ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Systems@ETH Zürich

Master’s Thesis Nr. 49

Systems Group, Department of Computer Science, ETH Zurich

in collaboration with

Amadeus IT Group SA

Flight Delay Prediction

by

Vincent Martinez

Supervised by

Prof. Donald Kossmann
Prof. Andreas Krause
Charles-Antoine Robelin

September 2011 – March 2012

inf Informatik
Computer Science

amadeus
Your technology partner

Abstract

Flight delays are quite frequent (19% of the US domestic flights arrive more than 15 minutes late), and are a major source of frustration and cost for the passengers. As we will see, some flights are more frequently delayed than others, and there is an interest in providing this information to travelers. As delays are a stochastic phenomenon, it is interesting to study their entire probability distributions, instead of looking for an average value.

This master's thesis proposes models to estimate delay probability distribution, based on a method called kernel density estimation and its extensions. These are data-driven methods, meaning that it does not try to model the underlying processes, but only consider past observations.

Our models, of increasing complexity, have been implemented, optimized and evaluated on a large scale, using several years of records of US domestic flights delays. During the evaluation, we will measure the good performance of some of the models to predict delay distributions, in spite of the intrinsic difficulty of measuring the goodness of fit between a probability distribution and the corresponding random experiment.

Acknowledgments

First, I would like to thank Amadeus for the opportunity I had to conduct my master thesis in the IT industry. Especially, I want to thank Charles-Antoine Robelin, my supervisor for this thesis, and head of the Search department of the Operational Research and Innovation division, to which I was attached. In addition to support and advises for my project, he gave me an interesting insight on the possible applications of my work, and more generally on the scientific, technical and innovative challenges faced by his team.

I also want to thank Prof. Andreas Krause for the directions and feedback he gave me during this thesis, his constant availability, and his ability to popularize his broad knowledge in the fields of statistics and data mining.

I am also very grateful to Prof. Donald Kossmann, for having set up the partnership with Amadeus and made this master thesis possible. I also enjoyed his open-mindedness, his curiosity about my work, and his experience regarding the way of conducting such a project.

Finally, I want to thank all the members of the ORI division at Amadeus, for their ability to integrate newcomers, their constant friendliness and their will to share their knowledge, which made this internship very enjoyable.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

The continuous increase of storage capacities and computational power is currently pulling the development of data analytics. Indeed, companies (and especially IT-intensive ones) are collecting massive volume of data (often referred as Big Data), such as web logs, customer information, production and sales tracking, etc.

Analyzing these datasets, with data mining algorithms for example, allows the extraction of information that can help a company to gain knowledge (for example on customers' behaviors) or to use the information as a basis for new products or services.

Amadeus, historically in charge of transaction processing for the travel and tourism industry, is developing new products to enhance the customer experience during the process of searching for a trip. In addition to the list of possible flight connections for a journey, a piece of information that can be provided is the risk of missing a connection. Knowing this probability can help the traveler to choose the best route, and the travel agent to adapt its suggestions or even prices.

In order to evaluate the risk of missing a connection, we need like to know the probability of the incoming flight being too late to be able to catch the second flight, taking into account the incompressible time necessary to go from the arrival gate to the departure gate of the second flight (possibly including immigration control). Models already exist to estimate the gate-to-gate transfer time. The goal of this master thesis is to build a model for the prediction of flight arrival delays.

1.2 Problem Statement

As explained, the goal of this project is to estimate the probability of any flight to be more than x minutes late, for any x being the difference between the total connection time and the time to go to the departure gate.

Moreover, as we would like to give this information to the customer during the search and reservation process, the model will have to give long-term predictions, up to several months forward, and will not take into account short-term effects, like current weather or traffic situation.

This model will be based on the unique public large dataset of flight delays, provided by the Bureau of Transportation Statistics of the United States Department of Transportation. This dataset is only composed of USA domestic flights, with data from 1995 for all major airlines.

1.3 Contribution

The prediction of short term delays (for the next hours or so) is already a largely explored field. Indeed, using information about weather conditions, airports congestion and current flight delays allows quite accurate predictions of future delays, as some parameters influencing them are known, even if they still have a random component). For example, the website FlightCaster exploit several sources of information (airports, airlines, weather and possibly historical data) to provide probabilities of being on-time, less than one hour late or more than one hour late, to travelers. However, this website is using the same estimations for all the flights when no short-term information is available.

A lot of researches have also been conducted on the management and propagation of flight delays, focused on traffic management systems. Mueller and Chatterji [1] tried to model the departure, en-route and arrival delays with Normal or Poisson distributions, that could possibly be taken into account in traffic management systems. Those rough models do not take into account any characteristic of the flights, but only give global trends. In another article focused on Ground Delay Programs improvement by Allan et al [2], are studied in details the meteorological conditions and their impact on on-ground and flight delays.

On a short-term perspective, an article by Zonglei et al [3] presents predictions of the overall traffic status on an airport (percentage of delayed flights), using decision trees and neural networks.

Finally, Tu & Ball tried to estimate in [4] the departure delay distribution by modeling the underlying mechanisms, with three components: a seasonal trend, a daily trend and a random residual, fitted using genetic algorithm. Their method seems quite expensive to compute and is not extensively tested. However it is close to the goal of my project, so we will try to compare our future results with the one presented in this article.

This project was specifically focused on customer long-term information. Its main contributions are the selection of the optimal models with the most relevant parameters, their optimization regarding the specificities of this project, and also their efficient implementations, given the large amount of data we wanted to exploit.

1.4 Thesis Structure

We will first introduce the concepts and methods used during my thesis, covering both the models construction and their evaluation. Then we will present the dataset used for this project, and we will try to identify the main factors influencing the delays.

These factors will then be combined in different ways and with different methods (mainly kernel density estimation, kernel conditional density estimation and decision tree), and we will also describe how to implement these models efficiently.

Finally, we will present different methods to measure the performances of the models. These methods will be first used to optimize the models, in order to get the best possible predictions, and then we will evaluate them in different use cases.

Chapter 2

Related Works

2.1 Kernel Density Estimation

2.1.1 Definition

The main goal of this project is to estimate the probability density of the delay from the data we had. As we do not want to make any assumption about this density or the factors influencing it, we had to use non-parametric statistics. Indeed, the orientation of this project was not to model separately the influence of different parameters on the delay.

If we have samples following an unknown probability density, the most common way of estimating this density is the histogram. However, its shape highly depends on the number of bins, their widths and their positions.

A much smoother and more accurate way of estimating a probability density is called Kernel Density Estimation, also known as Parzen-Rosenblatt window method, and extensively described by Silverman in [5]. It is composed of a sum of kernels, centered on each observation (see Figure 2.1). This data-centered method avoids the problem of number and positions of the bins we have with the histogram.

A kernel is a symmetric probability density function used for smoothing. The simple idea behind the smoothing is to consider that if we observed in a dataset a delay of 10 minutes, it means that the probability of being 10 minutes late is high, but we also give some probability to the possibility of being 9 or 11 minutes late, and maybe 8 and 12 minutes too. This width of influence of an observation is call the bandwidth, or smoothing parameter.

Mathematically speaking, if we have some observations $(x_i)_{1\leq i\leq n}$, we can compute the kernel density estimator as:

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i)$$

where $K_h(x) = \frac{1}{h}K\left(\frac{x}{h}\right)$ with K a kernel function, and h the bandwidth.

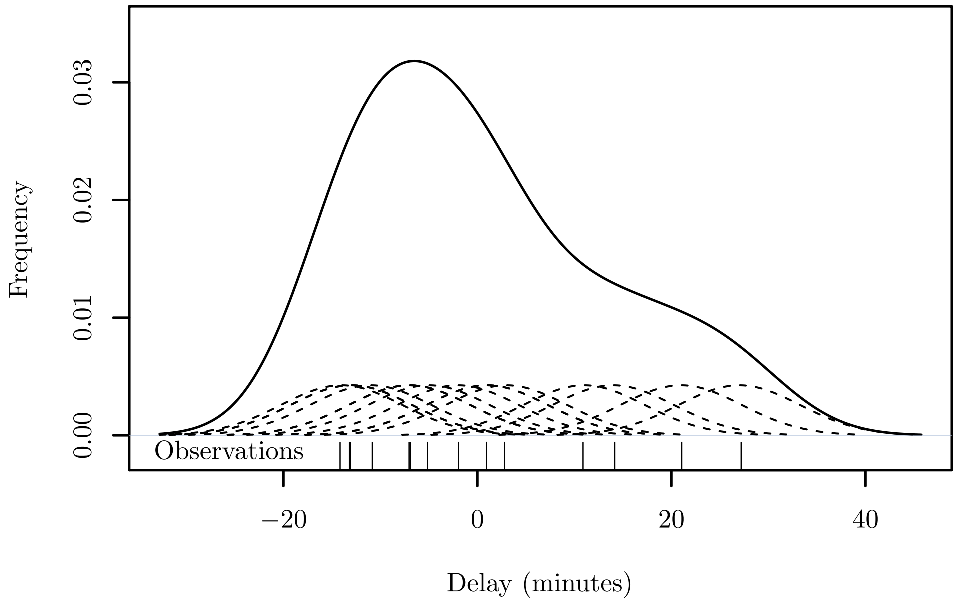

Figure 2.1: Kernel Density Estimation construction

This summation can be accelerated by using fast Fourier transform, considering that this kernel estimation is a convolution of the data with the kernel function (described by Silverman in [5]).

The most common kernel is the Gaussian kernel, following the normal distribution. However, it has an infinite support (i.e., it has a strictly positive value for all real numbers), so it may be useful (mainly for computation efficiency) to use a finite-support kernel, such as the Epanechnikov kernel, defined in Table 2.1 and represented in Figure 2.2. The Epanechnikov kernel had been specifically designed to minimize the approximation error of kernel density estimation [6]. The cumulative distribution function is the probability of observing a value smaller or equal to x: $\mathrm{CDF}_K(x) = P(X \leq x) = \int_{-\infty}^{x} K(y)dy$.

Kernel name	$K(x)$	$\mathrm{CDF}_K(x)$
Gaussian	$\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}$	$\frac{1}{2}\left(1+\mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)\right)$
Epanechnikov	$\frac{3}{4}(1-x^2)\ \mathbf{1}_{\{\lvert x\rvert\leq 1\}}$	$-\frac{x^3}{4}+\frac{3}{4}x+\frac{1}{2}$

Table 2.1: Mathematical definitions of kernel functions

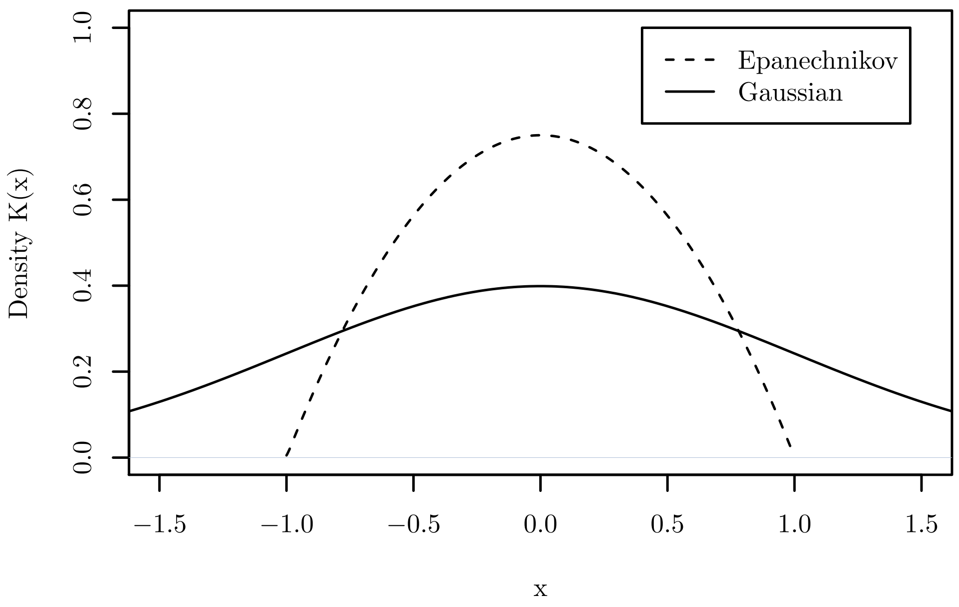

Figure 2.2: Gaussian and Epanechnikov kernels of bandwidth 1

2.1.2 Bandwidth Optimization

The challenge with kernel density estimation is to find the right level of smoothing. The bandwidth h changes the width of the kernel function: $K_h(x) = \frac{1}{h}K\left(\frac{x}{h}\right)$, which is still a probability density function (its integral is 1). We can also notice that $\mathrm{CDF}_{K_h}(x) = \int_{-\infty}^{x} K_h(y)dy = \int_{-\infty}^{x} \frac{1}{h}K(\frac{y}{h})dy = \int_{-\infty}^{\frac{x}{h}} K(y)dy = \mathrm{CDF}_K(\frac{x}{h})$

An over-smoothed estimation produces a flat density which does not provide any useful information. On the contrary, an under-smoothed estimation has very high peaks on each observation and is null elsewhere, which is not be a realistic density.

There are three main methods to compute the optimal bandwidth, that is to say the bandwidth such as the kernel density estimation fits the best the unknown density followed by the data. The details and proofs of the methods are not relevant for this project, and are described in [5].

The first optimization method is called mean integrated squared error minimization and tries to minimize $E(\int(\hat{f} - f)^2)$. However, the real distribution function f is unknown, but its expectation can be estimated from $\hat{f}$.

The second method for optimizing the bandwidth is the likelihood cross-validation, which tries to maximize the probability given to each sample x_i by a model $\hat{f_{-i}}$ trained on all the other samples: $\frac{1}{n}\sum_{i=1}^{n} \log \hat{f_{-i}}(x_i)$.

The third method introduced by Sheather & Jones in [7] is not based on cross-validation, but on solving an equation defining analytically the bandwidth that minimizes the asymptotic mean integrated squared error.

In addition to these computational methods that can be expensive, Silverman's rule of thumb [5] allows a quick evaluation of what would be the optimal bandwidth if the distribution function to be estimated was a normal distribution:

$$h = 1.06\hat{\sigma}n^{-1/5} \tag{2.1}$$

with $\hat{\sigma}$ the standard deviation of the observations. This approximation is also re-used in [8].

Silverman also proposed a variant of this rule of thumb, avoiding over-smoothing in case of a long-tailed or skewed distribution:

$$h = 0.9 \min\left(\hat{\sigma}, \frac{IQR}{1.34}\right) n^{-1/5} \tag{2.2}$$

IQR being the inter-quartile range (difference between the 75th and the 25th percentiles).

As a side note, the kernel density estimation method can be extended to multivariate kernel density estimation, which can for example estimate the bivariate probability distribution of events characterized by two observations. Another extension, called adaptive kernel density estimation, uses a variable bandwidth depending on the observations (for example, by using a larger bandwidth in low density regions).

2.2 Kernel Conditional Density Estimation

2.2.1 Definition

Kernel density estimation method can also be extended to take into account conditional probabilities, that is to say estimating the probability distribution of a variable that depends on another variable. In our project, we would like to estimate the distribution of the delay, taking into account some conditions, like the airline, the route or the hour of a flight. We will see in the following chapter that these factors play an important role in the delays.

The interest of this method is that, in addition to a smoothing on the dimension of the observed variable (like in kernel density estimation), we add a smoothing in the conditions dimension: for example, the probability distribution of the delays of flights arriving at 14:00 can provide information about the distribution of delays of flights arriving at 13:45 or 14:15, all else being equal. Indeed, we can consider that for a small variation of the conditions, the overall behavior of delays is almost similar. Consequently, observations with similar conditions provide some information, and can in some way fill gaps in sparse data.

As previously, one of the challenges is to find the optimal level of smoothing in the conditions dimension. Another issue is to identify relevant conditions to include in the model. Moreover, discrete and continuous parameters can be relevant and have to be treated differently.

We consider a set of observations $(\mathbf{x}_i, y_i)_{1\leq i\leq n}$, composed of a vector of numerical conditions $\mathbf{x}_i$ (that can be the departure time, the day of week or the month, in our case) and the observed delay of the flight y_i. From this dataset, the kernel conditional density estimation [9, 10] defines a way to compute the probability of a delay y given the conditions $\mathbf{x}$:

$$\hat{f}(y \mid \mathbf{x}) = \frac{\sum_{i=1}^{n} K_{h_1}(y - y_i) K_{h_2}(\|\mathbf{x} - \mathbf{x}_i\|)}{\sum_{i=1}^{n} K_{h_2}(\|\mathbf{x} - \mathbf{x}_i\|)} \tag{2.3}$$

where $K_h(x) = \frac{1}{h^d} K\left(\frac{x}{h}\right)$ with K a kernel function.

By applying this formula to a wide range of possible delays y, it is possible to get the full probability distribution, for any specific conditions $\mathbf{x}$.

This model depends on three parameters: h_1 and h_2 (the bandwidths for delay and conditions, respectively), and the kernel function K, common to both smoothing dimensions. The bandwidth h_2 is common for all the dimensions of the conditions, which are treated together when computing the distance $\|\mathbf{x} - \mathbf{x}_i\|$. As a consequence, the data have to be normalized, by subtracting the mean and dividing by the standard deviation, so as to get data centered on 0 and with a standard deviation of 1. Otherwise, a bandwidth of 1 would not smooth in the same way, for example an hour condition (having values between 0 and 24) and a day of week condition (between 1 and 7).

2.2.2 Bandwidths Optimization

We now have to optimize h_1 and h_2, so that the estimator fits the underlying distribution, and is able to overcome sparse data. As for kernel density estimation, most optimization methods are based on likelihood maximization, or mean squared error minimization.

The goad of cross-validated likelihood maximization is to maximize the average probability given to each observation by the model trained on the others $n-1$ records. The highest this probability is, the better the model predicts the given observations. The log-likelihood computation presented by Holmes in [10] requires the computation of $v(i,j) = K_{h_1}(y_i - y_j) K_{h_2}(\|\mathbf{x}_i - \mathbf{x}_j\|)$, for each $i \neq j$, which requires $O(n^2)$ computation of this formula.

One of the computational optimization proposed is to recursively split the data along all dimensions (conditions and delay) and to store the bounding boxes of each split. Based on the bounding boxes, we can see that records i and j in two distinct boxes have a $v(i,j)$ null or small enough. In this case, we can avoid the computation of $v(i,j)$ for all pairs of values in the two boxes.

The second optimization proposed in [10], called Monte-Carlo approach, looks at samples of records in the box rather than the boxes boundaries, and estimate $v(i, j)$ using bootstrap resampling.

Other articles focus on mean squared error minimization [11, 12]. However, these approaches seem complex and may not be scalable to large datasets, and we will see later that a fine tuning of the bandwidths is not very important for our use case.

2.3 Kolmogorov-Smirnov Test

We now introduce some methods that will be used to evaluate our models, by comparing a predicted probability distribution with the distribution observed on test samples.

The Kolmogorov-Smirnov test is used to compare a sample of observations with a probability distribution and to measure the goodness of fit of the sample to the distribution. It is non-parametric, meaning it does not rely on any assumption about the observations or the distribution.

From a sample of n observations (X_i), we can define their empirical distribution function $\hat{F}(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{\{x \geq X_i\}}$, $\mathbf{1}$ being the indicator function. A concrete example will be seen later on Figure 3.2, page 20.

The Kolmogorov-Smirnov statistic D measures the supremum distance between the empirical distribution function $\hat{F}(x)$ of a sample, and a given reference probability distribution $F(x)$: $D = \sup_x \left| \hat{F}(x) - F(x) \right|$. This can be seen on Figure 2.3 with a uniform reference distribution. Intuitively, a small distance D indicates a good fit between the sample and the reference distribution.

The null hypothesis of the Kolmogorov-Smirnov test is that the sample follows the reference distribution. We can associate to an observed statistic D, the probability of observing a statistic as large or larger than D if the null hypothesis were true. This probability can be computed using the Kolmogorov distribution [13], and is called the p-value. A p-value close to 0 (the significance level is often fixed at 5%) means that the null hypothesis can be rejected (i.e., it is very unlikely that the sample follows the reference distribution). Otherwise, we cannot reject the null hypothesis, and the p-value gives an indication of the goodness of fit, as it is proportional to D.

The p-value associated to a statistic D depends on the sample size: indeed, according to the law of large numbers, the larger the sample is, the more it should fit its reference distribution, and therefore, the smaller should be the statistic D. This relation can be seen on Figure 2.4, and is the reason why we will use the p-value has the indicator of goodness of fit, instead of the statistic D which does not allow comparison of samples of different sizes.

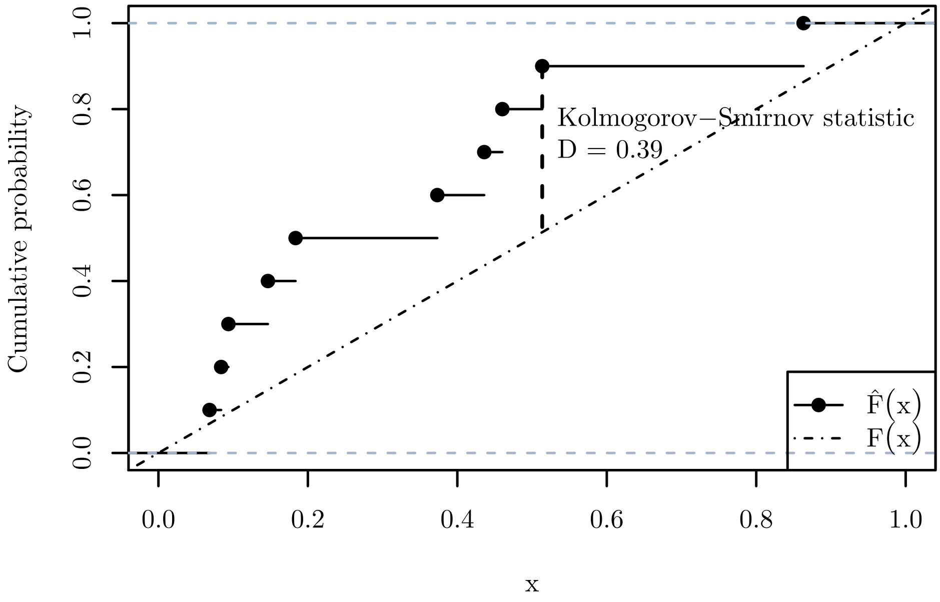

Figure 2.3: Empirical Distribution Function of a random uniform sample, and its Kolmogorov-Smirnov distance to a uniform distribution

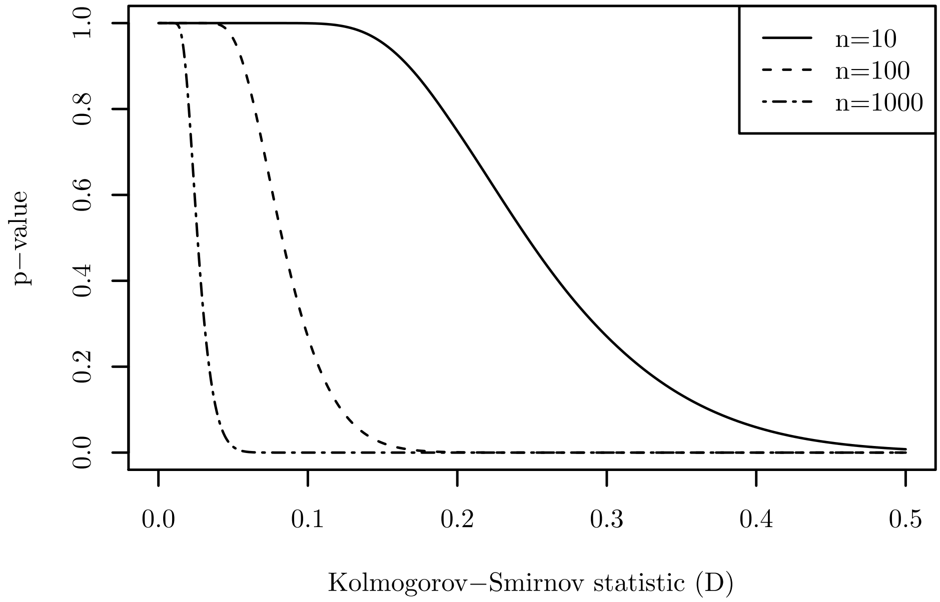

Figure 2.4: Relation between Kolmogorov-Smirnov statistic and p-value, for different sample sizes

2.4 Receiver Operating Characteristic

Instead of predicting the full delay probability distribution, we may also be interested in a simpler predictor giving probabilities of being delayed or not (we will define later the limit between the two classes). Such a predictor is called a binary classifier, and gives a prediction using a set of parameters associated to the observation.

A binary classifier can be discrete, when it returns only True (delayed, in our case) or False (not delayed). In this case, the performance of the classifier can be easily measured by comparing the returned classes with the classes of the test samples, using a confusion matrix and derived measures (specificity, sensitivity, accuracy, etc.).

A good classifier should have a high True Positive Rate (which is the ratio of correctly predicted delays to the total number of actual delays; a TPR of 1 meaning that all delays were well predicted), and a low False Positive Rate (which is the ratio of on-time samples predicted as delayed, to the total number of on-time samples; a FPR of 0 meaning that no on-time samples were predicted as delayed).

Some binary classifiers are continuous, and return a probability of begin delayed $P(\text{delayed})$ and a probability of not being delayed. As described in [14], such a classifier can be turned into a discrete classifier by setting a threshold probability $p_{\text{threshold}}$, in a way that if $P(\text{delayed}) > p_{\text{threshold}}$, the classifier returns "delayed", and otherwise "not delayed". As a consequence, each value of $p_{\text{threshold}}$ produces a different discrete classifier, with different performance.

The performance of all the discrete classifiers derived from a continuous classifier can be visualized on a Receiver Operating Characteristic curve (later referred at ROC curve), representing the True Positive Rate versus the False Positive Rate of each classifier (see Figure 2.5).

If the TPR is smaller than the FPR, it means that the inverse classifier (returning the opposite answer) would have a better performance. Equals TPR and FPR (diagonal line) indicate that the classifier is equivalent to a random guess. Finally, a perfect classifier (TPR = 1 and FPR = 0) would be in the top left-hand corner.

As a consequence, we can measure the performance of a classifier by looking at how close to the top left-hand corner it can perform: this is computed by calculating the area under the ROC curve (abbreviated as AUC). Following the previous discussion, a random classifier would have an AUC of 0.5, and a perfect classifier an AUC of 1. The AUC also allows a comparison of the performance of several classifiers, easier than by comparing the ROC curves. For example, the ROC curve of Figure 2.5 has an AUC of 0.62.

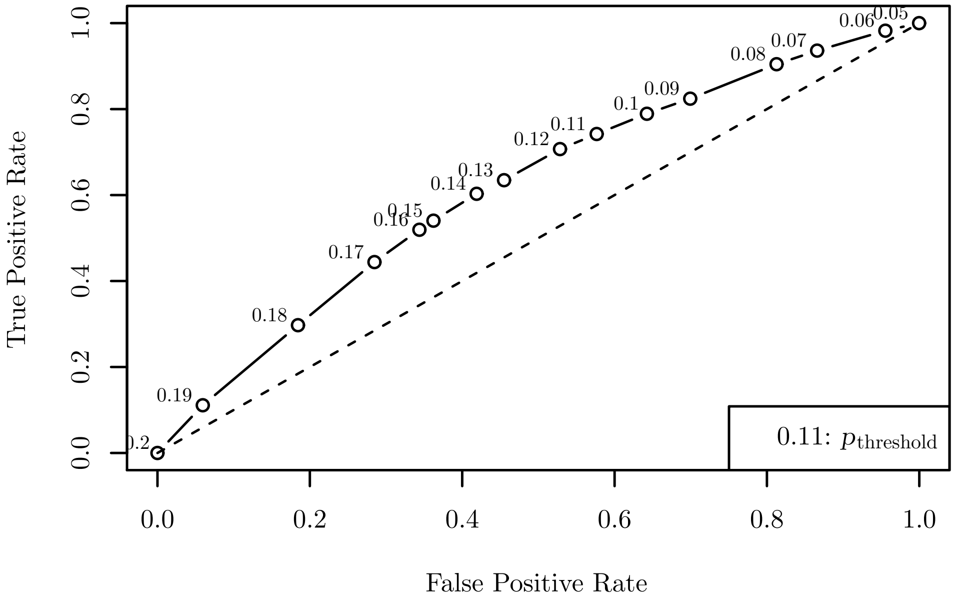

Figure 2.5: Example of a ROC curve, with TPR and FPR values for different $p_{\text{threshold}}$

Chapter 3

Data Exploration

Before going into the details of the delay prediction models, we will describe the dataset used. This way, we can get a better idea of the phenomenon we would like to estimate, how we can measure it and which factors can have an influence on it.

3.1 Dataset Description

The dataset used during this project is composed of publicly available data, collected by the Bureau of Transportation Statistics, a US federal agency. This dataset is composed of records of all USA domestic flights of major airlines, from 1995 to 2010, with two different perspectives, one focused on the departures and the other one on the arrivals. More precisely, the concerned airlines are the one having at least 1 percent of total domestic scheduled-service passenger revenues, plus two air carriers that report voluntarily. The records cover nonstop scheduled-service flights between points within the United States (including territories).

For each flight were recorded:

- identification information: flight number, date and tail number (aircraft identification)
- flight information: carrier code (like AA for American Airlines), origin and destination airports codes
- time information: scheduled and actual departure (or arrival) hour, and their difference being the departure or arrival delay (with a one-minute precision)
- flight duration information: scheduled and actual duration
- on-ground information: taxiing duration and take-off or landing hour
- delay reason (if available): part of the delay that can be attributed to the carrier, to the weather, to the air traffic control, to security reasons or to late arrival of the aircraft from the previous flight.

Regarding the goal of this project, only the arrival delays will be considered. For the delay prediction model, only some of these parameters will be taken into account: the origin, destination and airline (categorical parameters), and the hour and date, which are continuous numeric parameters. Numeric parameters can be used as conditions for kernel conditional density estimation, but the categorical parameters have to be treated in a different way.

The data had to be filtered, to remove incomplete records, and records of cancelled flights, which were out of the scope of this project.

3.2 Data Management

Records from one year represent around 900MB of data, for around 10 to 14 million records. As we need a simple and fast execution of queries on this dataset, especially to avoid parsing large data files, we used a MySQL database to store the dataset. As the database was only used for read operations, the necessary indexes could be created, without worrying about the cost of index updates. The indexes were especially useful when looking for records of flights of specific origin and destination.

Moreover, MySQL provides an in-memory database engine, allowing much faster operations than on disk. By restraining the data to useful attributes and only arrival delays, the database size was reduced to around 250MB per year of data, that easily fits into main memory.

The other advantage of the database was that it could be easily interfaced with a `R` program, with some packages [15, 16] allowing to execute SQL queries and extract the results in a format adapted to `R`.

3.3 Facts and Figures

3.3.1 Generalities

As the behavior of airlines (e.g., scheduled duration of flights) and of traffic management systems can evolve over the years, this first statistical study of the dataset is conducted with 2010 records, which is already a reasonable amount of data.

First of all, we can have a look on the overall distribution of arrival delays, for all routes and airlines, in 2010 (around 6.3 million records). We observe on Figure 3.1 a minute-based histogram of the center of this distribution (the full distribution going from 127 minutes early to 1632 minutes late). We measure a median of -4 minutes, but a mean of 4.5 minutes and a standard deviation of 35.8 minutes, being sensitive to extreme values of delay. As we can see, there is a heavier tail on the right-hand side.

The Figure 3.2 represents the cumulative sum of the histogram, called Empirical Cumulative Distribution Function. We can see that the probability of an arrival delay less than or equal to 0 minute is 62%. We can also see more clearly the heavier tail on the positive delays side. This simple density is already a basic model of the delay probability distribution.

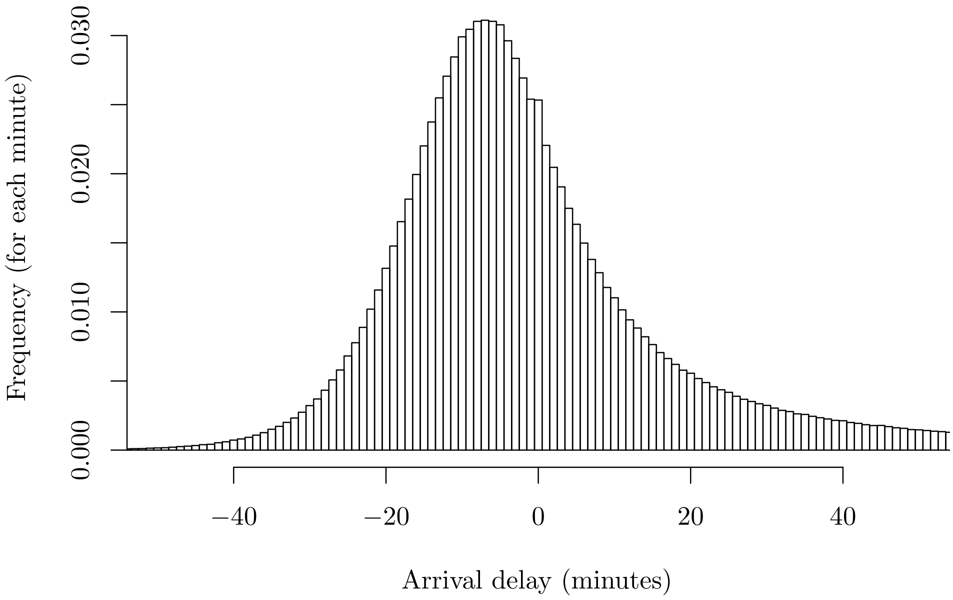

Figure 3.1: Histogram of arrival delay frequencies, for all 2010 records

3.3.2 Relevant Factors

To get an insight on the values of delay and their reasons, Figures 3.3 and 3.4 present the distribution of arrival delays, characterized by the 5th percentile, the median and the 95th percentile (in order to remove the outliers and because the long tails are less easy to compare).

Regarding the delay per scheduled arrival hour (Figure 3.3), we can observe for example that for flights arriving at 7am (local time at destination), 90% of the flights are between 23 minutes early and 28 minutes late. But at 9pm, this interval has to go up to 88 minutes late to cover 90% of the flights. There are few flights arriving between midnight and 6am, but for the rest of the day, there is a clear trend that more delays occur in the evening, even if the median stays just below 0.

Regarding the delay per airline (Figure 3.4), we can observe for example that Hawaiian Airlines performs very well. On the contrary, the low-cost carrier JetBlue Airlines has a very wide delay interval to cover 90% of flight delays.

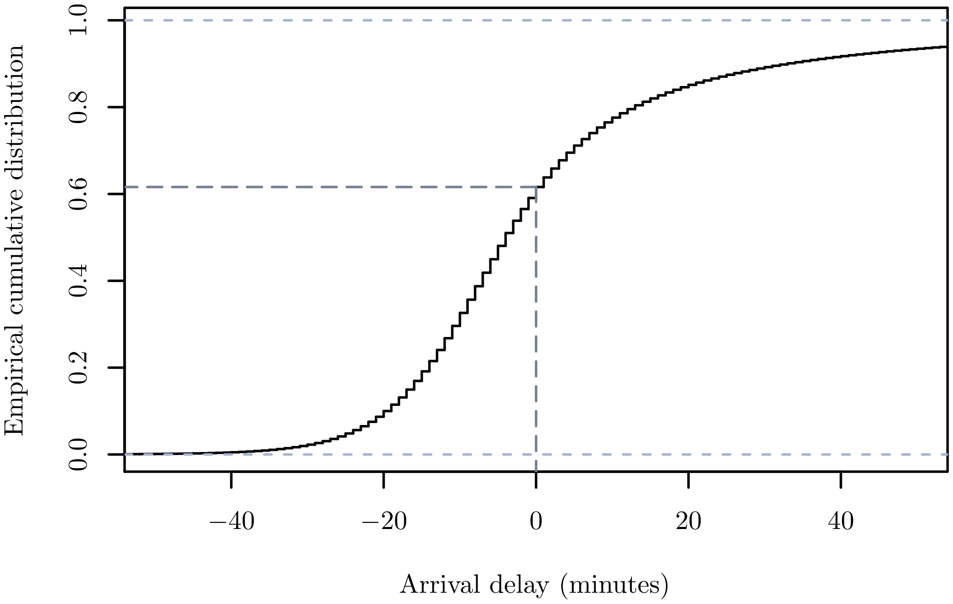

Figure 3.2: Empirical cumulative distribution of arrival delays, for all 2010 records

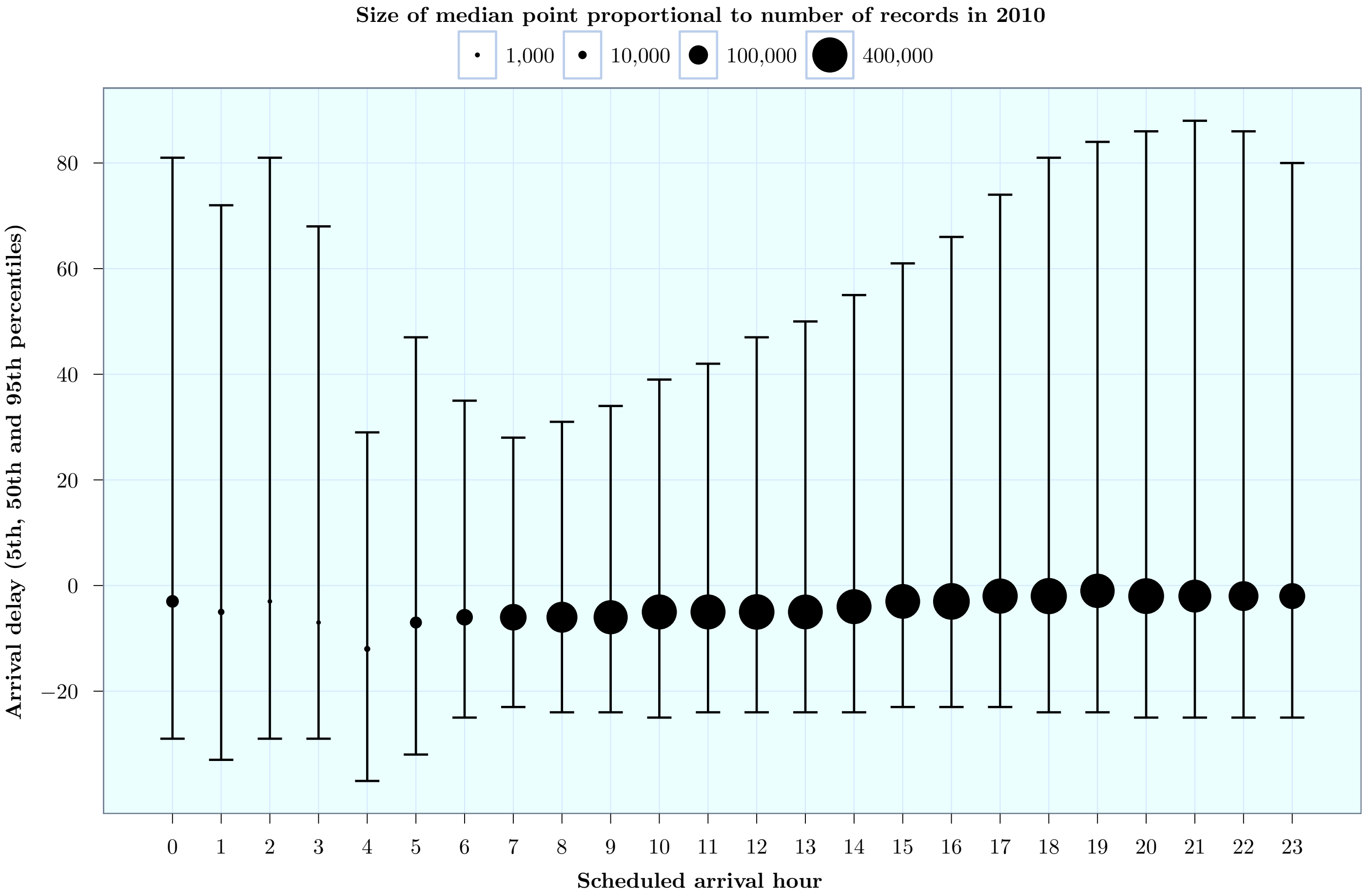

Figure 3.3: Arrival delay distribution per arrival hour, for all 2010 records

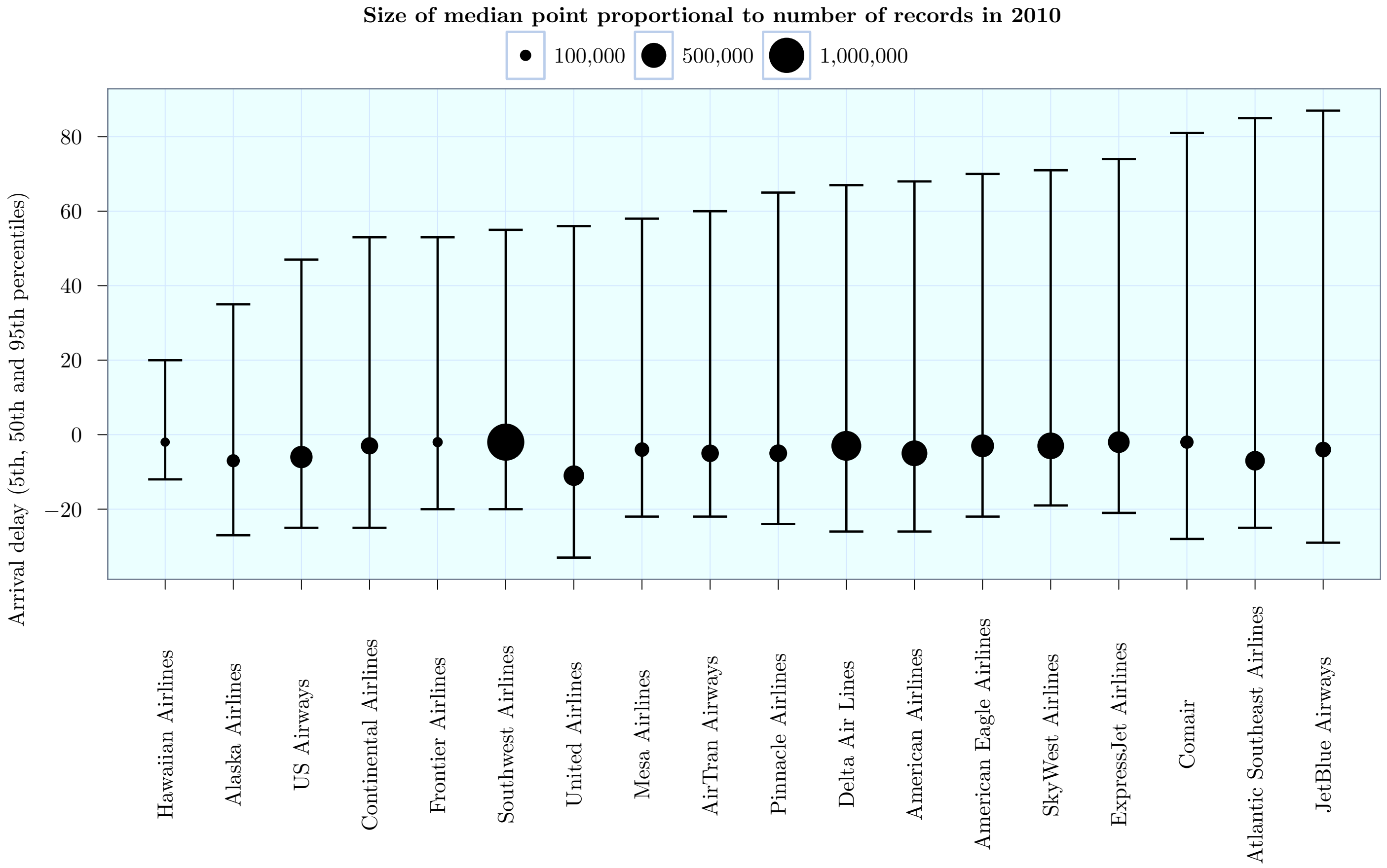

Figure 3.4: Arrival delay distribution per airline, for all 2010 records (ordered by 95th percentile)

This important variation of delay distributions with the hour or the airline is the reason why a sophisticated prediction model has to be built, in order to take into account all the parameters influencing the delay (or at least the most significant ones). The departure and arrival airports are also very important factors, but with around 300 airports, it is more difficult to visualize. However, as an illustration, Figure 3.5 represents the probability distribution of the arrival delays, during 2010, in three major airports with similar traffic (around 150,000 records per year): Salt Lake City, Detroit and San Francisco. We can observe, depending on the airport, different behaviors in terms of number of early flights, weight of the right-hand tail and also most frequent delay.

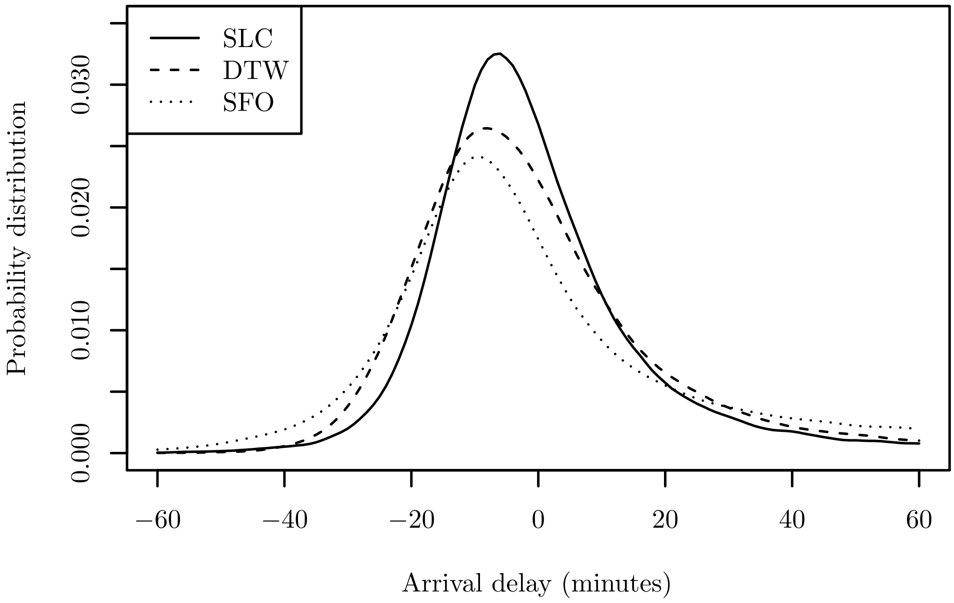

Figure 3.5: Arrival delay probability distribution, for all 2010 records and three airports

We can also observe on Table 3.1, the percentage of delayed flights and average delay of the top ten airports. We can see that the behavior in terms of delay does not only depends on the airport traffic or geographical position. San Francisco airport has the worst delays among the top ten airports in 2010, and only few small airports are worse.

Airport	Delayed flights ($\geq$ 15 min)	Average delay	Flights
Atlanta ATL	19.4%	5.9 min	405 254
Chicago ORD	19.9%	5.3 min	304 913
Dallas DFW	16.4%	2.9 min	262 904
Denver DEN	15.4%	0.8 min	235 759
Los Angeles LAX	17.4%	2.0 min	197 376
Houston IAH	17.0%	4.2 min	181 723
Phoenix PHX	14.0%	0.6 min	179 370
Detroit DTW	20.4%	5.5 min	156 893
Las Vegas LAS	16.9%	1.5 min	144 256
San Francisco SFO	27.1%	10.8 min	137 058
Global	18.6%	4.5 min	6 297 527

Table 3.1: Arrival delays for top ten airports and in average (2010 data)

Chapter 4

Models Description and Implementation

We have previously described the methods that can be used for density estimation. We have also seen what were the important factors influencing the delays. In this chapter, we will put this together, and describe how to apply the different methods, with which parameters, on which scale, and how they can be implemented efficiently.

4.1 Basic Prediction Models

The first models we consider are simple models, meaning that they do not take into account any characteristic of the flight to establish its delay probability distribution. We will see later, in Section 4.3, how such simple predictors can be put together to build a more complex model.

4.1.1 Empirical Cumulative Distribution Function

A very simple delay prediction model can be built from the empirical cumulative distribution function represented on Figure 3.2. This model takes into account none of the specificities of the flight, but represents an average behavior. This trade-off between quantity and specificity of data will be challenging during this entire project. We will see later, in Chapter 5, the performance of this simple model.

4.1.2 Kernel Density Estimation

The kernel density estimation is a very common method, and consequently is a standard function of many statistical programming languages, such as `R` which was used for this project.

I also implemented this method by myself, but the existing functions in `R` were much more optimized. As this method is very common, and only considered as a baseline for this project, I decided not to focus my effort here, but instead try to use the best of existing implementations. As a consequence, we will compare the existing implementations, present in two `R` packages, with various forms of kernels and several bandwidth optimization methods, in order to find the method that provide the best results for our use cases.

The default package “stats” can use many kernel forms (Gaussian and Epanechnikov presented previously, but also rectangular, triangular, biweight and cosine kernels). It also implements the following bandwidth selection methods: two rules of thumb (presented in Equation(2.1) and (2.2)), biased and unbiased cross-validation [17], and the method of Sheather & Jones [7] (based on analytical equation solving).

The other package, “ks” (Kernel smoothing) [18] implements plug-in selector [19] and smoothed cross-validation [20] optimization methods, and always uses a Gaussian kernel.

All these methods will be evaluated in Section 5.2.1, using both Gaussian and Epanechnikov kernels (the most common and efficient ones). We will try to find the method that best fits our problems.

4.2 Conditional Prediction Model

As seen in Chapter 3, the schedule arrival hour of the flight is an important condition determining the arrival delay. By making the hypothesis that the distribution of delay is continuous with the hour (meaning that a small change in the hour results in a small change in the delay distribution), we can integrate the scheduled arrival hour in our model, using Kernel Conditional Density Estimation. Other conditions, like the month or the day of week can also be integrated in kernel conditional density estimation, by considering them continuous. We will see in Chapter 5 if this improves or not the predictions.

4.2.1 Description

As a reminder, the formula for the kernel conditional density estimation trained on a dataset $(\mathbf{x}_i, y_i)_{1 \leq i \leq n}$, of conditions $\mathbf{x}_i$ and delay y_i, is:

$$\hat{f}(y \mid \mathbf{x}) = \frac{\sum_{i=1}^{n} K_{h_1}(y - y_i) K_{h_2}(\|\mathbf{x} - \mathbf{x}_i\|)}{\sum_{i=1}^{n} K_{h_2}(\|\mathbf{x} - \mathbf{x}_i\|)} \tag{4.1}$$

where $K_h(x) = \frac{1}{h^d} K\left(\frac{x}{h}\right)$.

Parameters like hour, month and day are cyclic, meaning that the distance between 23:00 and 1:00 is 2 hours (and not 22 hours that would be given by a simple absolute difference). Therefore, we implemented a computation of the distance $\|\mathbf{x}_i - \mathbf{x}_j\|$ that takes into account the need of applying a kind of modulo in the distance.

The Kolmogorov-Smirnov test and evaluation as a classifier, described previously, require the computation of the cumulative distribution function, instead of the probability density given by the previous equation (4.1). Given the definition of cumulative distribution function, we can compute, for a test sample $(\mathbf{x}^*, y^*)$, the cumulative probability (noted CP) for the delay y^*, given the conditions $\mathbf{x}^*$:

$$\begin{aligned} \mathrm{CP}(\mathbf{x}^*, y^*) &= \int_{-\infty}^{y^*} \hat{f}(y \mid \mathbf{x}^*) dy \\ &= \int_{-\infty}^{y^*} \frac{\sum_{i=1}^{n} K_{h_1}(y - y_i) K_{h_2}(\|\mathbf{x}^* - \mathbf{x}_i\|)}{\sum_{i=1}^{n} K_{h_2}(\|\mathbf{x}^* - \mathbf{x}_i\|)} dy \end{aligned} \tag{4.2}$$

4.2.2 Implementation Details

As the kernel conditional density estimation is the basis of the next steps of my project, it is interesting to optimize the computation of the Equation (4.2).

The `R` programming language [21] was mainly used for this project, because it is well-adapted to statistics, manipulation and representation [22] of large amounts of data. One of the main challenges faced by new `R` programmers is to get used to the vectorization: most of the computations can run directly on vectors or matrix, and this provides much faster results than the loops and iterations common in other languages.

`R` has also a large catalogue of packages offering advanced functionality. One interesting capability, provided by the package "parallel", was the exploitation of several CPUs (or CPU cores), which are very common on desktop computer or servers nowadays. As most of the computations had almost no inter-dependencies, they could to run in parallel without too much overhead.

Another very useful package is called Rcpp [23], and enables `R` functions to call compiled `C++`, that can be executed much faster (up to several orders of magnitude) than interpreted `R` code, especially for computation-intensive functions. The computation of the kernel conditional density estimation (Equations (4.1) and (4.2)) were implemented in `C++`, keeping all the data preparation, manipulation and post-computation analysis in `R`.

Delay Kernel

As seen in Equation (4.2), for each test sample $(\mathbf{x}^*, y^*)$, we compute the cumulative probability given by the model to the delay y^*, given the conditions $\mathbf{x}^*$. Instead of computing the integral by numerical integration, we can simplify this computation:

$$\begin{aligned}
\mathrm{CP}(\mathbf{x}^*, y^*) &= \int_{-\infty}^{y^*} dy \frac{\sum_{i=1}^{n} K_{h_1}(y - y_i) K_{h_2}(\|\mathbf{x}^* - \mathbf{x}_i\|)}{\sum_{i=1}^{n} K_{h_2}(\|\mathbf{x}^* - \mathbf{x}_i\|)} \\
&= \frac{\sum_{i=1}^{n} \int_{-\infty}^{y^*} (K_{h_1}(y - y_i) dy)\, K_{h_2}(\|\mathbf{x}^* - \mathbf{x}_i\|)}{\sum_{i=1}^{n} K_{h_2}(\|\mathbf{x}^* - \mathbf{x}_i\|)} \\
&= \frac{\sum_{i=1}^{n} \mathrm{CDF}_{K_{h_1}}(y^* - y_i) K_{h_2}(\|\mathbf{x}^* - \mathbf{x}_i\|)}{\sum_{i=1}^{n} K_{h_2}(\|\mathbf{x}^* - \mathbf{x}_i\|)}
\end{aligned}$$

The Cumulative Distribution Functions of Epanechnikov and Gaussian kernels were presented in Table 2.1, and their computations are straightforward. Moreover, as delays have a minute-based precision, y^* does not take an infinite range of values, but only few hundreds. As a consequence, we can save some computations by keeping in memory the values of the vector $\mathrm{CDF}_{K_{h_1}}(y^* - y_i)$ and reusing them for all test sample that have the same delay y^*. Another more memory-efficient solution is to sort the test samples by their delay y^*. In this case, we do not have to record the vector $\mathrm{CDF}_{K_{h_1}}(y^* - y_i)$ (which can be very long when using a large training dataset) for all values y^*: we can only test if the delay of the current test sample is the same as in the previous one, and reuse the value in this case.

Conditions Kernel

The next computational optimization concerns the computation of the condition kernel $K_{h_2}(\|\mathbf{x}^* - \mathbf{x}_i\|)$ in the case of a finite-support kernel (like the Epanechnikov kernel). Indeed, this value is null in many cases, when $\|\mathbf{x}^* - \mathbf{x}_i\| \geq h_2$. To minimize the number of computations, we have to focus only on points $\mathbf{x}_i$ close to $\mathbf{x}^*$ by less than h_2: this is called the fixed-radius neighbors problem. There is a large range of solutions to this problem [24]. The solution implemented is the one based on a grid, being simple to implement and populate, with a low memory cost and enough improvement for our use case. The idea is to divide the space of $\mathbf{x}$, that can have one or several dimensions (hour, day or month for example), into a grid of cells of size h_2, covering all the possible values of the conditions.

This grid is initialized by indexing the training data contained in each cell of the grid. Then for each test sample $\mathbf{x}^*$, we compute the condition kernel $K_{h_2}(\|\mathbf{x}^* - \mathbf{x}_i\|)$ only for training samples $\mathbf{x}_i$ that are in the same cell as $\mathbf{x}^*$ or in the neighbor cells. This avoids a lot of computations, and we finally have only few non-zero condition kernels to execute the computation of the cumulative probabilities described previously.

However, as the portion of space covered by the condition is bounded (like 0 to 24 hours), the number of training samples in the neighborhood of a test sample is directly proportional to the total number of training samples. If we have n training samples and m test samples, even with the optimization, the computation still has a complexity of $O(mn)$, but with a much smaller constant (that can be divided for example by around 100, with grid of size 15 minutes, over the 24 hours interval), at the cost of the initialization of the grid.

4.2.3 Parameters Optimization

The kernel conditional density estimation depends on three parameters: h_1 and h_2 (the bandwidths for delay and conditions, respectively), and the kernel function K, which have to be optimized when using this model. We will see later that Gaussian and Epanechnikov kernels provide estimators of similar performance. As Epanechnikov kernel is faster to compute (as described above), we will focus on this kernel for the rest of this project.

Regarding the optimization of the two bandwidths, I first implemented and analyzed in details the method proposed in [10], based on cross-validated likelihood maximization. The computational optimization, based on boundary boxes, allowed very good performance improvement for small bandwidths and finite-support kernels.

However, this likelihood maximization often led to small bandwidths, giving peak probabilities to the common delay values (which have a one-minute precision). Moreover, as most of the evaluations were based on the Kolmogorov-Smirnov test of goodness of fit, it did not make sense to optimize the model on a different metric that the one used during the evaluation phase.

As a consequence, the parameterization was conducted using the same method as for evaluation, described in the following chapter. The optimization concerned the combination of values of h_1 and h_2. Moreover, we will see that the performance of the models are continuous with the bandwidths, so even an educated guess of the bandwidth can already provide good results: for example, we can imagine that a good bandwidth for delay is not be 30 seconds or 1 hour, only by looking at the shape of the distribution in Figure 3.1. I spent some time on trying to finely optimize the bandwidths, before realizing that for my use case, it was not worth the effort, due to the intrinsic variability of delays influencing also the performance measurements.

4.3 Global Prediction Models

Up to now, we have described several predictors of delay probability distributions, based on some training dataset, and possibly taking into account continuous parameters, like the hour, day of week or month of flights. However, we have seen in Chapter 3 that categorical parameters like the origin, destination and airline, have also some influence on the delays. In order to build a more general model, we should now be able to handle categorical parameters.

4.3.1 Unified Models

As a baseline for evaluation, we use two simple models, based on a single predictor, ignoring the categorical parameters.

First of all, the empirical cumulative distribution function, as we have seen on Figure 3.2, can be used as a unique model, if we build it with all records and used it to predict delays, whatever the conditions (hour, route, etc.).

We can also use as a general predictor, a kernel conditional density estimator, trained on all data and used to predict delays, taking into account some conditions, like the arrival hour, and possibly the day of week and the month.

4.3.2 Route-based Models

The straightforward way to deal with categorical parameters is to use a specific predictor for each combination of these parameters. This way, the specificities of each combination are taken into account and not lost among all the rest of the data. However, as a consequence, we have less data to train the predictor.

The categorical parameters we consider for each flight are the origin, the destination and the airline, forming a specific route (operationally called a "leg"). The records from 2010 are composed of around 8600 routes, among which around 7250 have more than 50 annual records (i.e., at least one flight per week). The specificities of the airports may be their traffic or usual weather conditions for example. For the airlines, the specificities can be the tightness of their schedules, the quality of maintenance and the accuracy of flight duration prediction for example.

The predictor, used for each combination of parameters, can be a kernel density estimator, or a kernel conditional density estimator (with conditions on arrival hour, and possibly the day of week and month). One of the challenges with this method is to decide whether to optimize the bandwidth for each predictor, or to fix the same bandwidths for all of them.

4.3.3 Models using other Combinations of Parameters

However, we can imagine that a categorical parameter may have a greater influence on delays that the others. In this case, we can build a model composed of a predictor for each value of this important parameter, for example the destination airport. It reduces heavily the number of predictors to train, as we have around 300 airports, or 18 airlines, instead of almost 8000 routes.

We can also consider a model based on a predictor per combination of two parameters, like origin and destination and ignoring the airline. We will evaluate in the following chapter all these possible models.

4.3.4 Top-down Tree

Up to now, we have two kinds of models: models composed of a predictor per combination of parameters (e.g., route), and models with a unique predictor, identical for all records. As a compromise between very specific and very general models, we tried to build a model by grouping together routes having a “similar” behavior in terms of delay. Thus, we can expect gathering more data (that should improve the density estimations), at the cost of a loss of specificity.

Several approaches were considered for this grouping. The first approach was a bottom-up aggregation: grouping together a pair of routes, if each one has a good fit with a predictor trained on samples from the two routes, and repeating this aggregation. One of the problems with this method would have been the large number of couples of routes to evaluate (as we have around 8000 routes in the dataset). For the same reason, it would have been difficult to use common clustering methods.

The method we considered is based on the approach used to build decision trees: at each level of the tree, we look for the best variable (i.e., the one that separates the set into the most homogeneous subsets), and we split the data depending on the value of this variable (that can be numerical or categorical).

Concretely, at each step, we build a kernel conditional density estimator E, trained on the whole subset, with a condition on the arrival hour. Then for each parameter p (among origin, destination, airline, month, day of week) and for each value v_p of p, we select test samples such as $p = v_p$, evaluate their goodness-of-fit with the estimator E, and classify them as fitting or not fitting the estimator.

Finally, we choose as a splitting variable, the parameter p producing the best separation, meaning the largest subsets of fitting and non-fitting elements. Such a binary split can seem a bit rough, considering the diversity of behaviors that can exist among the non-fitting elements ; however it would have been difficult to quantify and group the different deviations to the average model.

This top-down construction has to be stopped at some depth, otherwise it might continue to split the dataset up to a leaf per route, or even finer if we consider the months and days of week. For the evaluation, we used trees of depth 1 to 6, meaning 2 to 64 leaves, each associated to a kernel conditional density estimator.

We tried to improve this method by considering, instead of the arrival delay (in minutes), a relative delay, meaning the ratio of the delay and the scheduled duration of the flight. This could enable an easier grouping of long and short haul flights, because we can imagine for example that a delay of 30 minutes is more frequent for a 6-hours flight than for a 45-minutes flight.

Chapter 5

Models Optimization and Evaluation

Up to now, we have described the methods and the different models we are interested in. Before evaluating and comparing them, we have to optimize their parameters, mainly the bandwidths. As explained previously, we will use for optimization the same metrics as for evaluation. These metrics will be introduced first, before presenting the results of the optimizations and comparing the optimized models.

5.1 Evaluation Methodologies

5.1.1 Kolmogorov-Smirnov Test

One of the evaluation measure we used is based on the cumulative distribution function (CDF). Considering the probability integral transform theorem, we know that if a random variable X follows a distribution such as its cumulative distribution function is F_X, then the random variable $Y = F_X(X)$ has a uniform distribution over $[0;1]$. (This enables, by inverse transform sampling, generating samples following a given CDF by applying the inverse of the CDF function to a uniformly distributed variable.)

The evaluation method used measures the uniformity of the cumulative probabilities given to the test samples by the model, using the Kolmogorov-Smirnov test. However, the goodness of fit obviously depends on the sample, so this measure has to be repeated enough times to get exploitable results.

This test can first be used to compare the distribution of samples of a specific flight to their predicted distributions. It can also be used to test for the uniformity of the cumulative probabilities given to any test samples by a more general model, composed for example of a predictor per route.

5.1.2 Likelihood and Confidence Interval

A common measure of the performance of a model is to look at the likelihood of the test samples $\{x_i\}$, given the model: we would like to maximize $\mathcal{L} = \prod P_{model}(X = x_i)$ (here we consider the probability density, and not cumulative probabilities). More practically, the average log-likelihood is often used: $\ell = \frac{1}{n} \ln \mathcal{L} = \frac{1}{n} \sum \log P_{model}(X = x_i)$. In addition, as the probability given by the model to some test samples can be zero, a prior uniform probability is usually added to the model probability (for example with a 0.01 coefficient), to avoid null likelihood.

Another aggregated measure we can use, and which was also used in [4], is to derive a confidence interval from the model: for example, 80% of the cumulative probabilities given by the model should be between 0.10 and 0.90. By computing the proportion of probabilities in such an interval, we can measure how close to the theoretical 80% the model is. The results for the optimized model presented in [4] were that 81.35% and 90.34% of the validation data were falling in the 80 and 90% confidence intervals.

5.1.3 Area Under the ROC Curve

As introduced in Section 2.4, we may also be interested in a simpler predictor, returning only a probability of being late or not. The performance of such a predictor is commonly measured by its Area Under ROC Curve (referred as AUC), a number between 0.5 (random classifier) and 1 (perfect classifier). This computation is implemented in `R` using the “ROCR” package [25].

As our density estimation models have a continuous output, we can define a family of binary classifiers, by varying $p_{\text{threshold}}$, in which each classifier returns “delayed” if $P(\text{delayed}) \ > \ p_{\text{threshold}}$.

We also have to define what means being delayed: we define a value τ such as $P(\text{delayed}) = P(X \geq \tau) = 1 - \text{CP}(\tau)$, CP being the cumulative probability.

This way, we can either evaluate a model for a fixed τ (for example 60 minutes), or look for the value τ offering the best classification performance, in terms of AUC.

The AUC measure cannot be applied to unconditional global models (like the empirical cumulative distribution function). Indeed, to a value of τ is attributed always the same probability, so all samples will be in the same class, and the TPR and FPR have the same value (both 0 or 1).

5.2 Models Optimization

5.2.1 Selection of the Optimization Method for Kernel Density Estimation

As we have seen previously, several optimization methods for kernel density estimation are implemented in R standard packages. We evaluated each one, using the Kolmogorov-Smirnov test, to select the one that best fits our needs.

These methods were evaluated for a model composed of a kernel density estimator per route. As explained previously, the Kolmogorov-Smirnov test has to be repeated several times to draw precise conclusions. Consequently, for each route, we repeated 50 times the following test: train and optimize each density estimator on a sample of 90% of one year of data, use each estimator to compute the cumulative probabilities given to the 10% of test samples, and compute the Kolmogorov-Smirnov statistic and p-value for the uniformity of these cumulative probabilities, for each estimator.

The results of this evaluation are presented in Table 5.1, showing the average p-value, for each combination of kernel and bandwidth optimization method. In this table and the following ones, the best performance is underlined. The best method appears to be the equation solving based method of Sheather & Jones [7], associated with an Epanechnikov kernel. However, its performance are very close to the other method by Sheather & Jones, and the unbiased cross-validation.

As a side note, these methods are quite fast, around 0.5 ms for the rules of thumb and around 5 ms for the others with 1000 records, which is the average amount of data per route for one year of data. However, the cross-validation methods and the Sheather & Jones methods have a computation time in $O(n^2)$, the others being almost linear.

Optimization method	Gaussian	Epanechnikov
$1.06\hat{\sigma}n^{-1/5}$ (Eq. (2.1))	0.4229	0.4219
$0.9\min\left(\hat{\sigma}, \frac{IQR}{1.34}\right)n^{-1/5}$ (Eq. (2.2))	0.4284	0.4281
Unbiased cross-validation	0.4300	0.4301
Biased cross-validation	0.4294	0.4292
Sheather & Jones (equation solving)	0.4315	0.43170
Sheather & Jones (direct plug-in)	0.4315	0.43169
Plug-in selector [19]	0.4315	0.4303
Smoothed cross-validation [20]	0.4316	0.4301

(All standard errors are close to 0.0005)

Table 5.1: Average p-value for Kolmogorov-Smirnov test of kernel density estimation route-based models

We can also notice that there is not much difference between Gaussian and Epanechnikov kernels performance using the same bandwidth. In addition, Epanechnikov kernel is faster to compute (around 5 times faster than a Gaussian kernel) thanks its finite support and the improvements described previously. As a consequence, we will only use the Epanechnikov kernel for the rest of the evaluations.

5.2.2 Optimal Combination of Categorical Parameters

We have seen in the previous chapter that we could define models composed on several predictors, for each value of one or several categorical parameters. This way, each predictor handles the possible different behavior for each value of a parameter.

We considered as relevant parameters, the origin, destination and airline, and also the month and day of week of the flight. We built models for different combinations of these parameters, in order to find the most relevant one. The models were composed of kernel conditional density estimators with a condition on the scheduled arrival hour. Indeed, as we have seen that the hour is the most important parameter, it is included in all models.

The predictors were trained on 4 years of data, with $h_1 = 4$ min and $h_2 = 1$ h. The test we conducted was to measure the AUC at $\tau = 60$ min, using 1000 test samples. This test was repeated 40 times. The results, in terms of average AUC, are presented on Figure 5.1.

We can see that taking into account month or day results in a lower average AUC, but it may come from a lack of data. However, as we are already using 4 years of data, in a real application, we may not want to wait so long before building a delay prediction model, and during this period, the delay behavior may also change.

We can also observe that the different combinations of origin, destination and airline have similar average AUC, or at least the standard errors of mean are overlapping. We will consider for the next steps, the combination of origin, destination and airline. Indeed, we will see the importance of the airline when building the decision tree, and for other performance measures. In addition, for origin and destination covered by several airlines, we usually observe important variations of the average delay with the airline.

5.2.3 Optimization of Training Data Quantity

Description

The dataset used was composed on records from 1995, and we can imagine that using more data to train the model would enhance them. However, the behavior of airlines (e.g., scheduled duration of flights) and of traffic management systems can evolve over years, so using too old data may not provide useful information. As an example, the average scheduled duration

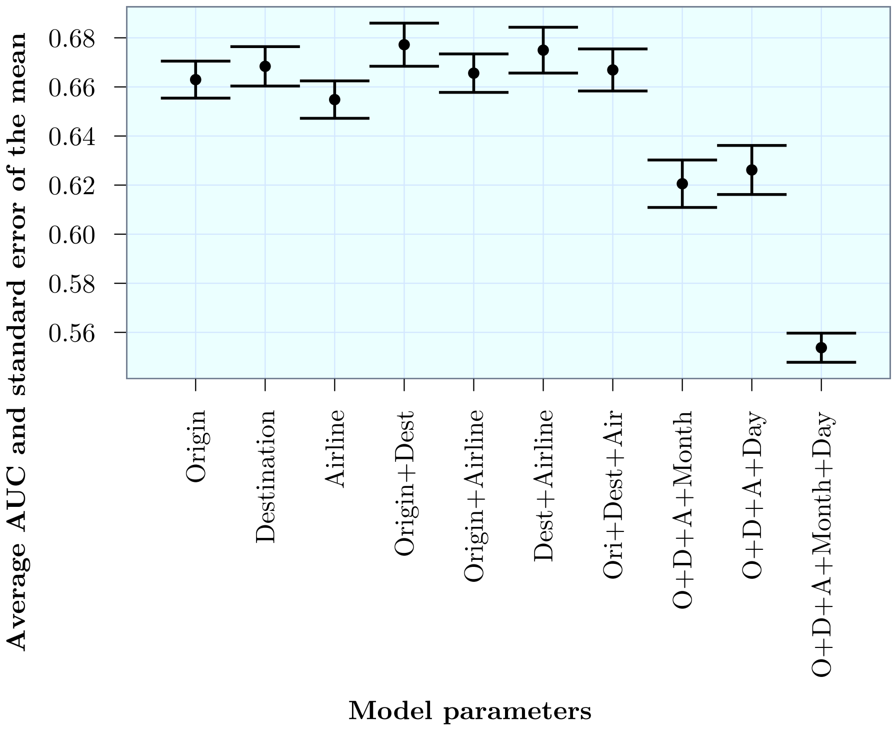

Figure 5.1: Average AUC for models with different combinations of categorical parameters

of flights regularly increased from 118 minutes in 1995 to 133 minutes in 2011. As a consequence, we tried to find the optimal number of years to be used for training the models.

The models used to conduct this test were: the empirical cumulative distribution function (ECDF), a global kernel conditional density estimation (with condition on the scheduled arrival hour), a kernel density estimator (KDE) per route, and a kernel conditional density estimator (KCDE) per route. The KCDE models were built using fixed bandwidth ($h_1 = 4$ min and $h_2 = 10$ min), usually providing good performance with the Kolmogorov-Smirnov test.

We used the years 1996 to 2011 as test years, and one to six preceding years as training data (this limitation was set to keep reasonable computation times). For each test year, and each set of training years, we trained the four models listed above, and evaluated them (using the Kolmogorov-Smirnov test described previously) on samples of the test year.

As our goal is to be able to predict the delay distribution of a specific flight, we reproduced this situation for the evaluation: for each one of the 10,000 evaluations, we randomly selected a route and a specific flight (i.e., a specific arrival hour, with a small margin to take into account small schedule adjustments). Then we evaluated the model on a sample of the records from this flight, and we measured the average goodness of fit (measured with the p-value, between 0 and 1), for the different models, and the different training and test years.

Results

On Figure 5.2 is represented the difference of average p-value when using 2 to 6 years of training data, with the average p-value of a model using only one year of data. We did not represent the absolute p-value, as it varies with the experiments, whereas its variations are easier to analyze.

We can see that using more data does not improve global models. As they are composed of only one predictor taking into account all training data, we can imagine that one year of data is already enough to build a consistent model, and that recent data provides a better basis for prediction.

On the contrary, the model composed of a kernel conditional density estimator per route is improved when using more data. Indeed, this model, composed of a predictor per route and having a condition on the arrival hour, spreads the records a lot, and as a consequence, needs more data to provide more consistent predictions.

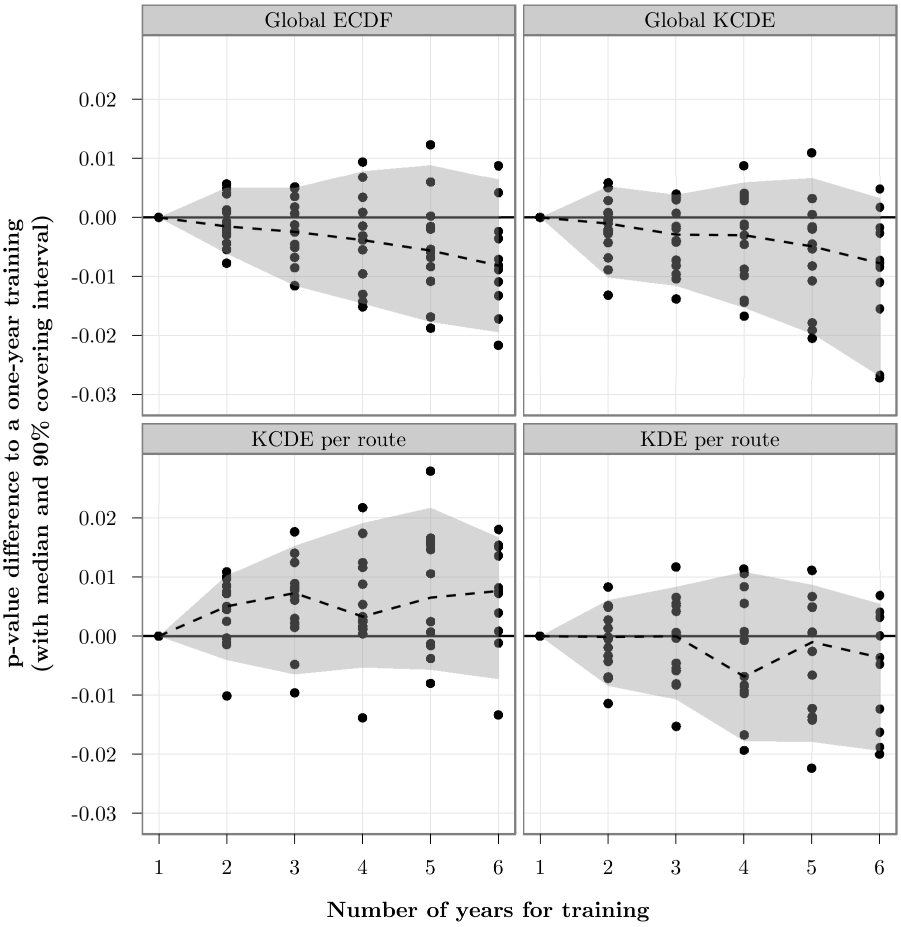

Figure 5.2: Goodness of fit improvement when using training data from several years

5.2.4 Optimization of Route-Based Kernel Conditional Density Estimation

The main model we would like to optimize is the model composed of a kernel conditional density estimator per route. We will use several methods to optimize its parameters, in order to see if the optimal parameters are identical for each evaluation method. We use the same bandwidths for all routes. Indeed, we first tried to optimize the bandwidths separately for each route, but the results highly depended on the training and testing samples used, and the gain was negligible in comparison with using common bandwidths.

As we use separate years for training and testing the model, the flight schedules may change. As a consequence, it may happen that some arrival hours do not have a corresponding density estimation, especially when the smoothing in the condition dimension is not important. In this case, we use, as a density estimation, a weighted average of the two closest (in terms of hour) density estimations available.

Likelihood Maximization Method

We will first of all try to find the optimal bandwidths by likelihood maximization. Using 4 years as training data and 1000 test samples from 2011, we measured 50 times the log-likelihood given to the test samples by several route-based models, using different bandwidths for delay and conditions kernel. The results are presented in Table 5.2. The highest log-likelihood is observed for $h_1 = 10$ min and $h_2 = 6$ h.

	Delay bandwidth h_1			
h_2	1 min	10 min	30 min	60 min
1 min	-8.11	-5.05	-4.73	-4.90
15 min	-5.78	-4.52	-4.55	-4.82
1 h	-5.18	-4.44	-4.52	-4.81
6 h	-4.87	-4.40	-4.51	-4.81
12 h	-4.83	-4.40	-4.52	-4.81

(All standard errors, except the first one, are less than 0.01)

Table 5.2: Average log-likelihood for kernel conditional density estimators per route, with several bandwidths

Confidence Interval Coverage Optimization

As a comparison, we conducted the same measures, with the same parameters and same test samples, but this time we measured the proportion of cumulative probabilities in the 80 and 90% confidence intervals. The results are presented in Table 5.3. We observe that several combinations of parameters cover quite well the 80 and 90% confidence intervals, and that large delay bandwidths produce too flat estimations.

	Delay bandwidth h_1			
h_2	1 min	10 min	30 min	60 min
1 min	67.3 ; 76.9	72.4 ; 81.8	84.6 ; 90.5	91.7 ; 94.4
15 min	76.0 ; 86.0	79.0 ; 88.5	88.2 ; 93.6	93.0 ; 95.6
1 h	78.5 ; 88.2	81.1 ; $\underline{90.1}$	89.1 ; 94.4	93.4 ; 95.9
6 h	79.6 ; 89.3	82.0 ; 91.1	89.6 ; 94.9	93.5 ; 96.0
12 h	$\underline{79.6}$; 89.5	82.0 ; 91.2	89.5 ; 94.9	93.4 ; 96.0

(All standard errors are less than 0.2%)

Table 5.3: Confidence intervals at 80 and 90%, for kernel conditional density estimators per route, with several bandwidths

Kolmogorov-Smirnov Test Optimization

Still using the same protocol and the same data, we applied a Kolmogorov-Smirnov test of uniformity to the cumulative probabilities given by the models. The average p-value of these tests are presented in Table 5.4. The standard errors of the mean are going up to 0.04, but this is normal considering the inherent variability of the p-value. We observe a p-value of 0.48 for $h_1 = 1$ min and $h_2 = 1$ h, which is very good, and similar to the average p-value we can observe when testing the uniformity of data randomly generated following a uniform distribution.

	Delay bandwidth h_1			
h_2	1 min	10 min	30 min	60 min
1 min	0.00	0.01	0.00	0.00
15 min	0.22	0.34	0.00	0.00
1 h	$\underline{0.48}$	0.30	0.00	0.00
6 h	0.42	0.21	0.00	0.00
12 h	0.42	0.18	0.00	0.00

Table 5.4: Average p-values for kernel conditional density estimators per route, with several bandwidths

The Kolmogorov-Smirnov test can also be used to measure the goodness of fit of the predicted distribution with the test samples, for a specific flight.

This way, we can be sure that the predicted distributions are good for each flight, and not only in average.

As the schedule may change during a year, we consider that a flight is composed of all records of one route in a 20 minutes interval. Considering 2000 flights and taking 50% of the records of each flight, we can measure the uniformity of the cumulative probabilities given by these records by the models. We observed that, as previously, delay bandwidths of 1 and then 10 minutes perform the best. Regarding the condition bandwidth, a smoothing of 6 or 12 hours provide very close results, with an average p-value of 0.18 in the best case.

Area Under ROC Curve Maximization

We can also try to optimize the parameters using the AUC, which evaluates the model as a classifier. We would like to optimize the two bandwidths h_1 (for the delay kernel) and h_2 (for the conditions kernel) on a large scale, and also the number of years of data used to train the models. This way, we would be able to see if the optimal bandwidths and size of the training dataset are correlated. The models were tested on 2011 data, and trained on all the records of 1, 3, 5 or 7 years, finishing on 2010.

The following test was repeated 100 times: for 5000 random samples from 2011, the estimator corresponding to each route was trained, and the cumulative probabilities given to a set of τ values were computed. These values gave the probability of being delayed, that was then compared to the effective delay on the test samples. Finally, we obtain the AUC for each model (having different parameters) and for each value τ (going from 0 to 180 min, with an increment of 10 min).

The τ value providing most frequently the best AUC was $\tau = 60$ min. The Figure 5.3 represents the average AUC for all combinations of h_1, h_2 and the size of training dataset. The standard error, always less than 0.003, is plotted only for few records, in order not to overload the graph.

We can first observe that h_1, the bandwidth of the delay kernel, does not have a major influence on the AUC (the four curves are close to each other). Surprisingly, a bandwidth of 60 min provides the best results, even if the resulting distribution is quite flat, considering the usual delay values.

We can also observe that using a large training set improves the AUC when h_2 is small. Indeed, if there is not much smoothing by the condition kernel, the data are very sparse and only few records are available to build the density estimator at each value of the condition. Using more data consequently helps to build better estimates. When the smoothing becomes more important, a larger training set does not provide large additional performance, and can even decrease the average AUC when using old data.

Finally, we observe that the bandwidth of the conditions kernel has an optimal value at 6 hours (with a very coarse grain but we can expect the

AUC to be continuous with the parameters). A larger smoothing would lose the interesting variation of the delay distribution with the hour, and a smaller smoothing suffers probably from overfitting and lack of data. An additional test has shown that the optimal bandwidth was actually close to 4 hours, but the difference was negligible.

As a conclusion for the optimization of the model based on a kernel conditional density estimation per route, we have seen that the optimal parameters depend on the performance measure used. However, the bandwidths $h_1 = 10$ min and $h_2 = 1$ h appear to be quite good compromise, and will be used for the comparison with other models.

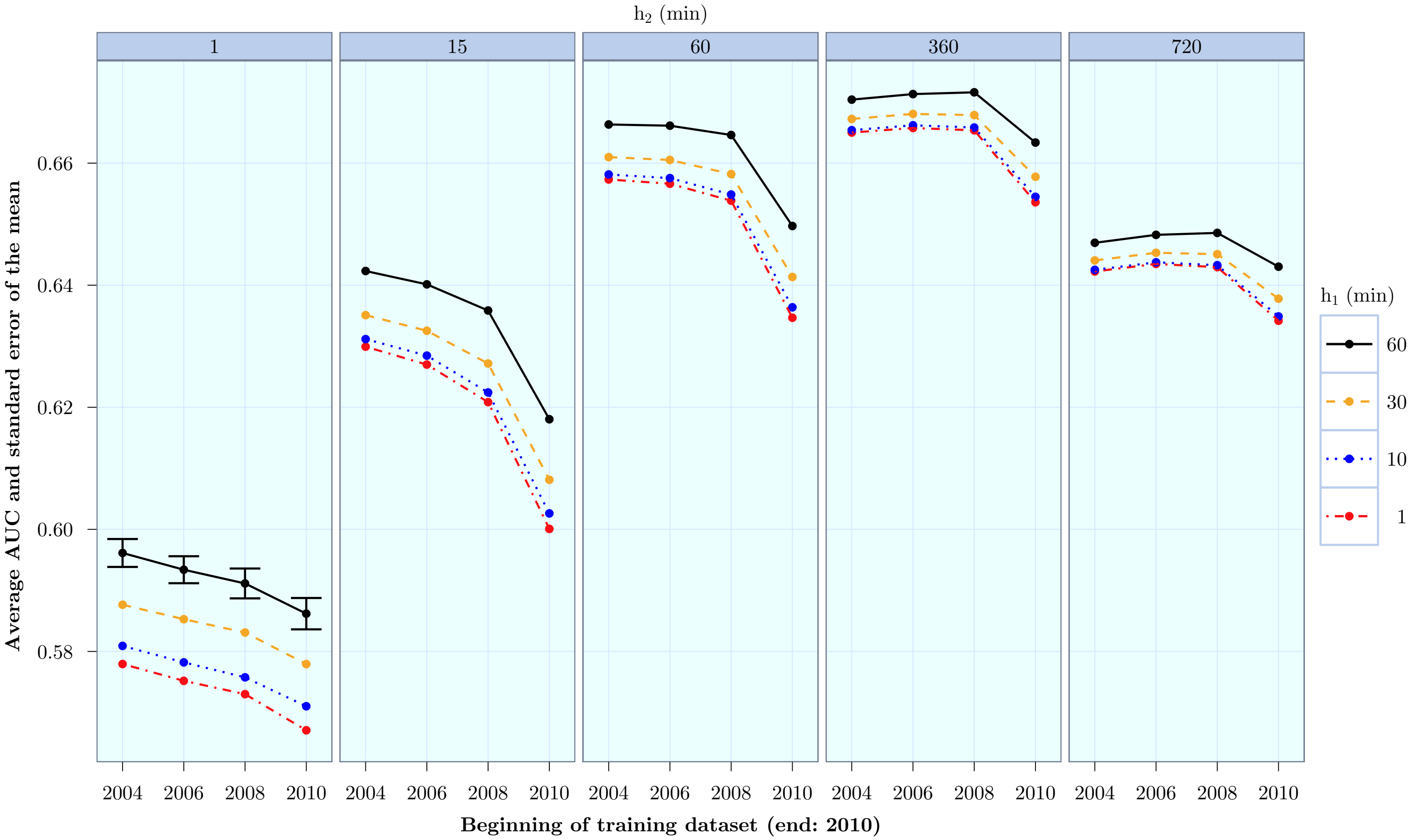

Figure 5.3: Average AUC for $\tau = 60$ min, for route-based kernel conditional density estimators using different parameters

5.2.5 Top-down Tree Construction

Our main model composed of a kernel conditional density estimator per route is now optimized. Another more complex model we presented and we would like to build and optimize is based on a decision tree, which aim is to group together routes having similar delay distributions.

Tree Construction

This tree was built using the Kolmogorov-Smirnov test as a performance measure. At each level was trained a kernel conditional density estimator with the whole dataset (or a large sample of it) corresponding to the leaf, with fixed bandwidths. Then we have to measure the goodness of fit of subsets corresponding to each value of each parameter, to the average model. Given the intrinsic variability of the p-value of the Kolmogorov-Smirnov test, this goodness of fit measure was repeated, for example 50 times on a sample of 10% of the subset records.

In case of a perfect fit, the p-value is a random variable having a uniform distribution between 0 and 1. As we cannot expect a perfect fit between a subset and the leaf average predictor, we allowed some of the tests to have a p-value lower than expected. The threshold for a subset to be considered as fitting the average predictor was that the 25th-percentile of the p-values must be above 0.1 (instead of the 0.25 expected for a perfect fit).

Finally, the splitting parameter is the one producing the most important separation between subsets fitting or not-fitting the average model.

The tree was built using the origin, destination, airline, month and day of week as parameters, which allowed finding the most important parameters for delay prediction. We can see on Figure 5.4, the splitting parameters for a part of the tree: the most important parameter is the destination, then the airline, then depending on the airline, either the origin and destination, etc. The values corresponding to each decision are not indicated, due to the high number of values (300 airports, 18 airlines).

We also trained and evaluated a tree taking into account the relative delay (i.e., the delay divided by the scheduled duration of the flight) instead of the absolute delay value. The order of the separation parameters was quite different, the most important one being the airline. However, a comparative evaluation has shown that a tree based on the absolute delay provided better predictions, especially in terms of confidence intervals coverage, and AUC.

Depths Comparison

We can expect that a deeper tree produces better estimates, as the training data are more specific. We compared models composed on a kernel conditional density estimator per leaf, with a condition on the scheduled arrival hour, for trees of depths 1 to 6. The predictors were trained on 4 years of

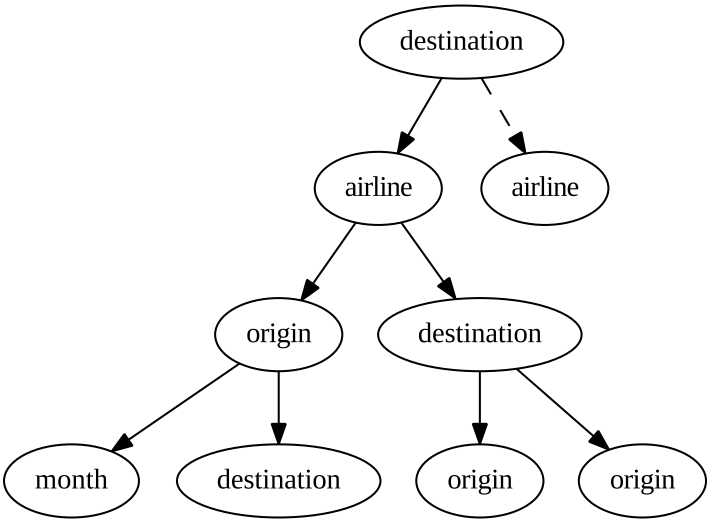

Figure 5.4: Top-down tree: first separation parameters

data, with $h_1 = 1$ min and $h_2 = 60$ min. The evaluation was repeated 100 times, using 5000 test samples. The models were compared using several measures: the log-likelihood, the confidence intervals and the area under ROC curve at $\tau = 60$ min.

The results are presented on Figure 5.5. We observe that a deeper tree enhances clearly the confidence intervals at 80 and 90%, being closer to the theoretical values. The AUC is also improved by a deeper tree. We can also see that a depth of 6 provides already a good coverage of the confidence intervals and a slower increase of the AUC. Building a deeper tree may improve a bit these results, but would be much more expensive to create.

However, the log-likelihood surprisingly decreases when the depth increases. As a deeper tree has less training data for each leaf, we can think that some rare value are consequently given a 0 probability, so the log-likelihood is penalized by the prior probabilities. Indeed, if we remove the test samples that were given a 0 probability, the log-likelihood increases with the depth.

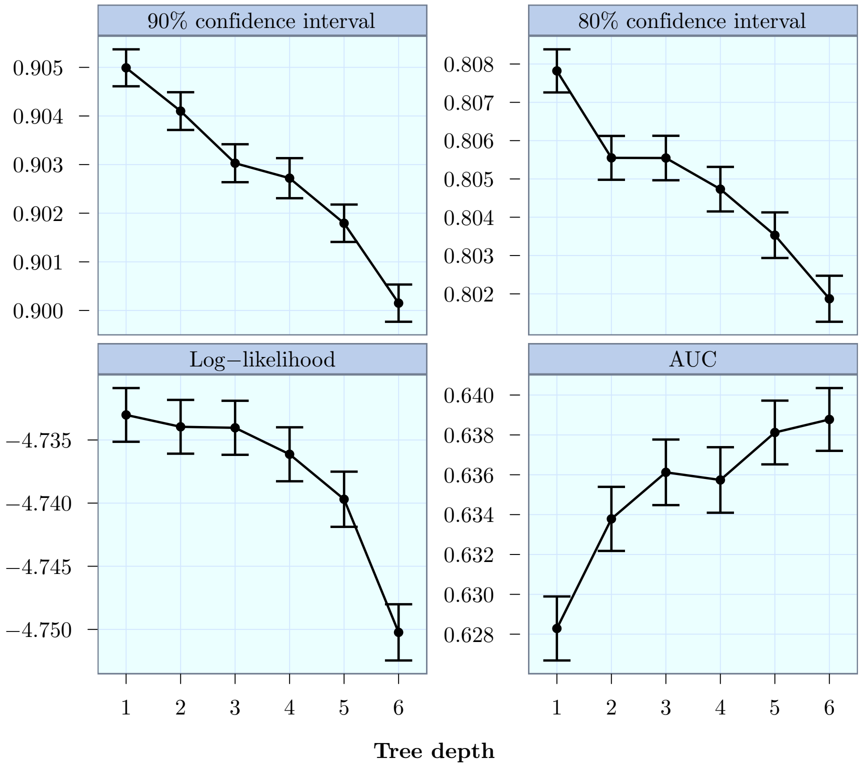

Figure 5.5: Performance evaluation of trees of different depths

5.3 Models Comparison

We have optimized the different models we were interested in. We will now compare them, using the different evaluation methods we have already seen, to find the best model for flight delay prediction.

5.3.1 Models Description

The optimized models we compare can be divided in three categories. The first group of models are composed of a single predictor, ignoring the categorical parameters:

1. Empirical Cumulative Distribution Function (ECDF), trained on 2010 data
2. Kernel Conditional Density Estimator, with a condition on the scheduled arrival hour, trained on 2010 data, $h_1 = 5$ min and $h_2 = 1$ h
3. Kernel Conditional Density Estimator, with conditions on the scheduled arrival hour, month and day of week, trained on 4 years of data, $h_1 = 5$ min and h_2 equivalent to 1 h on the hour dimension, before the scaling of all the conditions

The second group are models composed of a predictor per route (combination of origin, destination and airline):

4. Kernel Density Estimator per route, trained on 3 years of data, using the bandwidth optimization method of Sheather & Jones (equation solving).
5. Kernel Conditional Density Estimator per route, with a condition on the scheduled arrival hour, trained on 4 years of data, with $h_1 = 10$ min and $h_2 = 1$ h (as a compromise for all the evaluation methods)

Finally, the third group are models derived from the tree presented previously (taking into account the origin, destination, airline, day of week and month as splitting parameters), and are composed of a predictor for each of the 64 leaves:

6. Kernel Density Estimator per route, trained on 2010 records, using the optimization method of Equation (2.2) (the method of Sheather & Jones being too costly to compute)
7. Kernel Conditional Density Estimator per leaf, with a condition on the scheduled arrival hour, trained on 2010 records, $h_1 = 15$ min and $h_2 = 1$ h

5.3.2 General Evaluation

We compared these models using the measures we introduced previously: Area Under ROC Curve (AUC) at $\tau = 60$ min, 80 and 90% confidence intervals, and Kolmogorov-Smirnov (KS) test of uniformity of the cumulative probabilities. The likelihood were not be studied here, because we have seen it has some unexpected behavior and it is the only method that does not use cumulative probabilities, that are more relevant for our practical use case.

The test was conducted 100 times, using 5000 random test samples from 2011 data. The results are presented on Figure 5.6. Regarding the coverage of confidence intervals, we can see that the models based on the tree are the closest to the theoretical values.

Regarding the Kolmogorov-Smirnov p-value, the model composed of a kernel density estimation per route provides the best performance. The other models have quite low average p-values, compared to what was previously observed, but this is in part due to the large test set used (5000 samples), that make harder for the cumulative probabilities to fit precisely a uniform distribution.

Regarding the AUC, the model composed on a kernel conditional density estimator per route gives the best performance. Regarding the Empirical Cumulative Distribution Function, it has an AUC of 0.5, because as explained previously, all the records are classified the same way, so the TPR and FPR are either both equal to 0 or both equal to 1.

As a conclusion, the best model depends on the measure. However, the tree-based kernel conditional density estimation (7) and the route-based kernel density estimation (4) may be good compromises.

As an illustration of the classifier performance, we can consider the route-based kernel conditional density estimation model. With $\tau = 60$ min, we measured an AUC around 0.68. The best (in terms of Phi coefficient) classifier obtained was when defining $p_{\text{threshold}} = 0.098$, meaning that an observation was classified as "delayed" if $P(X \geq 60 \text{ min}) \geq 0.098$. In this case, we obtained a True Positive Rate of 38.6% and a False Positive Rate of 16.1%.

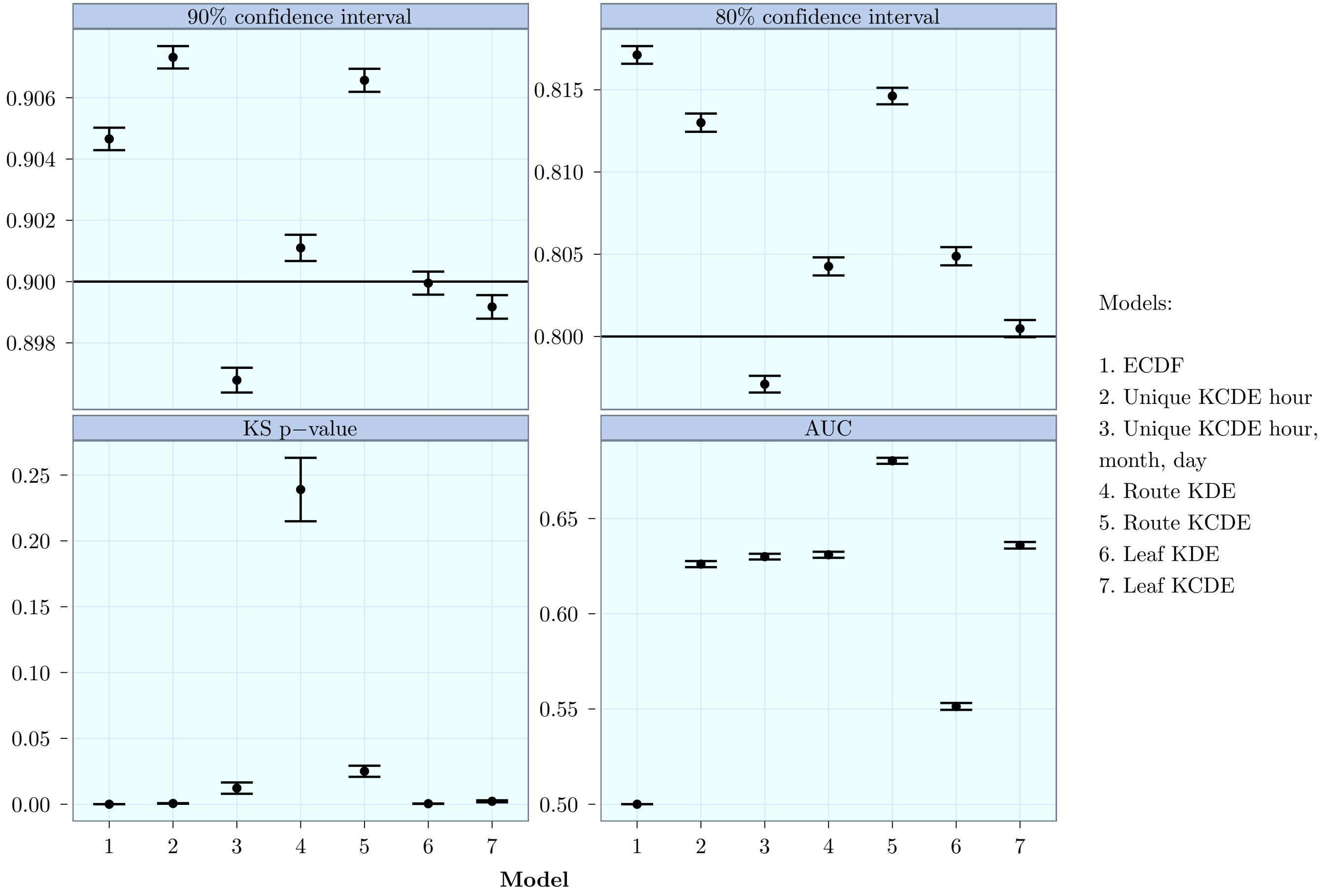

Figure 5.6: Performance evaluations of the optimized models

5.3.3 Evaluation on Flight Basis

In addition to the previous measures that concerned the cumulative probabilities given to a random set of test samples, we now focus on specific flights. We selected, for a combination of origin, destination and airline, a specific flight, meaning a small interval of arrival hour (in order to handle small schedule variations). Then we compared the distribution of the delay observations on these samples, to the predicted probability distribution for this specific flight.

We randomly selected 20,000 routes and a random flight on each route, then a sample of 50% of the observations for this flight (in order to allow some variability). Using a Kolmogorov-Smirnov test of goodness of fit, we usually consider that a p-value above 0.05 implies that we cannot reject the hypothesis that the observations fit the predicted distribution. The results are presented in Table 5.5.

Model	Percentage of p-values above 0.05
1. ECDF	41.4%
2. Unique KCDE hour	44.1%
3. Unique KCDE hour, month, day	45.7%
4. Route KDE	46.8%
5. Route KCDE	49.3%
6. Leaf KDE	41.8%
7. Leaf KCDE	44.2%

Table 5.5: Percentage of flights delay prediction having a Kolmogorov-Smirnov p-value above 0.05, for the optimized model

We can observe that the model based on a kernel conditional density estimator per route performs the best. However, obtaining a good p-value with a large test sample is very difficult, and it is also interesting to look at the difference between the prediction and the observation in a way which is easier to understand and interpret.

This can be done by measuring the root-mean-square deviation (RMSD) between the predicted distribution and the distribution of the test samples. We measured the RMSD of the 25th, 50th and 75th percentiles of the distribution, for 5000 flights, and this test was repeated 100 times. The results are presented on Figure 5.7. The value can appear to be quite high, but this may be to the sensibility of RMSD to outliers. As previously, the best model depends on the measure, the best compromises being probably the kernel density estimation route-based model, and the global kernel conditional density estimator.

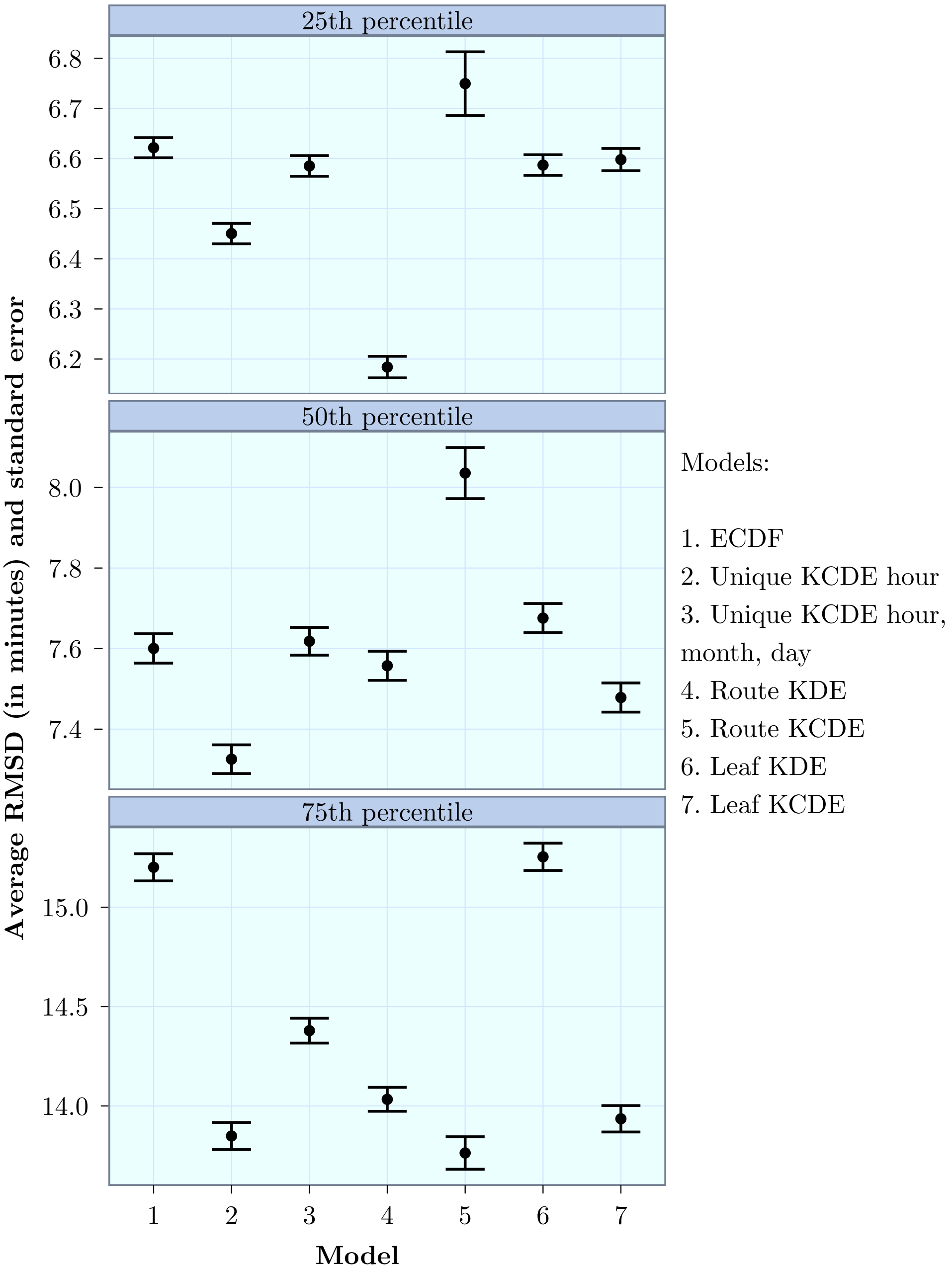

Figure 5.7: Root-mean-square deviation of 25th, 50th and 75th percentiles of flight delay prediction, for the optimized models

Chapter 6

Conclusion and Future Work

6.1 Conclusion

During this master's thesis, we have explored different prediction models and various evaluation method. By measuring the performance of the models using real data, we have seen interesting results on the predictability of the delays. The best delay prediction method appeared to be the most specific one, which takes into account all the combination of categorical parameters and a condition on the arrival hour.

The performances of the models were challenging to evaluate, due to the variety of measures used, and the different parameterizations adapted to them. However the predictions obtained appeared to be better than the one seen in the literature or used by FlightCaster (their long-term prediction being only based on the empirical cumulative distribution).

The kernel density estimation method was a very interesting method to learn and manipulate. Being a data-centered method, it can be used each time we want to reconstruct a probabilistic model from some observations.

6.2 Future Work

The models presented in this master's thesis could be improved in several ways. First of all, more complex density estimation methods could be used, using for example variable bandwidths, dynamically adapting the smoothing degree to the local density.

The models could also be refined, in order to give specific predictions for some period of time, like holidays and days off, during which a lot of people are travelling. We can also imagine building an expert system, which will use a combination of the different models we have seen, to select the best one for different use cases.

We could also model the phenomenon more precisely instead of looking only at the distribution of past data. We can for instance build separate models per time period, per type of aircraft, per airline, per region, and then grouping them into a general model. This way, we may be able to predict the delays of a new flight, without needing several months of data to build a prediction model.

Another step forward would be to generalize the model to flights of the entire world, or at least to exploit more data sources, to build more complete predictions.

Finally, the most interesting step would be to integrate such a model into a flight booking tool, to provide the delay prediction to future passengers, even if this would require a strong confidence in the information provided, considering the possible impact in terms of reservations.

Bibliography

Publications

[1] E.R. Mueller and G.B. Chatterji. “Analysis of aircraft arrival and departure delay characteristics”. In: *Proceedings of the AIAA Aircraft Technology, Integration, and Operations (ATIO) Conference, Los Angeles, CA*. 2002.

[2] SS Allan et al. “Analysis of delay causality at Newark International Airport”. In: *4th USA/Europe Air Traffic Management R&D Seminar*. 2001.

[3] Lu Zonglei, Wang Jiandong, and Zheng Guansheng. “A new method to alarm large scale of flights delay based on machine learning”. In: *Knowledge Acquisition and Modeling, 2008. KAM ’08. International Symposium on*. 2008, pp. 589–592.

[4] Y. Tu, M.O. Ball, and W.S. Jank. “Estimating flight departure delay distributions – a statistical approach with long-term trend and short-term pattern”. In: *Journal of the American Statistical Association* 103.481 (2008), pp. 112–125.

[5] B.W. Silverman. *Density estimation for statistics and data analysis*. Vol. 26. Chapman & Hall/CRC, 1986.

[6] V.A. Epanechnikov. “Nonparametric estimation of a multidimensional probability density”. In: *Teoriya Veroyatnostei i ee Primeneniya* 14.1 (1969), pp. 156–161.

[7] S.J. Sheather and M.C. Jones. “A reliable data-based bandwidth selection method for kernel density estimation”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1991), pp. 683–690.

[8] D.W. Scott. *Multivariate density estimation*. Vol. 139. Wiley Online Library, 1992.

[9] J.G. De Gooijer and D. Zerom. “On conditional density estimation”. In: *Statistica Neerlandica* 57.2 (2003), pp. 159–176.

[10] M.P. Holmes, A.G. Gray, and C.L. Isbell Jr. “Fast kernel conditional density estimation: a dual-tree Monte Carlo approach”. In: *Computational Statistics & Data Analysis* 54.7 (2010), pp. 1707–1718.

[11] J. Fan and T.H. Yim. “A crossvalidation method for estimating conditional densities”. In: *Biometrika* 91.4 (2004), pp. 819–834.

[12] M. Sugiyama et al. “Conditional density estimation via least-squares density ratio estimation”. In: *AISTATS 2010* (2010), pp. 781–788.

[13] J. Wang, W.W. Tsang, and G. Marsaglia. “Evaluating Kolmogorov’s distribution”. In: *Journal of Statistical Software* 8.i18 (2003).

[14] T. Fawcett. “ROC graphs: Notes and practical considerations for researchers”. In: *ReCALL* 31.HPL-2003-4 (2004), pp. 1–38.

[17] D.W. Scott and G.R. Terrell. “Biased and unbiased cross-validation in density estimation”. In: *Journal of the American Statistical Association* (1987), pp. 1131–1146.

[19] MP Wand and C. Jones. “Multivariate plug-in bandwidth selection”. In: *Computational Statistics* 9.2 (1994), pp. 97–116.

[20] MC Jones, J.S. Marron, and B.U. Park. “A simple root n bandwidth selector”. In: *The Annals of Statistics* 19.4 (1991), pp. 1919–1932.

[24] J.L. Bentley. *A survey of techniques for fixed radius near neighbor searching*. Tech. rep. SLAC-186 and STAN-CS-75-513. Stanford Linear Accelerator Center, 1975.

Implementation References

[15] David A. James and Saikat DebRoy. *RMySQL: R interface to the MySQL database*. R package version 0.8-0. 2011.

[16] R Special Interest Group on Databases. *DBI: R Database Interface*. R package version 0.2-5. 2009.

[18] Tarn Duong. *ks: Kernel smoothing*. R package version 1.8.5. 2011.

[21] R Development Core Team. *R: A language and environment for statistical computing*. ISBN 3-900051-07-0. R Foundation for Statistical Computing. Vienna, Austria, 2011.

[22] Hadley Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009. ISBN: 978-0-387-98140-6.

[23] Dirk Eddelbuettel and Romain François. “Rcpp: Seamless R and C++ integration”. In: *Journal of Statistical Software* 40.8 (2011), pp. 1–18.

[25] Tobias Sing et al. *ROCR: Visualizing the performance of scoring classifiers*. R package version 1.0-4. 2009.