Especialização em *Business Intelligence*
Unidade Curricular de Análise de Dados
Ano Letivo de 2016/17
2º Semestre

2015 Flight Delays and Cancellations

Beatriz Loureiro (A68876), Hugo Rodrigues (A73476), João Fontes (A71184), Pedro Lino (A66823)

Abril, 2017

Data de Recepção	
Responsável	
Avaliação	
Observações	

2015 Flight Delays and Cancellations

Beatriz Loureiro (A68876), Hugo Rodrigues (A73476), João Fontes (A71184), Pedro Lino (A66823)

Abril, 2017

Resumo

Um dos desafios de Aprendizagem Máquina passa pela construção de programas de computador que se melhorem automaticamente através de dados recolhidos e com a experiência absorvida de vários casos ao longo do tempo. Para isso, foi criado uma série de técnicas que nos permitem extrair conhecimento através de dados armazenados estruturadamente em bases de dados, *datasets*, etc. O objetivo deste trabalho foi estudar as diferentes metodologias de extração de conhecimento e, no final, estudar um *dataset* recorrendo à ferramenta R como auxílio nessa tarefa. No final tiramos as nossas próprias conclusões e dificuldades na aplicação destas tecnicas de extração de conhecimento.

Área de Aplicação: *Business Intelligence*, *Data* Science, Extração de Conhecimento de Dados, *Data Mining*

Palavras-Chave: Extração de Conhecimento, R, Classificação, Clustering, Regras de Associação

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

1.1. Contextualização

Um dos desafios da Aprendizagem Máquina passa pela construção de programas de computador que melhorem automaticamente com a experiência. Para tal, algoritmos que especificam, passo a passo, como resolver um problema, não são capazes de, por exemplo, encontrar padrões num grande conjunto de dados, pelo simples fato de que, de momento, é impossível codificar tantas e complexas características e todos os padrões possíveis para que o computador os encontre.

A partir desta noção de reconhecimento de padrões, podemos introduzir o conceito de *Machine Learning* [1], ou, traduzindo, Aprendizagem Máquina. Este conceito surge quando se pretende responder a questões mais complexas sobre um conjunto de dados. Tendo como exemplo a informação de vendas de uma loja, um SGBD consegue dizer quantas pessoas compraram um produto, mas não consegue identificar conjuntos de produtos que são frequentemente comprados em conjunto [2]. Para isso, é necessário usar um conjunto de técnicas que consigam classificar, prever, agrupar e extrair padrões destes dados, poupando assim os seres humanos dessa tarefa cansativa e de elevada dificuldade.

1.2. Apresentação do Caso de Estudo

Tendo sido propostos pelo docente 3 *datasets*, foi-nos atribuído o *dataset* "2015 Flight Delays and Cancellations", disponível em [3], que contém os dados dos voos comerciais realizados no ano de 2015 nos EUA. Este *dataset* está dividido em 3 ficheiros CSV:

- ***airlines.csv*** - contém os códigos IATA das companhias aéreas e os seus respetivos nomes;
- ***airports.csv*** - contém os códigos IATA dos aeroportos, bem como os seus respetivos nomes e localizaqção geográfica (cidade, estado, país e coordenadas geográficas)
- ***flights.csv*** - contém os dados dos voos comerciais realizados nos EUA no ano de 2015, desde a data de realização, a companhia aérea que o realizou, aeroportos de origem e destino, tempos espectáveis e reais de partida, de voo, chegada, tempos de atraso, tempos em que as rodas do avião saem do aeroporto de origem e em que chegam ao aeroporto de destino e os tempos de embarque e desembarque.

1.3. Motivação e Objetivos

Com o forte crescimento da tecnologia, hoje em dia, todos os serviços procuram modernizar as suas infraestruturas tecnológicas por forma a satisfazer melhor tanto os seus clientes como a poder ter o melhor encaixe financeiro possível com o menor esforço. Uma análise do volume de negócios e da BD de clientes permite a uma empresa atingir estes objetivos e obter previsões sobre o seu futuro operacional.

Com isto em mente, podemos verificar que este *dataset* é propício para análise, uma vez que apresenta uma grande quantidade de dados (aproximadamente 6 milhões de registos de voos), o que leva a que as previsões e conclusões tiradas a partir deles possam apresentar uma maior exatidão.

Tendo como objetivo principal descobrir conhecimento específico sobre os vôs nos EUA no ano de 2015, podemos definir alguns objetivos de análise para este *dataset* com vista a atingir esse fim como sendo:

- Descobrir qual a altura do ano mais propícia a haver menos atrasos nos voos, usando as capacidades de visualização de gráficos da ferramenta R;
- Descobrir que companhias conseguem atingir maior velocidade no tratamento de um voo (tempo de voo mais atrasos), usando também as técnicas de visualização de gráficos da ferramenta R;
- Criar um modelo que possa prever o atraso de um qualquer voo, usando técnicas de Regressão e Classificação de dados, testando diversos modelos e verificando a sua exatidão;
- Agrupar os aeroportos mediante os atrasos presentes nos voos que servem, usando técnicas de Clustering;
- Procurar padrões nas relações entre aeroportos e companhias aéreas, para procurar possíveis causas para os atrasos verificados, usando para isso a análise de Regras de Associação.

1.4. Estrutura do Relatório

A estrutura do relatório segue uma sequência lógica baseada na metodologia de DM largamente usada na indústria, denominada CRISP-DM, sendo que os capítulos do relatório retratam os diversos passos desta metodologia.

No primeiro capítulo será possível encontrar a primeira fase de análise do *dataset* e das suas particularidades, fazendo-se assim uma análise exploratória dos dados a usar.

O segundo capítulo retrata o processo de preparação dos dados para os processos de análise e as fases de desenvolvimento dos modelos para responder às questões apresentadas como objetivo de análise.

O terceiro capítulo apresenta a avaliação dos modelos obtidos, usando diversas métricas de erro que nos permitam avaliar a exatidão e assertividade dos modelos criados.

O quarto capítulo irá retratar como foi feita a implementação dos modelos de resposta às questões apresentadas como objetivo na ferramenta R.

Por fim, irá ser feita uma síntese de todo o trabalho realizado no âmbito deste trabalho, bem como algumas linhas de orientação para a realização do trabalho futuro.

2. Descrição e Compreensão da Natureza dos Dados

Neste capítulo vamos analisar o o conjunto de dados proposto, "2015 Flight Delays and Cancellations", disponibilizado através da plataforma Kaggle em [3]. Este *dataset* apresenta-nos um conjunto de dados relativo aos voos no ano de 2015 nos EUA, disponibilizados pelo *Department of Transportation*.

2.1. Descrição dos atributos do dataset

Os dados são disponibilizados em 3 ficheiros CSV, pelo que, de maneira a podermos analisar os seus atributos, começamos por carrega-los para a plataforma R.

```
# Read datasets
airlines <- read.csv ("flight-delays/airlines.csv")
airports <- read.csv ("flight-delays/airports.csv")
flights <- read.csv ("flight-delays/flights.csv")
```

Após carregar os ficheiros, podemos assumir que estes ficheiros resultaram de exportações de tabelas de uma BD Relacional, podemos uni-los usando a função *merge*, equivalente a um *join* numa BD Relacional. É necessário alterar o nome da coluna que representa o nome da companhia aérea no *dataset airlines*, uma vez que iria causar um conflito de nomes e esse nome não iria aparecer no *dataset flights* após a execução da função *merge*.

```
colnames(airlines) <- c("IATA_CODE", "AIRLINE_NAME")
flights <- merge(flights, airports, by.x = "ORIGIN_AIRPORT", by.y =
"IATA_CODE")
flights <- merge(flights, airports, by.x = "DESTINATION_AIRPORT", by.y =
"IATA_CODE")
flights <- merge(flights, airlines, by.x = "AIRLINE", by.y = "IATA_CODE")
```

Assim, temos um só *data frame* para analisar, simplificando as operações que teremos de realizar.

De seguida, listamos os atributos presentes no *dataset*, apresentando os valores que podem tomar e uma breve explicação dos mesmos [4].

- **AIRLINE** - código IATA da companhia aérea que efetuou o voo;
- **DESTINATION_AIRPORT; ORIGIN_AIRPORT** - códigos IATA dos aeroportos de destino e origem;
- **YEAR** - ano em que se realizou o voo (sempre 2015);
- **MONTH** - mês em que se realizou o voo (1 a 12);
- **DAY** - dia em que e realizou o voo (1 a 28/30/31);
- **DAY_OF_WEEK** - dia da semana em que se realizou o voo (1 - Domingo, …, 7 - Sábado);
- **FLIGHT_NUMBER** - identificador numérico que identifica cada voo;
- **TAIL_NUMBER** - identificador numérico que identifica a cauda do voo;
- **SCHEDULED_DEPARTURE; DEPARTURE_TIME; DEPARTURE_DELAY** - hora espectável e real de partida do voo e respetivo atraso (todas as horas são representadas por HHMM (representa a hora HH:MM), todos os atrasos são representados em minutos);
- **TAXI_OUT; TAXI_IN** - tempos em minutos entre o começo do embarque e a saída das rodas do avião do aeroporto de origem e a chegada das rodas do avião ao aeroporto de destino e o final do desembarque;
- **WHEELS_OFF; WHEELS_ON** - horas em que as rodas do avião saem do avião de origem e chegam ao aeroporto de destino;
- **SCHEDULED_TIME; ELAPSED_TIME; AIR_TIME** - tempos espectáveis, reais e de voo, em minutos, do avião;
- **DISTANCE** - distância percorrida, em kilómetros;
- **SCHEDULED_ARRIVAL; ARRIVAL_TIME; ARRIVAL_DELAY** - hora espectável e real de chegada do voo e respetivo atraso
- **DIVERTED; CANCELLED, CANCELLATION_REASON** - valores booleanos que indicam se o voo foi desviado, cancelado e, caso cancelado, a razão para o seu cancelamento;
- **AIR_SYSTEM_DELAY; SECURITY_DELAY; AIRLINE_DELAY; LATE_AIRCRAFT_DELAY WEATHER_DELAY** - tempos de atraso nos diversos esstágios do voo, desde tempos de atraso no *check-in*, na segurança, atrasos da companhia aérea, na chegada atrasada do avião ou por causa das condições climatéricas.

Os atributos descritos até este ponto são os atributos nativos do *dataset flights*. Os atributos seguintes foram adicionados a este *dataset* após a execução da função *merge*.

- **AIRPORT.{x,y}** - nomes dos aeroportos de origem e destino;

- **CITY.{x,y}; STATE.{x,y}; COUNTRY.{x,y}** - localização dos aeroportos de origem e destino (cidade, estado e país (sempre EUA));
- **LATITUDE.{x,y}; LONGITUDE.{x,y}** - coordenadas geográficas dos aeroportos de origem e destino;
- **AIRLINE_NAME** - nome da companhia aérea que efetuou o voo.

2.2. Análise Exploratória

Começamos por analisar como estão distribuídos os valores que representam tempos e atrasos, usando para isso a função *summary* do R.

```
time.att <- c("SCHEDULED_TIME","ELAPSED_TIME","AIR_TIME","ARRIVAL_DELAY",
              "AIR_SYSTEM_DELAY","SECURITY_DELAY","AIRLINE_DELAY",
              "LATE_AIRCRAFT_DELAY","WEATHER_DELAY")
summary(flights[, time.att])
```

Com isto, é-nos possível auferir a distribuição de valores que estes dados apresentam. O *output* da função é o seguinte:

```
SCHEDULED_TIME  ELAPSED_TIME     AIR_TIME         ARRIVAL_DELAY     AIR_SYSTEM_DELAY  SECURITY_DELAY
Min.   : 18.0   Min.   : 14.0    Min.   :  7.0    Min.   : -87.00   Min.   :   0      Min.   :  0
1st Qu.: 85.0   1st Qu.: 82.0    1st Qu.: 60.0    1st Qu.: -13.00   1st Qu.:   0      1st Qu.:  0
Median :123.0   Median :119.0    Median : 94.0    Median :  -5.00   Median :   2      Median :  0
Mean   :141.8   Mean   :137.2    Mean   :113.7    Mean   :   4.89   Mean   :  13      Mean   :  0
3rd Qu.:174.0   3rd Qu.:169.0    3rd Qu.:144.0    3rd Qu.:   8.00   3rd Qu.:  18      3rd Qu.:  0
Max.   :718.0   Max.   :766.0    Max.   :690.0    Max.   :1971.00   Max.   :1134      Max.   :573
NA's   :6       NA's   :101784   NA's   :101784   NA's   :101784    NA's   :4329554   NA's   :4329554

AIRLINE_DELAY     LATE_AIRCRAFT_DELAY WEATHER_DELAY
Min.   :   0      Min.   :   0        Min.   :   0
1st Qu.:   0      1st Qu.:   0        1st Qu.:   0
Median :   2      Median :   4        Median :   0
Mean   :  19      Mean   :  24        Mean   :   3
3rd Qu.:  19      3rd Qu.:  30        3rd Qu.:   0
Max.   :1971      Max.   :1331        Max.   :1211
NA's   :4329554   NA's   :4329554     NA's   :4329554
```

Podemos verificar que os dados apresentam muitos valores nulos, algo que irá dificultar a extração de conhecimento mais à frente e e que terão de ser tratados.

Podemos verificar ainda que os valores não nulos são muito díspares, pois apresentam grandes diferenças entre os valores máximos e os valores representativos do 3º quartil, o que vai gerar uma grande quantidade de *outliers*. Podemos explicar esta disparidade de valores pelo pouco ou nenhum

atraso dos voos, o que leva a que os atrasos tenham um valor próximo de 0, levando assim a uma concentração dos dados aí.

Para comprovarmos as afirmações feitas anteriormente, podemos usar um *boxplot* ou histogramas para auferir melhor qual é a distribuição dos dados. Estes gráficos podem ser encontrados no anexo I. Através da análise destes gráficos podemos comprovar a existência de uma grande dispersão dos dados e de um grande número de *outliers*, como foi referido anteriormente.

Após alguma alguma pesquisa nos *kernels* da plataforma *Kaggle*, reparamos que alguns dos códigos dos aeroportos descritos nor atributos ORIGIN_AIRPORT e DESTINATION_AIRPORT não estão coerentes com os presentes no *dataset airports*. Estes códigos terão de ser tratados também na fase de processamento de dados.

A título de curiosidade, usamos o pacote *maps* da plataforma R para podermos visualizar a localização dos aeroportos presentes no *dataset airports* no mapa dos EUA.

```
map("usa")
title("Airports")
points(airports$LONGITUDE, airports$LATITUDE, col="red", cex=0.75)
```

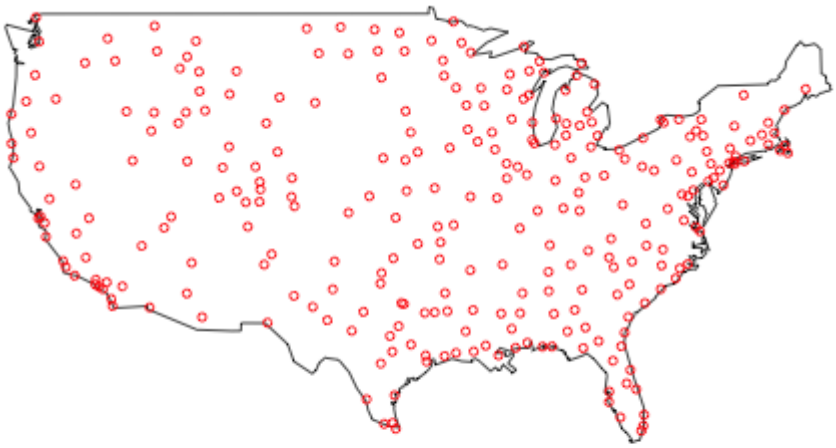

Figura 1 – Localização dos Aeroportos no Mapa dos EUA

3. Desenvolvimento dos Modelos

Neste capítulo vamos apresentar uma explicação de como foram pensados e desenvolvidos os modelos de classificação, regressão, segmentação e associação com vista a resolver as questões formuladas na motivação para a resolução deste problema.

Antes de iniciar com a resolução de qualquer questão anteriormente formulada, é necessário preparar os dados para iniciar a extração de conhecimento a partir destes. Sendo assim, é necessário repor os códigos IATA dos aeroportos em falta, uma vez que levaria a erros na extração de conhecimento destes. Esta reposição foi conseguida com recurso a um *kernel* consultado na plataforma *Kaggle*, criado por Scott A. Miller. Este *kernel* pode ser encontrado em anexo. A utilização desta função leva ao seguinte código R:

```
flights$ORIGIN_AIRPORT <- id.to.iata(flights$ORIGIN_AIRPORT)
flights$DESTINATION_AIRPORT <- id.to.iata(flights$DESTINATION_AIRPORT)
```

Outro tratamento que foi necessário executar nestes dados foi a remoção de valores nulos. No caso da ferramenta R, estes influenciam negativamente as operações executadas, pelo que foi necessário atribuir-lhes um valor por defeito de modo a podermos executar as operações desejadas. Sendo assim, decidimo-nos pela introdução do valor 0 no lugar dos valores nulos, uma vez que, assumindo que se não há um registo para o valor, então este é igual a 0. Isto verifica-se no caso dos valores de atraso, uma vez que não se regista um atraso de um voo quando ele não aconteceu. A remoção dos valores nulos foi feita usando o seguinte código:

```
flights[is.na(flights)] <- 0
```

Após se terem tratado os valores nulos, decidimos adicionar 2 atributos, um que indica o atraso total do voo e outro que nos indica se o voo atrasou ou não. Isso foi possível graças ao seguinte código:

```
flights[, "DELAY"] <- rowSums(flights[, delay.att])
flights[, "DELAYED"] <- ifelse(flights$DELAY > 0, 0, 1)
```

3.1. Melhor altura do ano para viajar

Aquando da formulação inicial desta questão, decidimos que usaríamos as questões de visualização de gráficos da ferramenta R para podermos responder a esta questão. Esta escolha prende-se com a razão de a ferramenta R apresentar grandes capacidades de criação de gráficos, pelo que apenas teríamos de obter os dados necessários e instruir a ferramenta R a construir o gráfico pretendido. Neste caso, o gráfico usado seria um gráfico de barras que conteria as médias dos atrasos em função de cada mês. Com isto, é possível averiguar quais os melhores meses do ano para se viajar.

3.2. Melhor companhia aérea onde se viajar

Para a resolução desta questão, inicialmente, foi definida também a utilização das ferramentas de produção de gráficos da ferramenta R para nos apresentar a informação necessária. Sendo assim, é apenas necessário selecionar a informação a usar, sendo esta os tempos médios de tratamento de um voo (atrasos mais tempo de voo) para cada companhia aérea. Uma vez que estes tempos não nos permitiam ter uma perceção de qual a melhor companhia aérea, pois estas podem apenas fazer voos de curta distância que, consequentemente, vão ter um tempo de voo baixo, decidimo-nos pela seleção também dos valores de atraso por cada companhia aérea. Com isto, podemos averiguar quais as companhias aéreas que têm mais tendência a atrasar os seus voos.

3.3. Prever atrasos de voos

Aquando da formulação desta questão, foi definida a utilização de técnicas de Regressão e Classificação para a sua resolução. Com isto, decidimos dividir esta questão em duas etapas:

- Inicialmente desenvolver um modelo que pudesse prever se um voo se ia atrasar ou não, ou seja, prever qual o valor do atributo DELAYED. Esta etapa iria envolver técnicas de classificação, uma vez que estamos a tentar prever uma variável discreta (apenas com os valores sim ou não). Com isto, era espectável o teste de vários modelos com vista a verificar qual deles se adequava melhor a este caso. Neste caso, decidimos testar os modelos de Regressão Linear, de Árvores de Decisão e *Naïve Bayes*;
- Por fim, desenvolver um modelo que pudesse prever qual iria ser o atraso de um voo. Neste caso, a técnica a usar seria a Regressão, uma vez que pretendemos prever o valor de uma variável contínua, da qual não era espectável a obtenção de um valor exato, mas sim um valor com erro mínimo. Assim, usamos a Regressão Linear para prever este valor e métricas de erro para avaliar a exatidão do nosso modelo.

3.4. Agrupar aeroportos de acordo com os atrasos

Nesta questão, foi equacionada a utilização de técnicas de *Clustering* para obter a sua resposta, uma vez que estas técnicas agrupam os dados mediante a distância entre eles. Com isto, decidimos testar 2 valores diferentes para o número de clusters:

- Inicialmente agrupar os valores em 3 *clusters* (BOM, MEDIO, MAU), usando para isso os algoritmos de *Clustering* Hierárquico e *K-Means*;
- Por fim, agrupar os valores em 5 *clusters* (MUITO BOM, BOM, MEDIO, MAU, MUITO MAU), usando também para isso os algoritmos de *Clustering* Hierárquico e *K-Means.*

Estes algoritmos permitem-nos obter então uma classificação para os aeroportos mediante os valores de atrasos médios que apresentam.

3.5. Procurar padrões entre aeroportos e companhias aéreas

Por fim, esta questão procura encontrar padrões entre aeroportos e companhias aérea em busca de causas para os atrasos originados. Sendo assim, a técnica usada foi a obtenção de Regras de Associação usando para isso o algoritmo *Apriori*, que nos permite obter assim regras com antecedente e consequente através de métricas como suporte e confiança. Após a obtenção destas regras, elas seriam filtradas de modo a obtermos as regras que originam como consequente um atraso grande (classes “LARGE” e “HUGE”), de modo a identificarmos assim as causas para esses atrasos.

4. Implementação dos Modelos

Neste capítulo vamos apresentar como foi feita a implementação com recurso à ferramenta R com vista à resolução das questões formuladas na motivação para este trabalho.

4.1. Melhor altura do ano para viajar

Esta questão tornou-se muito simples de se resolver graças às ferramentas ricas de visualização de dados da ferramenta R.

Os dados usados para responder a esta questão foram os dados dos atrasos médios por cada mês do ano civil. Sendo assim, foram usadas as seguintes instruções R para os obter:

Após a obtenção destes dados, apenas é necessário usar o comando *barplot* para se obter um gráfico de barras que represente estes dados.

```
# Get the mean of all delays in all months
months <- seq(1, 12, 1)
months.delays <- rep(0, 12)
for (month in months) {
  x <- flights[flights[, "MONTH"] == month, delay.att]
  months.delays[month] <- sum(x)/nrow(x)
}

# Plot the resulting data in a barplot
barplot(months.delays,
        las = 0,
        names.arg = months,
        col = "orange",
        xlab = "Months",
        ylab = "Delay Mean",
        main = "Delays by Month")
```

O gráfico obtido foi o seguinte:

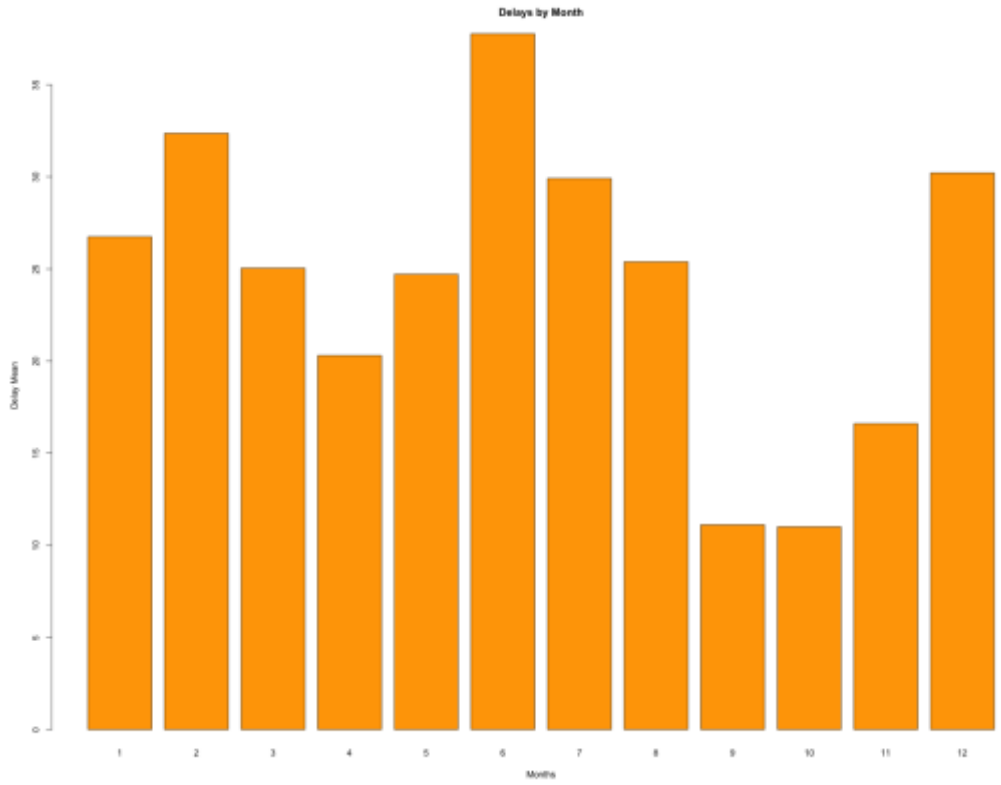

Figura 2 – Gráfico de Atrasos Médios por Mês

Como podemos observar, as piores alturas para se viajar de avião são Fevereiro, Junho e Dezembro. Isto pode ser explicado a partir de épocas festivas, como o Natal e o Carnaval, e épocas de férias, que levam as pessoas a viajar para outras partes do país para disfrutar dos seus tempos de férias. Com este gráfico, temos assim a resposta à nossa pergunta, uma vez que podemos facilmente observar que os meses de Setembro, Outubro e Novembro apresentam os valores médios de atraso mais baixo no ano, tornando assim a época do Outono como a melhor época para se viajar nos EUA.

4.2. Melhor companhia aérea onde se viajar

No caso desta pergunta a resposta também foi facilmente obtida graças às ferramentas de visualização da plataforma R, uma vez que nos permitiu obter uma rápida representação visual dos dados que necessitávamos.

Os dados necessários para obtermos esta resposta são os tempos médios de despacho de um voo e os atrasos médios por companhia aérea. Com isto, e usando para isso também o comando *barplot*, podemos visualizar quais as companhias aéreas que despacham os voos mais rapidamente e aquelas que se costumam atrasar mais e menos. O código necessário para obter esta resposta foi o seguinte:

```
# Get the sum of all delay times
airline.codes <- airlines$IATA_CODE
airline.times <- rep(0, length(airline.codes))
for (i in 1:length(airline.codes)) {
  x <- flights[flights[, "AIRLINE"] == airline.codes[i], delay.att]
  airline.times[i] <- sum(x)/nrow(x)
}

# Plot the resulting data in a barplot
barplot(airline.times,
        las = 0,
        names.arg = airline.codes,
        col = "orange",
        xlab = "Airline Codes",
        ylab = "Delay Mean",
        main = "Airline Mean Delay Times")

# Get the sum of all times
airline.times <- rep(0, length(airline.codes))
for (i in 1:length(airline.codes)) {
  x <- flights[flights[, "AIRLINE"] == airline.codes[i], time.att]
  airline.times[i] <- sum(x)/nrow(x)
}

# Plot the resulting data in a barplot
barplot(airline.times,
        las = 0,
        names.arg = airline.codes,
        col = "orange",
        xlab = "Airline Codes",
        ylab = "Mean Dispatch Time",
        main = "Airline Mean Flight Dispatch Times")
```

Com isto, os gráficos obtidos foram os seguintes:

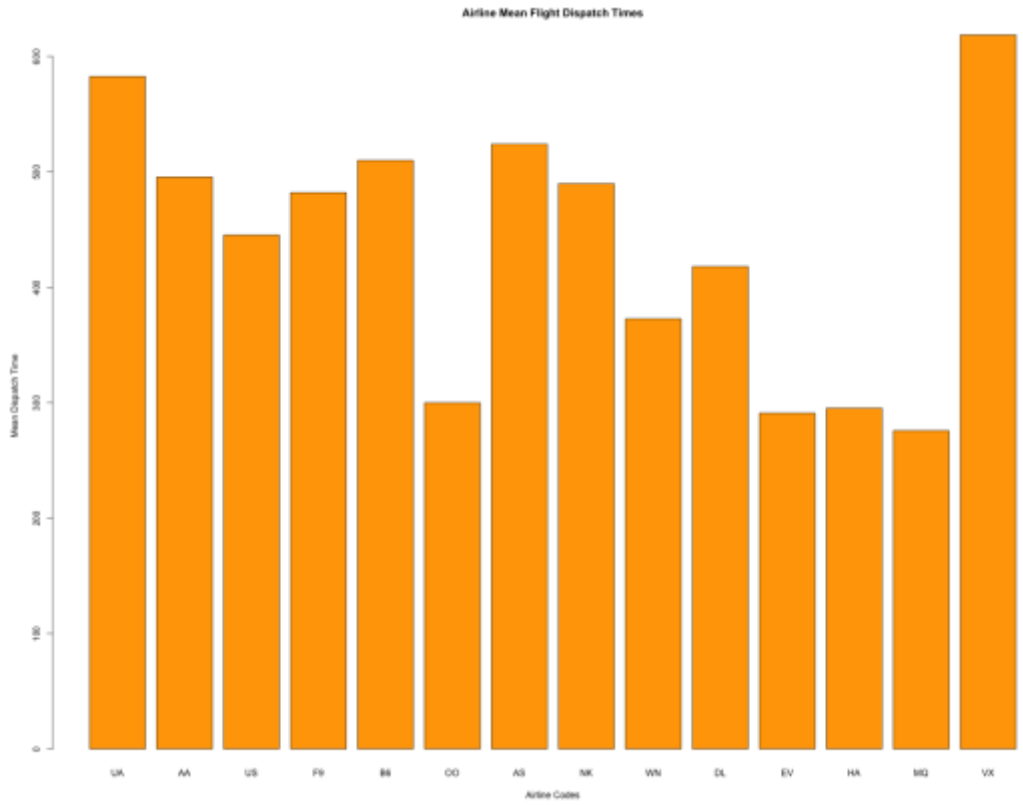

Figura 3 – Gráfico de tempo Médio de voo por Companhia Aérea

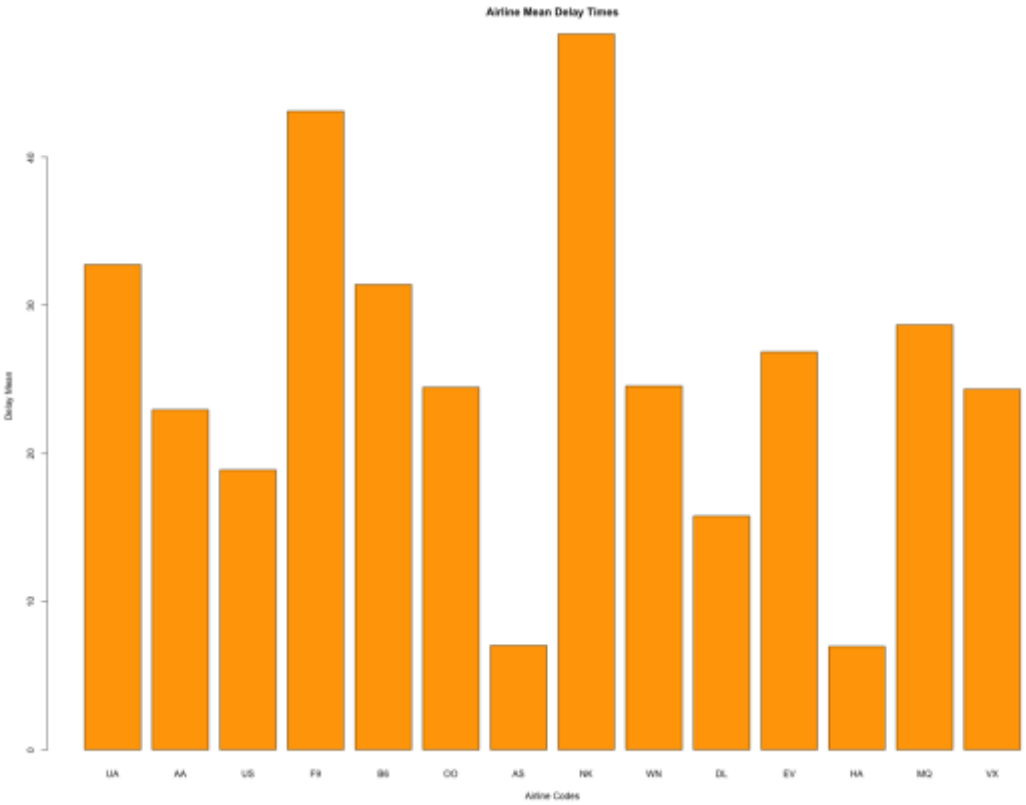

Figura 4 – Gráfico de Atrasos Médios por Companhia Aérea

Nestes gráficos podemos observar que apenas os tempos de voo não nos iriam ajudar a responder à questão, uma vez que os tempos de voo não refletem os tempos de atraso dos voos, representados no segundo gráfico. Como podemos ver, as companhias aéreas com menores atrasos são a *Alaska Airlines Inc.* e a *Hawaiian Airlines Inc.* Sendo assim, é de esperar que voos usando estas companhias aéreas apresentem atrasos muitos baixos em todos os seus estágios de voo. No outro extremo encontram-se as companhias *Spirit Air Lines* e a *Frontier Airlines Inc.*, que apresentam os maiores atrasos médios de todas as companhias aéreas.

4.3. Prever atrasos de voos

No caso desta pergunta, como já foi referido, este problema foi dividido em duas partes: uma parte de classificação, que prevê se um voo foi atrasado ou não, e uma parte de regressão, que determina qual o atraso previsto para um voo. É possível encontrar coisas em comum entre estas duas abordagens de Classificação e Regressão, uma vez que temos de selecionar para as duas quais os atributos a usar para a criação dos modelos e particionar o *dataset* inicial em dados de treino e dados de teste. Para isto, foi usada uma amostra de 20% (~1M de registos) do *dataset*, uma vez que os tempos de criação dos modelos eram impraticáveis.

```
# Get attributes to use for prediction
pred.att <- c("AIRLINE", "DESTINATION_AIRPORT", "ORIGIN_AIRPORT", "MONTH",
              "DAY", "DAY_OF_WEEK", "SCHEDULED_DEPARTURE",
              "SCHEDULED_TIME", "SCHEDULED_ARRIVAL")

# Take a sample of the dataset to speed up models construction
# (20%) ~ 1M records
set.seed(123456)
sample <- sample(1:nrow(flights),
                 size = ceiling(0.2 * (nrow(flights))),
                 replace = FALSE)
flights.sample <- flights[sample, c(pred.att, "DELAY", "DELAYED")]

# Generate train and test datasets
# 2/3 train, 1/3 test
set.seed(123456)
train <- createDataPartition(flights.sample$DELAYED,
                             p = 2/3,
                             list = FALSE)
```

```
flights.train <- flights.sample[train, ]
flights.test <- flights.sample[-train, ]
```

4.3.1 Prever se um voo vai ser atrasado

Tendo os dados de treino e de teste separados, podemos agora passar à criação dos modelos de classificação de voos, usando para isso três técnicas: regressão linear, árvores de decisão e naïve Bayes.

Regressão Linear

A regressão linear é um método muito usado para analisar dados, não só na área de regressão, mas também para classificar registos. Este tipo de modelos procura fazer uma aproximação estatística de uma reta aos dados de treino, criando assim uma função linear que tenta minimizar o erro. Uma vez que a regressão linear permite prever valores entre 0 e 1 por exemplo e, tratando este problema como probabilidades, podemos assumir que valores previstos acima de 0.5 representam a classe YES e abaixo disso representam a classe NO. Sendo assim, o código R necessário foi o seguinte:

```
# Build regression model
flights.regr <- lm(DELAYED ~ .,
                   data = flights.train[, c(pred.att, "DELAYED")])
```

Árvores de Decisão

As árvores de decisão são uns dos modelos mais usados para classificar dados, para além de serem extremamente simples de se compreender tendo um modelo à nossa frente. Estas estruturas procuram classificar registos a partir de um conjunto de decisões à medida que se “desce” na árvore, sendo que as folhas representam a classe atribuída ao registo. Neste caso, foi usado o método de criação de árvores “anova”, que faz uma análise da variância dos valores, permitindo assim obter valores mais precisos aquando da classificação. De modo a criarmos uma árvore de decisão para este caso, recorremos ao seguinte código R:

```
# Create the tree model
flights.tree = rpart(DELAYED ~ .,
                     data = flights.train[, c(pred.att, "DELAYED")],
                     method = "anova")
```

De modo a podermos visualizar a árvore criada, usamos o comando *rpart.plot* e obtivemos o seguinte modelo:

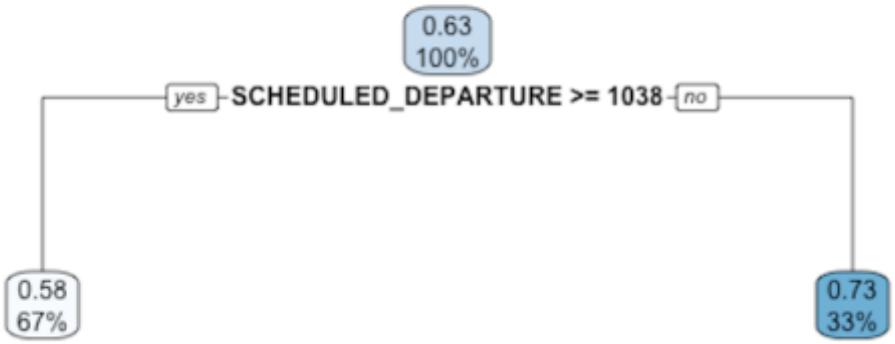

Figura 5 – Modelo da Árvore de Decisão

Naïve Bayes

O modelo *Naïve Bayes* baseia-se no teorema de probabilidades condicionadas criado por Thomas Bayes, e atribuiu a classe mediante o cálculo das classes com maior probabilidade de acontecerem. Como este modelo trabalha com probabilidades, então todos os valores que usam caracteres têm de ser categorias, pelo que têm de ser convertidos através da função *as.factor*. O modelo prevê uma classe, pelo que temos de usar também a função *as.*factor para que os valores da classe sejam reconhecidos como tal. Sendo assim, o código necessário para criar um modelo para este caso foi o seguinte:

```
# Convert all the character features to factor
flights.train[sapply(flights.train, is.character)] <-
  lapply(flights.train[sapply(flights.train, is.character)], as.factor)
flights.test[sapply(flights.test, is.character)] <-
  lapply(flights.test[sapply(flights.test, is.character)], as.factor)

# Create the probabilistic model
# DELAYED needs to be converted to factor because of the Naive Bayes
#  definition
# "Computes the conditional a-posterior probabilities of a **categorical**
#  class variable given independent predictor variables using the Bayes
#  rule."
flights.nb = naiveBayes(as.factor(DELAYED) ~ .,
                        data = flights.train[, c(pred.att, "DELAYED")])
```

4.3.2 Prever o atraso de um voo

Como já foi referido anteriormente, aqui procuramos prever uma variável contínua, logo teremos de usar técnicas de regressão para a executar. Neste caso, usamos a regressão linear para prever o valor do atraso de um voo, usando as métricas SSE[1], RMSE[2] e MAE[3] para avaliar o erro do modelo.

Sendo assim, foi criado mais um modelo de regressão idêntico ao criado para responder à pergunta anterior, mas desta vez com capacidade para prever a variável DELAY em vez da variável DELAYED. O código necessário foi o seguinte:

```
# Build regression model
flights.delayed.regr <- lm(DELAY ~ .,
                           data = flights.train[, c(pred.att, "DELAY")])
```

4.4. Agrupar aeroportos de acordo com os atrasos

Nesta questão procuramos agrupar os aeroportos mediante os valores dos atributos de atrasos, vendo assim quais os aeroportos que são mais parecidos entre si. Dividimos esta questão em 2 questões auxiliares, equivalentes entre si: agrupar os aeroportos através dos valores de atraso quando são o aeroporto de origem e quando são o aeroporto de destino. Como são equivalentes entre si, apresentamos aqui apenas o procedimento para agrupar aeroportos de origem.

Inicialmente foi necessário preparar o *dataset* de modo a que este pudesse reproduzir os valores de atraso e os aeroportos. Para isso, usamos o seguinte código R:

```
# Start by group origin airports
# Start by doing hierarchical clustering
# Prepare dataset
airports.origin.times <- data.frame(AIRPORT = integer(),
  ARRIVAL_DELAY = integer(), AIR_SYSTEM_DELAY = integer(),
  SECURITY_DELAY = integer(), AIRLINE_DELAY = integer(),
  DEPARTURE_DELAY = integer(), LATE_AIRCRAFT_DELAY = integer(),
  WEATHER_DELAY = integer())
```

[1] Sum of squared errors – Soma dos erros quadrados

[2] Root mean square error – Raiz quadrada da media do quadrado do erro

[3] Mean absolute error – Média do erro absoluto

```
i = 1
for (airport in airports$IATA_CODE) {
  x <- colMeans(flights[flights[, "ORIGIN_AIRPORT"] ==
                                                    airport, delay.att])
  airports.origin.times[i, ] <- c(airport, x)
  i <- i+1
}
```

Clustering hierárquico

Após a preparação dos dados a utilizar no processo de segmentação, é necessário construir as matrizes de distâncias, de modo a podermos utilizar o processo de *clustering* hierárquico. Esta construção foi feita com recurso ao seguinte código R:

```
# Build distance matrices
airports.origin.euclidean.dist <- dist(airports.origin.times[, delay.att],
                                       method = "euclidean")
airports.origin.euclidean.dist.full <-
  as.matrix(dist(airports.origin.times[, delay.att],
                 method = "euclidean", = TRUE, upper = TRUE))
```

A partir disto, podemos aplicar o processo de *clustering* hierárquico e obter um dendograma que nos mostra como se relacionam os diversos aeroportos.

```
# Build hierarchical cluster
airports.origin.hclust <- hclust(airports.origin.euclidean.dist,
                                 method = "complete")
```

Uma vez que queríamos dividir os aeroportos em 3 (BAD, AVERAGE, GOOD) ou 5 (HORRIBLE, BAD, AVERAGE, GOOD, EXCELLENT) categorias, usamos a função *rect.hclust*, que desenha retângulos a separar os *clusters* dentro de um dendograma. Para isso, usamos o seguinte código R:

```
# Plot the dendrogram
plot(airports.origin.hclust,
     airports.origin.times$AIRPORT,
     cex = 0.6)
```

```
# Place rectangles separating clusters
rect.hclust(airports.origin.hclust,
            k = 3,
            border = "red")

# Plot the dendrogram
plot(airports.origin.hclust,
     airports.origin.times$AIRPORT,
     cex = 0.6)

# Place rectangles separating clusters
rect.hclust(airports.origin.hclust,
            k = 5,
            border = "red")

# Plot the heatmap
heatmap(airports.origin.euclidean.dist.full,
        labRow = airports.origin.times$AIRPORT,
        labCol = airports.origin.times$AIRPORT,
        cexRow = 0.6, cexCol = 0.6)
```

Clustering K-Means

Após a utilização do algoritmo de *clustering* hierárquico, procedemos à utilização do algoritmo *K-Means*, para obter mais uma opção no agrupamento dos aeroportos. Sendo assim, utilizamos o agrupamento *K-Means* através da utilização do seguinte código R:

```
# Run kmeans function with:
# 3 centers - BAD, AVERAGE, GOOD
# 5 centers - HORRIBLE, BAD, AVERAGE, GOOD, EXCELLENT
airports.origin.kmeans.3 <- kmeans(airports.origin.times[, delay.att],
                                   centers = 3, nstart = 50)
airports.origin.kmeans.5 <- kmeans(airports.origin.times[, delay.att],
                                   centers = 5, nstart = 50)

# Plot clusters
clusplot(airports.origin.times[, delay.att],
         airports.origin.kmeans.3$cluster,
```

```
          color = TRUE, shade = TRUE,
          labels = 2, lines = 0)
clusplot(airports.origin.times[, delay.att],
          airports.origin.kmeans.5$cluster,
          color = TRUE, = TRUE,
          labels = 2, = 0)
```

Com isto obtivemos representações visuais dos *clusters* resultantes da aplicação do algoritmo *K-Means*, que podem ser encontradas em anexo. Este algoritmo foi usado com 3 e 5 centróides, possibilitando assim o agrupamento dos aeroportos em níveis de atraso diferentes, tal como já tinha sido usado na poda do dendograma resultante do *clustering* hierárquico.

4.5. Procurar padrões entre aeroportos e companhias aéreas

Por último, nesta questão procurou-se encontrar correspondências entre aeroportos (origem e destino), companhias aéreas e valores altos de atraso. Sendo assim, apenas foram selecionados os atributos que representam estes valores. Após isso, iniciamos a fase de preparação dos dados para análise fazendo uma discretização dos valores de atraso em 5 intervalos de igual frequência (None, Small, Medium, Large, Huge). Por fim, foi necessário transformar os dados em transações, uma vez que são a representação necessário para utilizar o algoritmo Apriori. O código usado foi o seguinte:

```
# Select airport, airline and delay attributes
flights.association.data <-
 flights[, c("ORIGIN_AIRPORT", "DESTINATION_AIRPORT", "AIRLINE", "DELAY")]

# Discretize DELAY attribute in 5 intervals of equal frequency
#  (None, Small, Medium, Large & Huge)
flights.association.data[, "DELAY"] <-
  discretize(flights.association.data[, "DELAY"],
             method = "frequency",
             categories = 5,
             labels = c("None", "Small", "Medium", "Large", "Huge"))

# Transform data frame to transactions
flights.association.data[, c(1,2)] <- (flights.association.data[, c(1,2)],
                                       as.factor)
flights.association.data <- as(flights.association.data,
                               "transactions")
```

Tendo os dados preparados, podemos agora correr o algoritmo Apriori e analisar as regras devolvidas. Para este caso utilizamos um suporte mínimo de valor baixo, uma vez que o tamanho do *dataset* não permite a utilização de suportes altos. Quanto à confiança mínima, usamos o valor 0.4 (40%).

```
# Run Apriori algorithm
# We need to use a small support because of the dataset size
flights.rules <- apriori(flights.association.data,
                         parameter = list(support = 0.000001,
                                          confidence = 0.4))
```

Após correr o algoritmo, obtemos um conjunto de regras que associam os valores de aeroportos, companhia aéreas e categorias de atrasos. Neste caso, procuramos regras que contenham no seu consequente uma categoria de atraso elevada. Sendo assim, selecionamos as regras usando o seguinte código R:

```
# Select rules that present great delay and analyze them
for (att in c("Large","Huge")) {
  flights.rules.sub =
    subset(flights.rules, subset =
      (rhs %in% paste("DELAY=", att, sep="")))
  inspect(sort(flights.rules.sub, by = "lift")[1:10])
  # Plot rules
  plot(sort(flights.rules.sub,
            by = "lift")[1:10],
       method = "graph", control = list(type = "items"))
  # Write delay rules to file
  write(sort(flights.rules.sub,
             by = "lift"),
        file = paste("questions/flights.rules.delays.",
                     att, ".csv", sep = ""),
        sep = "\t", quote = TRUE, row.names = FALSE)
}
```

Podemos observar que as regras geradas têm altos valores de confiança, pelo que podemos acreditar que o que elas nos ditam poderá acontecer no futuro, evitando assim as combinações de aeroportos e companhias aéreas ditadas nas regras. Para ajudar na visualização destas regras, foram criados diagramas visuais que as explicam, que podem ser encontrados em anexo.

5. Avaliação dos Modelos

Após a criação dos modelos de resposta às questões colocadas na motivação para este trabalho, é agora altura de os avaliar e procurar saber se realmente respondem às questões colocadas.

5.1. Melhor altura do ano para viajar

Neste caso, o modelo baseia-se em factos concretos, pelo que "contra factos não há argumentos". Sendo assim podemos assumir que o gráfico criado permite responder à questão colocada e permite-nos assim ver qual a melhor altura do ano para se poder viajar e evitar assim grandes atrasos.

5.2. Melhor companhia aérea onde se viajar

Esta questão suscitou dúvidas sobre se seria respondida pela abordagem inicial (tempos médios de despacho de um voo por companhia aérea) idealizada. Tal veio-se a verificar verdade com a aplicação da segunda abordagem (tempos médios de atraso por companhia aérea), uma vez que as companhias que demoram mais tempo a despachar um voo não são necessariamente as companhias com maiores atrasos, isto porque podem ser companhias de longo curso. Sendo assim, podemos assumir que a nossa segunda abordagem nos informa sobre quais as companhias aéreas a frequentar e quais devemos evitar, com vista a garantir o menor tempo de atraso possível.

5.3. Prever atrasos de voos

Esta questão mostrou-se como a mais intrigante e mais aliciante deste trabalho, uma vez que usamos técnicas de *Machine Learning* para prever valores que nos poderiam evitar de verificar as duas questões formuladas acima e ganhar assim algum tempo para outras tarefas. Sendo assim, passamos agora para uma avaliação dos modelos criados e pela escolha de qual dos modelos seria o mais fiável para entregar a um possível cliente.

Regressão Linear

Tendo treinado este modelo usando apenas um conjunto de dados de treino, recorremos ao conjunto de dados de teste reservado para esse efeito. No final, foi imprimida a matriz de confusão criada com recurso ao *package caret*. O resultado obtido foi o seguinte:

```
Confusion Matrix and Statistics

          Reference
Prediction      0      1
         0  20890  17542
         1 122371 227135

               Accuracy : 0.6393
                 95% CI : (0.6378, 0.6409)
    No Information Rate : 0.6307
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.0874
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9283
            Specificity : 0.1458
         Pos Pred Value : 0.6499
         Neg Pred Value : 0.5436
             Prevalence : 0.6307
         Detection Rate : 0.5855
   Detection Prevalence : 0.9009
      Balanced Accuracy : 0.5371

       'Positive' Class : 1
```

…

Árvores de Decisão

…

```
Confusion Matrix and Statistics

          Reference
Prediction      0      1
         0 108161 150578
         1  35100  94099

               Accuracy : 0.5214
                 95% CI : (0.5198, 0.5229)
    No Information Rate : 0.6307
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1196
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.3846
            Specificity : 0.7550
         Pos Pred Value : 0.7283
         Neg Pred Value : 0.4180
             Prevalence : 0.6307
         Detection Rate : 0.2426
   Detection Prevalence : 0.3330
      Balanced Accuracy : 0.5698

       'Positive' Class : 1
```

…

Naïve Bayes

…

```
Confusion Matrix and Statistics

          Reference
Prediction      0      1
         0  43999  44514
         1  99262 200163

               Accuracy : 0.6294
```

```
                95% CI : (0.6279, 0.6309)
    No Information Rate : 0.6307
    P-Value [Acc > NIR] : 0.9568

                  Kappa : 0.136
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8181
            Specificity : 0.3071
         Pos Pred Value : 0.6685
         Neg Pred Value : 0.4971
             Prevalence : 0.6307
         Detection Rate : 0.5160
   Detection Prevalence : 0.7718
      Balanced Accuracy : 0.5626

       'Positive' Class : 1
```

…

5.4. Agrupar aeroportos de acordo com os atrasos

5.5. Procurar padrões entre aeroportos e companhias aéreas

6. Conclusões e Trabalho Futuro

6.1. Avaliação do Processo de Trabalho

6.2. Avaliação do Sistema Desenvolvido

6.3. Evolução do Sistema

Bibliografia

[1] João Gama, André Ponce De Leon Carvalho, Katti Faceli, Ana Carolina Lorena, and Márcia Oliveira, 2015. *Extração de Conhecimento de Dados - Data Mining.* Lisboa: Edições Sílabo.

[2] Jiawei Han, Micheline Kamber, and Jian Pei, 2011. *Data mining: Concepts and techniques (the Morgan Kaufmann series in data management systems).* Amsterdam: Morgan Kaufmann Publishers In.

[3] Kaggle. (2017). *2015 Flight Delays and Cancellations | Kaggle*. [Online] Disponível em: https://www.kaggle.com/usdot/flight-delays [Acedido em 10 Abril 2017].

[4] Transtats.bts.gov. (2017). *RITA | BTS | Transtats*. [Online] Disponível em: https://www.transtats.bts.gov/Fields.asp?Table_ID=236 [Acedido em 10 Abril 2017].

Lista de Siglas e Acrónimos

BD	Base de Dados
CRISP-DM	*CRoss Industry Standard Process for Data Mining*
CSV	*Comma-Separated Values*
DM	*Data Mining*
EUA	Estados Unidos da América
IATA	*International Air Transport Association*
SGBD	Sistema de Gestão de Bases de Dados

Anexos

I. Análise da Dispersão dos Dados Temporais

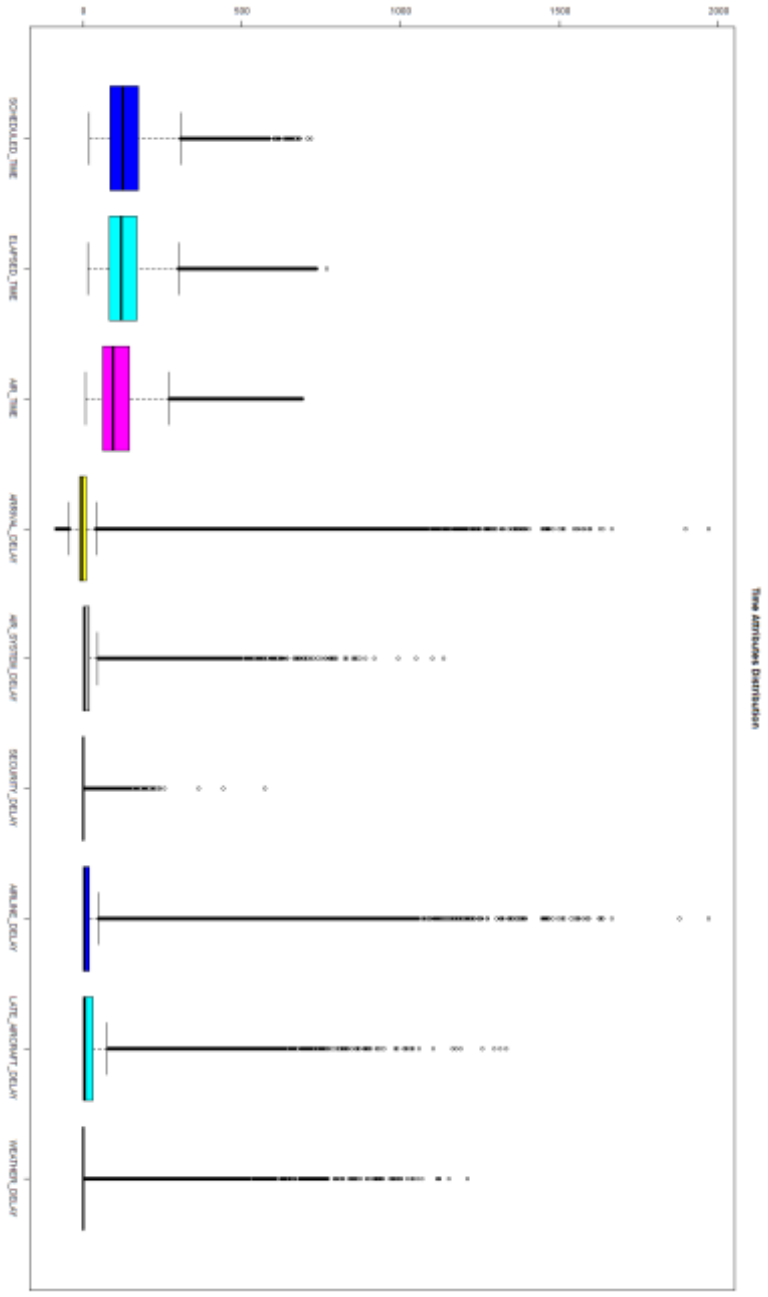

Figura 6 – *Boxplot* de Dispersão dos Dados Temporais

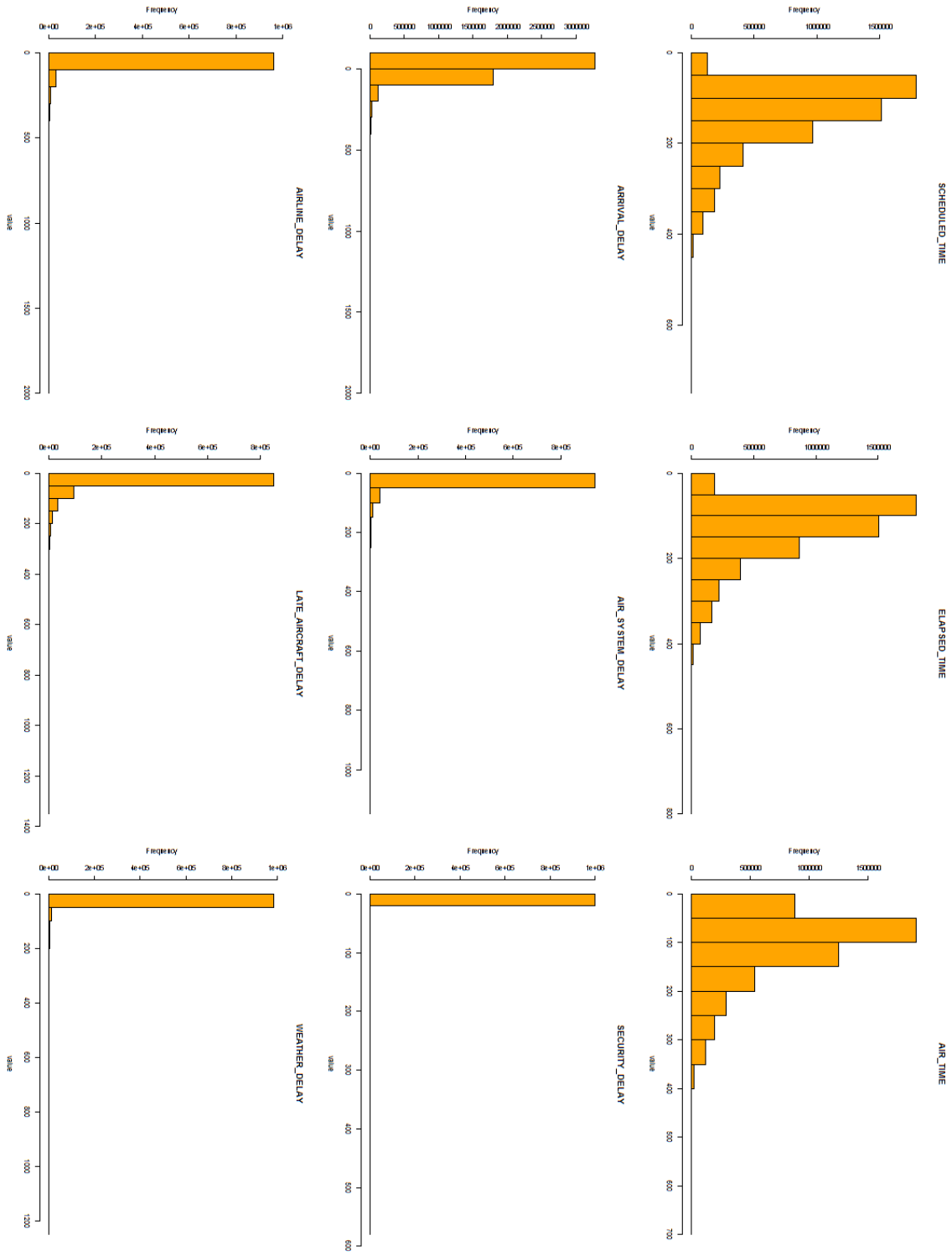

Figura 7 – Histogramas dos Atributos Temporais

II. Kernel de Atualização dos Códigos dos Aeroportos

```
id.to.iata <- function (x) {
  # Step 0: Initial setup
  # Load the incoming vector into a data frame, so we can use dplyr
  #  functions to join things up.
  df <- data.frame(ID = as.character (x),
                   stringsAsFactors = FALSE)
  # Store the number of records - used to make sure joins do not introduce
duplicates
  num_records = nrow(df)
  # Step 1: Add the Description to the base data.
  dfAirportID <- read.csv
 ("https://www.transtats.bts.gov/Download_Lookup.asp?Lookup=L_AIRPORT_ID",
  colClasses = c("character", "character"),
  col.names = c("AirportID", "Description"))
  df <- dplyr::left_join(df, dfAirportID, by=c("ID" = "AirportID"))
  # Step 2: Use Description to add the IATA_CODE to the base data.
  dfAirport <- read.csv
  ("https://www.transtats.bts.gov/Download_Lookup.asp?Lookup=L_AIRPORT",
   colClasses = c("character", "character"),
   col.names = c("IATA_CODE", "Description"))
  # There are duplicated airports. To solve this problem, clear out codes
  #  discontinued before 2015.
  # BSM was discontinued in 1999.
  # The IATA does not use NYL for Yuma, it uses YUM. So remove NYL.
  dfAirport <- dfAirport[! (dfAirport$IATA_CODE %in% c('BSM', 'NYL')),]
  df <- dplyr::left_join(df, dfAirport, by="Description")
  # Step 3: Make sure we have the same number of rows that we started with
  # If this error is triggered, steps will need to be made to eliminate
  #  duplicate key values.
```

```
  if (num_records != nrow (df)) {
    stop ("Due to duplicates in the data, the number of records has
changed.")
  }
  # Step 4: In cases where we didn't get a matching IATA_CODE, copy over
  #  the original value
  df$ID <- dplyr::coalesce (df$IATA_CODE, df$ID)
  # Step 5: We are all done. Return the results.
  return (df$ID)
}
```