REPORT SD201

Project : Prediction of flights delay

Members : Mohamed Jarmouni, Amine Azamman,
Quentin Bernhard, Jonathan Guerin

Table of contents

Problem

Our goal is to try to predict when a flight is delayed at arrival. In order to do that we are going to implement some algorithms that we saw in class.

Dataset

Why did we choose this data set ?

We started looking at various datasets that interested us. We firstly wanted to study and classify astral bodies but the dataset was not big enough so we decided to move on. Then we looked at market finance data. We could easily get a huge amount of data (for instance we had access to intraday crude oil prices from 2002 to 2022) which represented more than a million data points. The issue was that the data would have been hard to study so we finally decided to settle with a flight delay dataset.

This dataset is composed of a large amount of flight information between 2018 and 2022. There are almost 100 features we can study and more than 30 million flights. So we finally decided to study this dataset because of its number of datapoints (features and number of flights) but also because the topic caught our attention. In fact, when one travels, having a flight that is delayed is always an issue and a burden. So having a way to predict if a certain flight will be delayed (and by how much time) is something that could be really useful.

Cleaning the dataset

Seeing how big the dataset was, we had to make sure to clean the dataset before doing anything else.

To clean the data we had to decide which features we wanted to keep. We decided to keep 11 so that the clustering and regression algorithms could work well. Here is the list of the features we decided to keep :

- Date (DD/MM/YYYY format, we will have to convert it to a TIMESTAMP in order for the algorithms to work)
- Airline name (unique for each airline)
- Origin (Three letter code that indicates which airport the flight has taken off)
- Destination (Three letter code that indicates which airport the flight has landed on)
- Canceled (if the flight was canceled or not, boolean)
- DepTime (the actual time of departure in hh:mm)
- DepDelay (the delay of departure in minutes, if the delay is negative it is set to 0)
- ArrTime (the actual time of departure in hh:mm)
- ArrDelay (the delay of arrival in minutes, if the delay is negative it is set to 0)
- AirTime (duration of the flight in minutes)
- Distance (distance in miles between the 2 airports)
- FlightNumber (code for the flight number)

We remarked that the dataset was not “clean”, meaning that there were NaN values so we ran a data.dropna() to get rid of the flights that did not have arrival delays.

Once we had kept the features we were interested in we had to choose which datasets to keep as the number of datapoints was still too big. At first we decided to keep only certain airlines flights thinking it was going to decrease the number of datapoints to a reasonable size. We decided to keep the flights of these airlines : 'Endeavor Air Inc.', 'Delta Air Lines Inc.', 'Southwest Airlines Co.', 'Alaska Airlines Inc.', 'Hawaiian Airlines Inc.', 'SkyWest Airlines Inc.', 'Comair Inc.', 'Spirit Air Lines'.

At first we wanted to make this process with a python notebook to help allow a better understanding of our code. The issue is that notebooks are much slower than .py files, and seeing the size of the dataset we had to finally settle with a .py file. Although it was faster, we still had methods that were taking more than 10 minutes to run. In fact, we discovered that the dataframe.drop(indexes) method was really slow so we instead used the method data[~data.index.isin(indexToDrop)].copy() which allowed us to drop the rows we did not want in a much faster way.

Once we were done we still had 3.2 million flights so we decided to do a scramble and chose random flights from the already cleaned dataset. We wanted to have around 10.000 flights so we randomly selected this number with the help of pandas. We finally had a dataset that we could work with.

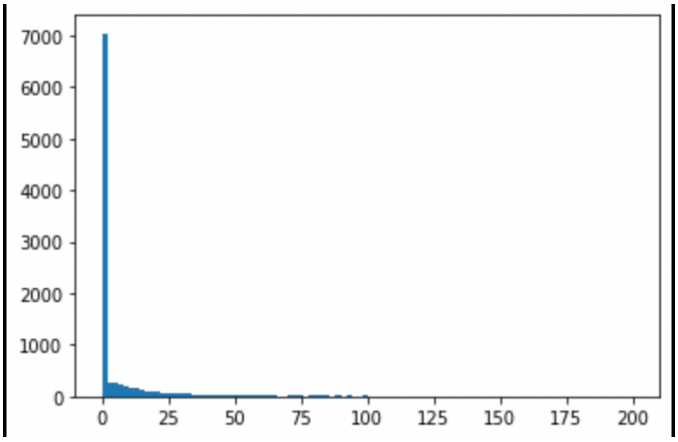
Statistical analysis of the data

First of all we decided to calculate and plot the correlation matrix to see how each variable is related to one another :

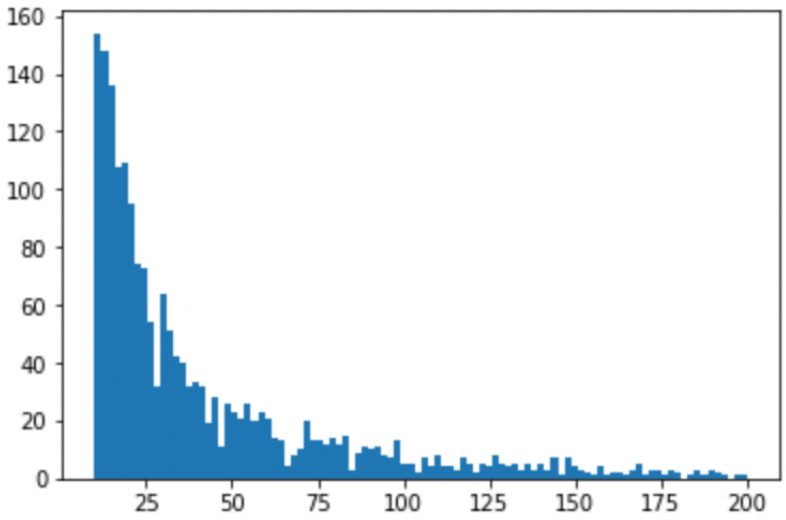
	Airline	Origin	Destination	DepTime	DepDelay	ArrTime	ArrDelay	AirTime	Distance	FlightNumber
Airline	1.00	0.01	0.01	-0.00	-0.00	0.01	0.01	-0.18	-0.19	0.43
Origin	0.01	1.00	0.04	-0.03	0.02	0.01	0.02	0.08	0.10	-0.03
Destination	0.01	0.04	1.00	0.02	0.02	0.02	0.02	0.12	0.11	-0.05
DepTime	-0.00	-0.03	0.02	1.00	0.11	0.69	0.09	-0.04	-0.04	0.01
DepDelay	-0.00	0.02	0.02	0.11	1.00	0.01	0.98	0.01	0.01	0.04
ArrTime	0.01	0.01	0.02	0.69	0.01	1.00	0.00	0.03	0.03	0.01
ArrDelay	0.01	0.02	0.02	0.09	0.98	0.00	1.00	0.01	0.01	0.05
AirTime	-0.18	0.08	0.12	-0.04	0.01	0.03	0.01	1.00	0.98	-0.30
Distance	-0.19	0.10	0.11	-0.04	0.01	0.03	0.01	0.98	1.00	-0.32
FlightNumber	0.43	-0.03	-0.05	0.01	0.04	0.01	0.05	-0.30	-0.32	1.00

We can see that departure time and arrival time are quite correlated which seems logical, that Distance and Airtime are also correlated (but the correlation is not equal to one which shows that sometimes the flights are delayed, maybe due to bad weather conditions etc…) and arrival delay and departure delay are also correlated which once again seems completely logical.

We can then try to calculate how many flights are delayed : We find that out of 10000 flights, 3099 are delayed at arrival. But we can be more precise and plot the histogram of the arrival delays :



We can see most of the delays are lower than 10 minutes. We may want to study how the ones higher than 10 minutes as they are the ones which really count :



Intuitively, we can think that the number of delays follows a power law (this was demonstrated in the paper “Statistical characterization of airplane delays” by Evangelos Mitsokapas).

Then we calculated the mean of the arrival delays which we found is equal to 11,4 minutes. We also found thanks to the quantiles that 10% of flights had a delay higher than 30 minutes which seems a lot knowing how much money airlines lose because of delays.

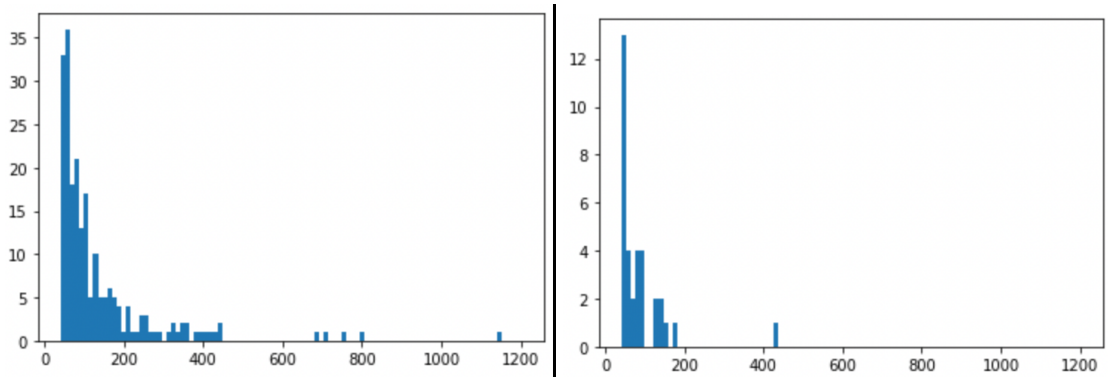
But a more interesting analysis would be to calculate the delays depending on the airline.

```
Delta Air Lines Inc. : Mean =  9.35  , Percentage of delays =  0.26
Southwest Airlines Co. : Mean =  10.8  , Percentage of delays =  0.34
Spirit Air Lines : Mean =  9.17  , Percentage of delays =  0.27
SkyWest Airlines Inc. : Mean =  16.8  , Percentage of delays =  0.31
Alaska Airlines Inc. : Mean =  8.52  , Percentage of delays =  0.33
Comair Inc. : Mean =  11.0  , Percentage of delays =  0.31
Endeavor Air Inc. : Mean =  9.06  , Percentage of delays =  0.24
Hawaiian Airlines Inc. : Mean =  6.73  , Percentage of delays =  0.32
```

We can see that Skywest has a way higher mean than the other companies without having much more flights delayed. By calculating the 0.9 quantile :

```
Delta Air Lines Inc. : 0.9 quantile =  18.0
Southwest Airlines Co. : 0.9 quantile =  31.0
Spirit Air Lines : 0.9 quantile =  24.8
SkyWest Airlines Inc. : 0.9 quantile =  42.0
Alaska Airlines Inc. : 0.9 quantile =  23.0
Comair Inc. : 0.9 quantile =  29.0
Endeavor Air Inc. : 0.9 quantile =  19.0
Hawaiian Airlines Inc. : 0.9 quantile =  21.6
```

We can then understand that the flights of SkyWest that are delayed, are delayed by much more than for the other companies. We can indeed confirm that by looking at the histograms of SkyWest (left) compared to Alaska Airlines (right) :



Analysis

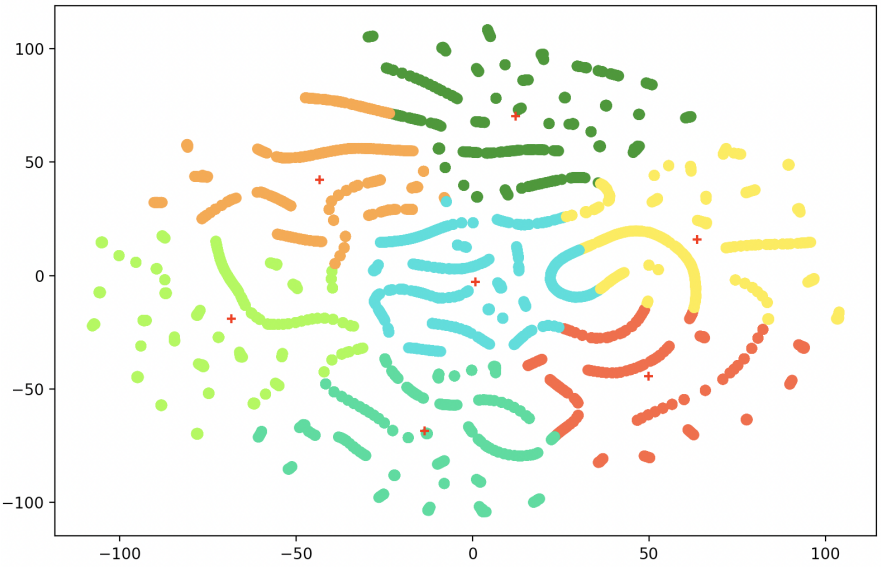
Clustering

Our goal with trying to cluster the data was to see if we could predict a flight delay by seeing if it was part of a certain cluster of delayed flights. This meant that we had to “drop” the delay features as it would not make sense to try to predict the delay if the delay was already a feature. “Dropping” the delay features means that we did not take into account the delay of departure, delay of arrival, airtime and arrival time as all of these features are directly related to the delay of a flight so it would not make sense to have them. This left us with only a few features : the airline, the date and time of departure, the airports and the distance between airports.

To make the clustering we still had to clean the data as all the features should be floats in order for the algorithm to work. Thus, we had to create python functions to convert the date and the airlines and airports to a number. Once we had done all of this, we could run the kmeans++ algorithm. In order to visualize the results we ran a TSNE

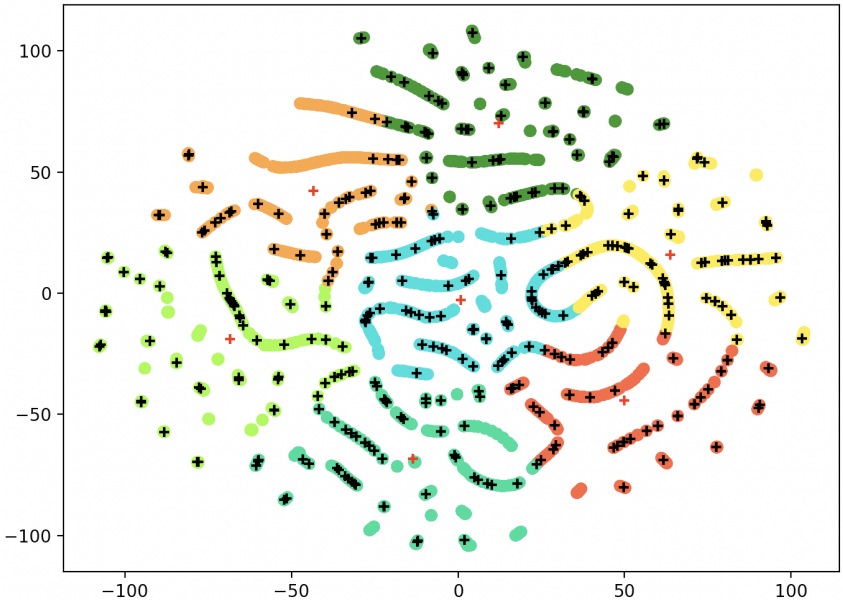
from the sklearn module, which lets us “compact” all of the features in a 2D space to facilitate visualization. This means that from the 6 features we had left, the algorithm transforms each data point into a 2 feature point. Kmeans++ has different inputs, the most important ones are the number of clusters we want to get but also the number of times the algorithm is run (n_init_). Seeing how the algorithm performed we chose to have 7 clusters and an n_init_ of 10. To find the best number of clusters we used the elbow method with SSE. We find that the optimal value is equal to 7 but the SSE is quite bad : 7561129.0. Thus we thought the results were not going to be good

Once we ran the kmeans algorithm we got this plot :



Each red cross is the center of each cluster and the points that have the same color are in the same cluster. The first thing that comes to mind when seeing the repartition are the lines, as if they were snakes. We found out that these lines correspond to the flights made by the same airline the same day.

Then we wanted to know if the clusters gave us an indication on whether the flight would be delayed or not. For that, we had to change the way we plotted the results to show when a flight was delayed or not. This led us to the next plotting : (the points with a black cross represent the flights that had more than 60 minutes of arrival delay)



We can see that there are no real clusters of delayed flights. When trying with other parameters for the TSNE and the Kmeans++ algorithms we still had the same results.

In conclusion, trying to predict the delay of a flight with the help of the kmeans algorithm does not seem to work. We could maybe have used an algorithm different from TSNE but seeing how little information we have about each flight, it does not seem likely that we could have gotten better results.

Classification

We now propose to focus on a different subset of data including the following columns ['Date', 'Airline', 'Origin', 'Destination', 'Cancelled', 'CRSDepTime', 'DepTime', 'ArrTime', 'ArrDelay', 'AirTime', 'CRSElapsedTime', 'ActualElapsedTime', 'Distance', 'DayOfWeek', 'Operating_Airline', 'Tail_Number', 'FlightNumber', 'OriginAirportID', 'OriginCityName', 'OriginStateName', 'DestAirportID', 'DestCityName', 'DestStateName', 'DepDel15', 'DepartureDelaGroups', 'DepTimeBlk', 'CRSArrTime', 'ArrDel15', 'ArrivalDelayGroups', 'ArrTimeBlk', 'DistanceGroup'].

We also proposes to begin with a bigger set of rows, of 60000.

We clean the data, convert time into timestamps to transform it into a real number, so that we can use to classify on and transform all data that are labels into corresponding int representing theirs classes using LabelEncoder().fit_transform.

We then separate the data into a training set, a test set(15%) and a validation set(5%)/

We implemented and tested the majority classifier but the most interesting part was the fit the model of a KNN-classifier with n_neighbors=3 and choosing the classes as DepDelay, we thus consider that each new time of delay constitutes a whole new classe. By training over X_train and y_train, we are able to get an precision of 0.6328491364520188, which is really good because we can see that even if the classifier is predicting that the plane will leave lately, it is counted as a fail he doesn't predict the time of delay correctly, sometime actually close from ground truth.

```
knn = KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train, y_train)
knn_preds = knn.predict(X_test)
```

On veut désormais tester la précision de ce classificateur

```
difference = y_test.subtract(knn_preds, fill_value=0)
a = difference[difference != 0].shape[0]
print("le ration de réussite (precision) est "+str((len(difference)-a)/len(difference)))
```

```
le ration de réussite (precision) est 0.6359373212855999
```

We then try to predict if the plane will be cancelled or not 1 = True = plane has been cancelled. We now use decision trees that hopeful get a good answer. And we do because the Decision tree is able to guess if the flight will be cancelled with a precision of 1.0.

```
from sklearn.tree import DecisionTreeClassifier, plot_tree

dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
dt_preds = dt.predict(X_test)
```

```
difference = y_test.subtract(dt_preds, fill_value=0)
a = difference[difference != 0].shape[0]
print("le ration de réussite (precision) est "+str((len(difference)-a)/len(difference)))
```

```
le ration de réussite (precision) est 1.0
```

However, now we can wonder if the Tree hasn't overfitted to the data, the problem is that the data here is big (40 000 of training) so this is not an obvious argument.