Flight Prices Prediction to Determine the Right Time to Buy Airline Tickets

Adrián Cerveró Orero

May 2021

Abstract

Flight prices fluctuate constantly, so buying one at one time or another can lead to considerable price differences. This project uses machine learning classification techniques in order to predict at a given time, considering properties of a flight, whether one should book the flight or wait for a better price. Using flights from 10 routes, the final model has been capable of getting around 10% of the original ticket price saved on average.

1. Introduction

Everyone who has bought airline tickets knows how prices change unexpectedly. Airlines use very sophisticated strategies and techniques known as 'revenue management'. These algorithms try to achieve the highest possible profit on a flight and take into account a lot of factors such as demand, the cost of aircraft maintenance, length of the flight, date and time of the flight, fuel, remaining seats… [1]

The reason for such a complicated system is that each flight only has a set number of seats to sell, so airlines have to regulate demand. When demand exceeds capacity, the airline may increase prices in order to decrease the rate seats are filling. A seat that remains empty represents a loss of revenue for the airline, so it will try to sell that seat at any price to minimize losses.

The goal of this project is to understand how airline ticket prices change over time and extract these factors that influence these price tendencies. Then, build a system capable of helping travelers make purchasing decisions by predicting if a given flight price will drop in the future or not.

Most of these factors are internal to the airlines and not accessible to the average customer. This makes our objective more difficult. For this project, I choose to focus on factors that are visible to anyone (such as departure day, departure time, the current price of the flight, etc…) and build a classifier that predicts the binary class of whether the customer should wait for a price drop or not.

2. Related Work

The issue of flight pricing has been studied since 2003 [2]. And it has had different ways of approaching it. In this chapter, we will first present several studies that are related to the subject of our project. Then, we will look at some existing tools that use these types of techniques in a real-world context.

In 2003, Etzioni et al. proposed a model that aims to tell users the perfect timing to buy the ticket or not [2]. They use just data from two routes: Los Angeles to Boston and Seattle to

Washington, D.C. Their model used 5 features which are: flight number, hours until departure date, airline, price and route. The final result was achieved by using an ensemble method and voting. Then they used the 'buy' or 'wait' suggestions to calculate the cost of each simulated passenger. The total amount of money saved using his strategy can reach 61.8%. However, their solution is not able to tell when the flight will reach the minimum price.

Similar to Etzioni et al. work, Bingchuan and Yudong [3] used a Bayes classifier to calculate the probability of a price change in the next hours or days. They focused on the Shanghai-Tokyo route, and selected only flights at a specific time of 9-10 am. To do this, they collected data for a whole year and then built a system capable of forecasting a smart buy-or-wait suggestion to customers. Their model reaches an average of 95.42% accuracy.

Groves and Gini proposed a regression model in 2011 [4]. Their system has two steps; in the first, they used a regression model to make predictions of the price of a flight on a given day, secondly, they set a threshold at which if the price is below that value, the passenger should buy the ticket. Otherwise the passenger should wait. They conclude that, given sufficient publicly-observable information, it is possible to predict airline ticket prices sufficiently to reduce costs for customers.

These types of techniques have been applied by some online travel agencies such as Kayak[1]. Although they themselves suggest to take these predictions as complementary and not determinant for the final decision of the customer [10]. They make a prediction about the next days indicating whether the price will drop in this period or not.

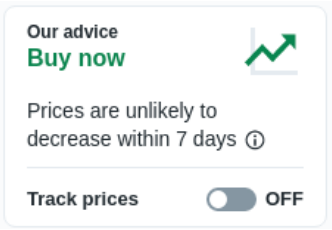


Figure 2.1: Kayak's purchasing suggestion

Google Flights[2] follows another approach: it indicates if the price is low compared to previous prices for that route and departure date.

Figure 2.2: Google Flights purchasing suggestions

[1] https://www.kayak.com/
[2] https://www.google.com/travel/

The Airhint[3] approach is the closest to this project. Their models predict the probability of price drop and make a suggestion to the customer. They have an individual model for each airline and only make predictions for low-cost airlines like Ryanair.

3. Methodology

3.1. Data Collection

I have wanted the data used for this project to be collected by me. On the one hand, to face the difficulties involved, and on the other hand, because this type of data such as historical flight prices are not openly available on the internet.

The data consists of flights made between February 1 and the end of April. They are flights traveling through 10 selected routes. All routes are from Spain and I have tried to make them varied in terms of distance/duration. The routes are the following:

- Madrid (MAD) - New York (JFK)
- Madrid (MAD) - London (LHR)
- Madrid (MAD) - Mexico City (MEX)
- Madrid (MAD) - Buenos Aires (EZE)
- Madrid (MAD) - Barcelona (BCN)
- Madrid (MAD) - Tenerife (TFN)
- Barcelona (BCN) - London (LGW)
- Barcelona (BCN) - Palma de Mallorca (PMI)
- Barcelona (BCN) - Rome (FCO)
- Barcelona (BCN) - Amsterdam (AMS)

As past flight data it is not freely and openly available on the internet. The only option is to collect daily data of current flights over a period of time in an automated way.
To extract the data there were two main approaches. Either use a web scraper to collect data from an online travel agency website directly (such as Skyscanner or Kayak), or use an API from one of these agencies. Finally, I opted for the second option and I used the API offered by Kiwi[4], as it seemed more comfortable and easy since I had already worked with API's in the past. The Kiwi API documentation is available at the following link: https://docs.kiwi.com/.

For the automation part I used Google Cloud to deploy a virtual machine with my collection data script[5] and using cron-log Linux functions. Flight data has been requested every day at the same hour for three months and stored automatically in a SQL Database powered by Google Cloud as well.

---

[3] https://www.airhint.com/
[4] https://www.kiwi.com/
[5] Available on repo: /src/scripts/collect_data.py

Such the API returns many variables about each flight, I have selected the ones that could be more useful for the problem. They are the following:

- Origin Airport
- Origin City
- Destination Airport
- Destination City
- Departure Day
- Departure Time
- Arrival Time
- Route
- Airline
- Seats
- Flight Nº
- Collection Date (when data was taken)

3.2. Data preparation

Before building new features and preparing the data for training the model. I perform an exploratory data analysis which can be found at the notebooks folder in the Github repository.

New features created from existing ones during the process:

- Days to Departure
- Session (flight departure at morning, evening or night)
- Day of Week (weekday of the departure day (monday, thursday...))

When it came to approaching the objective of this project, I tried several approaches:

1. A regression approach in which we predict the price of a flight on a particular day. Then, given a traveler who wants to buy a ticket on a given day check the price of each day until the departure of the flight. If the price drops, return wait, otherwise recommend to buy now.

I discarded this approach because it seemed too costly to make so many predictions per traveler and the error would cascade.

2. The second option is a regression model that predicts the exact number of days to wait. In this approach, we have to check each flight if the price drops in the remaining days to departure and compute the days between current day and the day with the minimum price.

Initially, I chose this approach but the results were poor and very few travelers were able to save money. It turned out to be very difficult to predict the exact days to wait. So I tried a different approach.

3. Classify the data into “Buy” or “Wait”. This way we only have to predict one binary number, making our goal more manageable. Although the disadvantage is we lose the way to predict the days to wait directly.

This last option is the approach followed for the rest of the project.

As for predicting the days to wait, we can only use the historical prices to estimate when the price will be minimum. While trying to implement the first option, I realized that at the individual level a lot of information is lost since there are flights for which we have only a few observations in our dataset. That is, we do not have the complete price path of a particular flight over time.

Therefore, I chose to aggregate the flights to compensate for this lack of data. We will use the flight groups to estimate when a flight will reach the minimum price. That is, if a group of flights reaches the lowest price 15 days before departure. We will estimate that any flight belonging to that group will also do so in the same way. We will use these estimates to subsequently calculate the savings or losses of the travelers.

Aggregating the flights

First of all, we have divided the data into two blocks. The last 30 days of flights have been used as test/valid and the rest as train. The train flights are the ones that we are going to grouping. We’ll use the test/valid flights to simulate passengers purchases.

For grouping the data we have used the features: ‘orig-dest’, ‘airline’, ‘session’ and ‘days_until_departure’.

	orig-dest	airline	session	days_until_dep	min	median	q25	competition	fly_duration	customPrice	wait	prob
9115	BCN-LGW	AF	night	34	151.0	151.0	151.0	2	9.166667	152.0000	0	0.733333
14231	BCN-PMI	IB	evening	77	95.0	102.0	95.0	4	17.041667	101.1125	1	0.866667
3481	BCN-AMS	TO	evening	6	230.0	230.0	230.0	1	18.666667	210.3500	1	0.241379
32471	MAD-MEX	LH	night	56	272.0	345.0	318.0	31	20.430108	320.4125	1	0.937500
1028	BCN-AMS	DL	night	6	197.0	197.0	197.0	1	2.500000	190.0000	0	0.333333

Figure 3.1: Dataframe with flights grouped

We also created a new feature that it is the count of each group as a measure of a competition factor. Then, we have to combine prices. And for this, there are also several options. At first, we simply used the first quartile as a function to aggregate the flight prices. This is because I had seen with the EDA, the minimum prices were very irregular and they do not follow a clear trend. And as we are interested in low prices I chose the first quartile.

Later on, I decided to add a new feature that combines prices called customPrice. This is the price giving more weight to recent days prices. This idea for combining price is extracted

from this paper [9]. This method has turned out to work better than using the first quantile in our model. The formula for calculating this customPrice is:

customPrice = w * (First quartile for entire time period) + (1-w) * (First quartile of last x days)

The best results have been obtained with w = 0.8 and x = 10 in our final model.

After we choose the feature to represent price for each group, we need to label the data with wait or buy values. We simply check for each group if the price drops in the next days. In this step we introduce a variable 'min_drop_per' which is the threshold that the difference between the current price and the price after waiting has to be overcome to be worthwhile to wait. This variable was quite significant in the final results.

Using the labels we create another important feature for the model: 'prob'. This variable measures the percentage of 'wait' in similar groups. We use this variable as some kind of measure for the probability of a flight price decrease. In Figure 3.2 we can see how, in general, this variable decreases until departure day as the departure day approaches as we can expect.

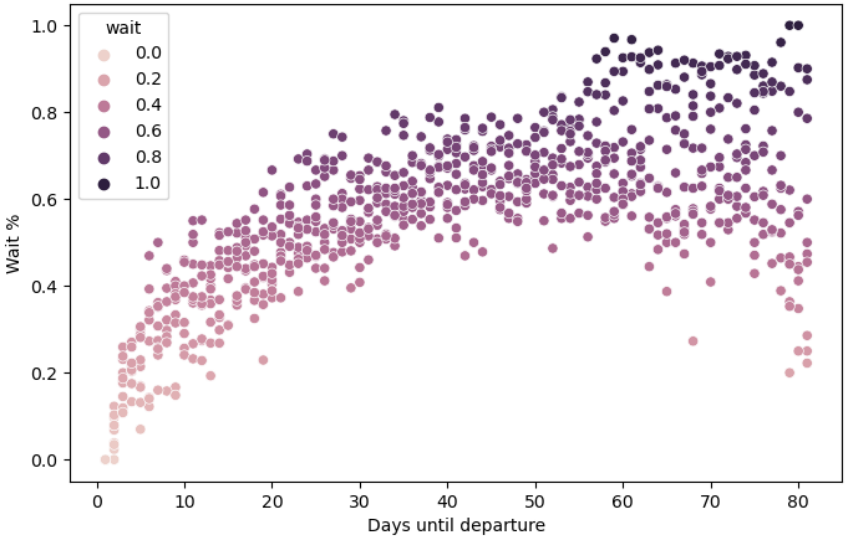


Figure 3.2: prob feature correlation with days until departure. Flights have been grouped by 'orig-dest' and 'days_until_dep' for visualization.

Days to wait

To estimate the days to wait we have created a dataframe (Fig. 3.3) containing a series of estimated prices for each group of flights and according to the days remaining to departure.

	orig-dest	airline	days_bins	price_est
8665	MAD-EZE	FR	(46, 51]	330.50
7736	MAD-BCN	TO	(41, 46]	104.25
2957	BCN-FCO	U2	(66, 71]	95.00
12498	MAD-LHR	SK	(11, 16]	254.00
5733	BCN-PMI	LY	(26, 31]	319.75

Figure 3.3: “price_bins” dataframe. Needed to estimate days to wait.

The variable 'days_until_dep' has been split into bins. There are two reasons for this. First, because it is quite difficult to predict the exact day so we decided to give the customer a range of approximate days to wait. Second, the groups data is too sparse to calculate estimated prices, this is because not all airlines have flight prices for all values of 'days_until_dep'.

3.3. Model

We have created two Python classes to carry out the model training experiments: PriceEstimatorTrainer and PriceEstimator.

- **PriceEstimatorTrainer**: This class has been used to perform the training experiments, evaluate on validation and test, hypertuning, simulate the travelers and compute the money saved...

- **PriceEstimator**: This class was created for the front-end/model interaction. It receives a trained model as input parameter and the output are several values: wait/buy, probability of flight price drop, and days to wait.

Metrics chosen

Before choosing a metric to evaluate the model. The aim must be clarified. For this project it has been decided that the main goal is to maximize the money saved by passengers.

The classes are imbalanced so accuracy was ignored, hence, we will look at f1-score, precision and recall.

During the process of choosing a model. It was quite difficult to come up with a model that was really accurate and would generate a lot of savings. It was found that generally the higher the f1-score, the lower the savings. This is because the higher the accuracy the more false positives the model generated. False positives have turned out to be the main cause of losses for a traveler. This is expected, since a false positive is a customer who has been advised to wait when the price is not really going to decrease.

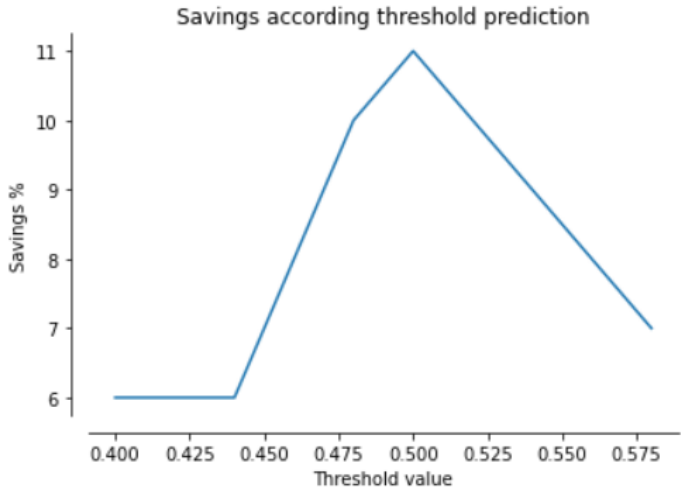
For this reason, it has been decided to give priority to models that generate very few false positives. That is to say a model with a high precision even if it is sacrificing accuracy or recall. In this way many false positives are generated but they are preferable because they do not lead to loss of money.

In short, we have chosen a fairly conservative model that predicts 'buy' most of the time but when it predicts 'wait' it does so with quite good precision. Therefore, the metrics chosen to select the model have been the precision and the average money saved per traveler (as a percentage since the price of tickets varies a lot among routes).

Anyway, a model with a higher f1-score but lower savings has also been built (model B) as an alternative to the proposed model (model A)

Parameters

The parameters we have tried to tweak are, on the one hand, the parameters of the sklearn model itself and, on the other hand, we have experimented with adjusting the threshold of the probabilities when making the predictions to try to reduce false positives. Although we have finally stayed with the default 0.5.



We have also adjusted the different variables that we discussed earlier in the data preparation phase. The ones that had the most impact on the model were “min_drop_per” and “w” used to calculate the customPrice feature.

Also, we have tested different features to group the flights. The big problem that has been presented in this project is that as soon as we group using many different variables the groups are empty or with very few flights.

3.4. Results

We have tested different classification models. The best results have been obtained by a Random Forest, so this has been chosen as the final model.

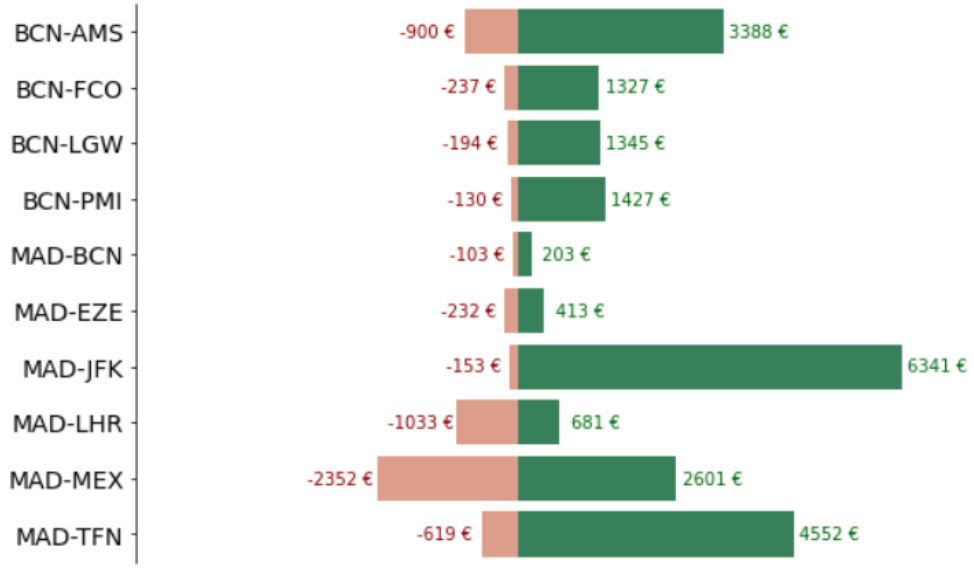
Metric	RandomForest	LogisticRegression	XGBoost	KNN
Precision (valid)	0.77	0.68	0.66	0.67
Savings* (valid)	11%	7%	5%	6%
Precision (test)	0.74	0.69	0.64	0.69
Savings (test)	10%	6%	4%	5%

Total results on test with final model (model A):

Model	Savings (k€)	Losses (k€)	Mean (%)	Accuracy	f1-score	Precision	Recall
RandomForestClassifier	22.3	-6.0	10.0	43.1%	30.76%	74.57%	19.38%

Results on test with final model by route (model A):

orig-dest	Savings (k€)	Losses (k€)	Mean Savings (€)	Savings Percentage	Wait predicted
BCN-AMS	3.4	-0.9	8.89	8.42%	18.17%
BCN-FCO	1.3	-0.2	7.52	11.44%	11.18%
BCN-LGW	1.3	-0.2	12.51	13.72%	21.65%
BCN-PMI	1.4	-0.1	5.71	23.82%	33.78%
MAD-BCN	0.2	-0.1	2.27	2.44%	4.84%
MAD-EZE	0.4	-0.2	5.32	1.12%	25.95%
MAD-JFK	6.3	-0.2	23.00	11.17%	18.96%
MAD-LHR	0.7	-1.0	-2.19	-5.93%	12.73%
MAD-MEX	2.6	-2.4	1.77	0.25%	10.23%
MAD-TFN	4.6	-0.6	13.02	12.36%	31.36%



- As can be observed, savings are greater than losses on most routes except on Madrid-London (MAD-LHR).
- The maximum percentage of money saved by a traveler is 74% of the original ticket price.
- Of the 10,000 simulated passengers, only 1695 (17 %) were predicted to wait. Of these 1695 wait passengers 1264 (74.6 %) were predicted correctly.

As an alternative, the results of model B are shown below.

Model	Savings (k€)	Losses (k€)	Mean (%)	Accuracy	f1-score	Precision	Recall
RandomForestClassifier	84.5	-56.8	4.0	62.02%	73.28%	70.01%	76.87%

orig-dest	Savings (k€)	Losses (k€)	Mean Savings (€)	Savings Percentage	Wait predicted
BCN-AMS	11.6	-4.7	6.18	4.07%	74.23%
BCN-FCO	7.8	-3.0	5.43	5.51%	67.22%
BCN-LGW	5.0	-1.4	10.28	8.1%	79.37%
BCN-PMI	4.2	-0.3	6.79	22.54%	81.65%
MAD-BCN	1.7	-0.9	1.95	1.83%	44.2%
MAD-EZE	3.0	-0.3	20.84	5.88%	97.74%
MAD-JFK	12.4	-17.1	-3.57	-2.36%	94.04%
MAD-LHR	6.8	-2.6	6.56	4.46%	50.84%
MAD-MEX	21.4	-24.0	-2.01	-1.4%	96.1%
MAD-TFN	10.7	-2.6	10.96	11.12%	73.93%

Conclusions

- In this project I have been able to go through all the phases of a typical data science project.

- The final model has at least managed to generate savings for the simulator travelers. This is not too bad considering that only 3 months of data were used.

- The API used was very limited in terms of the flights it returned, possibly because it is free. A real travel agency for example that wants to deploy a similar application for its customers would have much more and higher quality data. Despite these difficulties the final model looks "useful".

- The money saved most of the time by the travelers is too small to be worth the risk of waiting for a better price in a real environment. Even with perfect predictions.

- The final model is capable of not only suggesting to buy or wait but also suggesting the number of days to wait. Which is not usual in tools of this type in real-world apps.

Bibliography

[1] Pak, Kevin & Piersma, Nanda. (2002). “Airline Revenue Management”. Erasmus Research Institute of Management (ERIM), Research Paper.

[2] Etzioni, Oren & Knoblock, Craig & Tuchinda, Rattapoom & Yates, Alexander. (2003). To Buy or Not to Buy: Mining Airline Fare Data to Minimize Ticket Purchase Price. https://homes.cs.washington.edu/~etzioni/papers/kdd-2003.pdf

[3] B. Liu, Y. Tan and H. Zhou, "A Bayesian predictor of airline class seats based on multinomial event model," 2016 IEEE International Conference on Big Data (Big Data), 2016, pp. 1787-1791, doi: 10.1109/BigData.2016.7840795.

[4] Groves, William & Gini, Maria. (2013). An agent for optimizing airline ticket purchasing. 12th International Conference on Autonomous Agents and Multiagent Systems 2013, AAMAS 2013. 2. 1341-1342.

[5] Zouaoui, Faker & Rao, Bejugum. (2009). Dynamic pricing of opaque airline tickets. Journal of Revenue and Pricing Management. 8. 148-154. 10.1057/rpm.2008.50.

[6] Papadakis, M. Predicting Airfare Prices.

[7] Tziridis, Konstantinos & Kalampokas, Theofanis & Papakostas, George & Diamantaras, Kostas. (2017). Airfare Prices Prediction Using Machine Learning Techniques. 10.23919/EUSIPCO.2017.8081365.

[8] Polamuri, S. (2017). How the Random Forest Algorithm Works in Machine Learning. http://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/

[9] Joshi, Achyut & Sikaria, Himanshu & Devireddy, Tarun. (2017). Predicting Flight Prices in India.

[10] Price Trends & Tips Explanation (https://www.kayak.com/price-trend-explanation)