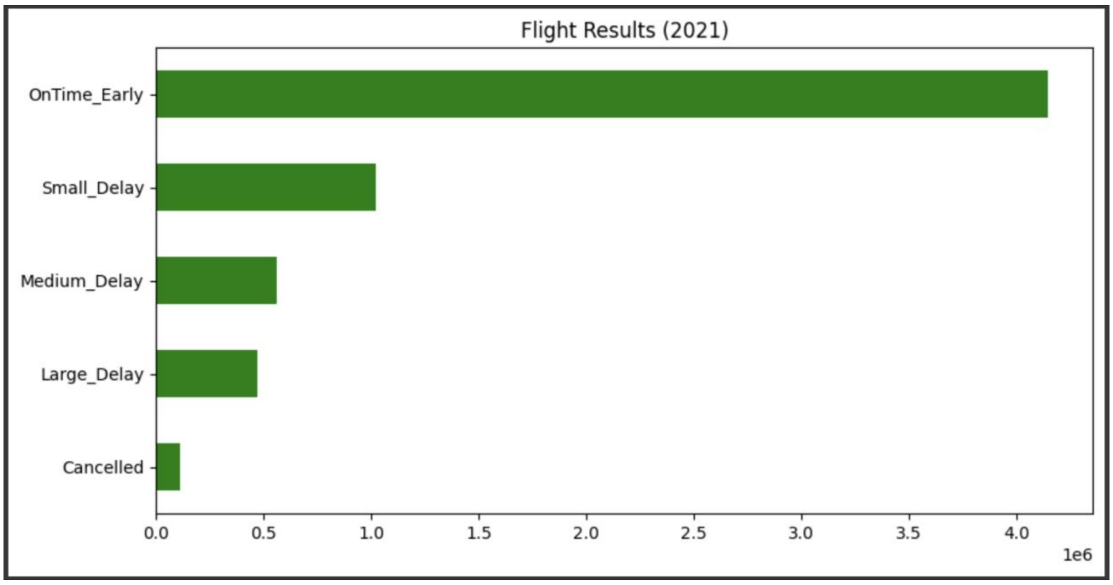
I aslo do the same processing for 2019-2022 datasets, the result I will attached below:



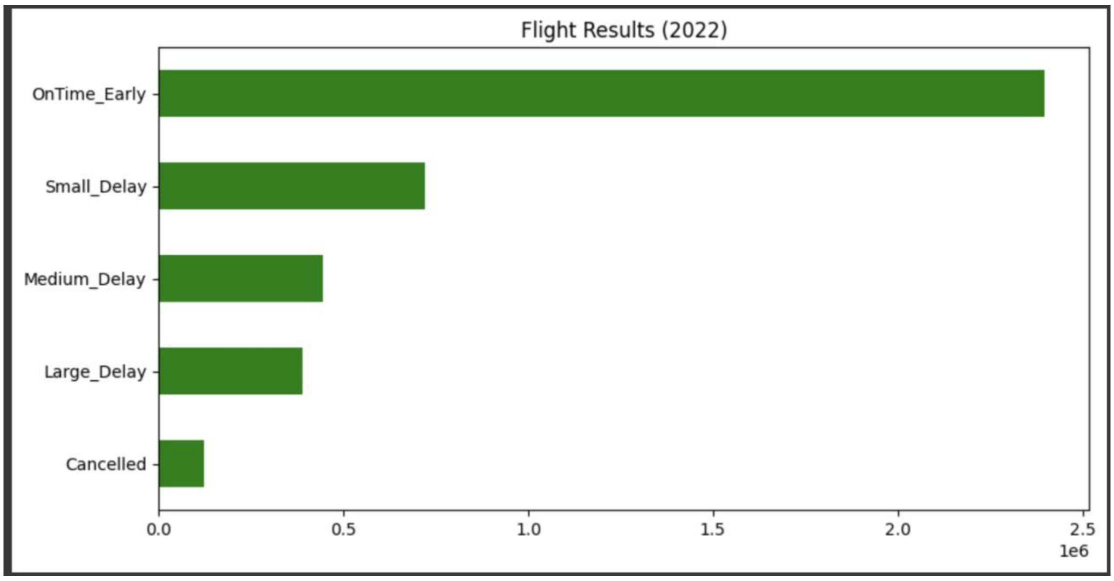
(Figure10: Flight Results(2019))



(Figure11: Flight Results(2020))



(Figure12: Flight Results(2021))

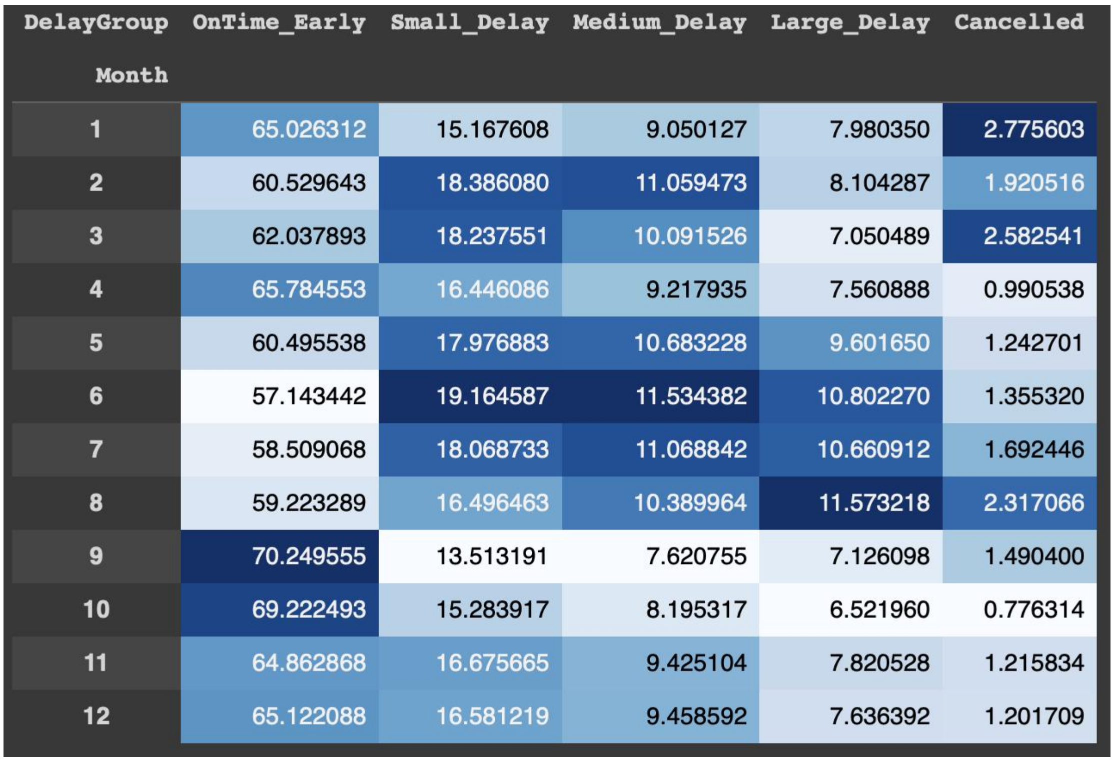


(Figure13: Flight Results(2022))

In the next section, I calculate the flight punctuality for each month of the same year and summarise data in a chart.

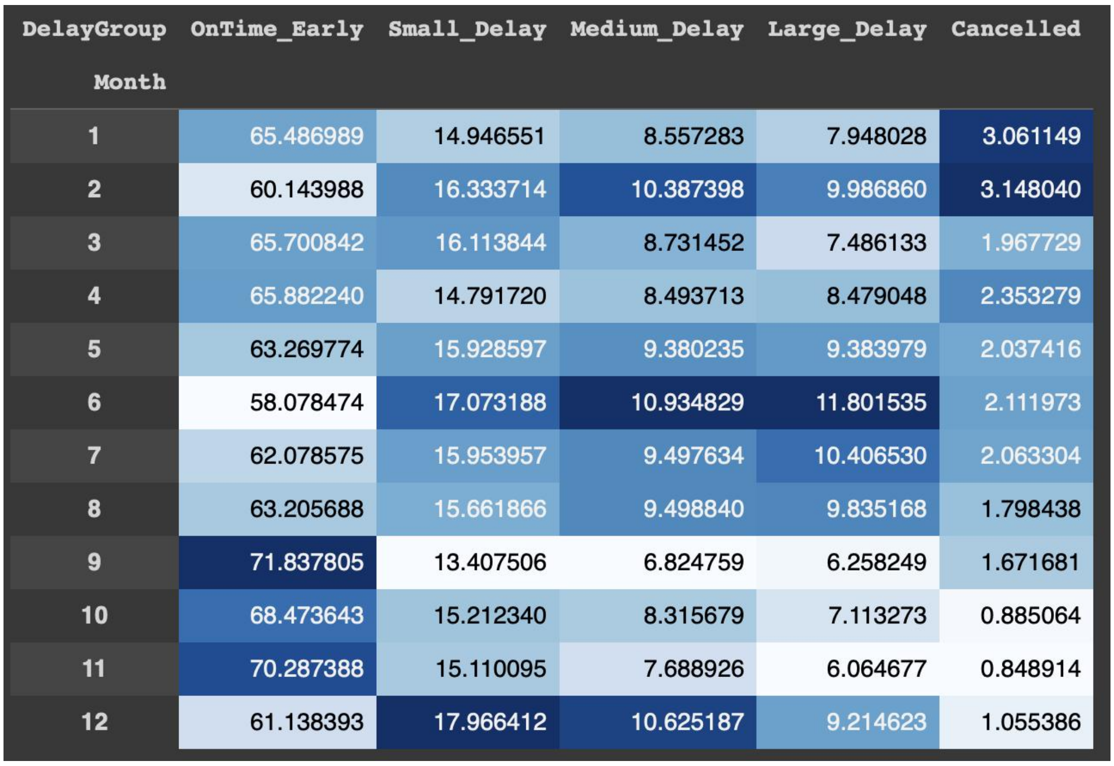
It first extracts the month portion from the dates in the FlightDate column via the dt.month property and creates a new column called Month. Then use the groupby command to aggregate the collated data and add constraints, then calculate the values and counts for each delay (On_time, Small_delay, Medium_delay, Large_delay and Cancelled) for each month, then normalise them to get the corresponding rates, and finally split the resulting Finally, the resulting DataFrame is split and the "DelayGroup" index is converted to a column. To visualise the data, multiply the result by 100 to convert it to a percentage. The darker the colour the more this state occurs during the month.

Same processing also finished in left four years datasets.



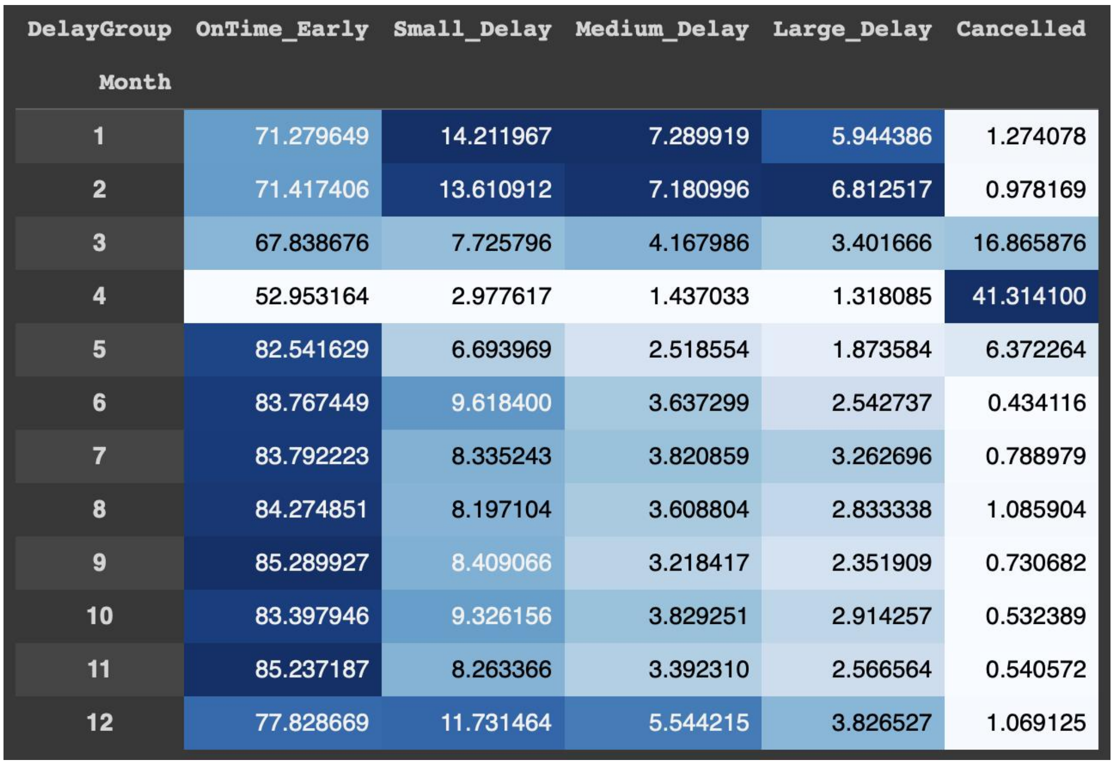
DelayGroup	OnTime_Early	Small_Delay	Medium_Delay	Large_Delay	Cancelled
Month					
1	65.026312	15.167608	9.050127	7.980350	2.775603
2	60.529643	18.386080	11.059473	8.104287	1.920516
3	62.037893	18.237551	10.091526	7.050489	2.582541
4	65.784553	16.446086	9.217935	7.560888	0.990538
5	60.495538	17.976883	10.683228	9.601650	1.242701
6	57.143442	19.164587	11.534382	10.802270	1.355320
7	58.509068	18.068733	11.068842	10.660912	1.692446
8	59.223289	16.496463	10.389964	11.573218	2.317066
9	70.249555	13.513191	7.620755	7.126098	1.490400
10	69.222493	15.283917	8.195317	6.521960	0.776314
11	64.862868	16.675665	9.425104	7.820528	1.215834
12	65.122088	16.581219	9.458592	7.636392	1.201709

(Figure14: Flight Results(2018))



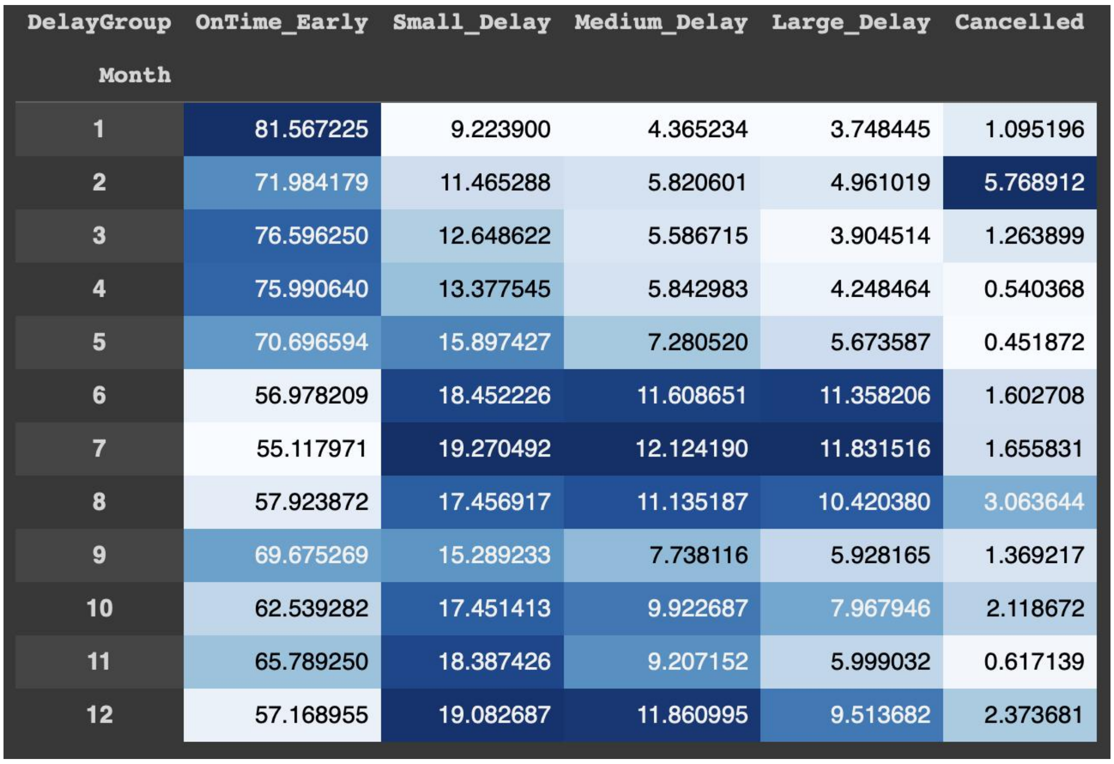
DelayGroup	OnTime_Early	Small_Delay	Medium_Delay	Large_Delay	Cancelled
Month					
1	65.486989	14.946551	8.557283	7.948028	3.061149
2	60.143988	16.333714	10.387398	9.986860	3.148040
3	65.700842	16.113844	8.731452	7.486133	1.967729
4	65.882240	14.791720	8.493713	8.479048	2.353279
5	63.269774	15.928597	9.380235	9.383979	2.037416
6	58.078474	17.073188	10.934829	11.801535	2.111973
7	62.078575	15.953957	9.497634	10.406530	2.063304
8	63.205688	15.661866	9.498840	9.835168	1.798438
9	71.837805	13.407506	6.824759	6.258249	1.671681
10	68.473643	15.212340	8.315679	7.113273	0.885064
11	70.287388	15.110095	7.688926	6.064677	0.848914
12	61.138393	17.966412	10.625187	9.214623	1.055386

(Figure15: Flight Results(2019))



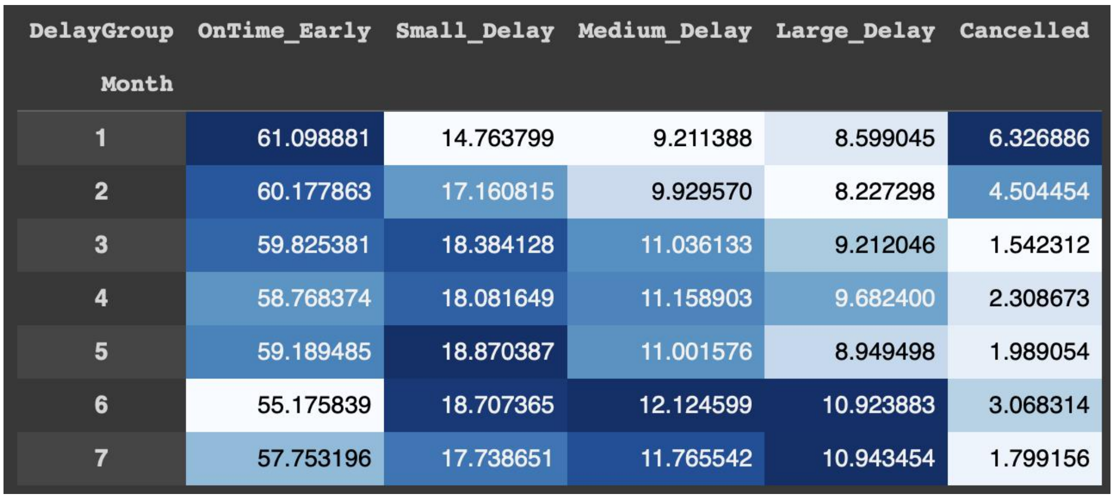
DelayGroup	OnTime_Early	Small_Delay	Medium_Delay	Large_Delay	Cancelled
Month					
1	71.279649	14.211967	7.289919	5.944386	1.274078
2	71.417406	13.610912	7.180996	6.812517	0.978169
3	67.838676	7.725796	4.167986	3.401666	16.865876
4	52.953164	2.977617	1.437033	1.318085	41.314100
5	82.541629	6.693969	2.518554	1.873584	6.372264
6	83.767449	9.618400	3.637299	2.542737	0.434116
7	83.792223	8.335243	3.820859	3.262696	0.788979
8	84.274851	8.197104	3.608804	2.833338	1.085904
9	85.289927	8.409066	3.218417	2.351909	0.730682
10	83.397946	9.326156	3.829251	2.914257	0.532389
11	85.237187	8.263366	3.392310	2.566564	0.540572
12	77.828669	11.731464	5.544215	3.826527	1.069125

(Figure16: Flight Results(2020))



DelayGroup	OnTime_Early	Small_Delay	Medium_Delay	Large_Delay	Cancelled
Month					
1	81.567225	9.223900	4.365234	3.748445	1.095196
2	71.984179	11.465288	5.820601	4.961019	5.768912
3	76.596250	12.648622	5.586715	3.904514	1.263899
4	75.990640	13.377545	5.842983	4.248464	0.540368
5	70.696594	15.897427	7.280520	5.673587	0.451872
6	56.978209	18.452226	11.608651	11.358206	1.602708
7	55.117971	19.270492	12.124190	11.831516	1.655831
8	57.923872	17.456917	11.135187	10.420380	3.063644
9	69.675269	15.289233	7.738116	5.928165	1.369217
10	62.539282	17.451413	9.922687	7.967946	2.118672
11	65.789250	18.387426	9.207152	5.999032	0.617139
12	57.168955	19.082687	11.860995	9.513682	2.373681

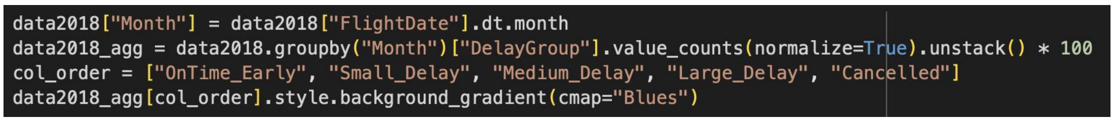
(Figure17: Flight Results(2021))



DelayGroup	OnTime_Early	Small_Delay	Medium_Delay	Large_Delay	Cancelled
Month					
1	61.098881	14.763799	9.211388	8.599045	6.326886
2	60.177863	17.160815	9.929570	8.227298	4.504454
3	59.825381	18.384128	11.036133	9.212046	1.542312
4	58.768374	18.081649	11.158903	9.682400	2.308673
5	59.189485	18.870387	11.001576	8.949498	1.989054
6	55.175839	18.707365	12.124599	10.923883	3.068314
7	57.753196	17.738651	11.765542	10.943454	1.799156

(Figure18: Flight Results(2022))

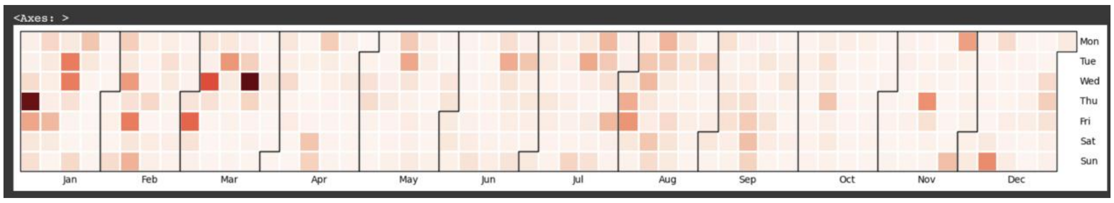
Example code below:



```
data2018["Month"] = data2018["FlightDate"].dt.month
data2018_agg = data2018.groupby("Month")["DelayGroup"].value_counts(normalize=True).unstack() * 100
col_order = ["OnTime_Early", "Small_Delay", "Medium_Delay", "Large_Delay", "Cancelled"]
data2018_agg[col_order].style.background_gradient(cmap="Blues")
```

(Figure19. Example code)

The following section displays the chart, again using the Aggregate command to analyse the flight statuses by week for each month of 2018, which represents the flight statuses that occur most frequently on each day of each week of each month. As in the previous section, the darker the colour, the higher the probability.



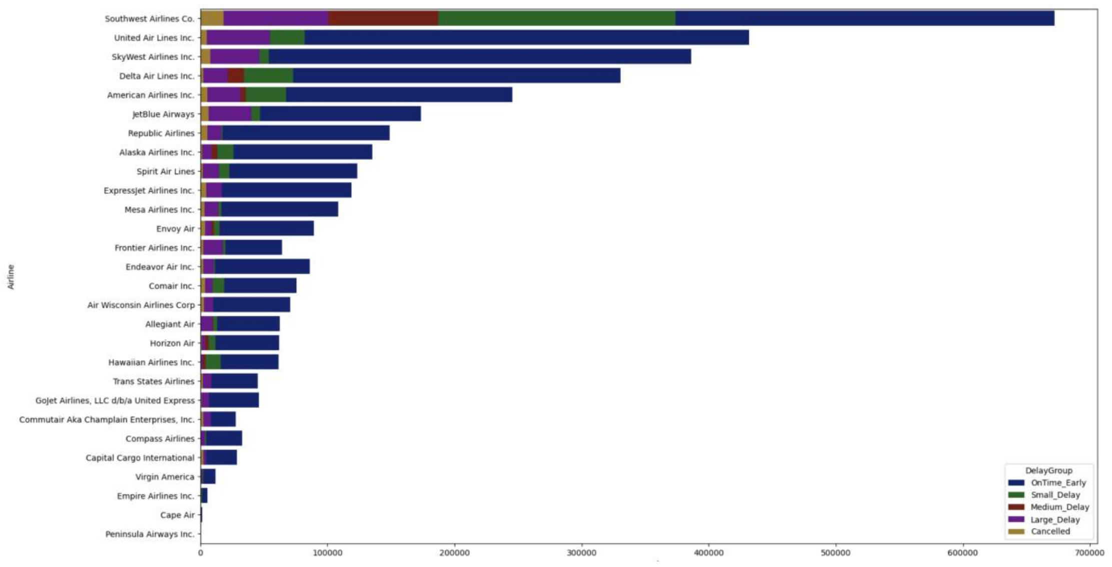
(Figure20. The output of each day in a week by month)

Example code below:

```
events = data2018.groupby('FlightDate')['Cancelled'].mean()
fig, ax = plt.subplots(figsize=(20,10))
calmap.yearplot(events, year=2018, monthly_border=True)
```

(Figure20. The output of each day in a week by month)

The chart shows the status of flights by airline (28 airlines in total), which allows us to visualise the ratio of each flight status for each airline and the percentage of the total number of flights in each status group. In this chart we can observe the airlines with a high percentage of on-time performance and small delays, thus selecting the top performers from the 28 airlines.



(Figure22. The flights status percentage of each airline)

4 MACHINE LEARNING AND PREDICTION

4.1 Prediction and model selection

Since a large number of variables and factors are involved in the flight status prediction problem, the problem can be considered as a multivariate problem.

For the prediction of this multivariate complex problem, Random Forest Model will be more suitable to deal with this kind of problem, so Random Forest Model is finally selected for this problem of flight status prediction.

4.2 Type of algorithms

The Random Forest model is an integrated row algorithm that improves the predictive performance of the model itself by combining multiple decision trees.

Prior to model architecture and data prediction, I chose the scikit-learn (often abbreviated to sklearn) library as the main tool library to call the model and train it. I chose the scikit-learn database because of the library's features such as powerful algorithms, API consistency, and inclusion of various data manipulation and visualisation tools.

The construction of the random forest model is divided into the following steps, but before building the model in the following steps, if you do not have a training dataset and a test dataset, you need to divide the dataset into a training dataset and a test dataset in advance (where there is a 7-3 split and an 8-2 split), usually the training dataset needs to contain 70% of the original dataset (another common split is the training dataset needs to contain 80% of the original dataset), and the test dataset needs to consist of 30% of the original dataset (in contrast, another common segmentation method is that the test data set needs to consist of 20% of the original dataset).

a. Data preparation: this step has been done in the previous Exploratory Data Analysis (EDA) section, so we can skip this part before model construction.
b. Feature Selection: select valid features and take precautions to prevent overfitting of the model.
c. Sampling: randomly draw the samples from the training set. Since the samples are randomly drawn, there may be duplicate data in the samples, but this phenomenon does not affect the subsequent training.

d. Decision Tree Construction: multiple decision trees are constructed using the CART (Classification and Regression Trees) algorithm as well as utilising previously drawn samples and selected feature sets. Splitting nodes during the training growth of decision trees can improve the accuracy of predictions by reducing entropy and squared error.
e. Making Predictions: The Random Forest model will aggregate the outputs from all the decision trees involved in making predictions and then take different approaches for different problems. For regression problems, the results of multiple decision trees will be averaged as the final prediction. For classification problems, voting will be used to select the result with the highest occurrence rate as the final prediction.
f. Output the final valuations
g. Model Evaluation: Measure performance through cross-validation or by using numerical values of performance metrics (e.g., accuracy, F1 scores, mean square error, and other metrics).

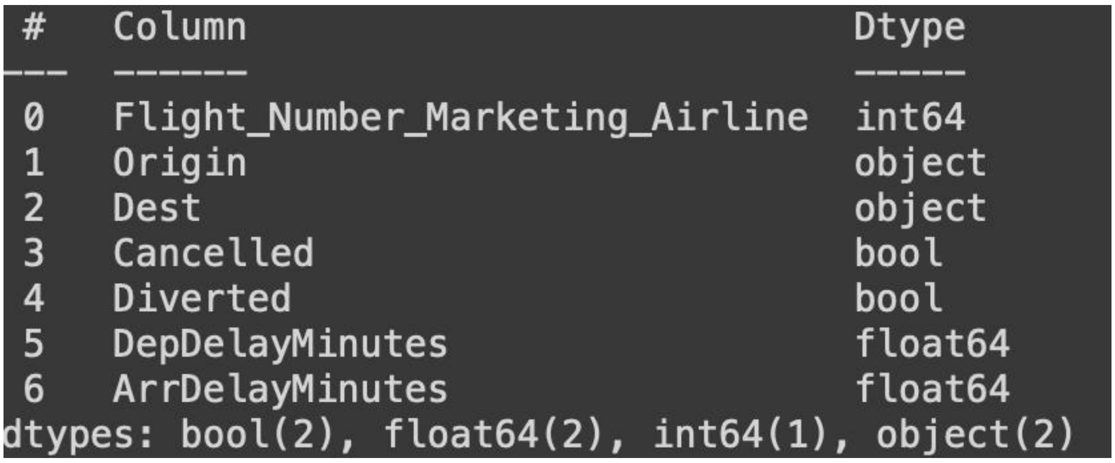
4.3 Feature selections

4.3.1 Feature columns selections

In this section I chose to use a parquet document because when making predictions you need to integrate five years of data while calling a large amount of data the parquet document as a columnar storage format is more suitable for efficiently storing larger datasets than the text format of a CSV document, which is better suited to the task of processing and analysing large amounts of data. Because of the large amount of data to be used in this study, both the pre-merge and post-merge datasets are large datasets, and the advantages of high performance and support for cross-platform operation of the parquet document are more suitable for this study. Moreover, parquet supports various compression and encoding methods, which makes the study more flexible and efficient as the dataset in this study contains a large number of complex data types and needs to be transformed for subsequent prediction model architecture.

I used pyarrow library for merging multiple parquet documents as follows, the combined file name is ‘Combined.parquet’.

As predictions need to be made based on the flight status of each flight over a number of years, the unique information of the flight number is a necessary choice. The columns required for the forecast chosen for this purpose are:
Flight_Number_Marketing_Airline(int64)
Origin(object)
Dest(object)
Cancelled(bool)
Diverted(bool)
DepDelayMinutes(float64)
ArrDelayMinutes(float64)

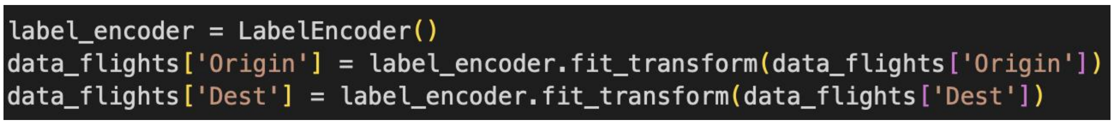


```
 #   Column                         Dtype
---  ------                         -----
 0   Flight_Number_Marketing_Airline  int64
 1   Origin                         object
 2   Dest                           object
 3   Cancelled                      bool
 4   Diverted                       bool
 5   DepDelayMinutes                float64
 6   ArrDelayMinutes                float64
dtypes: bool(2), float64(2), int64(1), object(2)
```

(Figure23. original data and data type)

4.3.2 Data type processing

Before we continue with the model architecture, firstly we need to make sure that the chosen data type can be applied to the random forest model. In the chosen data, the data types of the Origin and Dest columns are both object, so it is necessary to use the LabelEncoder class in the scikit-learn library to convert the class features to numeric classes for subsequent training and prediction.

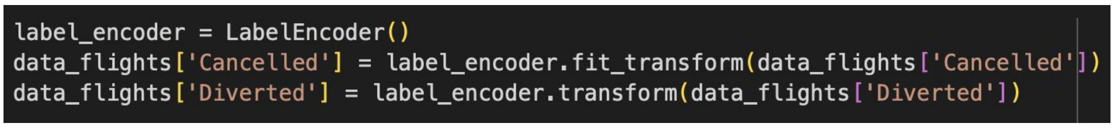


```
label_encoder = LabelEncoder()
data_flights['Origin'] = label_encoder.fit_transform(data_flights['Origin'])
data_flights['Dest'] = label_encoder.fit_transform(data_flights['Dest'])
```

(Figure24. data type conversion1)

The instance of LabelEncoder is created first, and then the values in the columns "Origin" and "Dest" are label encoded, converting the data type from an object to a numeric type.

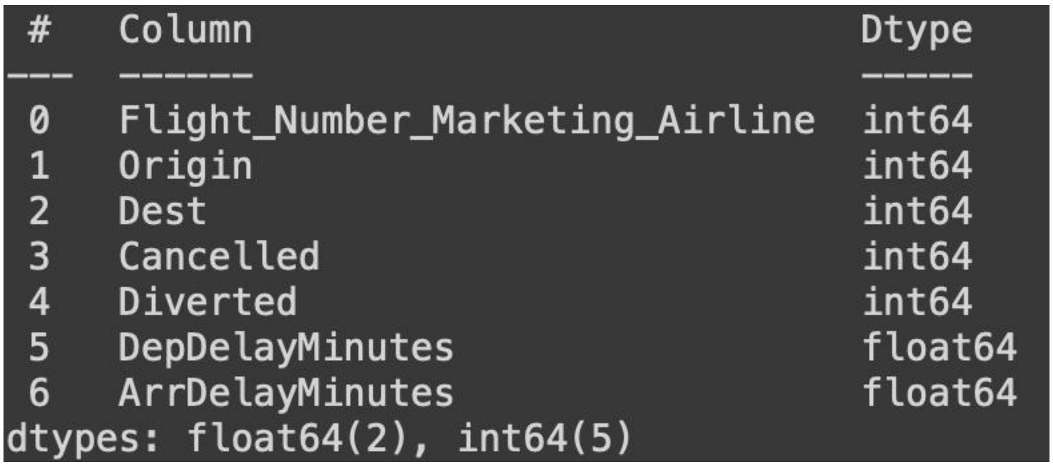
The other two bool columns, Canceled and Diverted, need to be encoded again with the LabelEncoder from the scikit-learn library. This encoding converts the bool values to numeric values, TRUE values to 0, and FALSE values to 1.



```
label_encoder = LabelEncoder()
data_flights['Cancelled'] = label_encoder.fit_transform(data_flights['Cancelled'])
data_flights['Diverted'] = label_encoder.transform(data_flights['Diverted'])
```

(Figure25. data type conversion2)

After data type processing, the selected data type is:



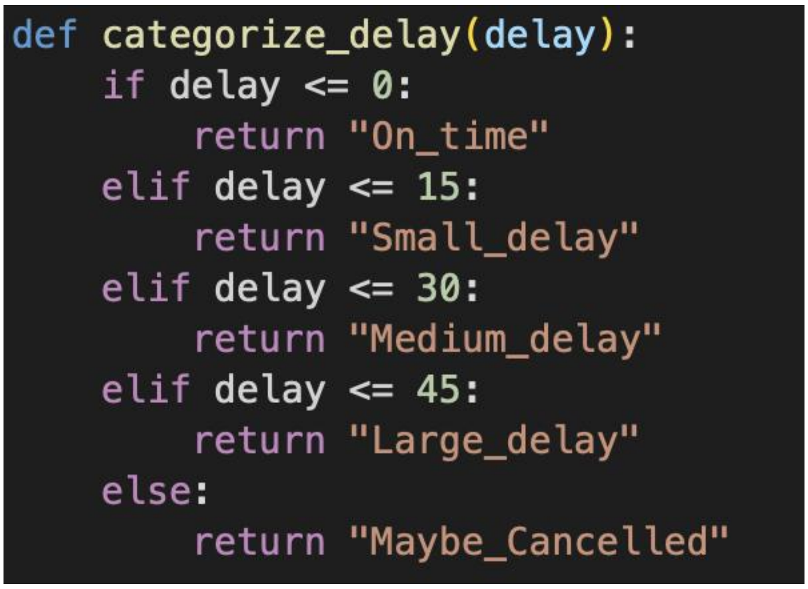
```
 #   Column                        Dtype
---  ------                        -----
 0   Flight_Number_Marketing_Airline  int64
 1   Origin                        int64
 2   Dest                          int64
 3   Cancelled                     int64
 4   Diverted                      int64
 5   DepDelayMinutes               float64
 6   ArrDelayMinutes               float64
dtypes: float64(2), int64(5)
```

(Figure26. data type after converting)

4.3.3 Data scala

In this step I first combine the departure and arrival delay times, add up the values to get a total DelayTime, and create a new column to store the corresponding time, and scale that time, normalising all flights without delays to a uniform scale data, and categorising the others according to the delay time on a scale.

The flights are classified according to the delay time. Flights with delays less than or equal to 0 minutes are classified as On_time, flights with delays greater than or equal to 0 and less than or equal to 15 minutes are classified as Small_delay, flights with delays greater than or equal to 15 minutes and less than or equal to 30 minutes are classified as Medium_delay. Flights with delays greater than or equal to 30 minutes and less than or equal to 45 minutes are classified as Large_delay. The rest are classified as Maybe_cancelled.



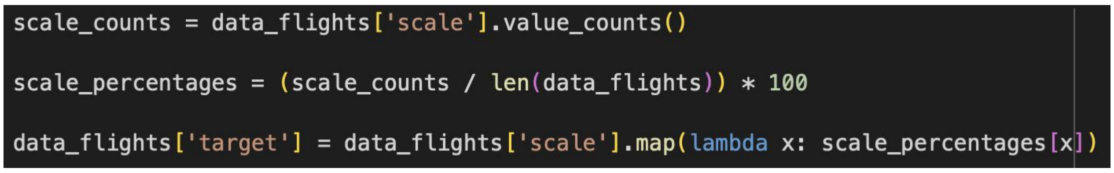
```
def categorize_delay(delay):
    if delay <= 0:
        return "On_time"
    elif delay <= 15:
        return "Small_delay"
    elif delay <= 30:
        return "Medium_delay"
    elif delay <= 45:
        return "Large_delay"
    else:
        return "Maybe_Cancelled"
```

(Figure27. data categorising)

To get the relative percentage values in the final prediction result, next assign a percentage to each unique value in the scala column.

To begin with, the number of occurrences of each unique value in the "scale" column of the "data_flights" dataframe is counted and the result is stored in a

Series object named scale_counts. The last line of code creates a new column called "target" and uses the mapping function to map each value in the "scale" column to a corresponding percentage value. So that the appropriate percentage is assigned to each row. This will convert each row's "scale" value to its corresponding percentage and store the result in the new "target" column.

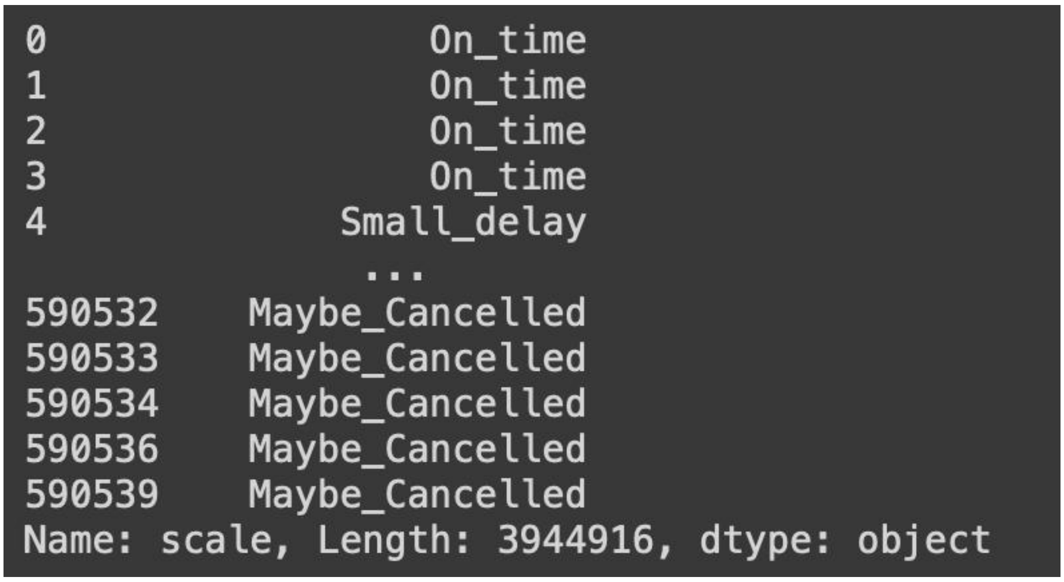


```
scale_counts = data_flights['scale'].value_counts()

scale_percentages = (scale_counts / len(data_flights)) * 100

data_flights['target'] = data_flights['scale'].map(lambda x: scale_percentages[x])
```

(Figure28. data scale)



```
0                  On_time
1                  On_time
2                  On_time
3                  On_time
4              Small_delay
                    ...
590532    Maybe_Cancelled
590533    Maybe_Cancelled
590534    Maybe_Cancelled
590536    Maybe_Cancelled
590539    Maybe_Cancelled
Name: scale, Length: 3944916, dtype: object
```

(Figure29. data scale showing)

4.4 Model training and validation

4.4.1 Selection of feature matrix (x) and target vector (y)

X (Feature Matrix): the feature matrix contains all the features or attributes, i.e. the input data used for training and prediction. For example, the following is used in the model:

Flight_Number_Marketing_Airline(int64)

Origin (int64)

Destination (int64)

Cancelled (int64)

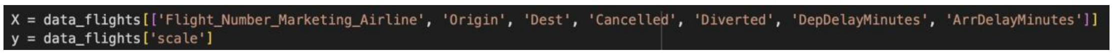
Transferred (int64)

Departure Delayed Minutes (float64)

Arrival delay minutes (float64)

The selection of these data allows the model to make predictions based on these features.

y (target vector): the target vector contains the target value or label associated with each data point in the feature matrix. The target value is usually the value we want the model to predict or classify, e.g. In this model, we want to use the target value to determine whether the future state of a flight is on time, small delay, medium delay or large delay.



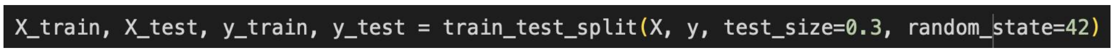
```
X = data_flights[['Flight_Number_Marketing_Airline', 'Origin', 'Dest', 'Cancelled', 'Diverted', 'DepDelayMinutes', 'ArrDelayMinutes']]
y = data_flights['scale']
```

(Figure30. Selection of feature matrix (x) and target vector (y))

4.4.2 Training and testing datasets divided

The original dataset is not divided into training set and test set, so in this step we will do the division of training set and test set, also used the train_test_split class in the scikit-learn library to help us to randomly divide the data, here we choose the conventional 7-3 division method, that is, 70% of the original dataset is the training set and 30% of the data is the test set.

Which random_state is the random seed parameter, which ensures randomness during the division of the dataset and allows the same results to be obtained, guaranteeing the stability of the repeated experiments.



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

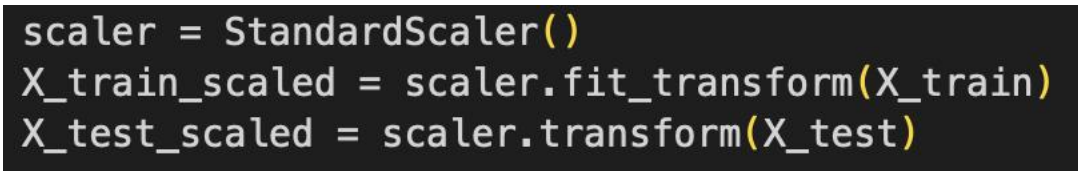
(Figure31. split the dataset)

4.4.3 Feature normalisation

This section will be normalised for features. This step requires the StandardScaler class from the scikit-learn library.

A Normaliser object named scaler is first created. Perform a normalisation operation by fitting and merging using the training dataset X_train. This process will get the mean and standard deviation of the data. Then normalise the features of the training set using fit_transform and store the results in X_train_scaled.

The normalisation aims to scale the values of the features to a standard normal distribution with mean 0 and standard deviation 1 (or similar). It seeks to reduce the influence of different data on the results, and this step ensures that more consistent results are obtained subsequently.



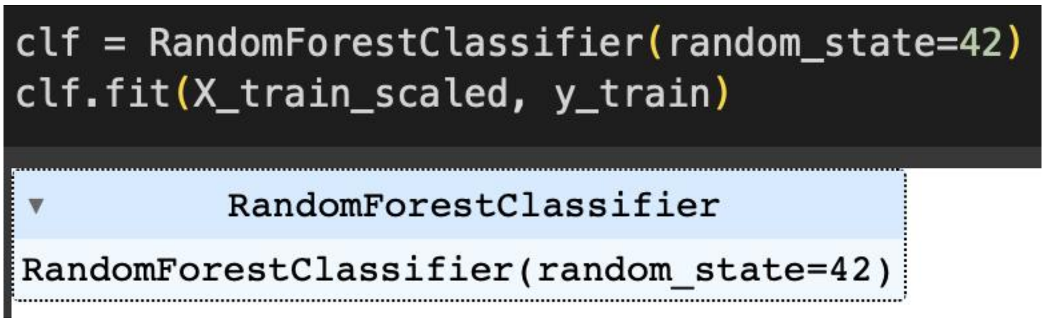
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

(Figure32. feature normalisation)

4.4.4 Creating and training models

In this section, the architecture and training of the Random Forest model will be executed.

The class RandomForestClassifier from the scikit-learn library will be used in this section. Random forest classification models are constructed by first creating a Random Forest Classifier object named clf. The constructed random forest model is then trained using a previously segmented training dataset. The model will predict the target value based on the previously input features.



```
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train_scaled, y_train)
```

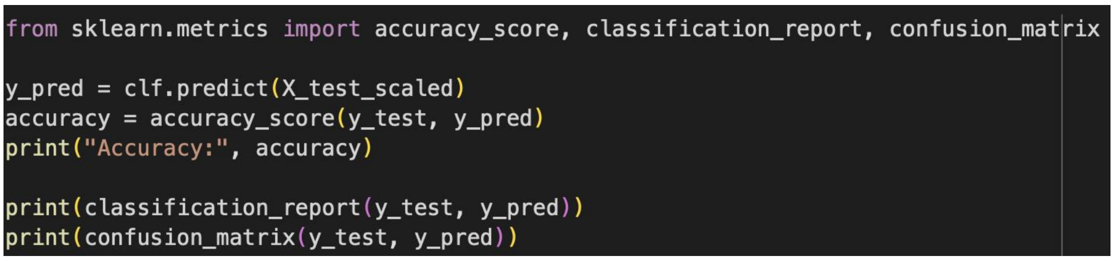
```
▾          RandomForestClassifier
RandomForestClassifier(random_state=42)
```

(Figure33. Creating and training models)

4.4.5 Model validation

This section requires the accuracy_score, classification_report, and confusion_matrix classes from the scikit-learn library.

Use the test set X_test_scaled to make predictions. Use the predict command to generate predictions for the model and store these predictions in y_pred. Model accuracy is then printed after calculation using accuracy_score function. Afterwards a classification report is generated using classification_report function which contains precision, recall, F1 score etc. which can be used to evaluate the model. Finally the confusion matrix is generated for further evaluation of the model accuracy.

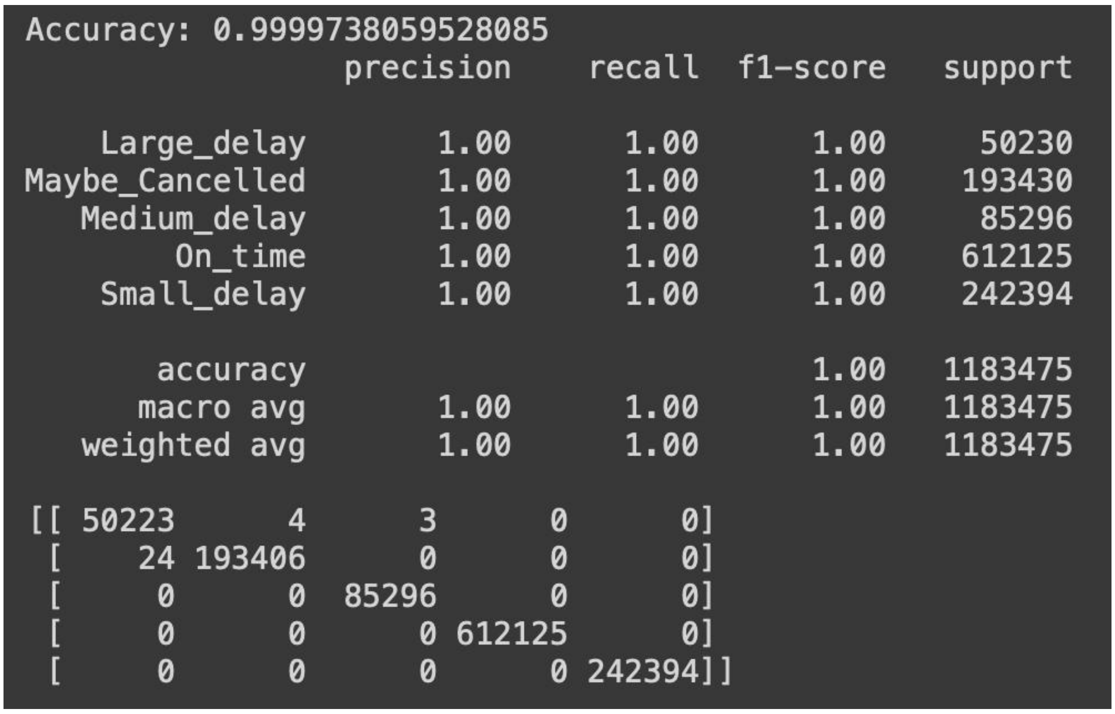


```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

y_pred = clf.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

(Figure34. model validation)



```
Accuracy: 0.9999738059528085
                 precision    recall  f1-score   support

    Large_delay       1.00      1.00      1.00     50230
Maybe_Cancelled       1.00      1.00      1.00    193430
   Medium_delay       1.00      1.00      1.00     85296
        On_time       1.00      1.00      1.00    612125
    Small_delay       1.00      1.00      1.00    242394

       accuracy                           1.00   1183475
      macro avg       1.00      1.00      1.00   1183475
   weighted avg       1.00      1.00      1.00   1183475

[[ 50223      4      3      0      0]
 [    24 193406      0      0      0]
 [     0      0  85296      0      0]
 [     0      0      0 612125      0]
 [     0      0      0      0 242394]]
```

(Figure35. validation result)

The output indicates that the accuracy of the model is about 99.997%, i.e., the model correctly predicts the vast majority of the samples. The model has an accuracy of 1.0, which would indicate that the model predicts the vast majority of cases accurately on each category. A recall of 1.0 would indicate that the model included all cases and did not miss any samples in any category. In summary, the performance of the model is excellent.

5 CONCLUSION

Combined with the Exploratory Data Analysis (EDA) component and machine learning model predictions, the flight status also becomes relatively known and controllable, saving time, material and labour costs at the root cause. Since the dataset used in this study lacks information on flight delays or cancellations due to weather, air traffic control and policy regulation, the predictions are not very accurate. If this information is included in the dataset and added to the model predictions for optimisation, it may be possible to get more accurate results and specific reasons for flight delays.

Therefore, because model predictions are not 100% accurate, airlines and airport operators, among others, will also need to use the experience gained from previous cases to face different scenarios in order to be able to adapt to sudden changes in the status of the flights in certain situations to meet the needs of the travellers.

REFERENCES

Rob Mulla(2022). Flight Status Prediction. https://www.kaggle.com/datasets/robikscube/flight-delay-dataset-20182022

Aviation master(2023, March 18). Aviation Industry Trends: An Overview of the Latest Developments and Strategies. https://amecet.in/blog/aviation-industry-trends-an-overview-of-the-latest-development-and-strategies/

Walter Leal Filho, Artie W. Ng, Ayyoob Sharifi, Jitka Janová, Pınar Gökçin Özuyar, Chinmai Hemani, Graeme Heyes, Dennis Njau & Izabela Rampasso(2022, September 10). Global tourism, climate change and energy sustainability: assessing carbon reduction mitigating measures from the aviation industry. https://pubmed.ncbi.nlm.nih.gov/36105893/

Shijin Wang, Jiewen Chu, Jiahao Li, Rongrong Duan(2022, April 1). Prediction of Arrival Flight Operation Strategies under Convective Weather Based on Trajectory Clustering. https://www.mdpi.com/2226-4310/9/4/189

Sia Gholami, Saba Khashe(2022, August 16). Flight Delay Prediction Using Deep Learning and Conversational Voice-Based Agents. https://asrjetsjournal.org/index.php/American_Scientific_Journal/article/view/7894

Rob Mulla(2022). Flight Dely - Exploratory Data Analysis [Twitch]. https://www.kaggle.com/code/robikscube/flight-delay-exploratory-data-analysis-twitch

Devashree Madhugiri(2023, September 15). Exploratory Data Analysis(EDA): Types, Tools, Process.
https://www.knowledgehut.com/blog/data-science/eda-data-science

Prajwal CN(2022, August 3). EDA-Exploratory Data Analysis: Using Python Functions.
https://www.digitalocean.com/community/tutorials/exploratory-data-analysis-python

aakarsha_chugh(2023, April 18). Label Encoding in Python.
https://www.geeksforgeeks.org/ml-label-encoding-of-datasets-in-python/

Krishni(2018, November 27). A Beginners Guide to Random Forest Regression.
https://medium.datadriveninvestor.com/random-forest-regression-9871bc9a25eb

Anu(2020, September 21). Random Forest Regression: A Complete Reference.
https://www.askpython.com/python/examples/random-forest-regression

Derrick Mwiti(2023, September 1). Random Forest Regression: When Does It Fail and Why?
https://neptune.ai/blog/random-forest-regression-when-does-it-fail-and-why

Piotr Płoński(2020, June 29). How to visualize a single Decision Tree from the Random Forest in Scikit-Learn(Python)?
https://mljar.com/blog/visualize-tree-from-random-forest/

Jason Brownlee(2021, April 27). How To Develop a Random Forest Ensemble in Python.
https://machinelearningmastery.com/random-forest-ensemble-in-python/