Course of Data Management - Professor Andrea Maurino - Academic year 2023/2024

Different airports and weather conditions: how flight delays change?

Adobati Simone, Ghiselli Ricci Alice, Mariani Chiara
Università degli Studi di Milano-Bicocca, CdLM Data Science

Abstract

Could weather conditions influence flight departure delays? The data analysed in this paper were collected using API, scraping and file integration. The data were cleaned and integrated to reach a uniform and easier structure in order to do a better analysis. The storage was implemented on a document-based NoSQL database called MongoDB. We obtained data from seven different airports, tracking the departure flights and weather conditions at the take-off. The goal of our analysis was to provide data to study a correlation between flight delays and weather conditions.

Contents

1 Introduction

In the realm of air transportation, flight delays is one of the most significant challenges faced by both airlines and passengers alike. These delays derives from a multitude of factors, including technical issues, airspace traffic and weather conditions. In addition to numerous inconveniences for travellers, flights delays also lead to increased costs for several companies.

In this analysis we focus on one of this possible factors, meteorological conditions at the departure airport, considering just non-commercial routes. The flights we tracked took-off from seven different airports: *Athens International Airport* (Athens, Greece), *El Dorado International Airport* (Bogotà, Columbia), *Miami International Airport* (Miami, USA), *Milan Malpensa Airport* (Milan, Italy), *Narita International Airport* (Narita, Japan), *Ninoy Aquino International Airport* (Manila, Philippines) and *Reykjavik Airport* (Reykjavik, Iceland). These data were collected by scraping the departures table in every website. We only selected fields which reflected the interests of our analysis. To enrich these data we downloaded a file containing airports acronyms, encoded in IATA format, and the respectively whole name of nearly all airports in the world.

The other fundamental elements for our analysis are weather conditions metrics associated to the city and time of the departure flight. A public API, named Open-Meteo (Open-Meteo, 2024), allowed us to fetch data of precipitation, cloud-cover and wind speed at two different altitudes. To make a connection between flights and weather we did scraping to obtain the longitude and the latitude of the airports.

This data choice was made to emphasize the possible correlation between flights delays and different weather conditions due to diverse geo-

graphical climate zones. Could weather conditions influence flight departure delays? To answer this research question, we implemented a data management system dividing the work in six phases, each one with a different aim, as follows:

- **Data acquisiton**: collecting flights data and weather information
- **Data cleaning**: considering only the non-superfluous features
- **Data integration and enrichment**: merging and adding information to create a unified dataset
- **Data storage**: storing data in a document-based database called MongoDB
- **Data quality**: measuring the quality of data by evaluating completeness, consistency and currency
- **Analysis and queries**: general study providing tools for further analysis

2 Data Acquisition

A core part of our project is to correctly acquire all the data necessary to track flights and weather conditions at departure.

We collected data starting from 7th of December 2023 to 14th of February 2024, having 121591 total flights. Some airports started later because we implemented one by one, the last one was Manila Airport which started on 19th of December 2023. Every website had different data maintaining time. For this reason to automatically retrieve the data, we chose to add a crontab job on an Ubuntu Server machine that executed the script every 30 minutes.

We firstly decided to choose seven airports based on the different geographical and climate areas which take place in the world. The climate zones and the respectively airport are:

- **Artic** climate: Reykjavik Airport
- **High mountain** climate: El Dorado International Airport
- **Mediterranean** climate: Athens International Airport
- **Subtropical** climate: Miami International Airport
- **Temperate continental** climate: Milan Malpensa Airport
- **Temperate oceanic** climate: Narita International Airport
- **Tropical** climate: Ninoy Aquino International Airport

There are different methods to obtain data from the internet, the ones we used are: scraping, for airport and flights related data, API, for climate related data, and file download for IATA codes acronyms.

2.1 API

Some information are available on the internet by public access, using a technique called API. This technology allows the user to interact with a third party web server making requests to obtain or insert data. Regarding weather condition information we found a public API called Open-Meteo. This service gives detailed weather conditions at precise point using geospatial coordinates and time.

The endpoint we used is *https://archive-api.open-meteo.com/v1/archive*. This request needs query parameters to identify the place where we want to get the weather, the specific date and time and the list of the weather characteristics of interest. Among all the available information of the API we chose to take into account precipitation [millimetres], cloud-cover [percentage] and wind speed at 10 metres and 100 metres of altitude [kilometres per hour].

In order to prevent a huge amount of requests to the API, we decided to make a request for each day for each airport. Collected data were stored in a Python dict and with a for-cycle we assigned each weather situation to each flight. This API provides weather conditions hourly, so, in case a flight departs in the middle of an hour, we implemented a weighted mean to approximate the conditions at a specific minute.

2.2 Scraping

Since there are no public APIs that distribute real-time data related to flights, we must adopt web scraping as the technique to obtain this type of data. To avoid the use of Selenium and to execute the requests faster, we examined the distinct requests that the browser makes when loading the airport page. By directly calling the endpoint which returns the data we needed, we were able to access these information faster and easier. The most probable types of response are HTML or JSON documents which need to be handled in different ways. With HTML we used BeautifulSoup, a Python module that converts

the HTML document in a tree-type structure, while JSON in Python is directly represented as a dict, so it doesn't need specific libraries or tools.

Now, we provide a detailed summary for each type of response the endpoints gave:

- HTML:
  - Reykjavik Airport (endpoint: *https://www.isavia.is/en/reykjavik-airport/flight-information/departures*)
  - Miami Airport (endpoint: *https://webvids.miami-airport.com/webfids/webfids*)
  - Narita Airport (endpoint: *https://www.narita-airport.jp/en/api/flight*)
- JSON:
  - Bogotà Airport (endpoint: *https://api.eldorado.aero/api/flights*)
  - Athens Airport (endpoint: *https://www.aia.gr/handlers/rtfiV2.ashx*)
  - Milan Airport (endpoint: *https://apiextra.seamilano.eu/ols-flights/v1/en/operative/flights/lists*)
  - Manila Airport (endpoint: *https://miaagov.online/flight-dep.json*)

Scraping rises some concerns about the data provided, in some websites is possible to find common information and different ones, so it is important to make a selection of the feature. For the aim of our analysis we took only six features: *flight-number*, *flight-status*, *scheduled-departure*, *actual-departure*, *airport-departure*, *airport-arrival*.

By doing scraping on airports websites, we could not obtain information about the geospatial airport coordinates. For this reason, we had to find a way to link the airports with their corresponding coordinates. The solution was to do scraping on a website whose endpoint is *https://photon.komoot.io/api*. This request returns latitude and longitude as response in JSON and it wants, as parameter, the string containing the name of the location.

2.3 File download

Every airport is encoded into an acronym using a standard called IATA. To obtain this list there were different ways, we decided to download it from a website (IATA, 2024). In this file there were the acronyms associated with the corresponding airport, the city where the airport is located and the state.

3 Data Cleaning

Data cleaning involves rectifying or eliminating erroneous, corrupted, improperly formatted or duplicated data within a dataset. Data can also be semantically superfluous for the research question, even though, they are considered correct in terms of form or syntax. For this reason, we chose to delete some attributes that could be available on the different websites.

We are going to show two examples, one for JSON response (code 1) and one for HTML response (figure 1). In Athens flights, as we can see in the code 1, there are many features that we can exclude from our data collection. We deleted all ID related attributes, all the boarding information and airline data, such as "*ID*", "*Gate*", "*AirlineName*".


```
{
    "ID": 860318199,
    "UID": "2140621218",
    "Session": 526424,
    "FlightNo": "TO 3521",
    "ScheduledTime": "14/02/2024 11:30",
    "EstimatedTime": "",
    "ActualTime": "11:47",
    "NextInfo": "",
    "Airline": "935874fc",
    "Airport": "4fb187ea",
    "Stop": "00000000",
    "FlightStatus": "00d0660c",
    "AirlineName": "Transavia France",
    "AirportName": "Paris ORY",
    "StopName": "",
    "FlightStateName": "Departed",
    "FlightStateColor": "07b336",
    "Gate": "C38",
    "Exit": "D",
    "CkifCkit": "32---35",
    "JFNO": ""
}
```


Listing 1: Athens flight data example

As shown in the figure 1, in the HTML response there are less features because, with this type of scraping, we can only get what is inside the HTML. Conversely from JSON, all the attributes which do not concern passengers' interests, such as code, IDs and technical data, are hidden.

Figure 1: Miami flight example

These are just examples, other airports responses are different, some have even more features, others have less. For each response we decided to keep only attributes that were useful for our research question, which are:

- **number**: code identifying the flight (i.e. "*EC03911*")
- **status**: identifying if the flight is departed or cancelled (i.e. *"DEPARTED"* or *"CANCELLED"*)
- **scheduledDep**: date and time of the scheduled departure (i.e. "*2023-12-07T06:00:00.000+00:00*")
- **actualDep**: date and time of the actual departure (i.e. "*2023-12-07T06:19:00.000+00:00*")
- **airportDep**: airport of departure (i.e. "*MXP*")
- **airportArr**: airport of arrival (i.e. "*Luxembourg Airport*")

The IATA file we downloaded contained some formatting errors and a missing name. The acronym "*EAP*" was associated to "*, Switzerland*" instead of "*Basel Mulhouse Freiburg, Switzerland*".

4 Data Integration and Enrichment

Once we have selected the appropriate features for our analysis, we improved our data by doing some integration and enrichment operations.

4.1 Data integration

The integration phase is about taking two or more different data, coming from different sources, and merge them directly or by doing some additional processes. Firstly, in order to represent each flight with the six attributes we selected, we needed to find where each value was located in each scraping response. This permitted to uniform the output of the process, which was a dataset containing all the flights data. The next part was to associate all the flights with the corresponding weather conditions at the take-off. To do this we needed to link the datasets by obtaining geospatial coordinates of each airport. This permitted to have the final dataset containing all the flights information and meteorological data respectively.

4.2 Data enrichment

On the other hand, data enrichment permits to add handy information to the dataset to improve the comprehension by increasing the value and the relevance of the information stored. The first step we made in this phase was to substitute the plain airport name in the "*airportArr*" attribute with a standardized name using IATA acronym. Then, to manage coherent data we standardized flight number. We did some different works for each airport, in some cases the number was already formed as "[airline code][flight number]" (i.e. "*EC03911*"), in other cases we had to put together these two values. Where the airline code acronym was already formed, we simply concatenated it with the flight number, instead, where the airline code was the plain name, we only took the capitalized letters to create the acronym.

5 Data Storage

After the data have been acquired, cleaned and integrated, it is crucial to choose and manage a correct way to store them. In our case, both SQL and NoSQL were plausible choices, so we decided to implement a NoSQL document-based database, MongoDB. This assumption is principally based on an easier way to implement the database and handle the data because it removes all the constraints that characterize SQL databases.

There are two ways to implement this type of database, a local one and a remote one. Our type of data acquisition is based on an automated script running on a server, therefore it was useful to adopt a remote database in or-

der to have all the data synchronized with all our devices. We used the service, offered by MongoDB platform, called Atlas, that provides a free to use database with some limitations, like storage limit or bandwidth limit.

To store the data, we created two collections inside the database called: **collflights** (containing all flights related data), **colliata** (containing all the acronyms and the relative airport). Collections diagrams can be found at the figure 2 and 3.

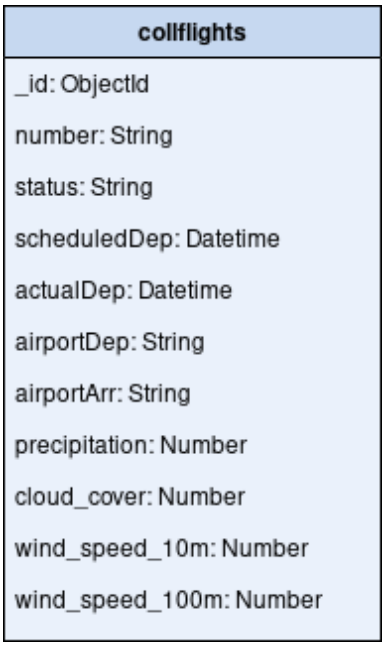

Figure 2: Flights collection

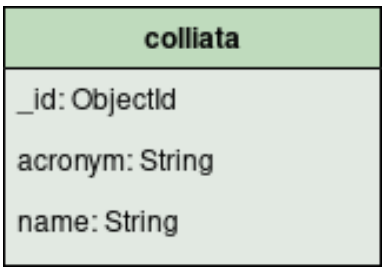

Figure 3: IATA collection

6 Data Quality

Data quality plays a pivotal role in data management process. Data quality dimensions help to recognise whether obtained data are in accordance with the purpose of the analysis. In our case, we appropriately selected different quality dimensions, such as *completeness*, *consistency* and *currency*.

6.1 Completeness

Completeness of a dataset is the coverage with which the observed phenomenon is represented in the dataset. The way we obtained and cleaned the data brought to possible null value only in the *actualDep* attribute. None values in this feature are not considered as missing values, but their meaning is that the flight was cancelled. The value obtained for *actualDep* is: **98,652%**.

6.2 Consistency

Consistency is the degree of how much data are uniform and adhere to defined rules and standards within a system or organization. In our case, this is reflected on the fact that all the airports' names must be codified according to IATA standard. This metric is calculated by taking the total number of arrival airports not coded in IATA standard and dividing it by the number of total flights. This can be caused by two factors: the IATA acronym can be missing in the collection or the conversion formula may skip an airport due to a mismatch. However, this is not a concern for the research question because the whole analysis is made on the departure. For our system, this is just a metric to evaluate the conversion criteria. The value obtained is: **90,260%**.

6.3 Currency

Currency is the speed with which data are updated compared to the corresponding phenomenon in the real world. This measure, in our case, depends on two different factors. The insertion of a new flight in the database is made at most 30 minutes after the departure of it, this is due to the script that runs every 30 minutes. The update of a flight with its weather conditions is done two days after the departure, this is caused by API limitations on newer data. Moreover, since there are missing details about the departure on the day the script runs, Malpensa Airport's flights are taken the day after.

7 Analysis and Queries

7.1 Descriptive analysis

To answer our research question, we firstly did a descriptive analysis to understand which one of the weather condition measures mostly influence flight delays. The correlation results are all very low, as seen in the table 1, meaning that the weather is not the fundamental cause of flight delays. In particular we analysed each

airport to verify which one of the four metereological measures had the greatest impact. As the table 1 shows, weather attributes do not affect each airport in the same way.

For example, Reykjavik (RKV), due to its geographical characteristics and its arctic climate, is heavily conditioned by precipitation and wind.

This led us to take into account for our analysis **wind speed at 100m** and **precipitation**, although they are not much high, they are the most effective measures which lead to a delay at the take-off. For this reason, to accomplish our task, future analysis will be based on these features.

	Precipitation	Cloud	Wind 10m	Wind 100m
ATH	0,015	0,003	0,043	0,038
BOG	0,004	-0,012	-0,002	-0,023
MIA	0,006	0,025	0,001	0,002
MXP	0,046	0,031	-0,026	-0,010
NRT	0,007	-0,013	0,027	0,006
RKV	0,126	0,071	0,008	0,066
RPLL	-0,001	0,011	-0,012	-0,007

Table 1: Correlation between flights delays and weather measures

7.2 Queries

Once we calculated the correlation, we ran some queries to fetch aggregated information from the database. We decided to implement eight different queries:

- **Query 1**: returns the count of flights for each airport (listing 2)
- **Query 2**: returns the average speed of wind at 100m for each airport (listing 3)
- **Query 3**: returns the average of precipitation for each airport (equal to listing 3, changing only "*$avg: $wind_speed_100m*" to "*$avg: $precipitation*")
- **Query 4**: returns the average delay for each airport
- **Query 5**: returns the average delay for each airport filtering flights where the wind speed at 100m is greater than the average of wind speed at 100m at the departure airport
- **Query 6**: returns the average delay for each airport filtering flights where the wind speed at 100m is less or equal than the average of wind speed at 100m at the departure airport
- **Query 7**: returns the average delay for each airport filtering flights where precipitation is greater than the average of precipitation at the departure airport
- **Query 8**: returns the average delay for each airport filtering flights where precipitation is less or equal than the average of precipitation at the departure airport

More in depth, query 4, 5, 6, 7 and 8 have all similar structure, therefore we decided to show only one code implementation, that is visible at the listing 4. The difference is in the *$match* object of the pipeline. In the query 4 this stage is not required because there is no filter needed on the flights. Query 5 and 6 implement this stage with two operators *$gt* and *$lte* respectively. To avoid code repetitions, we created a function called **meanToDict()** which takes three arguments: operator (*$lte* or *$gt*), attribute (*$wind_speed_100m* or *$precipitation*) and mean (response of query 2 or 3 based on the considered attribute). This function returns an array as shown in the listing 5, used to correctly execute the *$or* operator.

```
dbflights.collflights.aggregate([
  {
    "$group": {
      "_id": "$airportDep",
        "count": {
          "$sum": 1
        }
      }
    },
  {
    "$project": {
      "airport": "$_id",
      "_id": 0,
      "count": 1
    }
  }
])
```

Listing 2: Query to obtain the number of flights for each airport

```
dbflights.colliata.aggregate([
  {
    "$group": {
      "_id": "$airportDep",
      "mean": {
        "$avg": "$wind_speed_100m"
      }
    }
  },
  {
    "$project": {
      "airport": "$_id",
      "_id": 0,
      "mean": 1
    }
  }
])
```

Listing 3: Query to obtain the mean speed of wind at 100m for each airport

```
dbflights.collflights.aggregate([
  {
    "$match": {
      "$or": self.meanWindToDict("lte")
    }
  },
  {
    "$project": {
      "delays": {
        "$dateDiff": {
          "startDate": "$scheduledDep",
          "endDate": "$actualDep",
          "unit": "minute"
        }
      },
      "airportDep": 1,
        "_id": 0
      }
    },
    {
      "$group": {
        "_id": "$airportDep",
        "avgDelays": {
          "$avg": "$delays"
        }
      }
    },
    {
      "$project": {
        "avgDelays": 1,
        "airport": "$_id",
        "_id": 0
      }
    }])
```

Listing 4: Query to obtain mean of airport delays filtering on wind speed at 100m

```
[
  {
    'airportDep': 'MXP',
    'wind_speed_100m': {
      '$lte': 12.831892961099022
    }
  },
  //all other airports
  {
    'airportDep': 'RKV',
    'wind_speed_100m': {
      '$lte': 28.68754596491228
    }
  }
]
```

Listing 5: Example of meanToDict() response

The results we obtained, running the queries, are illustrated in the following bar charts, done by using Tableau software (Tableau, 2024). The figure 4, showing the total number of flights for each departure airport, the figures 5 and 7, representing the average of wind speed at 100 metres and precipitation for each airport respectively, provide a general view over the two aspects of our analysis, flights information and weather measures. More in details, what emerges from figure 4 is that there is a huge difference among total flights departed from Reykjavik and the count of flights from each other airport. The value associated to Reykjavik airport is 855 and this low value supports the fact that few flights leave every day. This data has a big relevance on the correlation with weather conditions: minimal change in weather leads to an exponential effect on flight delays.

This exploratory analysis gave us the key elements to make a comparison useful to a possible answer for our research question. We made a bar chart representing the comparison among delays occurred when wind speed is greater or less than each airport average wind speed, the latter shown in the figure 6. For a better comprehension, these values are also put adjacent to the column of delays calculated without filters on wind speed. A meaningful difference among these three values is evident in the Reykjavik and Bogotà airport column. On the contrary, for other airports the differences are minimum.

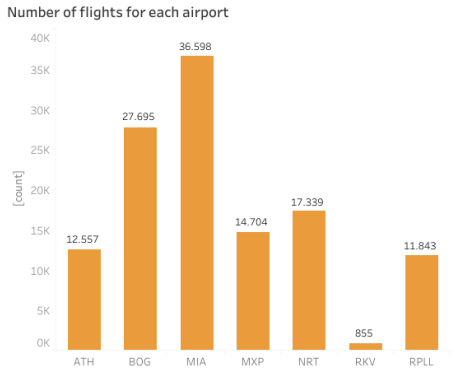

Figure 4: Number of flights for each airport

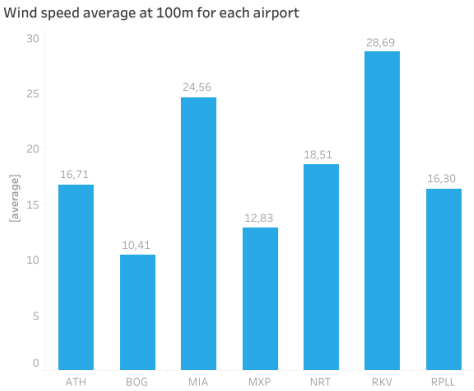

Figure 5: Wind speed average at 100 metres for each airport

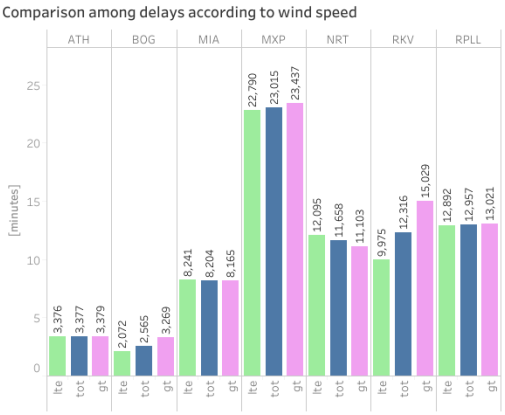

Figure 6: Comparison among delays according to wind speed

The same analysis we have done concerning the wind characteristic, we also made for the precipitation. The figure 7 shows the average of precipitation for each airport, Miami and Reykjavik have the greatest values while Athens has the lowest. Regarding the figure 8, it illustrates the delays comparison when precipitation is greater or less than each airport precipitation average. Some airports, such as, Bogotà, Malpensa, Narita, Reykjavik and Manila show similar results both in figure 6 and 8. On the contrary, in case of Miami, regarding the wind feature, it is possible to see a steady value between *lte*, *tot* and *gt* instead, concerning precipitation, the trend is upwards. Athens, in figure 8, shows a particular scenario: the increase of precipitation leads to a reduction of the delay.

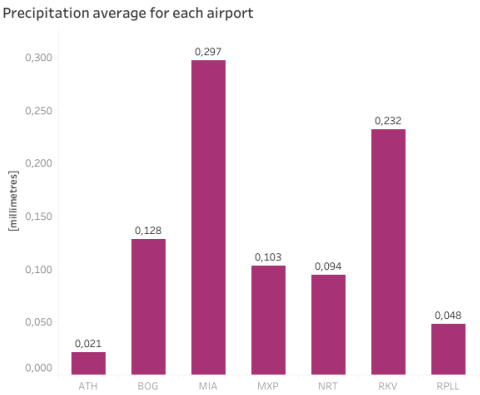

Figure 7: Precipitation average for each airport

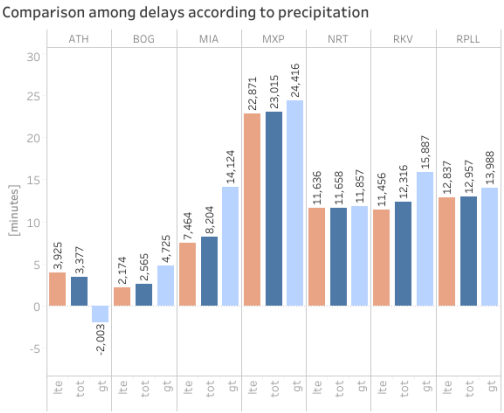

Figure 8: Comparison among delays according to precipitation

All the considerations made are based on the table 2, which shows the detailed percentage increase going from a low value of wind or precipitation to high value of wind or precipitation. We integrated our analysis with table 2, summarizing the outcome of the graphs, to highlights the most meaningful results we obtained and for an easier comprehension from a third-part viewer.

	Wind	Precipitation
ATH	+0,089%	-151,032%
BOG	+55,77%	+117,341%
MIA	-0.922%	+89,228%
MXP	+2,839%	+6,755%
NRT	-8,202%	+1,899%
RKV	+50,667%	+38,678%
RPLL	+1,001%	+8,966%

Table 2: Percentage increase from low to high wind and from low to high precipitation situations

These results can not be compared to the ones in the correlation matrix 1 due to the fact that those in correlation table are calculated considering if precipitation or wind occurred at the time of the flight departure.

8 Conclusion

Thanks to an exhaustive data acquisition and a coherent data integration, we were able to provide results on how specific weather conditions, due to the climate of departure city, can affect flight delays. For each of the seven airports located in different geographical zones, it was crucial to identify the necessary data to be collected, such as *scheduled departure time*, *actual departure time* and flight *status*. Regarding weather information, we have chosen measures that we thought had the greatest impact, such as *precipitation* and *wind speed*.

We implemented queries whose results highlight a correlation, although it is low, between *precipitation* and *wind speed* at 100 meters with the delay that flights accumulate at the departure.

In particular, Miami and Reykjavik airports, despite the fact that present high values of precipitation and wind speed average, the table 2 underlines that the percentages increase significantly. This leads to possible questions on how airports handle negative weather conditions.

Concerning the Athens airport, it shows a peculiar behavior. The delay does not only decrease as precipitation increases, but it also becomes negative.

What stands out from graph 6 and 8, Malpensa airport has the highest number of minutes of delay, nevertheless the percentage delay remains more or less constant in precipitation or wind conditions. The latter behavior occurs also for Manila and Narita airports. Conversely, Bogotà airport is the one that presents the greatest value of percentage both in precipitation and wind case.

All collected data and the analysis we have done with, permitted us to give a possible answer to our research question, pursuing step by step in this report.

For a further analysis, there are some considerations that can be done. Firstly, the period of data collection could be extended to a whole year to involve all the seasons for each flight departure city. In addition, information related to weather have limitations, due to the fact that we got data from a public API, maybe different to the source from which airports take official data. Another part of our study which could be improved, is finding a different data source, for weather conditions, that acquires data more frequently than every hour.

References

Athens. Greece. [Online; accessed 18-Feb-2024].

Bogotà. Columbia. [Online; accessed 18-Feb-2024].

IATA. 2024. [Online; accessed 18-Feb-2024].

Manila. Philippines. [Online; accessed 18-Feb-2024].

Miami. USA. [Online; accessed 18-Feb-2024].

Milan. Italy. [Online; accessed 18-Feb-2024].

Narita. Japan. [Online; accessed 18-Feb-2024].

Open-Meteo. 2024. [Online; accessed 18-Feb-2024].

Reykjavik. Iceland. [Online; accessed 18-Feb-2024].

Tableau. 2024. [Online; accessed 18-Feb-2024].