Universitat Autònoma de Barcelona

escola d'enginyeria

GRADO EN GESTIÓN AERONÁUTICA

PREDICCIÓN DE TIEMPOS DE TURNAROUND

Estudio del Aeropuerto de Barcelona – El Prat

Memoria del Trabajo de Fin de Grado

Realizado por

Aitor Romero Checa

y dirigido por

Liana Napalkova

Escuela de Ingeniería

Sabadell, Febrero de 2016

Página dejada en blanco intencionadamente.

Universitat Autònoma de Barcelona

escola d'enginyeria

El sotasignat, *Liana Napalkova*

Professor/a de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en/na *Aitor Romero Checa*

I per tal que consti firma la present.

Signat: .....

Sabadell, 10 de Febrer de 2016

INDICE

TABLA DE ILUSTRACIONES

INTRODUCCIÓN

En la actualidad existe una creciente saturación en los aeropuertos principalmente en las horas de mayor frecuencia así como en fechas específicas, donde se hace indispensable la necesidad del control de los tiempos del turnaround, de ello dependerá que las entradas y salidas de las aeronaves sucedan con fluidez, evitando así retrasos y otros inconvenientes derivados de estos retrasos. Para ello es importante observar y determinar qué factores toman un papel importante en el tiempo que conlleva el cambio de vuelo, para ello, no se puede dejar a la intuición, ni a la vez a estudios poco rigurosos, y se deberá establecer procesos estadísticos que determinen con el menor error posible cual son esas variables y cómo se puede reducir el impacto en el tiempo del turnaround.

Una vez que son determinados las principales variables, será posible generar aproximaciones y predicciones de aquello que pueda suceder en un escenario determinado, consiguiendo así saber dónde prestar atención y focalizar los esfuerzos en mejorar los valores de las variables que se deseen para minimizar el tiempo total y conseguir plena eficacia y eficiencia en los tiempos de turnaround con indiferentemente cual sea el escenario a afrontar.

El mercado aeronáutico se está encaminando principalmente por las soluciones Low Cost, estas compañías operan con un modelo de tiempos de escala, mucho más estrechos y sin margen que una compañía de bandera por ejemplo. Esto es debido a que cada minuto que el avión no está en el cielo, es dinero que pierde y/o deja de ganar la compañía, es por ello que es uno de los principales objetivos que persiguen las aerolíneas que venden los billetes a precios tan económicos, esos precios no serían posible si se ocasionan retrasos que no permitan que el avión tome vuelo con la mayor celeridad.

A la vez que la eficiencia se incrementa para dar un buen tiempo de turnaround, junto con un buen precio de billete, la demanda de estos vuelos, se incrementa, teniéndose que incrementar cada vez más la habilidad con la que se evita el retraso.

A menudo la eficiencia lograda no dependerá de la destreza de la parte interesada sino que también de factores externos.

Los retrasos que sufren los vuelos, en la práctica están generados por tres causas principales: la capacidad aeroportuaria y su utilización, los esfuerzos que realizan las aerolíneas en puntualidad, y las interacciones entre vuelos. Así que una parte residirá en los esfuerzos que haga otro agente, pero hay que recordar que la otra parte a la vez tendrá el interés de reducir el tiempo en el turnaround.

La predicción de los tiempos de turnaround es de gran importancia para las compañías aéreas, para que se lleve a cabo con éxito se necesitaran cada vez inputs de mayor calidad para que los resultados puedan ser usados con total certeza en la toma de decisiones de cualquier aerolínea, pudiéndose usar como herramienta, las herramientas que hoy en día son usadas en la Ciencia de Datos, y que evolucionan forma vertiginosa, obteniendo con ellas factores que no son visibles, así como correlaciones entre variable, entre otros

Objetivo y tareas del proyecto

El objetivo principal del estudio es la identificación de los factores que impactan con mayor fuerza en los tiempos de turnaround, así como una gestión automatizada de la obtención de datos relacionados con estos tiempos, a través de aplicativos que distribuyen informaciones relativas a los tiempos de los vuelos.

Por otro lado será importante determinar los métodos estadísticos que se llevaran a cabo para el correcto procesamiento de la información obtenida. Por último se deberá establecer que modelos predictivos ofrecen los resultados más precisos y con menor cantidad de error.

Para llevar a cabo el estudio, una visión general de las tareas a realizar serán las siguientes:

- Identificar el problema principal
- Seleccionar los entornos y los lenguajes con los que se trabajara para la generación de los códigos.
- Elaborar el procedimiento de obtención de datos, con su posterior limpieza y depuración de datos
- Identificar qué atributos, variables, o características son las de mayor relevancia, cuales afectan más a los tiempos del turnaround, y a la vez, cuáles pueden ser prescindidas del estudio ya que no tienen suficiente impacto en estos tiempos.
- Aplicar métodos estadísticos que permitirán la buena ejecución de un modelo predictivos
- Generar los resultados del modelo predictivo.

Valor Práctico

El valor del proyecto reside en la implementación de unas herramientas que permitan de una forma automatizada, el análisis en la operativa aeroportuaria del momento, dando a conocer en profundidad de qué forma afectan los factores y las variables al correcto funcionamiento de las partes implicadas en el proceso del turaround, conocer que consecuencias tienen las modificaciones en ciertos parámetros, así como crear predicciones con la máxima precisión posible, para conocer los recursos que serán necesarios así como el momento en que serán necesarios.

Esto, ante la creciente congestión que sufren los aeropuertos, está enfocado para la reducción de costes, a la vez que aprovechar de la mayor forma posible a la ventaja inherente que tiene la aviación, que es la velocidad.

Metodología

Como visión general de la metodología empleada en el proyecto, se mencionaran los programas y herramientas usadas en él, así como los modelos implementados en este, entre otros aspectos.

Primeramente, la información necesaria se ha recogido del portal FlightStats, de donde los usuarios finales pueden recabar datos de forma manual e individual de una forma fácil y gratuita, la parte de desarrollador ha permitido, la automatización de este proceso, mediante la programación de código.

Los datos recogidos son datos reales, del Aeropuerto del Prat de Llobregat (Barcelona), estos datos se recogieron para todos los vuelos comerciales registrados en la base de datos del

portal, que entraron y salieron del aeropuerto en los meses de Octubre y Noviembre de 2015, recogiendo un total aproximado de dos mil registros.

Para el almacenamiento de los datos se usa el formato de archivo CSV (Comma Separated Values), donde cualquier programa con hoja de Cálculo, es capaz de abrirlo y trabajar con la información que contiene.

Para extraer los datos de FlightStats e introducirlos en la hoja de Cálculo, se realiza mediante código escrito en lenguaje Python, y usando el entorno de desarrollo integrado JetBrains PYCharm, programa en código abierto. Donde se ha ido programando, en base al formato JSON proporcionado por el portal, el código para la extracción automática del código a la vez que el almacenamiento en la hoja de Cálculo.

A través de otro código en Python se ha procedido a la limpieza de los datos extraídos para asegurar su integridad y coherencia.

Otros códigos en Python junto con librerías adicionales, han permitido la realización de los gráficos, para facilitar el análisis visual.

Las librerías estadísticas permiten solventar problemas que derivarían en errores o predicciones que carezcan de precisión.

Para concluir la parte de programación se usaran varios modelos predictivos, entre ellos Random Forests, o Gradient Boosting Machine, para determinar, cual ofrece una predicción más precisa y con el menor índice de errores.

Una vez realizado esto, se extraerán las conclusiones sobre los datos ofrecido por los modelos de predicción.

Estructura del Proyecto

- **Introducción:** Descripción de la necesidad de eficiencia en los tiempos de turnaround
- **La definición del problema:** Análisis de la situación actual e identificación de carencias
- **Predicción de tiempos de turnaround usando métodos de "Ciencia de Datos":** Conocimiento de los diferentes modelos y estudios utilizados en la actualidad.
- **Estudio del Caso:** Elaboración del estudio
- **Conclusiones:** Que nuevos conocimientos se han generado con el proyecto.
- **Bibliografía:** Referencia a los materiales utilizados.
- **Anexo:** En el anexo se encuentran los diferentes códigos creados.

…

1. DEFINICIÓN DEL PROBLEMA

A las interminables colas para pasar cada uno de los controles del aeropuerto, así como el tiempo de realizar el Check-In, junto con el tiempo prudencial en que se debe estar presente en el aeropuerto, hay que unir los tiempos antes de despegar y después de despegar.

Cada día es más necesario realizar los procesos de embarque y desembarque buscando la mayor optimización posible, debido a que los intentos de rebajar cada vez más los precios en los billetes obliga a prescindir de todos aquellos costes que no sean necesarios o que se puedan suprimir, y uno de estos costes es el tiempo en el turnaround, ya que tiene un coste considerable para la aerolínea cada minuto de más.

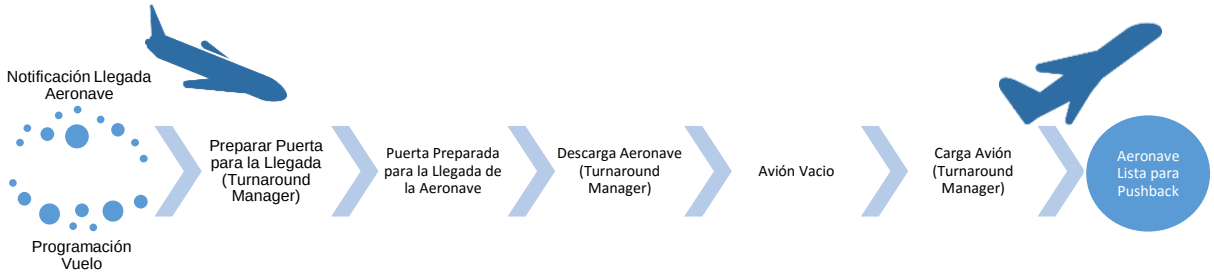


Figura 1 - Esquema Básico Turnaround Aeronave

El retraso en el proceso de turnaround está influenciado por la cantidad de tiempo reservada para este, la puntualidad en la llegada, entre otros, así como la eficiencia operacional de los servicios en tierra. Según la conclusión de la Performance Review Unit (PRU), indica que principalmente las llegadas con retrasos son originadas principalmente por salidas con retrasos. Esto conduce a la propagación de retrasos en toda la rotación de aeronaves y a la vez en toda la red de una compañía aérea, un retraso, provoca un nuevo retraso.

Según Eurocontrol el coste medio por minuto para la aerolínea por un retraso es tanto cuando el avión de pasajeros está retrasado en suelo o en el aire, y es dividido en dos valores.

Estos dos valores se diferencian por el hecho de que un valor es el coste directo para la aerolínea, y el restante es resultante de un coste indirecto teniendo en cuenta por ejemplo el coste de oportunidad de los pasajeros.

Value 1

Direct cost to an airline (for system wide use or individual airline studies)

	Delay cost per minute (€)	
Tactical with network effect (includes reactionary delay)	Ground	49.5
	Airborne	69.2
Strategic (buffer built into schedule)	Ground	16.7
	Airborne	51.0

(Adjusted to 2014 values)

Value 2

Overall cost to an airline (for individual airline studies, includes passenger opportunity cost)

	Delay cost per minute (€)	
Tactical with network effect (includes reactionary delay)	Ground	90.8
	Airborne	110.5

(Adjusted to 2014 values)

Figura 2 - Valores Recomendados Eurocontrol

Los valores **1** y **2** estan basados en la metodología de la Universidad de Westminster en nombre de la Performance Review Unit de Eurocontrol.

Los datos de la figura pertenecen al informe presentado en 2004 y actualizado en 2011, representando la valoración más reciente y completa del coste de los retrasos en el sistema de gestión del tráfico aéreo en Europa. Contiene una evaluación detallada de los costes de retraso para 12 tipos de aeronaves específicas a fin de derivar una estimación del coste medio por minuto de retraso en Europa.

Se calculan tres tipos de retrasos:

- **Retraso táctico y sin efecto de red:** el retraso inicial en sí, sin el efecto de la consiguiente demora causada.
- **Retraso táctico con el efecto de la red:** el retraso real incluyendo el efecto de la consiguiente demora (propagación de retrasos), causados ya sea a la aeronave incurrida en el retraso inicial o para otros aviones.
- **Retraso estratégico:** el "buffer" integrado en horarios de previsión de retrasos.

Para cada tipo de retraso, el coste del retraso se divide en:

- **Retraso en tierra:** fases de vuelo antes del despegue y después del aterrizaje.
- **Retraso en Vuelo:** fases en ruta y de gestión de la llegada, con exclusión d3\e la fase inicial de ascenso inicial de 3000 pies y desde 3000 pies hasta el aterrizaje.

La siguiente tabla detalla los componentes del coste de la demora en valores recomendados:

Delay costs per minute (€)	Tactical without network effect		Tactical with network effect		Strategic	
	Ground	Airborne	Ground	Airborne	Ground	Airborne
Fuel costs	0.1	19.8	0.2	19.8	1.3	24.0
Maintenance costs	0.5	1.0	0.6	1.1	-	11.7
Crew costs	7.2	7.2	8.6	8.6	8.4	8.4
Ground and passenger handling	-	-	-	-	-	-
Airport charges	0.5	-0.0	0.6	0.1	-	-
Aircraft ownership costs[2]	-	-	-	-	6.9	6.9
Passenger compensation	22.0	22.0	39.6	39.6	-	-
Direct cost to an airline	**30.3**	**50.0**	**49.5**	**69.2**	**16.7**	**51.0**
Passenger opportunity cost	22.9	22.9	41.2	41.2	-	-
Overall cost	**53.2**	**72.9**	**90.8**	**110.5**	**16.7**	**51.0**

(Adjusted to 2014 values)

Figura 3 - Costes Retrasos por Minuto

En 2013 el Centro Experimental de Eurocontrol en Bretigny, hizo un estudio de la propagación de los retrasos en el turnaround, siguiendo los datos de Air France y en numerosos aeropuertos franceses. Ellos crearon un modelo con el objetivo de explicar la progresión de los retrasos a través de los aeropuertos. El itinerario de las aeronaves fueron seguidas, a la vez que parámetros locales en cada aeropuerto, y conjuntamente se observó un grupos de posibles retrasos debidos a otras circunstancias que las locales, y todo ello fue implementado además de los efectos de los slots del ATFM y eventos excepcionales.

En primer lugar encontraron que la duración real del vuelo excede como media, hasta 6 minutos. Además se descubrieron que los retrasos cortos (entre tres y catorce minutos) resultan principalmente de un retraso anterior y/o de condiciones locales (retraso en la carga, tiempo de parada programada). Dando como conclusión que los efectos locales en los retrasos por si solos no pueden alcanzar los 15 minutos, si no son debidos a un cambio en el slot por parte del ATFM.

El hecho de que se produzca esta propagación e incluso aumento de los retrasos significa que siempre hay que mirar que ha pasado en el vuelo anterior, seguidamente si como el estudio reveló, se aprecia que por condiciones locales como máximo el retraso puede ser de 15 minutos, se puede llegar a entender que la propagación de retraso es la suma de los retrasos anteriores más los retrasos por condicionantes locales.

Todas las causas de retrasos se pueden dividir en retrasos primarios y de propagación, donde los segundos son aquellos causados por el retraso en la llegada del avión, pasajeros, carga, etc…Mientras que los primarios son aquellos que se dan lugar en el proceso del turnaround.

Algunos parámetros a tener en cuenta en el proceso son:

- Repostaje
- Catering
- Modelo de Avión
- Hora del Vuelo
- Limpieza
- Handling de Equipaje

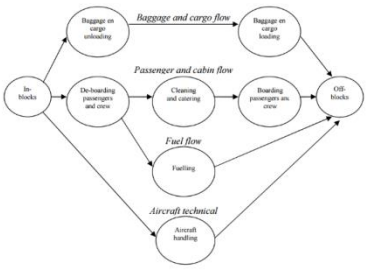


Figura 4 - Workflow de un Turnaround

2. PREDICCIÓN DE TIEMPOS DE TURNAROUND USANDO MÉTODOS DE "CIENCIA DE DATOS"

2.1 Descripción General "Ciencia de Datos"

2.1.1 Definición

La Ciencia de datos es un campo interdisciplinario que involucra los procesos y sistemas para extraer conocimiento o un mejor entendimiento de grandes volúmenes de datos en sus diferentes formas (estructurados o no estructurados) y diferentes formatos (.txt, .dat, .doc, .jpg, json, etc.).

Es una continuación de algunos campos de análisis de datos como son: la minería de datos y la analítica predictiva.

La ciencia de datos es un nuevo paradigma sobre el cual los investigadores se apoyan de sistemas y procesos que son muy diferentes a los utilizados en el pasado, como son modelos, ecuaciones, algoritmos, así como evaluación e interpretación de resultados.

La ciencia de datos ha resultado para muchos una disciplina de reciente creación, pero en realidad este concepto lo utilizo por primera vez el científico danés Peter Naur en la década de los sesenta como sustituto de las ciencias computacionales.

En 1974 se publicó el libro "Concise Survey of Computer Methods" donde se utiliza ampliamente el concepto "Ciencia de Datos", lo que permitió que se comenzara a utilizar más libremente entre el mundo académico y no fue hasta el 2008 que Jeff Hammerbacher y DJ Patil lo reutilizaron para definir sus propios trabajos realizados en Facebook y Linkedin, respectivamente, llevando a la práctica lo que advirtió Peter Drucker en 1993 con su libro La Sociedad Postcapitalista, en el que expone la necesidad de una nueva teoría económica donde se coloca el conocimiento en el centro de la producción de la riqueza. Además, recalco que la cantidad de conocimiento no era lo importante, sino la productividad que éste llegue a generar. En pocas palabras dijo que para una sociedad de la información el recurso básico seria el saber, y Hammerbacher y Patil lo entendieron muy bien, situación en la actualidad ha masificado más que nunca el concepto de ciencia de datos a pesar de tener más de 40 años.

Actualmente se ha intensificado el interés en la ciencia de datos debido fundamentalmente a:

- Disponibilidad de grandes cantidades de datos abiertos
- Accesibilidad a las plataformas de Cloud Computing y tecnologías de Big Data que permiten el procesamiento de grandes cantidades de datos (por ejemplo, Amazon Web Services, Google Cloud Platform, FIWARE…)

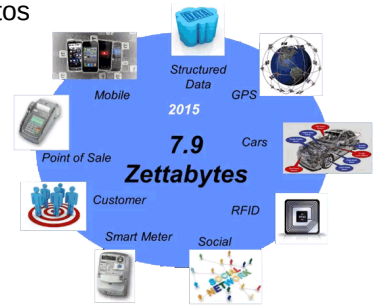


Figura 5 - Fuentes de Datos

Un dato importante a considerar, es que actualmente el 90% de los datos mundiales, han sido creados en los últimos dos años.

2.1.2 Términos relacionados

A menudo el término Ciencia de Datos, va muy unido con Business Intelligence (BI), y a la vez con el término más que conocido Big Data, cabe por ello matizar sus diferencias. Business Intelligence también se ha popularizado en estos tiempos, e incluso se ha llegado a utilizar de

manera indiscriminada con el concepto de ciencia de datos para referirse al análisis de datos, pero en realidad existen diferencias abismales entre dichos conceptos, a continuación se presenta alguna de sus diferencias.

Ciencia de datos:

- Trabaja en datos incompletos
- Los datos suelen estar desordenados
- Analiza los datos para ver qué información obtiene
- Grandes conjunto de datos que es un desafío administras
- Los hallazgos impulsan decisiones sobre operaciones y productos
- Aplicación de técnicas inteligentes como son Aprendizaje Automático y Minería de Datos

Business Intelligence (BI):

- Conjunto de datos completos
- Archivos de datos limpios
- Informa lo que dicen los datos
- Conjuntos de datos manejables
- Sus hallazgos miden el rendimiento pasado
- Énfasis en la creación de cuadros de mando y en la visualización de datos
- Típicamente BI no cubre técnicas inteligentes para la recuperación del conocimiento a partir de datos

Otro término que se relación frecuentemente con la ciencia de datos y BI es el de Big Data, de acuerdo con la guía de *Amazon Web Service* considera al Big Data como a una cantidad considerable de datos con dificultades para almacenarse en base de datos tradicionales, para procesarse en servidores estándar y para analizarse con aplicaciones habituales. Cuando hablamos de "Big Data" nos referimos a un volumen de datos que van desde los Terabytes (10^{12} Bytes) a los Zettabytes (10^{21}), junto con la relación de la variedad de datos, y la velocidad, siendo estas, las 3V's que lo forman. El Big data es el insumo esencial del científico de datos.

Por un lado en este proyecto no se trabajaran con datos totalmente limpios, ni de una forma archivada, por ello cabe descartar el Business Intelligence, y por otro la cantidad de datos (inferior a los 10,000 registros) hará que no sea necesario el uso de las técnicas y herramientas que hacen falta en Big Data (por ejemplo, Apache Hadoop, Spark, etc.).

Para la creación de modelos de toma de decisión inteligentes, en el proyecto ser aplicarán el uso de técnicas que aprendizaje automático (Machine Learning), cosa que no se considera en ámbitos como BI (Business Intelligence).

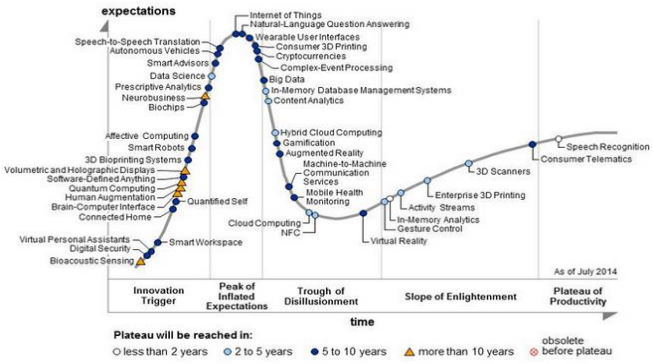


Figura 6 - Hype Cycle. Donde Big Data está en la última

2.1.3 Científico de Datos

Las personas que se dedican a la ciencia de datos se les conoce como científico de datos, de acuerdo con el proyecto Master in Data Science define al científico de datos como una mezcla de estadísticos, ingenieros en sistemas computacionales y pensadores creativos, con las siguientes habilidades:

- Recopilar, procesar y extraer valor de las diversas y extensas bases de datos.
- Imaginación para comprender, visualizar y comunicar sus conclusiones a los no científicos de datos.
- Capacidad para crear soluciones basadas en datos que aumentan los beneficios y reducen los costes.
- Los científicos de datos trabajan en todas las industrias y hacen frente a los grandes proyectos de datos en todos los niveles.

Usualmente es preciso que los científicos de datos, mezclen los conocimientos estadísticos con interfaces de programación de aplicaciones (APIs), bases de datos y extracción de datos, a la vez que los conocimientos de la ingeniería en sistemas computacionales, que todo ello servirá para analizar y encontrar datos con significado.

En la tesis del Doctor Benjamin Fry, se explicó que el proceso para comprender mejor a los datos comenzaba con una serie de números y el objetivo de responder pregunta sobre los datos, en cada fase del proceso que él propuso (adquirir, analizar, filtrar, extraer, representar, refinar e interactuar), se requieren de diferentes enfoques especializados que aporten una mejor comprensión de los datos.

Entre los enfoques que menciona el Doctor Fry están: ingenieros en sistemas, matemáticos, estadísticos, diseñadores gráficos, especialistas en visualización de la información y especialistas en interacciones hombre-máquina, mejor conocidos por sus siglas en inglés “HCI” (Human-Computer Interaction). Además, Fry afirmó que contar con diferentes enfoques especializados lejos de resolver el problema de entendimiento de datos, se convierte en parte del problema, ya que cada especialización conduce de manera aislada el problema y el camino hacia la solución se puede perder algo en cada transición del proceso.

2.1.4 Pasos claves en proyecto de Ciencia de Datos Metodología de siete etapas

Benjamin Fry propuso un método que consta de siete etapas que van desde la recolección de datos, hasta el usuario interactuando con estos. La secuencia se presenta a continuación:

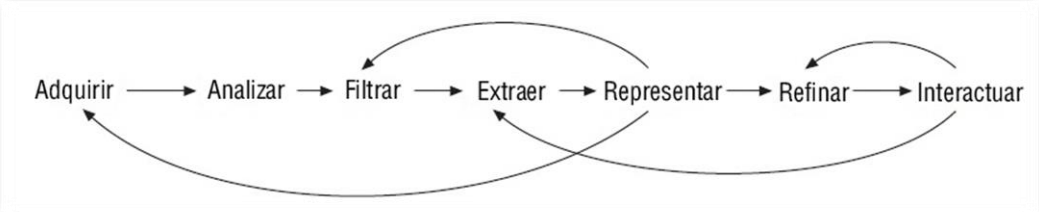


Figura 7 - Interacción 7 Etapas por Ben Fry

Dependiendo de los requerimientos, y de los resultados de cada paso, puede o no ser necesario regresar a pasos anteriores para modificar ciertas decisiones.

1. **Adquirir Datos**
   Para este paso, lo principal es tener una fuente confiable, proveniente de algún organismo oficial, instituto de estadística, estudio académico, etc.

2. **Analizar**
   En este paso, la información recopilada se ordena en estructuras, de modo que los datos adquiridos comiencen a adquirir una estructura.

3. **Filtrar**
   Este paso consiste en eliminar los antecedentes innecesarios.

4. **Extraer**
   Este paso involucra matemáticas, estadística, y minería de la información es decir, metodología científica aplicada.

5. **Representar**
   Este paso determina la forma básica que tomará el conjunto de datos.

6. **Refinar**
   En este paso se aplican métodos de diseño gráfico para fomentar la claridad de la representación, aumentando la atención sobre datos particulares (estableciendo jerarquías), o modificando atributos (como el color) para contribuir la legibilidad.

7. **Interactuar**
   En esta etapa se añade interacción, permitiendo al usuario controlar o explorar los datos, esto puede ser cambiando el punto de vista, u otras variables o características.

Processing

Dentro de los pasos de Adquisición de Datos, así como de Analizado o Fitrado entre otros, nos encontramos el Processing (o Procesado), para trabajar con mayor facilidad con los datos, se constituye de lenguaje de programación mayoritariamente en código abierto, y un Entorno de Desarrollo Integrado (o IDE por dud diglas en inglés). Fue construido para las comunidades de artes electrónicas, y de un diseño visual, con el propósito de enseñar lo básico de la programación computacional en un contexto visual.

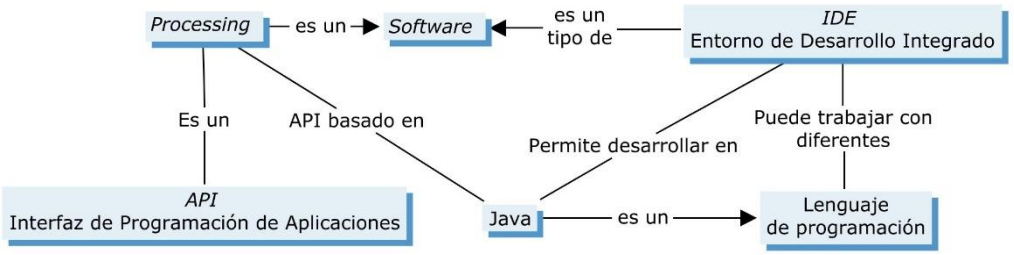


Figura 8 - Fases Processing

2.2 Entorno y Lenguaje

Para llevar a cabo la programación de los modelos predictivos y con esto, la limpieza de datos, estructuración de los mismos, etc. Se debe de acertar con la elección del entorno que será usado así como el lenguaje llevado a cabo en este.

Para situaciones con grandes cantidades de datos de una forma semi-estructurada, existen varias plataformas que son adecuadas para cumplir con esta finalidad, hay entornos más sencillos, y de otros más complicados, en el caso del actual estudio, para hacer una aproximación a estos tipos de análisis de datos, juntos con la creación de modelos predictivos, se ha tomado la decisión de elaborar el proyecto a través del Entorno Integrado de Desarrollo (IDE) JetBrains PyCharm, junto con el lenguaje requerido para la programación en esta plataforma, Python.

2.2.1 JetBrains PyCharm

PyCharm es un IDE (Integrated Development Environment) usado para programar en Python. Este proporciona herramientas de análisis de código, un depurador gráfico, testers de unidad integrados, integración con sistemas de control de versiones (VCSes), y soporta el desarrollo web a través de Django.

Figura 9 - Pycharm 4.0.5 Logo

PyCharm es muy versátil ya que es multiplataforma, pudiendo ser utilizado en Windows, Mac OS X y Linux. A parte de disponer una edición profesional, mediante el pago de una licencia, PyCharm dispone a la vez de una Community Edition, edición gratuita, esta es menos extensa que la versión profesional, pero que aun así se ajusta a la mayoría de proyectos con el alcance del que se desea realizar.

2.2.2 Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

Se trata de un lenguaje de programación multi-paradigma, significa a la vez que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.

Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. Es administrado por la Python Software Foundation.

Figura 10 - Logo Lenguaje Python

La popularidad de Python para la ciencia de datos se debe en gran parte a la fortaleza de sus bibliotecas principales (como son NumPy, SciPy, Pandas, Matplotlib, IPython), de alta productividad, para la creación de prototipos y la construcción de sistemas pequeños y reutilizables, y su fuerza como un lenguaje de programación de propósito general.

Desde que los científicos de datos también suelen estar implicados con una red de aplicaciones de una forma conjunta, programando para la web, realizando secuencias de comandos para las automatizaciones de da tareas de procesamiento de datos y otros procesos, es necesario el hecho de hacer todos estos trabajos, además del análisis real y modelado, en un solo lenguaje de programación.

Es un gran lenguaje que puede servir para desarrollar fácilmente tanto aplicaciones pequeñas como aplicaciones de gran envergadura. Destacan tres aspectos importantes de Python:

- **Alto nivel:** Ya que su sintaxis es relativamente sencilla, lo hace un gran lenguaje para aprender a programar. A su vez facilita el desarrollo de aplicaciones, ya que acorta el número de líneas de código a desarrollar con respecto a otros lenguajes de programación.
- **Multipropósito:** Puede ser usado para desarrollar tanto scripts sencillos como para desarrollar sitios webs dinámicos.

- **Posee una gran librería pre-instalada de apoyo:** Cabe la posibilidad que dentro de la librería ya estén desarrolladas muchas de las cosas comunes que se quieran hacer, así nos evitamos programar un módulo desde cero.

Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos).

Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en C o C++. Python puede incluirse en aplicaciones que necesitan una interfaz programable.

Por la gran similitud con la programación el lenguaje C, C++, hace que sea un lenguaje candidato muy atractivo para el proyecto.

2.3 Métodos Obtención de Datos

2.3.1 Introducción

Para desarrollar un estudio, y realizar actividades de análisis, es necesario utilizar un conjunto amplio de conocimientos, estos conocimientos deben ser encontrados por medio de un trabajo indagatorio sobre los objetos que se intenta conocer.

Cuando se empieza a preocuparse en el modo en que se adquieren los dichos conocimientos, o cuando se intenta encontrar conocimientos nuevos, se presentan cuestiones de variada índole, muchas de las cuales integran el campo de la Metodología.

Una vez se obtengan los indicadores de los elementos teóricos y definido el diseño de la investigación, será necesario definir que técnicas de recolección de datos se utilizaran para construir los instrumentos que permita obtenerlos.

Cualquier instrumento para la obtención de datos será eficaz si es útil para acercarse a los fenómenos y extraer la información que se necesite observar de ellos.

- La observación en sí debe cumplir algunos principios básicos, entre ellos:
- Debe tener un propósito especifico
- Debe ser planeada cuidadosa y sistemáticamente.
- Debe llevarse, por escrito, un control cuidadoso de la misma.
- Debe especificarse su duración y frecuencia.
- Debe seguir los principios básicos de validez y confiabilidad.

Para la observación del estudio que se presenta, se debe realizar de una forma telemática ya que una observación presencial no es posible, a la vez que no es precisa, esto se puede realizar de distintas formas que serán descritas a continuación, entre estas formas las más destacadas son el WEB Scraping, y las Interfaces de Programación de Aplicaciones (es decir API).

2.3.2 WEB Scraping

2.3.2.1 Definición

El WEB Scraping es una técnica utilizada mediante programas de software para extraer información de sitios web. Usualmente estos programas simulan la navegación de un humano en la *World Wide Web* ya sea utilizando el protocolo HTTP manualmente, o incrustando un navegador en la aplicación.

El WEB Scraping está muy relacionado con la indexación de la web, la cual indexa la información de la web utilizando un robot y es una técnica universal adoptada por la mayoría de los motores de búsqueda. Sin embargo, el WEB Scraping se enfoca más en la transformación de datos sin estructura en la web (como el formato HTML) en datos estructurados que pueden ser almacenados y analizados en una base de datos central, en una hoja de cálculo o en alguna otra fuente de almacenamiento. El termino WEB Scraping también está relacionado con la automatización de tareas en la Web, la cual simula la navegación de un humano utilizando un software de computadora. Alguno de los usos del WEB Scraping es la comparación de precios en tiendas, la monitorización de datos relacionados con el clima de cierta región. La detección de cambios en sitios webs y la integración de datos en sitios webs.

2.3.2.2 Técnicas

WEB Scraping es el proceso de recopilar información de forma automática de la Web. Es un campo con desarrollos activos, compartiendo un propósito en común con la visión Web semántica. Utiliza soluciones prácticas basadas en tecnologías existentes que son comúnmente *ad hoc*. Existen distintos niveles de automatización que las existentes tecnologías de WEB Scraping pueden brindar:

- *Copiar y pegar* humano: Algunas veces incluso las mejores técnicas de WEB Scraping no pueden reemplazar la examinación manual de un humano, y a veces esta puede ser la única vía de solución cuando el sitio que se tiene en mente pone ciertas barreras para prevenir que se creen software para realizar tareas automáticas en este.
- Uso de expresiones regulares: Una posible vía para extraer información de páginas webs pueden ser las expresiones regulares, aunque comúnmente no se recomienda utilizarlas para parsear el formato HTML.
- Protocolo HTTP: Páginas webs estáticas y dinámicas pueden ser obtenidas haciendo peticiones HTP al servidor remoto utilizando sockets, etc.
- Algoritmos de minería de datos: Muchos sitios webs tienen grandes colecciones de páginas generadas dinámicamente a partir de una base de datos. Datos de la misma categoría aparecen usualmente en páginas similares mediante un script o una plantilla. En la minería de datos, un programa detecta estas plantillas en un contexto específico y extrae su contenido.
- Parsers de HTML: Algunos lenguajes como XQuery y HTQL pueden ser utilizados para parsear documentos, recuperar y transformar el contenido de documentos HTML.
- Aplicaciones para WEB Scraping: Existen muchas aplicaciones disponibles que pueden ser utilizadas para personalizar soluciones de WEB Scraping. Estas aplicaciones pudieran reconocer automáticamente la estructura de cierta página o brindar una interfaz al usuario donde este pudiera seleccionar los campos que son de interés dentro del documento. De esta forma no es necesario escribir manualmente código para realizar estas tareas.
- Reconocimiento de información semántica: Las páginas que son analizadas podrían incluir metadatos o cierta información semántica como anotaciones o comentarios, los cuales pueden ser usados comúnmente. Si estas anotaciones están en las mismas páginas, como sucede con los microformatos, estas podrían ser de utilidad cuando parseamos el DOM del documento. En otro caso, las anotaciones, organizadas en una capa semántica, son almacenadas y manejadas de forma separada desde otras páginas, por lo que los scrapers pueden recuperar estos esquemas y las instrucciones desde esta capa antes de analizar los documentos.

2.3.3 API (Application Programming Interface)

2.3.3.1 Definición

Se conoce como API, la interfaz de programación de aplicaciones con la abreviación del inglés *Application Programming Interface*, que es el conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Generalmente son usadas en bibliotecas de programación.

2.3.3.2 Características

Una API representa la capacidad de comunicación entre componentes de software. Se trata del conjunto de llamadas a ciertas bibliotecas que ofrecen acceso a ciertos servicios desde los procesos y representa un método para conseguir abstracción en la programación, generalmente (aunque no necesariamente) entre los niveles o capas inferiores y los superiores del software. Uno de los principales propósitos de una API consiste en proporcionar un conjunto de funciones de uso general, por ejemplo, para dibujar ventanas o iconos en la pantalla. De esta forma los programadores se benefician de las ventajas de la API haciendo uso de su funcionalidad, evitándose el trabajo de programar todo desde el principio. Las API

asimismo son abstractas: el software que proporciona una cierta API generalmente es llamado la implementación de esa API.

2.3.3.3 Ejemplos

En la web, las API's son publicadas por sitios para brindar la posibilidad de realizar alguna acción o acceder a alguna característica o contenido que el sitio provee. A parte de la API de FlightStats que se explicará su funcionamiento de una forma más detallada a partir del punto 3, existen un gran número de otras API, algunos ejemplos de algunas que actualmente son conocidas y utilizadas por un gran número de usuarios son:

- **Amazon** Es un servicio totalmente gestionado que facilita a los desarrolladores la creación, la publicación, el mantenimiento, el monitoreo, y la protección de API a cualquier escala. Proporcionando acceso mediante programación a las funcionalidades de selección de productos de Amazon para que los desarrolladores puedan crear anuncios de productos Amazon en sitios webs.
- **Google Search** Esta API permite poner la búsqueda de Google en diferentes páginas web mediante JavaScript. Pudiendo incrustar un simple recuadro de búsqueda, mostrar resultados de búsqueda en las propias páginas o utilizar esos resultados de formas innovadoras.
- **Google Maps** Parecida a la API anterior la API de Google Maps, permite visualizar mapas y acceder a funciones completas de asignación incrustadas en páginas webs, como indicaciones precisas o Street View. Ayudando también a elaborar mapas personalizados y contenidos propios.
- **Flickr** La API de Flickr es una poderosa manera de interactuar con las cuentas de Flickr. Con la API, se puede leer casi rodos los datos asociados con fotos y juegos. También facilita la subida de fotos a través de la API.
- **Delicious** Probablemente Delicious API, proporcione una de las API más potentes y flexibles de la web. Ya que cada día se sirven más de mil millones de enlaces. Y se integra fácilmente en cualquier sitio web o aplicación, para de una manera fácil, leer, agregar, consultar, editar paginas favoritas, etc.

Figura 11 – API's con gran cantidad de usuarios

2.4 Exploración y Limpieza de Datos

2.4.1 ¿El porqué de la limpieza de datos?

Antes de usar los datos que se han recogido, hay que cerciorarse que estos se encuentren de una manera correcta, consistente, entre otros atributos.

Datos incorrectos o inconsistentes pueden derivar en falsas conclusiones y en inversiones mal dirigidas.

La limpieza de datos es un proceso que principalmente basa su acción en detectar y corregir –o borrar cuando sea necesario - los datos que sean corruptos o entradas con falta de precisión, todo ello en un sistema de datos, ya sea una tabla o una base de datos.

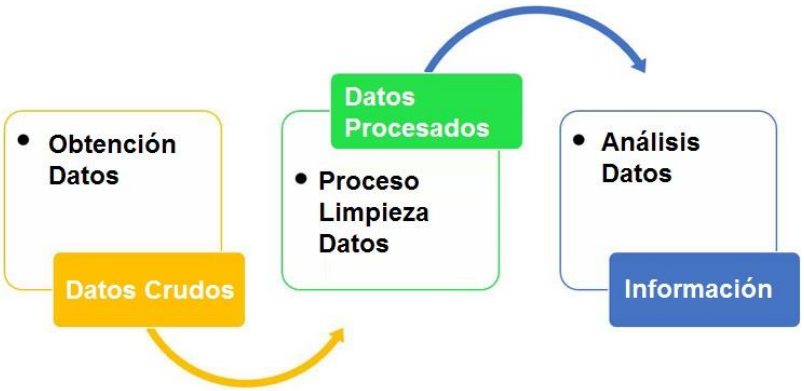


Figura 12 - Fases Limpieza Datos

2.4.2 Automatización

La limpieza de datos puede realizarse de una manera interactiva, a través de herramientas designadas para ello, donde principalmente consiste en transformar la información “cruda” en información valida, a través de mecanismos semi-automatizados, ya pudiendo ser basados en modelos estadísticos, usando algoritmos de ordenación, etc. En la actualidad existen muchas herramientas para realizar el proceso de la limpieza de datos.

O por otro lado la limpieza se puede realizar de una forma automatizada, mediante la escritura y ejecución de código, donde primero se plantean las necesidades que requerirá el programa para proceder a la escritura de un código que cumpla los requisitos de la limpieza.

2.4.3 Calidad en los datos

Una vez completada la limpieza, el conjunto de los datos será consistente con otros conjuntos de datos similares en el sistema. Las inconsistencias detectadas o eliminadas a veces han sido originadas por errores en el momento de entrar los datos por parte del usuario, por la corrupción de estos datos en el momento de la transferencia de los datos o su almacenaje,

En el proceso se suele ver implicada la eliminación de datos por errores tipográficos, o contrastar valores con otros, de una lista de entidades ya conocida, sin dejar lugar valores no incluidos en esta lista, por ejemplo.

Otro factor importante a tener en cuenta en algunas circunstancias, es suprimir aquellos registros de una forma íntegra por la falta de uno de sus campos, si este es indispensable para el estudio que se esté realizando.

2.4.5 Criterios

Para lograr unos datos de alta calidad, se necesita pasar un conjunto de criterios de calidad, estos son los siguientes:

- **Validez:** Es el grado en que los elementos se ajustan a las reglas o restricciones. Cuando es utilizada la tecnología para diseñar el sistema de obtención de datos, la validez es fácil de asegurar.
- **Entereza:** El grado en el que se conocen todos los elementos necesarios. Hecho que conlleva que sea difícil de fijar a la hora de la limpieza, ya que no se conocen los valores que no fueron obtenidos.
- **Consistencia:** El grado en que un conjunto de elementos son equivalentes entre los sistemas. La inconsistencia se produce cuando dos elementos de datos en el conjunto se contradicen entre sí.
- **Unicidad:** Aquello relacionado con los datos duplicados
- **Uniformidad:** El grado en el que se especifican valores de datos en todo el conjunto con las mismas unidades de medida entre todos los sistemas.
- **Densidad:** Es importante conocer la densidad, dada por el cociente de valores omitidos sobre el número de valores totales.
- **Exactitud:** Recoge que los datos cumplan los requisitos de integridad consistencia y densidad
- **Integridad:** El termino Integridad engloba, exactitud, consistencia y algunos aspectos de validación.

2.4.6 Proceso de Limpieza de Datos

– **Auditar los datos:** Los datos deben ser revisados a través de métodos estadísticos para descubrir anomalías y contradicciones. Esto finalmente da una indicación de las variables y sus respectivas posiciones

– **Definición del Flujo de Trabajo:** Una secuencia de operaciones trabaja en la detección y la eliminación de anomalías, esto es conocido como flujo de trabajo (o workflow). Para un correcto funcionamiento de este, se debe identificar las principales causas de las anomalías y errores de los datos.

Figura 13 - Ciclo Limpieza de Datos

– **Ejecución del Flujo de Trabajo:** Después tener un Flujo de Trabajo verificado y corregido, se procede a su ejecución.

– **Controles y Post-Proceso:** Los datos que no hayan podido ser corregido mediante la ejecución del Flujo de Trabajo deberán ser corregidos a mano, siempre y que se requiera de esos conjuntos de datos. Tras esta corrección, ser deberá ejecutar de nuevo el Flujo de Trabajo para finalmente revisar que se hayan cumplido las especificaciones del proyecto.

2.4.7 Métodos más utilizados en la limpieza de datos

- **Análisis:** Para la limpieza de los datos es necesario un previo análisis, que se realiza para encontrar los posibles errores en los datos. Este algoritmo, decidirá si es necesario declarar algunos conjuntos de datos como erróneos o no.
- **Transformación de datos:** La transformación de los dato hace posible la conversión de los datos en un formato deseado. Esto se lleva a cabo mediante conversiones de valor, funciones de transformación, así como normalización de valores numéricos, para que estos sean aceptable dentro de unas restricciones de unidades, tamaño, etcétera.
- **Eliminación de Duplicados:** Se suele requerir un algoritmo para la comprobación de la existencia de registros duplicados, por un lado, o duplicidades en aquellos campos que requieran una identificación única.

Figura 14 - Eliminación de Duplicados en CSV

- **Métodos Estadísticos:** Realizan la tarea del análisis de los datos a través de métodos estadísticos como pueden ser desviaciones, promedios, rangos etc. Estos métodos a la vez pueden ser de utilidad para manejar la inexistencia de algunos datos, que pueden ser substituidos manualmente o por métodos estadísticos como por ejemplo las interpolaciones.

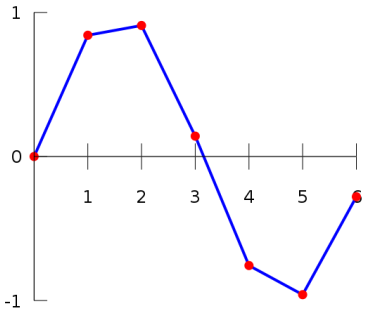


Figura 15 - Ejemplo Interpolación Lineal

2.5 Feature Engineering (Ingeniería de Variables)

2.5.1 Definición

Una breve definición la introduce como:

"La ingeniería de Variables es el proceso de la transformación de datos crudos en variables que representan mejor el problema de fondo de los modelos predictivos, resultando en una mejor exactitud del modelo predictivo, especialmente en los datos invisibles"

O por otro lado:

"La Ingeniería de Variables (Feature Engineering) es el proceso de usar el conocimiento del dominio de los datos, para crear variables que hace que los algoritmos de Aprendizaje Automático (Machine Learning) funcionen"

Este último hace referencia a poder prescindir de una Ingeniería de Variables que se deba construir de una manera manual, para dejarlo al Aprendizaje Automático.

Importancia de la Ingeniería de Variables

Es de suma importancia ya que el objetivo de un estudio usando métodos de análisis de datos, es conseguir los mejores resultados a través de un modelo predictivo, consiguiendo el mayor provecho a partir del material que uno ya dispone.

Esto incluye obtener los mejores resultados de los algoritmos que se están usando en el proyecto, implicando a la vez lograr el mayor rendimiento de estos algoritmos.

Las variables de los datos con los que se están trabajando, influirá directamente en el modelo predictivo que se use, así como en los resultados que se alcanzaran. Es decir que a mayor calidad de las variables que se preparen y se escojan, mejor serán los resultados que se lograran.

2.5.2 Proceso a realizar

- Lluvia de ideas, en lo que a Variables se refiere.
- Decidir qué Variables se van a crear.
- Creación de las Variables.
- Comprobación del funcionamiento de las Variables en el modelo.
- Mejorar las Variables si fuera necesario.
- Regresar a la lluvia de ideas y/o al proceso creativo de Variables hasta que se finalice el trabajo.

2.5.3 Relevancia de las Variables

Cada Característica tiene una relevancia, ya que probablemente en unas haya información que o se encuentra en ninguna de las otras.

Otras serán consideradas con una relevancia débil, ya que probablemente contenga información que ya contengan otras.

Y por último habrá aquellas que sean irrelevantes y puedan ser eliminadas del modelo.

2.5.4 Feature Extraction (Extracción de Variables)

Hay veces en que los datos recopilados en su forma “cruda” son demasiado numerosos para ser modelados directamente por algoritmos de modelado predictivo.

Algunos ejemplos comunes son las imágenes, sonidos, etc., que los datos tabulados ocuparían millones de atributos.

Feature Extraction es un proceso que reduce la dimensión automáticamente de estos tipos de observaciones, a un conjunto mucho más pequeño que podrá ser modelado con relativa facilidad.

Figura 16 - Etapas Extracción Características

Para los datos tabulares, se puede incluir métodos de proyección, así como el análisis de las componentes principales y también algunos métodos no supervisados. En el caso de imagen - a modo de curiosidad ya que el proyecto actual no se encuentra en este ámbito – un modo de desarrollar las características principales puede ser las líneas de la imagen, o por ejemplo la detección de bordes en esta.

La ventaja principal es que estos métodos son automáticos, aunque requieran de un diseño y construcción previos, estos resuelven los problemas de la gestión de grandes dimensiones de datos.

2.6 Análisis y modelos predictivos

2.6.1 Descripción General

Los modelos predictivos pueden ser descritos como un conjunto de técnicas matemáticas que combinadas de una forma adecuada, ayudan a obtener comportamientos futuro, a partir del análisis de los datos históricos y de los actuales que a su vez van creando modelos estadísticos cada vez más fiables.

El objetivo del modelo predictivo es evaluar y calcular la probabilidad de que una hecho similar tenga el mismo comportamiento o rendimiento en una muestra diferente. En ellos se buscan patrones de datos ocultos o que no son visibles de una manera fácil, y que respondan a preguntas sobre este comportamiento.

Gracias a los avances actuales respecto la ingeniería e informática, en el análisis de grandes cantidades de datos, los modelos actuales son capaces de acercarse al comportamiento humano, en situaciones o estímulos específicos.

A diferencia de lo que sucedía en el pasado con las herramientas de análisis predictivo, hoy en día ya no se requiere de usuarios con habilidades muy avanzadas para la implantación y uso de estos, todo ello debido a su popularización, que ha permitido herramientas más modernas que han ido mejorando sustancialmente el acceso a todo tipo de usuarios analistas.

Existe una cantidad creciente de herramientas disponibles para la ejecución de estudios en ámbito predictivo, des de las herramientas básicas que requieren de muy poca sofisticación del usuario, hasta aquellas que han sido diseñadas para usuarios expertos.

Los usuarios de las herramientas de análisis predictivo buscan sobre todo, que estas herramientas les permita analizar y visualizar la información de sus datos, de una forma que cualquier cargo de la empresa pueda extraer conocimiento útil para la toma de decisiones empresarial. Es por ello que normalmente la visualización de esta información útil se realiza a través de cuadros, gráficos, tablas, etc., que muestran sencillamente y de una forma muy gráfica los resultados de las predicciones.

2.6.2 Random Forests

2.6.2.1 Origen

Es un algoritmo que fue desarrollado por Leo Breiman y Afele Cutler aproximadamente en 1995. También es conocido como Selvas Aleatorias, y es una combinación de árboles predictores, de la manera que de cada árbol depende de los vectores de un vector aleatorio probado independientemente, y con la misma distribución para cada uno de estos.

2.6.2.2 Definición

Es una modificación sustancial de bagging (es un meta-algoritmo diseñado para mejorar la estabilidad y la precisión del aprendizaje automático) que forma una larga colección de árboles no correlacionados para posteriormente promediarlos.

El algoritmo Random Forests es sustancialmente más simple de entrenar y ajustar. Hecho que da popularidad al algoritmo y es ampliamente usado.

Los árboles son los candidatos ideales para el bagging, dado que ellos pueden registrar estructuras de interacción compleja en los datos, y si crecen de una manera suficientemente profunda, tendrán entonces relativamente baja parcialidad.

Producto de que los árboles son notoriamente ruidosos, ellos se benefician grandemente al promediar.

Cada árbol es construido usando el siguiente algoritmo:

1. Sea *N* el número de casos de prueba, *M* es el número de variables en el clasificador.
2. Sea *m* el número de variables de entrada a ser usado para determinar la decisión en un nodo dado, m debe ser mucho menor que *M*
3. Elegir un conjunto de entrenamiento para este árbol y usar el resto de los casos de prueba para estimar el error.
4. Para cada nodo del árbol, elegir aleatoriamente *m* variables en las cuales basar la decisión. Calcular la mejor partición a partir de las *m* variables del conjunto de entrenamiento.

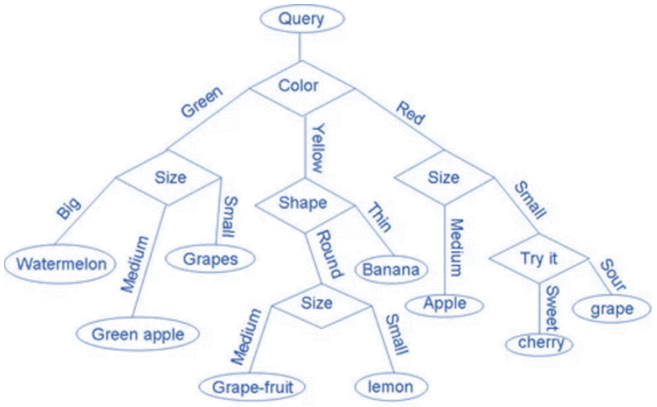


Figura 17 - Ejemplo Representación Random Forests

Para la predicción, un nuevo caso es empujado hacia abajo por el árbol. Luego se le asigna la etiqueta del nodo terminal donde termina. Este proceso es iterado por todos los árboles en el ensamblado, y la etiqueta que obtenga la mayor cantidad de incidencias es reportada como la predicción.

2.6.2.3 Ventajas Random Forests

- Es uno de los algoritmos de aprendizaje más certero que hay disponible. Para un set de datos lo suficientemente grande produce un clasificador muy acertado.
- Corre eficientemente en base de datos grandes.
- Puede manejar cientos de variables de entrada sin excluir ninguna.
- Da estimados de qué variables son importantes en la clasificación.
- Tiene un método eficaz para estimar datos perdidos y mantener la exactitud cuando una gran proporción de los datos está perdida.
- Computa los prototipos que dan información sobre la relación entre variables y la clasificación.
- Computa las proximidades entre los pares de casos que pueden usarse en los grupos, localizando valores atípicos, o (ascendiendo) dando vistas interesantes de los datos.
- Ofrece un método experimental para detectar las interacciones de las variables.

2.6.2.4 Desventajas

- Se ha observado que Random Forests sobre-ajusta en ciertos grupos de datos con tareas de clasificación/regresión ruidosas.
- A diferencia de los árboles de decisión, la clasificación hecha por Random Forests es difícil de interpretar por el hombre.
- Para los datos que incluyen variables categóricas con diferente número de niveles, Random Forests se parcializa a favor de esos atributos con más niveles. Por consiguiente, la posición que marca la variable no es fiable para este tipo de datos. Métodos como las permutaciones parciales se han usado para resolver el problema.
- Si los datos contienen grupos de atributos correlacionados con similar relevancia para el rendimiento, entonces los grupos más pequeños están favorecidos sobre los grupos más grandes.

2.6.3 Gradient Boosting Trees

2.6.3.1 Descripción General

Gradient Boosting es una técnica de Aprendizaje Automático (Machine Learning) para la regresión y clasificación de problemas, que crea un modelo predictivo en forma de conjunto de modelos débiles de predicción. Típicamente árboles de decisión.

Esta técnica construye el modelo de una manera escalonada en etapas, al igual que otros modelos estilo “Boosting”, y los generaliza al permitir la optimización de una función de pérdida diferenciable arbitraria.

La idea de la técnica Gradient Boosting, se originó por la observación de Leo Breiman, donde el Boosting puede ser interpretado como un algoritmo de optimización en una función de coste adecuada.

Los Gradient Boosting de regresión explícita, fueren desarrollados posteriormente, simultáneamente con el Gradient Boosting más general y funcional. Los dos últimos trabajos introdujeron la visión abstracta de impulsar algoritmos como algoritmos de Functional Gradient Descent (Descenso Gradual Funcional). Es decir, los algoritmos que optimizan un coste funcional sobre el espacio funcional por la elección de una función iterativa (Hipótesis Débil) que apunta en la dirección del gradiente negativo.

Este punto de vista funcional del “Boosting” lo ha llevado al desarrollo, a impulsar algoritmos en muchas áreas de aprendizaje automático y estadísticas más allá de la regresión y la clasificación.

En conclusión, el método Gradient Boosting representa un algoritmo de aprendizaje automático aplicable en ámbitos muy generales, y con el que se puede obtener un gran rendimiento.

2.6.3.2 Cantidad de categorías

Hay que tener en cuenta, y aunque no sea de aplicación en este proyecto, que para el procedimiento de la aplicación del Gradient Boosting, en la clasificación de problemas, requiere secuencias separadas de los arboles (Boosted Trees), cada una creada para cada categoría o clase.

Por lo tanto, el esfuerzo computacional en general se hace más grande, es decir, se multiplica, más de lo que es necesario para resolver un problema de predicción de regresión simple (para una sola variable dependiente continua).

Por lo tanto no sería acertado este método para analizar las variables dependientes categóricas (variables de clase) con más de, aproximadamente, 100 o más clases. Pasado este máximo, los cálculos realizados pueden necesitar una cantidad razonable de tiempo y esfuerzo.

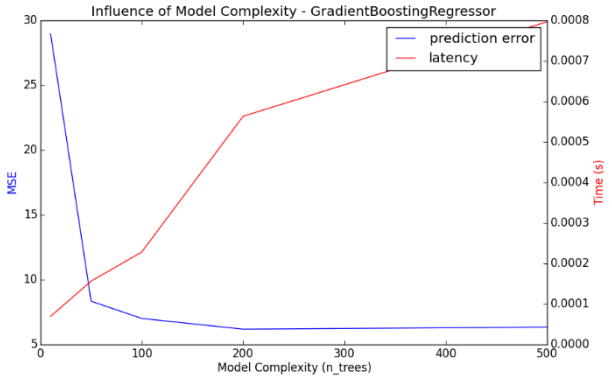


Figura 18 – Demostración cómo la complejidad del modelo influye tanto en la precisión de la predicción y el rendimiento computacional.

2.6.4 XGBoost

2.6.4.1 Descripción General

XGBoost es una implementación del ya conocido algoritmo Gradient Boosting. Este modelo es a veces descrito como una caja negra (blackbox), refiriéndose a que trabaja bien pero no es trivial entender como lo hace. Ciertamente, el modelo está construido por cientos hasta incluso miles de árboles de decisión. Por ello es muy difícil que un ser humano pueda ser capaz de tener una visión general del modelo.

Mientras XGBoost es conocido por sus capacidades de velocidad y precisión en la predicción, tamben es conocido por venir con varias funciones que ayudaran a que sea entendido el funcionamiento.

2.6.4.2 Entendiendo el modelado en XGBoost

Para construir un árbol, el conjunto de dato es dividido recursivamente varias veces. AL final del proceso se habrán obtenido grupos observatorios.

Cada división operativa es llamada *Split* (división).

Cada grupo en cada nivel de división se les llama r*amas*, y el nivel más profundo *hoja*.

En el modelo final, estas *hojas* se supone que deben ser tan puros como sea posible para cada árbol, es decir que cada *hoja* debe ser de una clase, aunque esto no sea logrado, es el objetivo que se debe tratar de conseguir, en un mínimo de divisiones.

No todos los *Split* son igual de importantes. Básicamente la primera división de un árbol tendrá un mayor impacto en la purea que, por ejemplo, la más profunda.

Intuitivamente, se entiende que la primera división hace la mayor parte del trabajo y que las siguientes divisiones se centran en partes más pequeñas del conjunto de datos que han sido clasificadas erróneamente en el primer árbol.

De la misma manera, en Boosting, se intenta optimizar la correcta clasificación en cada ronda, por lo tanto el primer árbol va a hacer la mayor parte del trabajo y lo siguientes árboles se centraran en los trabajos restantes, las partes que no han “aprendido” correctamente en los arboles anteriores.

La mejora que se ha llevado a cabo en cada división puede ser medida, esta es llamada *ganancia*.

Y por último para entender el funcionamiento de XGBoost hay que tener claro que cada división es hecha en una característica a un valor en concreto.

2.7 Selección de Variables para el Modelo Predictivo

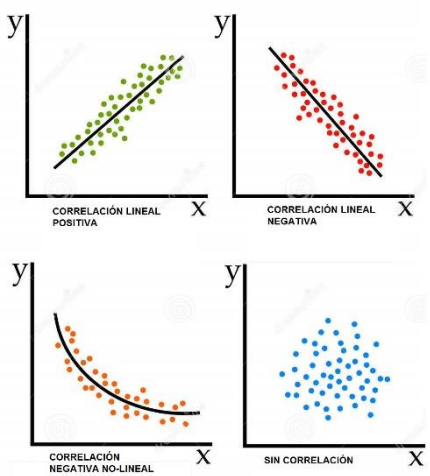
Como variables, en un estudio aeronáutico se pueden encuentrar campos como la puerta de embarque, la hora de llegada, y todos los campos de los parámetros de salida descritos anteriormente. Cada uno de estos puede tener una mayor influencia sobre los otros, y a la vez sobre el resultado final que en este caso sería el retraso de los turnaround. A esto se le llama *correlación*, que en probabilidad y estadística, la correlación indica la fuerza y la dirección de una relación lineal y proporcionalidad entre dos variables estadísticas. Se considera que dos variables cuantitativas están correlacionadas cuando los valores homónimos de la otra también: si hay dos variables (A y B), entonces existe correlación si al aumentar los valores de A lo hacen también los de B y viceversa. La correlación entre dos variables no implica por sí misma, ninguna relación de causalidad.

Una vez son definidos las variables y los métodos de predicción, es necesario crear un sub-conjunto final de variables que podrán ser usadas para las predicciones. Estas deberán ser solo aquellas variables que tengan una alta correlación con la variable dependiente (Y), mientras tenga una relativamente baja correlación con otras variables (X).

2.7.1 Fuerza, sentido y forma de la correlación

La relación entre dos variables cuantitativas queda representada mediante la línea de mejor ajuste, trazada a partir de la nube de puntos. Los principales componentes elementales de una línea de ajuste y, por lo tanto de una correlación son la fuerza, el sentido y la forma:

- La **fuerza** extrema según el caso, mide el grado en que la línea representa a la nube de puntos: si la nube es estrecha y alargada, se representa por una línea recta, lo que implica que la relación es *fuerte*; si la nube de puntos tiene una tendencia elíptica o circular, la relación es *débil*.
- El **sentido** mide la variación de los valores de B con respecto a A: si al crecer los valores de A lo hacen los de B, la relación es directa (pendiente positiva); si al crecer los valores de A disminuyen los de B, la relación es inversa (pendiente negativa).
- La **forma** establece el tipo de línea que define el mejor ajuste: la línea recta, la curva monotónica, o la curva no monotónica.



3. Estudio del Caso

3.1 Entorno y Lenguaje

Primero se procede a instalar el lenguaje, antes que el entorno de programación. Con lo cual se empieza con la configuración de Python.

Tanto para plataformas Mac OS como para Linux y Windows existen dos versiones activas que se pueden usar, la versión 2, y la versión 3, cada una con sus posteriores desarrollos menores.

La principal diferencia entre versiones es su estabilidad y madurez, y compatibilidad por ello será el elegido para el desarrollo del proyecto, debido que al pertenecer al grupo de Versión 2.X.XX, su mayor permanencia en el mercado le otorga cantidad de librerías existentes que pueden ser usadas con esta versión, con esto, facilitando mucho la labor de desarrollo.

En concreto 2.7 es la última versión de la serie 2.X, como los desarrolladores de Python se están centrando en el desarrollo de las serie 3.X, esto significa que la seguirá recibiendo correcciones de errores así como actualizaciones pero no habrá nuevas versiones con funciones completas nuevas.

Sin embargo, aunque hay un gran subconjunto común entre Python 2.7 y Python 3, muchos de los cambios que implica la migración a ese subconjunto común, se puede podrá automatizar de forma segura, es decir si en algún momento se pretende ejecutar el proyecto a través de versiones más nuevas, se podrá hacer sin problema y mantener así el código actualizado.

La instalación no tiene una elevada complejidad y solamente hay que seguir los pasos que sugieren los desarrolladores, es imprescindible instalar a posteriori librerías que serán usadas en el proyecto, como Matplotlib, Numpy, Scipy, etc.

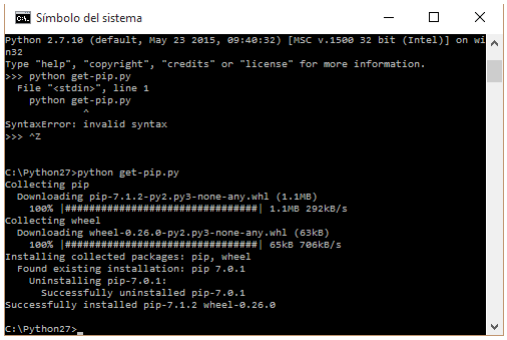


Figura 20 - Instalación Python

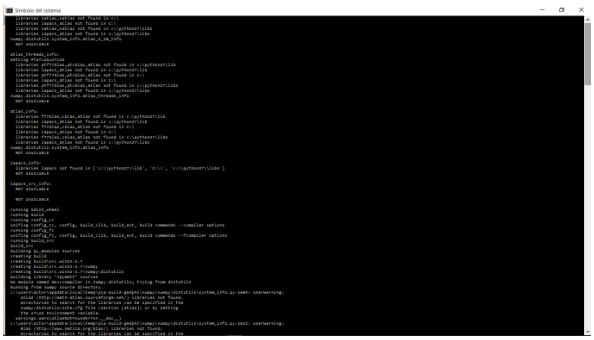


Figura 20 - Instalación Librerías

Para poder programar con facilidad, se procede a instalar PyCharm, que a diferencia de Python que se instala a través de la consola del sistema operativo, el entorno se instalará bajo un instalador ejecutable, de una manera sencilla.

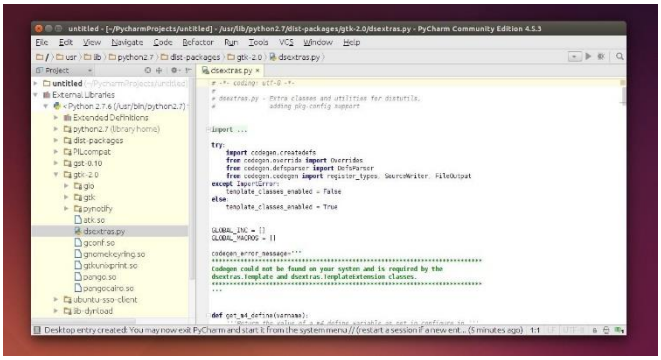


Figura 21 - Entorno PyCharm en Ubuntu

3.2 Obtención de datos

Para la consecución de datos, estos se obtendrán datos de todos los vuelos con destino u origen el Aeropuerto de Barcelona – El Prat, que presenten información relevante para el estudio, de esta manera se logrará conocer el origen de los retrasos en el turnaround.

Dicha información será recolectada de una manera semi-automatizada a través de una interfaz de programación de aplicaciones, o más conocido por su abreviación API *(Application Programming Interface)*, que consta de un conjunto de funciones y procedimientos, con la capacidad de comunicación entre diferentes softwares, evitando así una parte de la programación para la fase de obtención de Datos.

El API que se ha escogido para su utilización en el proyecto, será el de la sección Desarrollador (Developer) de FlightStats.

3.2.1 Sobre FlighStats.

FlightSats es líder en servicios de información de vuelos y aeropuertos a nivel global. Es fácil tener acceso a datos de FlightStats, ya que actualmente cuenta con aplicaciones móviles, web, así como aplicaciones de terceros impulsado por esta.

Los productos más significativos son:

- Estado de los vuelos en tiempo real.
- Seguimiento de vuelos basado en mapas
- Calificaciones de vuelo
- Información de aeropuertos
- Widgets
- XML API
- Aplicaciones de gestión de viajes
- Histórico de información de vuelos

Figura 22 - Logo Proveedor Información FlighStats Developer Center

La selección de este API es por la razón de que hoy en día es el que pone en conocimiento la mayor cantidad de datos, sin coste, sin embargo las cuentas en este proveedor de información caducan a los 30 días, y para la extracción de datos también suele imponer algunas restricciones de ámbito temporal, pero que será remediado para el caso, mediante la programación de código.

3.2.2 Obtención mediante API

Para la realización del estudio, se obtendrán datos de todos los vuelos con destino u origen Barcelona El Prat, que presenten información relevante para el estudio, para así poder extraer de estos, información necesaria para conocer los orígenes de los retrasos en el turnaround y la predicción de estos tiempos.

Dicha información será recolectada de una forma semi-automatizda a través de una interfaz de programación de aplicaciones, o más conocido por su abreviación API (*Application Programming Interface*), que es un conjunto de funciones y procedimientos, con la capacidad de comunicación entre diferentes software, evitando así una gran parte de programación.

Figura 23 - Proceso Alta FlightStats

El API utilizado será el de la sección de Desarrollador (*Developer*) de FlightStats una empresa que aplica estos datos a la visualización online de datos de vuelos, entre otras cosas, la selección de este API es por la razón que hoy en día es el que pone en conocimiento la mayor cantidad de datos, sin coste, sin embargo las cuentas en este proveedor de información caducan a los 30 días, y para la extracción de datos también suele tener algunas restricciones de ámbito temporal, pero que será remediado para el caso, mediante la ejecución de código.

3.2.2.1 Creación de la cuenta:

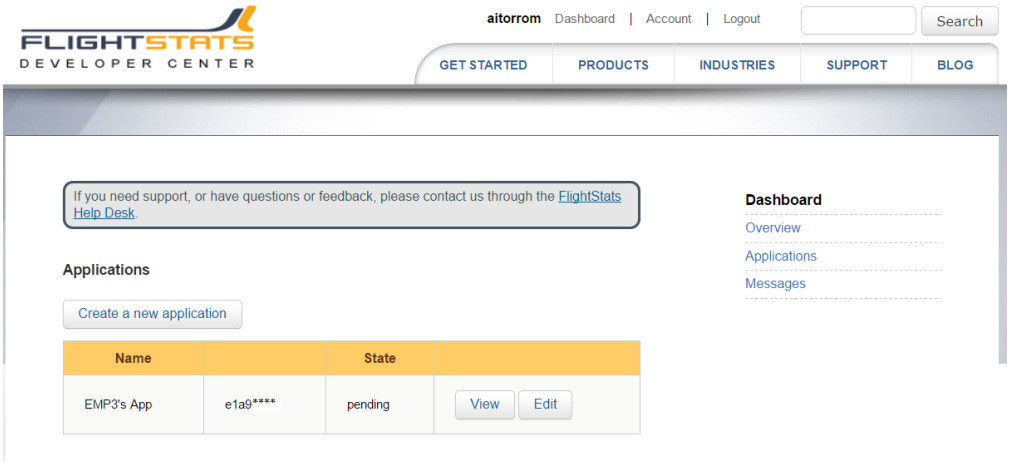


Figura 24 - Captura de Pantalla en el Proceso de Creación de la Cuenta

La información que se recoge, proviene a la vez de varias fuentes, principalmente de proveedores de información aeronáutica, hecho que brinda a la API de potencia, precisión y concisión. EL conjunto de datos será obtenido en formato JSON, ya que se ha considerado que es la idónea para el estudio. JSON es un formato derivado del JavaScript, que destaca por su simplicidad e idoneidad para ser usado como formato de intercambio de datos, así como si estos tienen un gran volumen.

De la cantidad de APIs orientadas a diferentes aspectos de información Aeronáutica, la usada principalmente para la ejecución del proyecto es la de Estado del Vuelo y Ruta (*Flights Status and Track*) esto cómo su nombre indica es desglosado en dos partes, que cada una de ella responde a determinadas preguntas:

Estado del Vuelo:

- ¿El vuelo va según el tiempo previsto?
- ¿De cuánto es el retraso?
- ¿Esta cancelado?
- ¿Cuál es la terminal y la puerta de embarque o de salida?
- ¿Qué tipo de avión es usado en este vuelo?

Estado de la Ruta:

- ¿Cuál es el camino planificado para este vuelo?
- ¿Dónde está actualmente?
- ¿Qué recorrido ha hecho?

De estas cuestiones se puede comprobar que este API de Flightstats brinda acceso a información de vuelo actual, que incluye los tiempos programados, estimados, y actuales, respecto salidas/llegadas, el equipo usado (Tipos de Avión), cálculos de retraso, información sobre la terminal, puertas, y cintas de recogida de equipaje.

Dentro de la API de Estado de Vuelo y Ruta, se pueden clasificar varios puntos de entrada de los cuales la información será clasificada de una forma u otra según el foco de interés, de los que ofrece, para hacer un estudio centrado en los tiempos de escala (*turnaround*), *Flight Track by Airport* es el que encaja, ya que no tiene en cuenta la totalidad del vuelo, sino el paso por el aeropuerto, esta información se extrae dentro del API Flight Status & Track, en la subdivisión airport/tracks.

3.1.1.1 Parámetros de entrada

Para que FlightStats pueda generar la información correcta, será necesario especificar unos parámetros para seleccionar los datos de interés.

Estos parámetros son los siguientes:

appId/appKey: El identificador y su clave, obtenidos previamente en el alta del servicio

airport: En este caso LEBL, ya que es el escogido para el estudio

year/month/hourofday: Respecto al momento del cual se quiere conocer datos

numHours: Cantidad de horas de la que se quiere conocer información, aquí radica una de las principales restricciones que plantea el servicio ya que el máximo son 6, pero es fácil mediante el cambio iterativo dentro el código

carrier: En el supuesto que se quiera filtrar por Operador Aéreo

codeType: Filtrado por códigos de vuelo

maxFlights: Cantidad máxima de vuelos que serán devueltos

extendedOptions: Opciones extra para modificar el comportamiento de la API

3.1.1.2 Parámetros de Salida

Los parámetros que contiene la información que es generada por el API, deben ser conocidos ya que de esos, posteriormente se obtendrán los cálculos, resultados, conclusiones y con ellos se realizarán trabajos para que la información sea más clara y concisa.

Estos parámetros son:

AIRPORT_RESOURCES.BAGGAGE Localización de la recogida del equipaje

AIRPORT_RESOURCES.ARRIVAL_TERMINAL La terminal, la cual el vuelo ha llegado o llegará

ARRIVAL_AIRPORT_FS_CODE Código de FlightStats para el aeropuerto de llegada

ARRIVAL_DATE.DATE LOCAL La fecha de llegada del vuelo en hora local y UTC. Este valor puede ser publicado también en otros como PUBLISHED_ARRIVAL o SCHEDULED_GATE_ARRIVAL.

CARRIER_FS_CODE Código único para identificar a la compañía aérea operadora del vuelo

DELAYS.ARRIVAL_GATE_DELAY_MINUTES Cálculo del retaso de llegada a la puerta (en minutos)

DELAYS.ARRIVAL_RUNWAY_DELAY_ MINUTES Cálculo del retraso de llegada a la pista (en minutos)

FLIGHT_DURATIONS.SCHEDULED_BLOCK_MINUTES Cálculo del tiempo programado entre bloques (puerta a puerta) (en minutos)

FLIGHT_DURATIONS.BLOCK_MINUTES Tiempo calculado entre bloques (puerta a puerta) (en minutos). Siempre y cuando esté disponible, de lo contrario, será una estimación precisa.

FLIGHT_DURATIONS.SCHEDULED_TAXI_IN_MINUTES Cálculo de la hora programada para el aterrizaje con el rodaje de entrada (Taxi In), es decir, desde la pista hasta la puerta (en minutos)

FLIGHT_DURATIONS.TAXI_IN_MINUTES Tiempo calculado para el aterrizaje con el rodaje de entrada (Taxi In), es decir, desde la pista hasta la puerta (en minutos), en caso de no ser conocido, será una estimación precisa.

FLIGHT_EQUIPMENT.SCHEDULED_EQUIPMENT_IATA_CODE Código IATA para el equipo programado para que realice el vuelo, como referencia para ser usado en el apéndice de IATA

FLIGHT_EQUIPMENT.TAIL_NUMBER Número de matrícula de la Aeronave que realiza el vuelo

FLIGHT_ID Código único que genera FlighStats para cada vuelo, que no hay que confundir con el número de vuelo

FLIGHT_NUMBER El número de identificación del vuelo así como los caracteres adicionales.

OPERATIONAL_TIMES.ACTUAL_GATE_ARRIVAL La hora real observada en la llegada a la puerta de la terminal.

OPERATIONAL_TIMES.ACTUAL_RUNWAY_ARRIVAL La hora real observada en la llegada a la pista de aterrizaje.

OPERATIONAL_TIMES.ESTIMATED_GATE_ARRIVAL Hora prevista según las observaciones, para la llegada a la puerta de la terminal

OPERATIONAL_TIMES.ESTIMATED_RUNWAY_ARRIVAL Hora prevista según las observaciones, para la llegada a la pista de aterrizaje.

OPERATIONAL_TIMES.PUBLISHED_ARRIVAL La hora publicada por la compañía aérea, de la llegada del vuelo.

OPERATIONAL_TIMES.SCHEDULED_GATE_ARRIVAL Hora de llegada prevista a la puerta de la terminal, es la misma hora que la hora publicada de llegada, pero este campo puede ser modificado si se detecta algún cambio.

SCHEDULE.FLIGHT_TYPE El tipo de servicio programado para el vuelo.

J	Scheduled Passenger(Normal Service)
S	Scheduled Passenger(Shuttle Service)
U	Scheduled Passenger(ServiceVehicle)
F	Scheduled Cargo/Mail(Loose loaded cargo and/or preloaded devices)
V	ScheduledCargo/Mail(Surface Vehicle)
M	Scheduled Cargo/Mail(MailOnly)
Q	Scheduled Passenger/Cargo inCabin
G	Non-scheduled Passenger(NormalService)
B	Non-scheduled Passenger(ShuttleService)
A	Non-scheduled Cargo/Mail
C	Charter(Passenger Only)
O	Charter(Special handling-Migrants/Immigrants)
H	Charter(Cargo and/or Mail)
L	Charter(Passenger and Cargo and/or Mail)
P	Non-revenue
T	Technical Test
K	Training
D	General Aviation
E	Special (FAA/Government)
W	Military
R	Additional Flights - Passenger/Cargo
Y	IATA Special Internal(Y)
Z	IATA Special Internal(Z)

Figura 25 - Tipo de Servicio Programado

SCHEDULE.SERVICE_CLASSES Tipos de clases de servicio para el vuelo (IATA)

A	Active
C	Canceled
D	Diverted
DN	Data source needed
L	Landed
NO	Not Operational
R	Redirected
S	Scheduled
U	Unknown

Figura 26 - Estado Actual del Vuelo

STATUS El estado actual del vuelo.

3.2.3 Estructuración de Datos

Una vez obtenidos los datos, estos están repartidos en dos archivos diferentes para un mismo aeropuerto (LEBL), estos archivos son:

- ***Flights_arrivals.csv***
- ***Flights_departures.csv***

El tipo de archivo es CSV (*Comma-Separated Values*), que almacena datos tabulados (números y textos) de una forma plana. Cada línea del archivo es un registro. Este registro está divido en varios campos que son separados a través de comas, en esto radica el nombre del archivo.

Gestionar archivos es fácil, y se puede crear una visualización rápida con programas como por ejemplo Excel que incluyen funcionalidades de tabulaciones delimitadas mediante comas.

El código ejecutado para reunir la información desde el API, es el que se muestra a continuación:

```
flights_arrivals = pd.read_csv("flights_arrivals.csv")
flights_departures = pd.read_csv("flights_departures.csv")
```

Figura 27 - Lectura de Archivos CSV

3.2.3.1 La fusión

Para conseguir datos relativos a los tiempos de escala (*Turnaround*), los datos de llegada de un vuelo se deben fusionar con los datos de salida de otro vuelo, esta fusión es una parte clave del estudio, ya que en la actualidad no existe una entrada independiente en FlightStats para la realización de estas consultas.

Se plantean distintos problemas a la hora de los parámetros usados en la fusión, ya que pocos campos identifican cual será el vuelo siguiente del Turnaround.

Por un lado y si la información es consistente el vuelo de salida, debe contener los mismos valores en **FLIGHT_EQUIPMENT.SCHEDULED_EQUIPMENT_IATA_CODE** de llegada que de salida, pero la gran mayoría de los parámetros serán diferentes, incluso aquellos que la lógica dicta que deben ser los mismos, como puede ser la puerta de embarque, ya que como ejemplo la llegada puede ser mediante pasarela, mientras que la salida puede ser en posición remota.

Los parámetros que se han considerado idóneas para la fusión de las dos tablas, son los siguientes:

FLIGHT_EQUIPMENT.TAIL_NUMBER Un Turnaround siempre involucre el mismo avión, y para que este sea exactamente el mismo, el único valor que no puede variar es la Matricula de la aeronave.

Para recolectar la información de los dos archivos CSV en una, dónde el punto en común es el TAIL_NUMBER, se procede con el siguiente código:

```
flights = flights_arrivals[column_names_arrivals].merge\
    (flights_departures[column_names_departures], on='flightEquipment.tailNumber')
```

Figura 28 - Unión Archivos CSV en uno

OPERATIONAL_TIMES Dentro de los tiempos operacionales, unidos por la misma matricula, será el vuelo de salida, aquel que con la misa matrícula tenga el tiempo de salida más próximo al de llegada.

Para lograr esto, se emplea el siguiente código:

```
flights_arrs = []
for index, row in flights_arrivals.iterrows():
    result = find_nearest(row,flights_departures,'flightEquipment.tailNumber')
    if len(result) > 0:
        flights_arrs.append(result)

flights_arrs = pd.DataFrame(flights_arrs)

flights = pd.merge(flights_arrs, flights_departures, \
                   on=['flightEquipment.tailNumber', 'operationalTimes.actualDateTime'])
```

Figura 29 - Vinculación Vuelo LLEGADA-SALIDA

El inconveniente que puede generar esta fusión y que hay que tener en cuenta es si el avión al llegar, su intención primaria no sea aquella que la de partir. Esto se contempla en vuelos donde posteriormente se realiza un mantenimiento prolongado, o es el último vuelo del día, y el avión descansa durante horas.

3.2.3.2 Datos No Disponibles

Unos resultados óptimos requieren de información completa y limpia.

Por un lado hay información, que a pesar de ser inexistente, por su naturaleza, puede ser trivial para el estudio, o por el contrario, de ser necesario, no se podrá utilizar un registro que omita alguno de los parámetros. El caso de la matrícula de la Aeronave (*Tail Number*) es totalmente necesario para el estudio, por ello se eliminaran todas las entradas de los dos archivos (previo a la fusión) para suprimir estas entradas, cuyo campo Tail Number no exista.

```
flights_arrivals = flights_arrivals.dropna(subset=['flightEquipment.tailNumber'])
flights_departures = flights_departures.dropna(subset=['flightEquipment.tailNumber'])
```

Por otro lado existe la posibilidad de información de la cual no se tienen datos de algunos de los registros pero se puede hacer una estimación, a través de una interpolación, dónde el método escogido y el que da mejores resultados es la interpolación lineal, con esto se consigue llenar campos vacíos e inconsistencias de información para poder realizar un estudio, obteniendo una fiabilidad más alta.

En un

```
flights[times_column_names] = flights[times_column_names].interpolate(method="linear")
```

tercer lugar se encuentran aquellos campos en las columnas de retraso, que por el hecho de no tener retraso o desconocerlo, estos aparecen en blanco.

Para el correcto funcionamiento de los algoritmos, es necesario que estos datos inexistentes se transformen en ceros, para ello previamente se definen las columnas que interese, en este caso las relacionadas con los retrasos. Con la siguiente línea de código, se definen las columnas y se rellenan las casillas correspondientes con los ceros necesarios:

```
delay_columns = [x for x in flights.columns.values.tolist() if
x.startswith("delays")]
flights[delay_columns] = flights[delay_columns].fillna(0)
```

Figura 30 - Definición Columnas y Completado de '0'

Y por último, el resto de filas que no serán de utilidad para el análisis, se eliminaran con el siguiente comando:

```
flights = flights.dropna()
```

Figura 31 - Eliminación Registros Inservibles

Una comprobación que hay que realizar posteriormente para cerciorarse de que no exista contenido con valores vacios, es el siguiente:

```
print flights.isnull().any()
```

Que imprime todos los atributos, e indica si hay alguno con campos vacíos, esta función retorna valores booleanos con lo cual se obtendrá un *True* en el caso de que aún reste algún valor incompleto, en caso correcto y contrario, aparecerá algo similar a la siguiente salida de Python:

```
…
airportResources.baggage_x                                  False
airportResources.arrivalTerminal_x                          False
delays.arrivalGateDelayMinutes_x                            False
delays.arrivalRunwayDelayMinutes_x                          False
…
```

Figura 32 - Comprobación Campos Vacíos

3.3 Análisis de los Datos

Una vez obtenida toda la información, esta contendrá datos semi-limpios, el propósito entonces será sacar conclusiones de ella, para ello será requerido un proceso de inspección, transformación de datos, con la meta de obtener información útil, que será práctica para la posterior toma de decisiones, conclusiones del estudio, entre otros.

Una buena forma de visualizar los datos es directamente en el archivo CSV, donde registro por registro se pueden examinar cada uno de los datos de cada vuelo, y así comprender ya en los datos limpios y íntegros, sus valores y extraer conocimiento de ellos.

Para la óptima visualización de los datos en necesario una hoja de cálculo que tabule los datos separados por comas, mejorando notoriamente la disposición de los datos así como su visualización.

	airportResources.baggage_x	airportResources.arrivalTerminal_x	arrivalAirportFsCode_x	carrierFsCode_x	delays.arrivalGateDelayMinutes_x	delays.arrivalRunwayDelayMinutes_x	flightDurations.blockMinutes_x	flightDurations.scheduledBlockMinutes_x	flightDurations.scheduledTaxiInMinutes_x	flightDurations.taxiInMinutes_x	flightEquipment.scheduledEquipmentIataCode_x	flightEquipment.tailNumber_x	flightId_x	flightNumber_x	schedule.flightType_x	schedule.serviceClasses_x	status_x	opera
0	1	BCN	AA	7	21	478	465	21	7	763	N342AN	608864020	66	J	RJY	L	400762	
14	1	BCN	VY	17	0	299	275	16.94118	6	320	EC-LRM	609397127	7845	J	RFJY	L	400842	
14	1	BCN	VY	0	0	274	265	16.88235	3	320	EC-MFM	609393778	7001	J	RFJY	L	400822	
11	1	BCN	VY	0	0	274	275	16.82353	4	320	EC-LZE	609336165	7767	J	RFJY	L	400837	
9	1	BCN	VY	0	0	47	50	16.58824	3	320	EC-KHN	609382264	3915	J	RFJY	L	400782	
10	1	BCN	VY	0	0	57	65	16.52941	2	320	EC-HGZ	609317996	1431	J	RFJY	L	400796	
8	1	BCN	VY	0	0	64	70	16.47059	6	320	EC-LQM	609311711	1311	J	RFJY	L	400798	
8	1	BCN	VY	0	0	81	90	16.41176	2	320	EC-KJD	609394100	2223	J	RFJY	L	400820	
11	1	BCN	VY	1	0	95	95	16.35294	5	320	EC-MBE	609361168	1293	J	RFJY	L	400836	
3	1	BCN	VY	0	0	91	95	16.29412	5	320	EC-LRA	609377857	8007	J	RFJY	L	400821	
11	1	BCN	VY	26	0	167	135	16.23529	5	320	EC-HQJ	609312162	8318	J	RFJY	L	400901	
3	1	BCN	OS	17	0	157	135	16.17647	11	320	OELBK	609400925	391	J	FJY	L	400892	
6	1	BCN	VY	3	0	193	190	16.11765	5	321	EC-MGY	609396302	3215	J	RFJY	L	400993	
9	1	BCN	VY	0	0	91	95	16.05882	5	320	EC-JSY	609387554	1676	J	RFJY	L	400833	
9	1	BCN	VY	0	0	103	110	16	3	320	EC-LVU	609341381	6109	J	RFJY	L	400858	
9	1	BCN	VY	0	0	82	85	15.82353	4	320	EC-LML	609378872	1572	J	RFJY	L	400848	
6	1	BCN	AF	10	0	105	100	15.76471	6	321	FGTAE	609324741	1148	J	RFJY	L	400890	
10	1	BCN	VY	0	0	72	75	15.70588	6	320	EC-JGM	609365879	1022	J	RFJY	L	400871	
8	1	BCN	VY	7	0	95	90	15.64706	5	320	EC-KCU	609394095	2253	J	RFJY	L	400897	
8	1	BCN	VY	0	0	87	95	15.58824	3	320	EC-LRY	609311035	2126	J	RFJY	L	400878	
8	1	BCN	VY	0	0	60	65	15.52941	3	320	EC-MCU	609317995	1439	J	RFJY	L	400870	
6	1	BCN	VY	9	0	59	50	15.41176	7	320	DABFB	609382256	3903	J	RFJY	L	400884	
10	1	BCN	VY	1	0	58	55	15.35294	3	320	EC-MDZ	609351656	3507	J	RFJY	L	400901	
9	1	BCN	VY	1	0	75	70	15.29412	3	320	EC-KDX	609311710	1301	J	RFJY	L	400916	
10	1	BCN	VY	13	0	88	75	15.23529	5	320	EC-MEQ	609365881	1002	J	RFJY	L	400943	

Figura 33 - Datos Archivo 'Flights' Tabulados

3.4 Análisis Visual

Para una correcta y amigable visualización de los datos, son necesarias herramientas que nos lo permitan, para ello, se utilizarán varias librerías de Python. Las que destacan por su propósito e importancia en este campo, y que por ello serán empleadas son las siguientes:

- **Seaborn:** Es una librería de visualización. Proporciona una interfaz de alto nivel para la elaboración de atractivos gráficos estadísticos.
- **Maptolib:** Es una librería para la creación de gráficos 2D, y que produce figuras para impresiones y para entornos interactivos.

Con la importación de estas librerías y la ejecución de código para la transformación de la información almacenada en los archivos CSV, en información agrupada y conveniente, se realizaran los siguientes gráficos, de los cuales podremos sacar información valiosa.

3.4.1 Elaboración Gráficos

3.4.1.1 Retrasos en el Turnaround por Modelo de Avión

En el grafico que se muestra, se puede apreciar como por un lado los aviones con fuselaje ancho y alta capacidad de pasajeros, tienden a sufrir más retrasos en el turnaround que los de fuselaje estrecho, como sucede con los 757-300, 767-400 o 330-200, cosa que no sucede con las aeronaves de media capacidad y principalmente operadas por compañías low-cost donde prima la rentabilidad, como son los modelos A320 (Vueling), B737 (Ryanair), que son los que registran los retrasos menores, en concreto los Boeing 737.

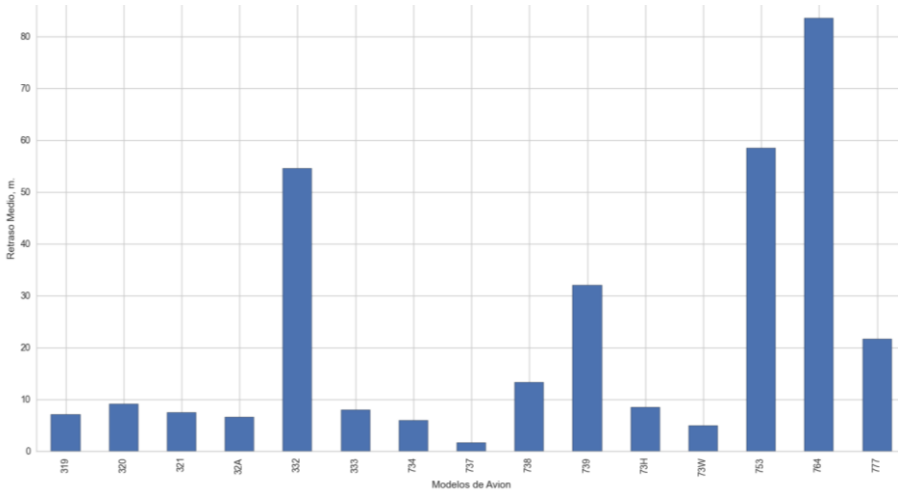


Figura 34 - Gráfico Retraso Turnaround por "Modelo de Avión"

Por otro lado se aprecia que excepto el puntual caso del modelo del fabricante americano operado por Ryanair, la gran mayoría de Boeing son propensos a sufrir mayores retrasos como se valora en la mitad derecha de la gráfica.

```
data_equipment['flightEquipment.scheduledEquipmentIataCode_x'] =
temp_flights_equipment['flightEquipment.scheduledEquipmentIataCode_x']
data_equipment['delays.departureGateDelayMinutes'] =
temp_flights_equipment['delays.departureGateDelayMinutes']
equipment_group = data_equipment.groupby('flightEquipment.scheduledEquipmentIataCode_x')
delays_totals = equipment_group.mean()
delays_totals.sort(columns='delays.departureGateDelayMinutes')
ax = delays_totals.plot(kind='bar', title ="Restrasos por Modelo de Avion",legend=False)
```

Figura 35 - Código Gráfico Retraso Turnaround por "Modelo de Avión"

3.4.1.2 Retrasos en el Turnaround por Compañías

Se pueden distinguir las aerolíneas en tres bloques según sus retrasos medios de turnaround, encabezando el bloque con las compañías que más retrasos sufren, entorno a los 60 minutos y los 90, se encuentran las siguientes:

4U – Germanwings

IZ – Arkia Israel Airlines

LY – Israel Airlines

UA – United Airlines

Dentro un Segundo bloque, con unos retrasos medios que oscilan entre los 20 y los 40 minutos, se encuentran, las siguientes compañías:

AA – American Airlines

TU – Tunisair

UX – Air Europa

UX – Air Europa

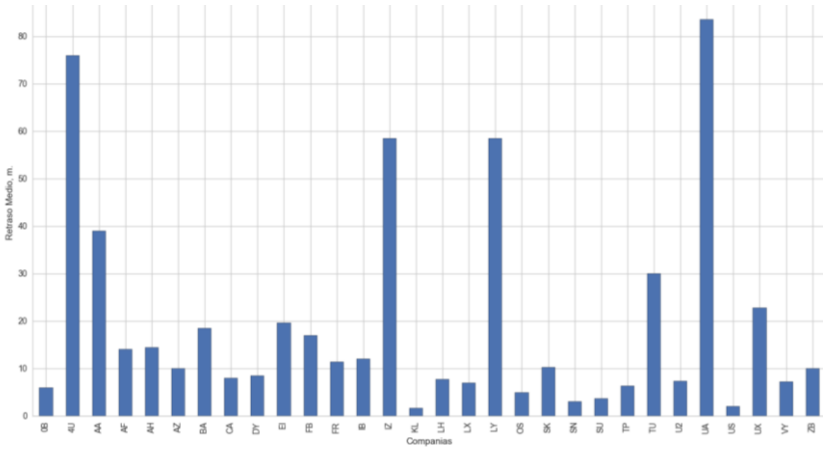


Figura 36 - Gráfico Retrasos Turnaround por "Compañías"

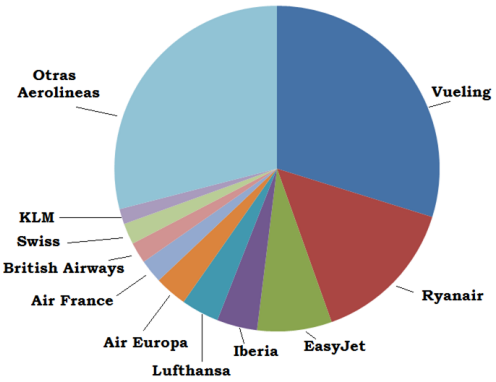


Figura 37 - Porcentajes Compañías BCN

Por último el bloque donde todas las demás compañías querrían estar es este tercero con retrasos cercanos a la puntualidad, donde se puede apreciar algunas con pocos minutos como KLM con una media de retraso de unos escasos 2-3 minutos:

KL – KLM Airlines

SN – Brussels Airlines

SU – Aeroflot

US – US Airlines

```
temp_airline = pd.DataFrame()
temp_airline['carrierFsCode_x'] = flights['carrierFsCode_x']
temp_airline['delays.departureGateDelayMinutes'] = flights['delays.departureGateDelayMinutes']
temp_airline = temp_airline.dropna()

data_airline = pd.DataFrame()
data_airline['carrierFsCode_x'] = temp_airline['carrierFsCode_x']
data_airline['delays.departureGateDelayMinutes'] = temp_airline['delays.departureGateDelayMinutes']
airline_group = data_airline.groupby('carrierFsCode_x')
delays_totals_airline = airline_group.mean()
delays_totals_airline.sort(columns='delays.departureGateDelayMinutes')
ax = delays_totals_airline.plot(kind='bar', title ="Retrasos por Companias",legend=False)
ax.set_xlabel("Companias")
ax.set_ylabel("Retraso Medio, m.")
plt.show()
```

Figura 38 - Código Gráfico Retrasos Turnaround por "Compañías"

3.4.1.3 Retrasos en el Turnaround por Puerta de Embarque

De una ojeada se puede apreciar que de las 80 puertas de embarque solo unas pocas experimentan retrasos diferentes al resto, y esto comprobado con los ficheros que genera el código Python confirma que son casos de retrasos puntuales.

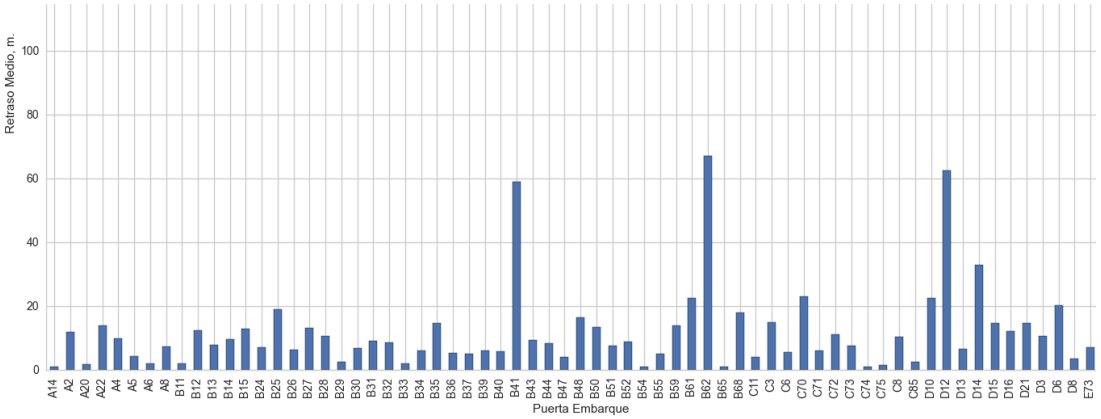


Figura 39 - Gráfico Retrasos Turnaround por "Puerta de Embarque"

Las demás puertas de embarque siguen un patrón similar que va de los pocos minutos de retraso, a prácticamente 15 minutos, siendo superados en raras ocasiones, esto junto el gran número de puertas hace que sea difícil justificar una tendencia de retraso debido a estas como parámetro de influencia.

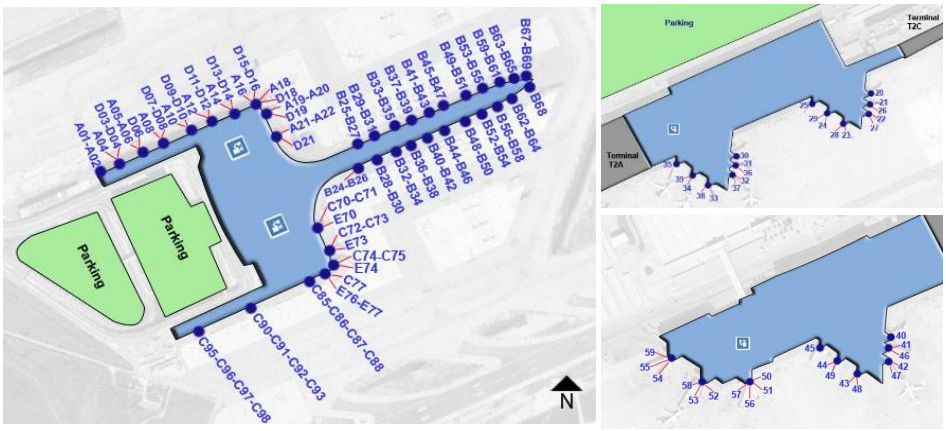


Figura 40 - Esquema Puertas Embarque BCN

```
temp_gate = pd.DataFrame()
temp_gate['airportResources.departureGate'] = flights['airportResources.departureGate']
temp_gate['delays.departureGateDelayMinutes'] = flights['delays.departureGateDelayMinutes']
temp_gate = temp_gate.dropna()

data_gate = pd.DataFrame()
data_gate['airportResources.departureGate'] = temp_gate['airportResources.departureGate']
data_gate['delays.departureGateDelayMinutes'] = temp_gate['delays.departureGateDelayMinutes']
gate_group = data_gate.groupby('airportResources.departureGate')
delays_totals_gate = gate_group.mean()
delays_totals_gate.sort(columns='delays.departureGateDelayMinutes')
ax = delays_totals_gate.plot(kind='bar', title ="Restrasos por Puerta Embarque",legend=False)
ax.set_xlabel("Puerta Embarque")
ax.set_ylabel("Retraso Medio, m.")
plt.show()
```

Figura 41 - Código Gráfico Retrasos Turnaround por "Puerta de Embarque"

3.4.1.4 Retrasos en el Turnaround por Duración de Vuelo

Los retrasos del turnaround, tal y como se interpreta en la gráfica, tienen una gran correspondencia con la duración de los vuelos, pudiendo deducir fácilmente que a mayor duración, mayor retraso, y efectivamente así es como sucede.

El agujero creado por los vuelos con unas duraciones aproximadas de 6-7 horas, o lo que es lo mismo de unos 360 a 420 minutos, no representan una buena eficacia y una inexistencia de estos retrasos, sino la falta de vuelos que operan con estos rangos de tiempo.

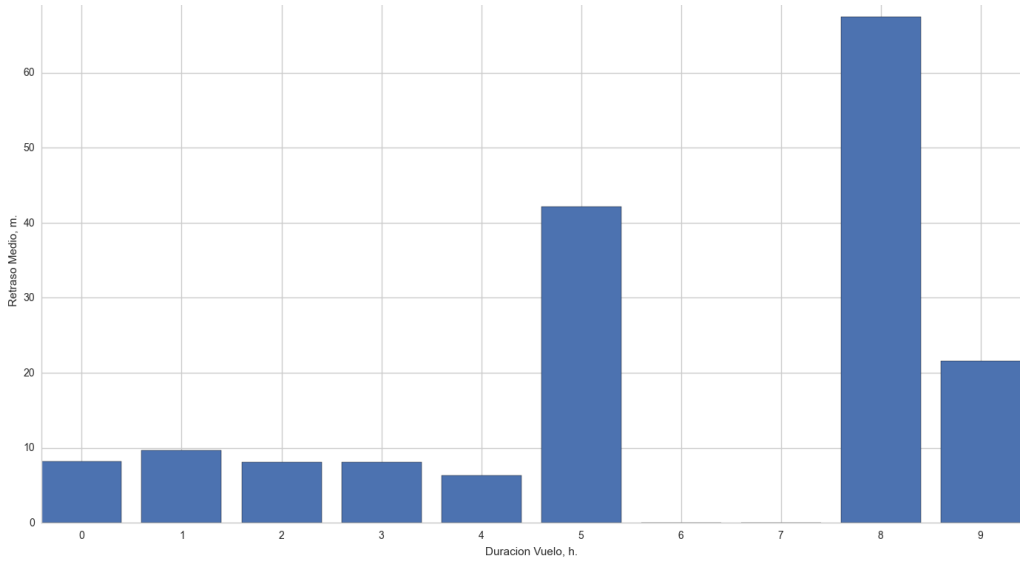


Figura 42 - Gráfico Retrasos Turnaround por "Duración Vuelo"

Para visualizar estos aspectos, se requiere del siguiente gráfico, pero antes de ello comprobar que un vuelo de entre 5 y 9 horas tiene un retraso en el turnaround hasta 6-7 veces mayor que los vuelos inferiores a estos tiempos, donde se sitúan todos muy a la par en un rango de 6 a 10 minutos.

Como se ha comentado, en la gráfica se muestra la asiduidad de los vuelo respecto sus duraciones, como cabe esperar de un aeropuerto como el de Barcelona son vuelos con unos tiempos de recorrido que se concentran en el rango de los 45 minutos a 145 minutos, ya que la gran mayoría de estos vuelos tienen como destino ciudades europeas.

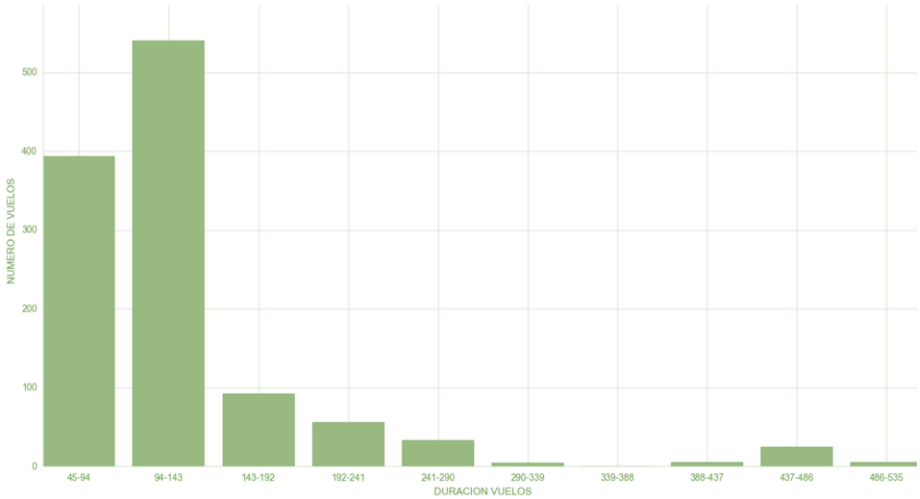


Figura 43 - Histograma Duración Vuelos

A partir de los 145 minutos de duración de vuelo, la cantidad de estos desciende drásticamente, hasta llegar a no haber vuelos en unos rangos de duración de entre 290 y 430 minutos.

A partir de los 430 minutos, en la gráfica se encuentra un ligero ascenso en las duraciones de entre 430 y 490, para luego volver a desaparecer los vuelos de estas duraciones, el ascenso puntual, que representan las 7-8 horas de vuelo es debido a la cantidad medianamente elevada de vuelos a Norte América, donde predominan vuelos a Nueva York, Chicago, Newark, todos ellos obviamente sin escalas. Con esta información complementaria al grafico anterior,

```
temp_duration = pd.DataFrame()
temp_duration['flightDurations.blockMinutes_x'] = flights['flightDurations.blockMinutes_x']
temp_duration['delays.departureGateDelayMinutes'] = flights['delays.departureGateDelayMinutes']
temp_duration = temp_duration.dropna()

barchart(temp_duration['flightDurations.blockMinutes_x'].values,temp_duration['delays.departure
GateDelayMinutes'].values,"Duracion Vuelo, h.", "Retraso Medio, m.")
```

Figura 44 - Código Gráfico Retrasos Turnaround por "Duración Vuelo"

extraemos los destinos principales con mayor atraso en el turnaround.

Los códigos usados para el gráfico de los retrasos según la duración de vuelo es el que crea un gráfico de barras, mientras que para el que se ha usado solamente con fines de saber el número de vuelos respecto su duración, sin tener en cuenta los retrasos, se ha creado código para que cree un histograma y así poder ver de un simple vistazo qué duraciones predominan y qué duraciones escasean. Pasajeros Totales Barcelona por Área Geográfica

```
histplot(flights['flightDurations.scheduledBlockMinutes_x'],'DURACION VUELOS','NUMERO DE VUELOS')
```

Figura 45 - Código Histograma Duración Vuelos

Como se puede apreciar en la siguiente gráfica con datos históricos, se ve como queda distribuido el total de pasajeros del aeropuerto de Barcelona según el destino, esto junto a la variación interanual. Todo ello tiene una gran relación con los gráficos y datos expuestos antes, como son las duraciones y frecuencia de los vuelos.

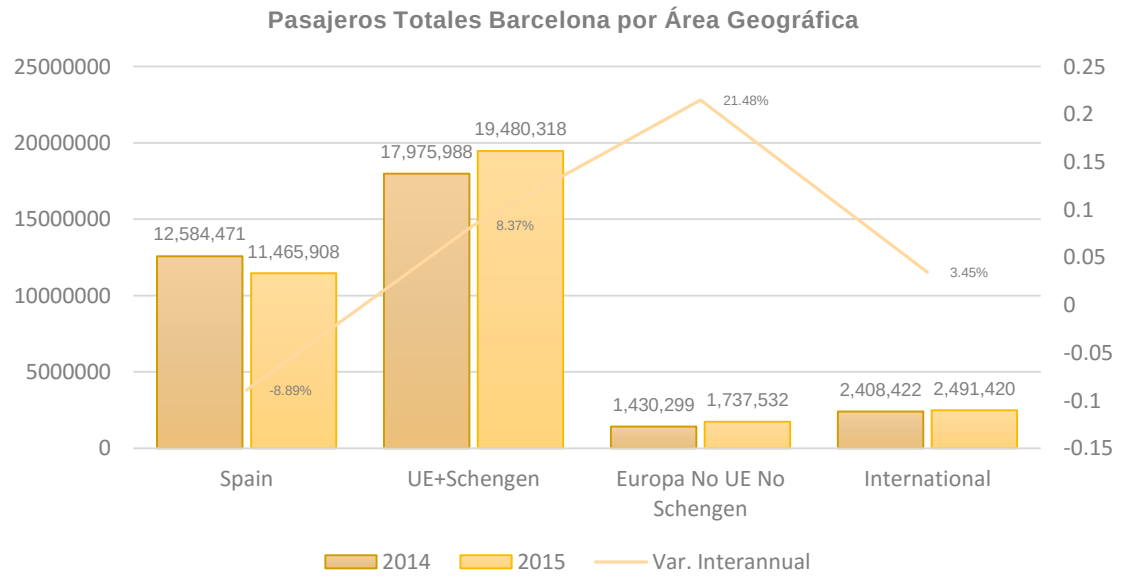


Figura 46 - Pasajeros Totales Barcelona por Área Geográfica

3.4.1.5 Retrasos en el Turnaround por Clases

Para entender el siguiente gráfico, hay que entender los diferentes tipos de clases que puede ofrecer un operador para un vuelo, estos los describe IATA (International Air Transport Association) bajo la siguiente clasificación:

- **R:** Primera Clase Premium (Suite, No usada con este fin)
- **F:** Primera Clase
- **J:** Clase Business
- **Y:** Clase turista o económica

IATA también recoge otros tipos de clases pero que no son utilizados por el API de Flightstats.

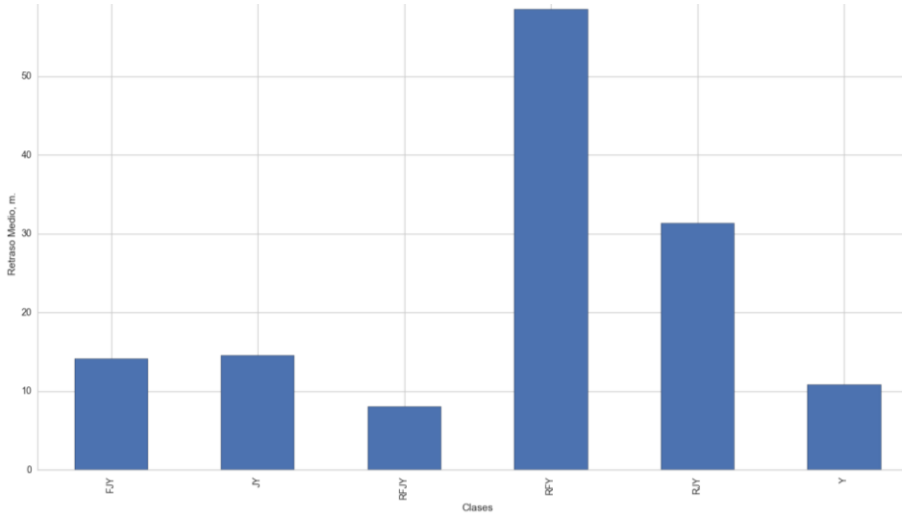


Figura 47 - Gráfico Retrasos Turnaround oir "Clases"

Existe una contradictoriedad con la Clase R, ya que es utilizada para servicio Suite de Primer Clase Premium, solo ofrecidas en pocas situaciones por compañías como Emirates y con aviones A380 que no operan en esta configuración en Barcelona por el momento. Y de tal manera es usada la Clase R por compañías que añaden un servicio extra como es el caso de

```
temp_class = pd.DataFrame()
temp_class['schedule.serviceClasses_x'] = flights['schedule.serviceClasses_x']
temp_class['delays.departureGateDelayMinutes'] = flights['delays.departureGateDelayMinutes']
temp_class = temp_class.dropna()

data_class = pd.DataFrame()
data_class['schedule.serviceClasses_x'] = temp_class['schedule.serviceClasses_x']
data_class['delays.departureGateDelayMinutes'] =
temp_class['delays.departureGateDelayMinutes']
class_group = data_class.groupby('schedule.serviceClasses_x')
delays_totals_class = class_group.mean()
delays_totals_class.sort(columns='delays.departureGateDelayMinutes')
ax = delays_totals_class.plot(kind='bar', title ="Restrasos por Tipos de
Clases",legend=False)
ax.set_xlabel("Clases")
ax.set_ylabel("Retraso Medio, m.")
plt.show()
```

Figura 48 - Código Gráfico Retrasos Turnaround por "Clases"

Prioridad al Embarque entre ellos, servicios característicos de las compañías Low cost.

Una vez esclarecido la clasificación podemos ver que dejando a un lado la Clase R, los retrasos del turnaround son mayores cuando existen clases superiores como la F, seguido de la J, ya que en vuelos donde solo existe la clase turista estos retrasos se ven disminuidos.

3.4.1.6 Retrasos en el Turnaround por Terminales

A Como resultado del fuerte crecimiento, en 2009 Barcelona amplió el aeropuerto con el fin de proporcionar un mejor servicio a más pasajeros.

Se construyó una nueva terminal (Terminal 1) a 4 km de la Terminal 2. La nueva Terminal 1 actualmente opera el 70% de los vuelos. Las antiguas terminales A, B y C, son ahora la Terminal 2, que opera aproximadamente el 30% de los vuelos.

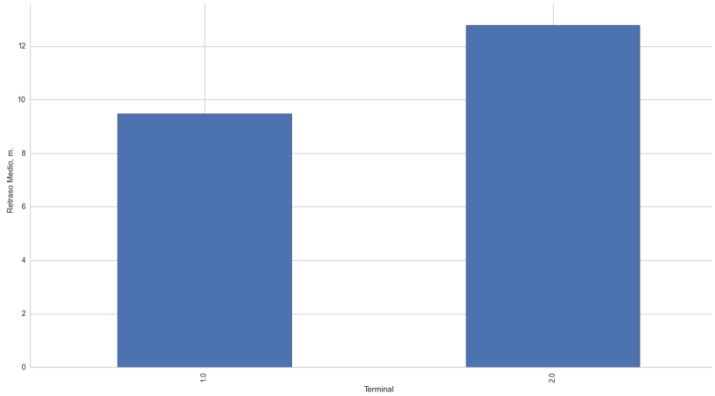


Figura 49 - Gráfico Retrasos Turnaround por

La actual organización de la nueva Terminal 1, así como su anterior planificación, ha hecho que aunque la diferencia de los retrasos en el turnaround de una a otra no sean ingentes, sí que se pueda observar que a pesar de que la Terminal 1 opera hasta el 70% de los pasajeros del aeropuerto, sufre unos retrasos en el turnaround de aproximadamente un 20% inferiores al de la Terminal 2.

Este descenso de los retrasos en el turnaround se consiguen gracias a que la Terminal 1, opera con unos índices de saturación muy bajos respectos a los que se ha acostumbrado a trabajar en la T2, ya que con la nueva terminal, el aeropuerto alcanzó la capacidad de manejar 55 millones de pasajeros al año y puede llegar hasta 90 operaciones a la hora.

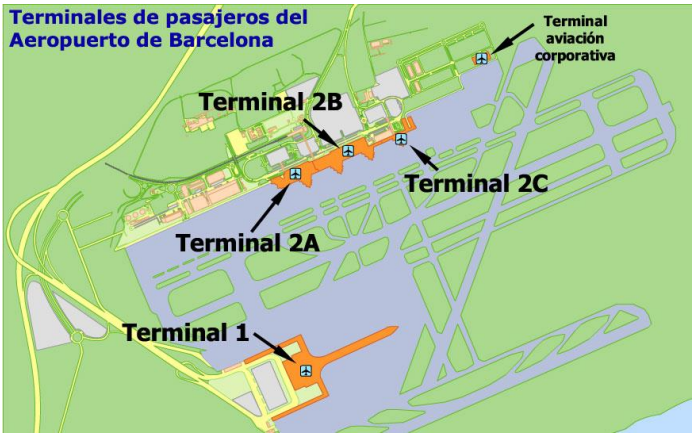


Figura 50 - Distribución Terminales de Pasajeros BCN

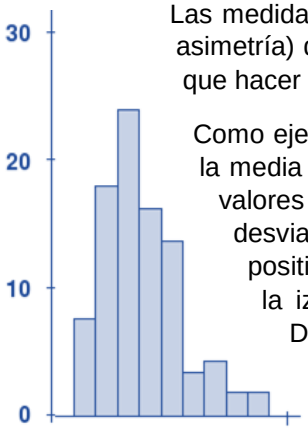
```
temp_terminal = pd.DataFrame()
temp_terminal['airportResources.arrivalTerminal'] = flights['airportResources.arrivalTerminal']
temp_terminal['delays.departureGateDelayMinutes'] = flights['delays.departureGateDelayMinutes']
temp_terminal = temp_terminal.dropna()

data_terminal = pd.DataFrame()
data_terminal['airportResources.arrivalTerminal'] = temp_terminal['airportResources.arrivalTerminal']
data_terminal['delays.departureGateDelayMinutes'] = temp_terminal['delays.departureGateDelayMinutes']
terminal_group = data_terminal.groupby('airportResources.arrivalTerminal')
delays_totals_terminal = terminal_group.mean()
delays_totals_terminal.sort(columns='delays.departureGateDelayMinutes')
ax = delays_totals_terminal.plot(kind='bar', title ="Restrasos por Terminal",legend=False)
ax.set_xlabel("Terminal")
ax.set_ylabel("Retraso Medio, m.")
plt.show()
```

Figura 51 - Código Gráfico Retrasos Turnaround por "Terminal"

3.5 Pre-procesado y Selección de Variables

3.5.1 Skewness (Asimetría)



Las medidas de asimetría son indicadores que permiten establecer el grado de simetría (o asimetría) que presenta una distribución de probabilidad de una variable aleatoria sin tener que hacer su representación gráfica.

Como eje de simetría consideramos una recta paralela al eje de ordenadas que pasa por la media de la distribución. Si una distribución es simétrica, existe el mismo números de valores a la derecha que a la izquierda de la media, por tanto, el mismo número de desviaciones con signo positivo que con signo negativo. Decimos que hay asimetría positiva (o a la derecha) si la "cola" a la derecha de la media es más larga que la de la izquierda, es decir, si hay valores más separados de la media a la derecha. Diremos, que hay asimetría negativa (o a la izquierda) si la "cola" a la izquierda de la media es más larga que la de la de la derecha, es decir, si hay valores más separados de la media a la izquierda.

3.5.1.1 Medidas de asimetría

Existen varias medidas de asimetría, a continuación se describirán las tres que actualmente se usan con mayor frecuencia, y se pueden encontrar ya integradas en librerías de programación.

3.5.1.2 Coeficiente de asimetría de Fisher

En teoría de la probabilidad y estadística, la medida de asimetría más utilizada parte del uso del tercer momento estándar. La razón de esto es que nos interesa mantener el sino de las desviaciones con respecto a la media, para obtener si son mayores las que ocurren a la derecha de la media que las de la izquierda. Sin embargo, no es buena idea tomar el momento estándar

$$\sum_{i=1}^{k} f_i(x_i - \mu) = \sum_{i=1}^{k} f_i x_i - \mu \sum_{i=1}^{k} f_i = \mu - \mu = 0$$

con respecto a la media de orden 1. Debido a que una simple suma de todas las desviaciones siempre es cero. En efecto, si por ejemplo los datos están agrupados en ***k*** clases, se tiene que:

En donde x_1 representa la marca de la clase ***i***-ésima y f_1 denota la frecuencia relativa de dicha clase. Por ello, lo más sencillo es tomar las desviaciones al cubo.

El coeficiente de asimetría de Fisher, representado por γ_1 se define como:

$$\gamma_1 = \frac{\mu_3}{\sigma^3},$$

Donde μ_3 es el tercer momento en torno a la media y σ es la desviación estándar.

Si $\gamma_1 > 0$, la distribución es asimétrica positiva o a la izquierda.

Si $\gamma_1 < 0$, la distribución es asimétrica negativa o la derecha.

Si la distribución es simétrica, entonces sabemos que $\gamma_1 = 0$.

El recíproco no es cierto: es un error común asegurar que si $\gamma_1 = 0$ entonces la distribución es simétrica (lo cual es falso).

3.5.1.3 Coeficiente de asimetría de Pearson

Sólo se puede utilizar en distribuciones uniformes, unimodales y moderadamente asimétricas. Se basa en que en distribuciones simétricas la media de la distribución es igual a la moda.

$$A_p = \frac{\mu - moda}{\sigma},$$

Donde **μ** es el momento central de orden 1, que corresponde a la media aritmética de la variable **X**.

Si la distribución es simétrica, **μ = moda** y A_p **= 0**. Si la distribución es asimétrica positive a la media se sitúa por encima de la moda y, por tanto, A_p **> 0**.

3.5.1.4 Coeficiente de asimetría de Bowley

Está basado en la posición de los cuartiles y la mediana, y utiliza la siguiente expresión:

$$A_B = \frac{Q_{3/4} + Q_{1/4} - 2Me}{Q_{3/4} - Q_{1/4}}$$

En una distribución simétrica el tercer cuartil estará a la misma distancia de la mediana que el primer cuartil. Por tanto A_B **= 0**.

Si la distribución es positiva o a la derecha, A_B **> 0**.

3.5.1.5 Utilidad

La asimétrica resulta útil en muchos campos. Muchos modelos simplistas asumen una distribución normal, esto es, simétrica en torno a la media. La distribución normal tiene una asimetría cero. Pero en realidad, los valores no son nunca perfectamente simétricos y la asimetría de la distribución proporciona una idea sobre si las desviaciones de la media son positivas o negativas. Una asimetría positiva implica que hay más valores distintos a la derecha de la media.

Las medidas de asimetría, sobre todo el coeficiente de asimetría de Fisher, junto con las medidas de apuntamiento o curtosis se utilizan para contrastar si se puede aceptar que una distribución estadística sigue la distribución normal. Esto es necesario para realizar numerosos contrastes estadísticos en la teoría de la inferencia estadística.

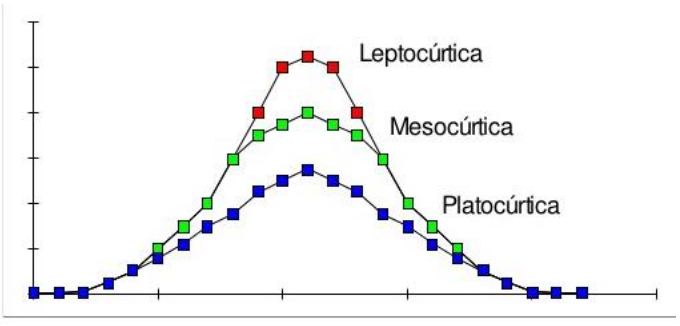


Figura 52 - Medidas de Apuntamiento y Curtosis

3.5.2 Selección de Variables y Pre-procesado

3.5.2.1 Carga de todas las Variables

Para un correcto procesamiento posterior de las variables y así poder conocer la importancia y el grado en que afecta cada una en el proceso del turnaround, se les debe asignar valores para poder conocer su desviación respecto a la media. De este modo se procede a cargar los datos de *'flights.csv'*.

```
df= pd.read_csv("flights.csv")
```

Figura 53 - Carga Datos FLIGHTS.CSV

Posteriormente se procede a seleccionar solo aquellas variables que serán usadas para el pre-proceso.

En primer lugar se seleccionan y se añaden las variables numéricas, estos no tienen la necesidad de ser transformado ya que están en el formato que se desea.

```
features.append ('airportResources.arrivalTerminal_x')
```

Figura 54 – Añadir Variables Numéricas

En segundo lugar, parecido al primer paso, pero esta vez con las variables categóricas que no están formadas por números, estos valores (comúnmente cadena de caracteres), deben ser transformados a valores numéricos para poder realizar cálculos y operaciones con ellos.

```
features.extend(['BaggageClaimCarrousel','CarrierFsCode','DepartureGate'])
le = LabelEncoder().fit(data['airportResources.baggage_x'])
data['BaggageClaimCarrousel']=le.transform(data['airportResources.baggage_x'])
le = LabelEncoder().fit(data['carrierFsCode_x'])
data['CarrierFsCode']=le.transform(data['carrierFsCode_x'])
le = LabelEncoder().fit(data['airportResources.departureGate_y'])
data['DepartureGate']=le.transform(data['airportResources.departureGate_y'])
```

Figura 55 – Añadir Variables Categóricas y su Transformación

En tercer y último lugar en la carga de éstas, se seleccionan los valores que contienen datos, y estos son descompuestos, para que cada campo pueda ser dividido en varias columnas, por ejemplo, donde inicialmente se tiene 2015-10-14, el objetivo es descomponerlo en 3 columnas, por 'Año'*(2015)*, 'Mes'*(10)*, y 'Día'*(14)*, de esta forma se obtienen columnas con datos significativos con los que se puede operar cómodamente, a diferencia de un campo con cadena de caracteres.

```
features.extend(['Month', 'Day', 'DayOfWeek', 'WeekOfYear', 'ArrivalTimeMinutes'])
data['Month'] = pd.to_datetime(data['arrivalDate.dateLocal_x']).dt.month
data['Day'] = pd.to_datetime(data['arrivalDate.dateLocal_x']).dt.day
data['DayOfWeek'] = pd.to_datetime(data['arrivalDate.dateLocal_x']).dt.dayofweek
data['WeekOfYear'] = pd.to_datetime(data['arrivalDate.dateLocal_x']).dt.weekofyear
data['ArrivalTimeMinutes'] = 60*pd.to_datetime(data['arrivalDate.dateLocal_x']).dt.hour
pd.to_datetime(data['arrivalDate.dateLocal_x']).dt.minute
```

Figura 56 – Añadir Fecha Descompuesta en Columnas Separadas

3.5.2.2 Pre-procesado de Variables

Cuando los datos de los factores predictores aparecen en diferentes rangos, dificulta lograr una estimación precisa de la predicción, es por ello que centrar y escalar es muy útil ya que de esta manera se consigue mayor estabilidad en los datos y lograr exactitud en los algoritmos de predicción.

Para realizar estos cálculos uno de los paquetes más idóneos es el paquete *sklearn.preprocessing*, ya que provee de gran cantidad de funcionalidades y transformadores de clases para cambiar vectores de variables en bruto en una representación que es mar adecuada para los estimadores.

Uno de los pasos fundamentales del pre-procesado de los datos, es el escalado de cada característica a un rango concreto, que es una alternativa a la estandarización, el escalado como su nombre bien dice, escala entre unos valores mínimos y máximos previamente dados, a menudo entre cero y uno.

La motivación principal para utilizar el método escalable es la robustez en muy pequeñas desviaciones estándar de las variables, y la preservación de cero en entradas que contienen datos escasos.

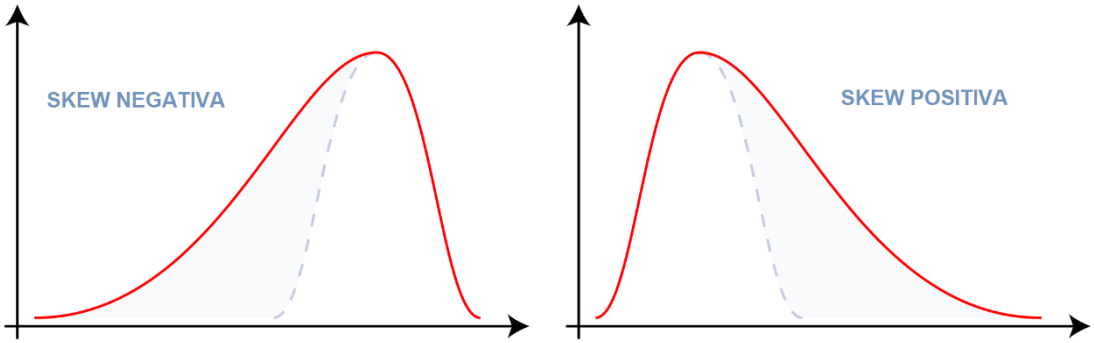


Figura 57 - Tipos de Skew

También, se debe realizar la transformación por la asimetría (*Skewness Transformation*), ya que la información se presenta frecuentemente de una forma asimétrica que dificulta el cálculo de las desviaciones. Consiguiendo así una distribución simétrica.

La asimetría (o *Skew*) es muy importante si el conjunto de datos de una característica es demasiado a la izquierda (*Skew Positiva*) o a la derecha (*Skew Negativa*), es decir, que no tiene simetría.

De la siguiente manera se puede comprobar la media y la desviación estándar de los datos aún sin procesar. Donde se observa la no-simetría.

```
for x in range(0, len(features)):
      plt.figure(x)
      plt.hist(df[features].values[:,
      plt.title(features[x])
      plt.xlabel("Value")
      plt.ylabel("Frequency")
      plt.show()
```

Figura 59 - Salida Gráficos con SKEW

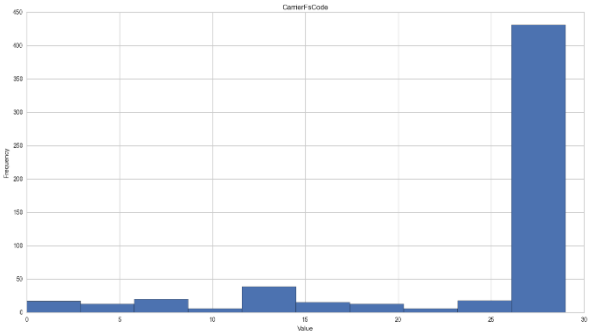


Figura 58 - Ejemplo de SKEWNESS en "Retrasos Compañías"

Una vez los datos son procesados, aplicando el centrado y el escalado a todas las variables, podemos observar que el código transforma los datos para ajustar los datos a los requeridos, donde la Desviación es **1**, y la Media parte del **0**. Como es por ejemplo el caso del gráfico anterior donde se ve una mayor cantidad de datos a la derecha, de aquí los siguientes resultados ya procesados:

```
airportResources.arrivalTerminal_x Mean:     -3.69E-17
airportResources.arrivalTerminal_x Std.Dev.: 1

CarrierFsCode Mean:    4.92E-17
CarrierFsCode Std.Dev.:       1

DepartureGate Mean:    -2.46E-17
DepartureGate Std.Dev.:       1

Day Mean:      4.92E-17
Day Std.Dev.:  1

ArrivalTimeMinutes Mean:      5.53E-17
ArrivalTimeMinutes Std.Dev.:  1
```

Figura 61 - Variables ya Transformadas

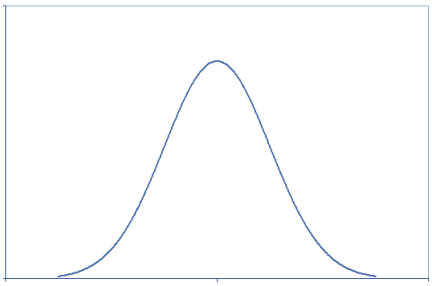


Figura 60 - Curva de una Distribución Normal

3.5.2.4 Selección de Variables

Para empezar es necesario especificar que variable se quiere predecir, en el caso actual no es otra que el tiempo de turnaround, en minutos, por ello:

```
turnaroundtime_minutes =
(60*pd.to_datetime(df['operationalTimes.actualGateDeparture.dateLocal_y']).dt.hour + \
pd.to_datetime(df['operationalTimes.actualGateDeparture.dateLocal_y']).dt.minute) - \
(60*pd.to_datetime(df['operationalTimes.actualGateArrival.dateLocal_x']).dt.hour + \
pd.to_datetime(df['operationalTimes.actualGateArrival.dateLocal_x']).dt.minute)
```

Figura 62 - Establecer Variable a Predecir

A continuación a través de Random Forests (ya presentado en el punto 2), junto con la mayoría de los parámetros, es decir, aquellos que pueden influenciar en un retraso en el turnaround, se empleará para poder determinar la importancia de cada una de las variables.

```
random.seed(111)
forest = ensemble.RandomForestRegressor(n_estimators=10,
min_samples_split=2, n_jobs=-1, random_state=111)
forest.fit(df_scaled, turnaroundtime_minutes)
feature_importance = forest.feature_importances_
```

Figura 63 - Montaje Algoritmo Definición de Importancia a Través de Random Forests

Para conocer la importancia de cada una de ellas se les dará los valores correspondientes, estos serán en una escala de porcentaje, donde 100 será el máximo.

```
feature_importance = 100.0 *
(feature_importance /
feature_importance.max())
```

Figura 64 - Establecer Rango de Importancia (En Porcentaje)

A la vez se deberán descartar aquellas variables que hayan adquirido un valor de importancia dentro el conjunto de datos, por debajo de un porcentaje dado, en concreto este número representara el porcentaje respecto al valor más importante. Es decir, como será del 2%, se descartaran todas aquellas variables que tengan un valor inferior al 2% de importancia de la que tiene la característica más importante.

```
fi_threshold = 2
```

Figura 65 -Establecmiento Umbral

Finalmente se harán otras operaciones como crear los índices de todas aquellas variables que por importancia se sitúen por encima del umbral establecido, posteriormente se deberá crear una lista con todas estas, ordenarlas de mayor importancia a menor, entre otros. Para finalmente acabar con la elaboración del gráfico de las variables más importantes en lo relativo a los tiempos de turnaround en el conjunto de los datos que se han obtenido.

```
pos = np.arange(sorted_idx.shape[0]) + .5
plt.subplot(1, 2, 2)
plt.barh(pos,
feature_importance[important_idx][sorted_idx],
align='center')
plt.yticks(pos,
important_features[sorted_idx])
plt.xlabel('Relative Importance')
plt.title('Variable Importance')
```

Figura 67 – Código Grafico Importancia Variables

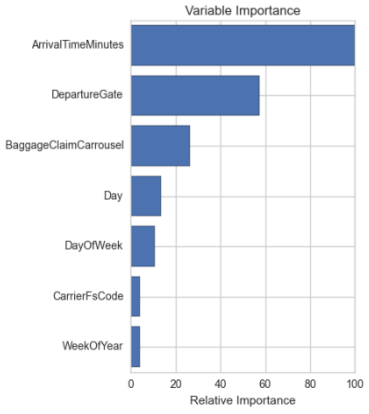


Figura 66 - Gráfico Importancia Variables

3.6 Modelo Predictivo

3.6.1 Evaluación del modelo usando RMSE (Root-Mean-Square Error)

Con los siguientes modelos predictivos se obtendrán los resultados en forma de RMSE (Root-Mean-Square Error) o lo que es lo mismo la Raíz del Error Cuadrático Medio, pudiendo ser a la vez RMSD (Root-Mean-Square Deviation) o lo que es lo mismo la Raíz de la Desviación Cuadrática Media

El Error Cuadrático Medio (MSE- Mean Square Error) en estadística se trata de un estimador que mide el promedia de los errores al cuadrado, es decir, la diferencia entre el estimador y lo que se estima. El MSE es una función de riesgo, correspondiente al valor esperado de la pérdida del error al cuadrado o perdida cuadrática. La diferencia se produce debido a la aleatoriedad o porque el estimador no tiene en cuenta la información que podría producir una estimación más precisa.

El MSE (Mean Square Error) es el segundo momento (sobre el origen) del error, y por lo tanto incorpora tanto la varianza del estimador así como su sesgo. Para un estimador insesgado el RMSE es la varianza del estimador. Al igual que la varianza, el RMSE tiene las mismas unidades de medida que el cuadrado de la cantidad que se estima.

En una analogía con la desviación estándar tomando la raíz cuadrada del RMSE produce el error de la raíz cuadrada de la media o la desviación de la raíz cuadrada media (RMSE o RMSD), que tiene las mismas unidades que la cantidad que se estima, para un estimador insesgado, el RMSE es la raíz cuadrada de la varianza, conocida como la desviación estándar.

3.6.1.1 Fórmula

El RMSD de un estimador $\hat{\theta}$ con respecto a un estimador parámetro θ es definido como la raíz cuadrada del error cuadrático medio:

$$\mathrm{RMSD}(\hat{\theta}) = \sqrt{\mathrm{MSE}(\hat{\theta})} = \sqrt{\mathrm{E}((\hat{\theta} - \theta)^2)}.$$

Figura 68 - Definición Fórmula RMSD

Para un estimador insesgado, el RMSD es la raíz cuadrada de la varianza, conocido como el error estándar.

El RMSD de los valores previstos $\hat{\mathbf{y}}_t$ para tiempos $\mathbf{y}$ de variable dependiente de una regresión se calcula para *n* diferentes predicciones como la raíz cuadrada de la media de los cuadrados de las desviaciones:

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{t=1}^{n} (\hat{y}_t - y)^2}{n}}.$$

Figura 69 - RMSD Para Valores $\hat{y}_t$

En algunas disciplinas, el RMSD se utiliza para comparar las diferencias entre dos cosas que pueden variar, ninguno de los cuales es aceptado como el "estándar". Por ejemplo, al medir la diferencia media entre dos series de tiempo $\boldsymbol{x_{1,t}}$ y $\boldsymbol{x_{2,t}}$, la fórmula se convierte en:

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{t=1}^{n} (x_{1,t} - x_{2,t})^2}{n}}.$$

Figura 70 - RMSD Medir Diferencia Entre Series

3.6.2 Predicción Random Forests

Se realizará el modelaje predictivo con el Algoritmo Random Forests, inicialmente se definirá tal y como se ha descrito anteriormente la Raíz del Error Medio Cuadrático, realizado a través del siguiente código:

```
def RMSE(y, yhat):
    rmspe = np.sqrt(np.mean((y - yhat)**2))
    return rmspe
```

Figura 71 - Código Definición RMSE

Los datos usados para poder realizar una correcta predicción son los datos ya pre-procesados donde se encuentran los valores que se han asignado por la importancia en que afecta al turnaround, valores con la desviación respecto a la media ya conocida, centrado en rangos, transformados para evitar la asimetría (Skewness), entre otros.

Estos datos se encuentran en el archivo CSV (Valores Separados por Comas) *flights_transformed.csv*:

```
df = pd.read_csv("flights_transformed.csv")
```

Figura 72 - Lectura des de CSV Transformado

Posteriormente a través de dos líneas de código se seleccionaran los predictores, que son todos aquellos que ayudarán a la predicción, como aparece en la primera línea *alldata*, y posteriormente el valor que se quiere predecir, en este caso el objetivo (*target*) será el mismo valor de *turnaround_time*.

```
alldata = df.ix[:,0:(len(df)-1)]
target = df['turnaround_time']
```

Figura 73 - Selección Predictores y Variables

3.6.2.1 Conjunto de Entrenamiento

Random Forests tiene dos maneras de reemplazar los valores perdidos. La primera forma es la más rápida. Si la variable no es categórica, el método calcula la media de todos los valores de esta variable en la clase **j** por ejemplo, entonces se utiliza este valor para reemplazar todos los valores perdidos de la variable. Sin embargo si la variable es categórica el reemplazo es el valor más frecuente en el conjunto de datos. Estos valores de reemplazo son llamados rellenos.

La segunda manera de reemplazar los valores perdidos es computacionalmente más difícil, pero da un rendimiento mayor que el primero, incluso con grandes cantidades de datos faltantes. Sustituye los valores solo en el conjunto de entrenamiento. Comienza haciendo un relleno áspero e inexacto de los valores perdidos, luego se ejecuta Random Forests y computa valores próximos.

Para crear un conjunto de entrenamiento se procede a realizar la siguiente ejecución:

```
m = forest.fit(train_fold, train_y)
```

Figura 74 - Conjunto de Entrenamiento Random Forests

3.6.2.2 Conjunto de Prueba

Dependiendo si el conjunto de prueba usa etiquetas o no, se utilizaran diferentes estrategias. En ambos casos se utilizaran los valores de relleno obtenidos por la ejecución en el conjunto de datos de Entrenamiento.

Se medirá la calidad que tiene el remplazo para ver que tasa de errores se asigna al conjunto de entrenamiento. Por ejemplo si el conjunto de prueba se extrae de la misma distribución del de entrenamiento este dará un porcentaje de error diferente si se hace de forma distinta. A medida que la proporción de los datos inexistentes crezca, usando rellenos, la distribución de la prueba se alejará cada vez más del conjunto de entrenamiento, a la vez que el ratio de error también aumentará.

Para establecer este Conjunto de Prueba se utilizará:

```
predicted_y = m.predict(test_fold)
```

Figura 75 - Conjunto de Prueba Random Forests

3.6.2.3 Establecimiento Conjuntos, Creación Algoritmo y Variaciones RMSE

Previo al establecimiento los conjuntos de Prueba y Entrenamiento, se debe establecer los porcentajes de cada uno de estos conjuntos:

```
train_fold, test_fold, train_y, test_y = cross_validation.train_test_split
(alldata, target, test_size=0.3, random_state=123)
```

Figura 76 - Porcentaje Conjuntos de Prueba

Dónde se aprecia que para el estudio un 30% de los datos de '*all_data'* se escogen para hacer las pruebas del Conjunto de Datos de Prueba (*Test Set*). Restando el 70% para el Conjunto de Datos de Entrenamiento.

```
forest = ensemble.RandomForestRegressor
(n_estimators=10, min_samples_split=2, n_jobs=-1)
```

Figura 77 - Parámetros Random Forests

Con esta línea el algoritmo de Random Forest se creará, con los parámetros establecidos, en este caso expuesto con *n_estimators* igual a **10** valor que indica el número de árboles que habrá en el algoritmo, y *min_samples_split* con un valor de **2** que representa el número mínimo de muestras requeridas para dividir un nodo interno.

Este código es ejecutado una gran cantidad de veces con diferentes valores en estos dos parámetros expuestos para observar que los Errores en la Predicción (RMSE) van variando según estos parámetros.

El resultado de los diferentes Errores en la Predicción (RMSE) con el modelo predictivo 'Random Forest' se muestra en la siguiente gráfica:

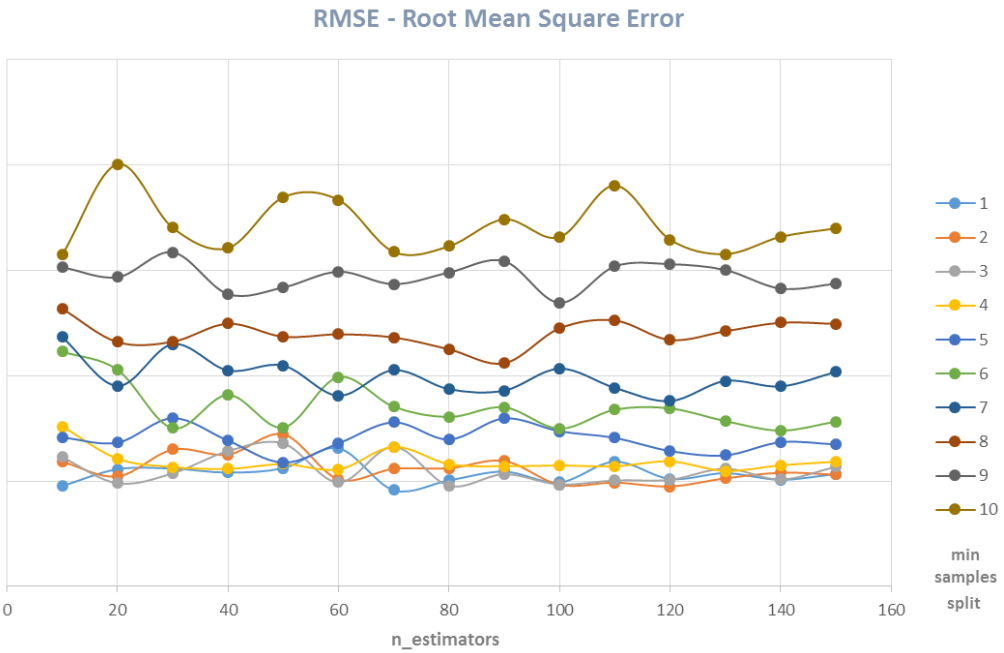


Figura 78 - Variaciones RMSE Según Parámetros con RANDOM FOREST

La siguiente gráfica muestra los valores a través del mismo procedimiento pero esta vez a través del modelo predictivo Gradient Boosting.

Como se aprecia, todas las líneas quedan sobrepuestas en una única línea debido a que todos los valores son prácticamente idénticos, esto significa que este modelo predictivo no se ve afectado por los *min_samples_split*, devolviendo casi valores exactos.

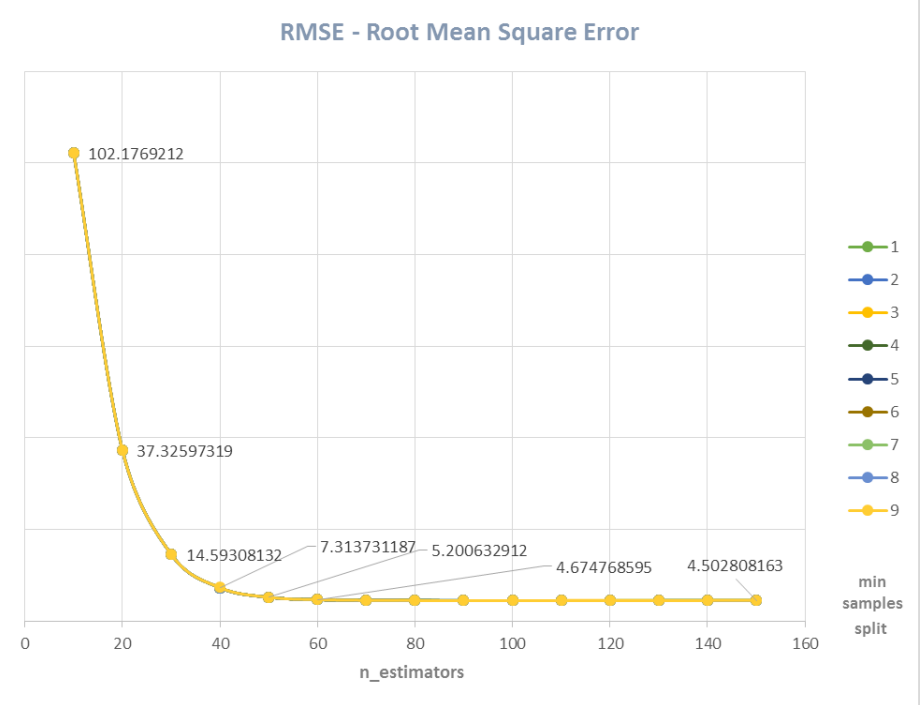


Figura 79 - Variaciones RMSE Según Parámetros con GRADIENT BOOSTING

Para la correcta visualización de los parámetros exactos es necesario acudir a los valores en la

num_min_splits \ n_estimators	1	2	3	4	5	6	7	8	9
10	102.1769212	102.1769212	102.1769212	102.1769212	102.1769212	102.1769212	102.1769212	102.1769212	102.1769212
20	37.32597319	37.32597319	37.32597319	37.32597319	37.32597319	37.32597319	37.32597319	37.32597319	37.32597319
30	14.59308132	14.59308132	14.59308132	14.59308132	14.59308132	14.59308132	14.59308132	14.59308132	14.59308132
40	7.284349957	7.284349957	7.284349957	7.284349957	7.284349957	7.284349957	7.284349957	7.284349957	7.313731187
50	5.199175317	5.199175317	5.199175317	5.199175317	5.199175317	5.199175317	5.199824019	5.195105741	5.200632912
60	4.684860249	4.684860249	4.684860249	4.682548348	4.682548348	4.680865193	4.677124994	4.671258115	4.674768595
70	4.556779179	4.556779179	4.556779179	4.547177027	4.546358384	4.543864717	4.551873989	4.529827406	4.540319468
80	4.516073839	4.516073839	4.517275574	4.50776183	4.510410084	4.504220626	4.511278609	4.492830973	4.50453953
90	4.496478309	4.496478309	4.491318432	4.480165615	4.482629627	4.480905179	4.487541534	4.469206241	4.488522503
100	4.491015689	4.491015689	4.493530186	4.47405304	4.480983823	4.476427447	4.486179695	4.461340458	4.48342041
110	4.493053302	4.493053302	4.498239794	4.48183241	4.479444909	4.476452779	4.493725948	4.462672983	4.480898558
120	4.492047083	4.492047083	4.504243563	4.474531414	4.484744761	4.476178205	4.501979361	4.466719823	4.488046309
130	4.499426598	4.499426598	4.508091269	4.477509544	4.489211755	4.472468891	4.504269313	4.467274036	4.487158558
140	4.504296776	4.504296776	4.509351263	4.481804347	4.492913643	4.47262915	4.507108875	4.467967183	4.499987929
150	4.500661285	4.500661285	4.515316729	4.479134793	4.496070812	4.470412947	4.517672912	4.470040115	4.502808163

Figura 80 - Hoja de Cálculo Valores RMSE Gradient Boosting

hoja de cálculo.

Finalmente, los valores importantes a considerar son los que generan el menor error posible, y en este caso el menor aparece en Gradient Boosting con un RMSE de 4.46134 a diferencia del valor 4.57021 generado por Random Forests.

El valor mínimo es logrado con los siguientes parámetros: **8 n_estiamators - 100 min_num_splits**

Siendo entonces Gradient Boosting el modelo predictivo más apropiado en las predicciones.

CONCLUSIONES

Como conclusión, tras la observación que el factor más importante en los retrasos en el turnaround es la hora en la que el turnaround tiene lugar, surge la idea de que la homogenización de las horas de partida de los vuelos repartidos en las horas operativas del aeropuerto, ayudaría a descongestionar las horas punta, y así disminuyendo el tiempo del turnaround, a la vez que los costes.

Las horas de los vuelos dependen de la demanda de los consumidores principalmente, pero a la vez una diferente franja horaria puede persuadir a un cliente, a través de su precio.

Esta congestión por franjas horarias, es un problema común en la mayor parte de la Unión Europea, y hasta el momento no se han tomado soluciones estrictas a ello, llevando a los aeropuertos a ambos escenarios simultáneamente en el día, escenarios de sobre congestión en horas punta, y escenarios de descongestión en horas de poco interés.

La Comisión Europea a través de Eurocontrol logró una ligera mejoría en la homogenización de las horas de vuelos, a pesar de ellos, poco más se ha hecho para mejorar.

No obstante la implementación de unas correctas políticas de tarifas de congestión en horas punta con el propósito de regular la demanda, solucionaría en gran parte problemas de congestión sin hacer inversiones de ampliaciones aeroportuarias.

Según Eurocontrol, sobre el año 2035 se perderán alrededor de dos millones de vuelos debido a la congestión de los aeropuertos Europeos. Algunos de los aeropuertos con más tráfico ya operan a plena capacidad, aunque algunos disponen aún de slots libres durante las horas valle, pero haciendo igualmente frente a grandes niveles de congestión en horas punta, que se suelen dar a primera hora de la mañana o ultima hora de la tarde.

Además de la información aprendida sobre cómo afectan estos problemas sobre el mundo de la aviación, se ha podido aprender a usar un nuevo lenguaje de programación muy adecuado para la recolección de datos así como su procesamiento, este lenguaje es Python, un proyecto de software libre y a través del entorno de programación PyCharm, otros conocimientos adquiridos son la aplicación del preocesado de datos así como las técnicas de aprendizaje automático (Machine Learning), técnicas de las que toman parte los algoritmos de Random Forest y XGBoost.

A través de las funciones estadísticas de las librerías adicionales, se han conseguido datos con buena calidad, y en un formato óptimo para su posterior tratamiento.

Como resultado, cabe destacar la importancia que tiene la correcta selección de las variables o variables, para saber determinar con certeza cuales tienen un rol determinante en el tiempo del turnaround.

Finalmente se ha podido comprobar que los modelos predictivos generan unos valores satisfactorios para la predicción de los tiempos del turarnound.

El acceso a información podría ser proporcionado de una manera más completa y a la vez de una forma mayormente formateada de la que actualmente se proporciona, hay carencias respecto a las fuentes de información de los portales de información aeronáutica. A parte de la baja calidad de algunos datos, sería deseable en un futuro próximo la inserción de más variables, como ejemplos cabe mencionar, número de pasajeros que permite la configuración de la aeronave, combustible previsto para el repostaje, saturación de las pistas, número de tractores pushback disponibles en el momento, etc. Qué puedan ser determinantes en el tiempo del turnaround que hoy en día no están disponibles de una forma informatizada y fácilmente accesible,

Aitor Romero Checa, a 25 de Enero de 2015

BIBLIOGRAFÍA

Edición Web

1. **Wikipedia - Data science**

   https://en.wikipedia.org/wiki/Data_science

   Fecha de consulta: 13 de Noviembre 2015

2. **Udacity - Data Science**

   https://www.edx.org/course/data-science-machine-learning-essentials-microsoft-dat203x-0

   Fecha de consulta: 13 de Noviembre 2015

3. **IBM - What is a Data Scientist?**

   http://www-01.ibm.com/software/data/infosphere/data-scientist/

   Fecha de consulta: 13 de Noviembre 2015

4. **Kdnuggets - Data Science of Process Mining – Understanding Complex**

   http://www.kdnuggets.com/2015/09/data-science-process-mining-understanding-complex-processes.html

   Fecha de consulta: 14 de Noviembre 2015

5. **Quora - Flow Process Data Scientist**

   https://www.quora.com/What-is-the-work-flow-or-process-of-a-data-scientist

   Fecha de consulta: 13 de Noviembre 2015

6. **SAS - 7 Steps for executing a successful data science strategy**

   http://www.sas.com/en_us/insights/articles/analytics/7-steps-for-executing-a-successful-data-science-strategy.html

   Fecha de consulta: 13 de Noviembre 2015

7. **Codecademy - Python**

   https://www.codecademy.com/es/learn/python

   Fecha de consulta: 14 de Noviembre 2015

8. **SAA - Methods of Gathering Data**

   http://www.saa.org/ForthePublic/Resources/EducationalResources/ForEducators/ArchaeologyforEducators/MethodsofGatheringData/tabid/1347/Default.aspx

   Fecha de consulta: 15 de Noviembre 2015

9. **Wikipedia - Python (programming language)**

   https://en.wikipedia.org/wiki/Python_(programming_language)

   Fecha de consulta: 15 de Noviembre 2015

10. **Learning Space Tool Kit - Data Gathering Tools**

    http://learningspacetoolkit.org/needs-assessment/data-gathering-tools-2/

    Fecha de consulta: 15 de Noviembre 2015

11. **WarWick - Methods Gathering Data**

    https://www2.warwick.ac.uk/services/ldc/resource/eguides/elearning/methods/

    Fecha de consulta: 20 de Noviembre 2015

12. **JetBrains - PyCharm**

    https://www.jetbrains.com/pycharm/

    Fecha de consulta: 20 de Noviembre 2015

13. **Wikipedia - Web scraping**

    https://en.wikipedia.org/wiki/Web_scraping

    Fecha de consulta: 20 de Noviembre 2015

14. **School of Data - Introducción al Web Scrapping**

    http://es.schoolofdata.org/introduccion-a-la-extraccion-de-datos-de-sitios-web-scraping/

    Fecha de consulta: 20 de Noviembre 2015

15. **GOOGLE - API MAPS**

https://developers.google.com/maps/?hl=es

Fecha de consulta: 21 de Noviembre 2015

16. **WeboPedia - API**

http://www.webopedia.com/TERM/A/API.html

Fecha de consulta: 21 de Noviembre 2015

17. **Analytics Vidhya - A Comprehensive guide to Data Exploration**

http://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/

Fecha de consulta: 21 de Noviembre 2015

18. **Machine Learning Mastery - How to Prepare Data for Machine Learning**

http://machinelearningmastery.com/how-to-prepare-data-for-machine-learning/

Fecha de consulta: 20 de Noviembre 2015

19. **Wikipedia - Feature engineering**

https://en.wikipedia.org/wiki/Feature_engineering

Fecha de consulta: 21 de Noviembre 2015

20. **Machine Learning Mastery - Discover Feature Engineering**

http://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/

Fecha de consulta: 21 de Noviembre 2015

21. **UFAL - Feature Engineering (PDF)**

https://ufal.mff.cuni.cz/~zabokrtsky/courses/npfl104/html/feature_engineering.pdf

Fecha de consulta: 22 de Noviembre 2015

22. **Wikipedia - Predictive Modelling**

https://en.wikipedia.org/wiki/Predictive_modelling

Fecha de consulta: 22 de Noviembre 2015

23. **Univeristy of Washington - Introduction to Boosted Trees**

https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf

Fecha de consulta: 22 de Noviembre 2015

24. **XGboost - Introduction to Boosted Trees**

http://xgboost.readthedocs.org/en/latest/model.html

Fecha de consulta: 22 de Noviembre 2015

25. **Wikipedia - Gradient Boosting**

https://en.wikipedia.org/wiki/Gradient_boosting

Fecha de consulta: 22 de Noviembre 2015

26. **Explorable - Correlación Estadística**

https://explorable.com/es/la-correlacion-estadistica

Fecha de consulta: 27 de Noviembre 2015

27. **Biblioteca Virtual VIREF - Correlación Variables**

http://viref.udea.edu.co/contenido/menu_alterno/apuntes/ac36-correlacion-variables.pdf

Fecha de consulta: 27 de Noviembre 2015

28. **FlighStats - Home**

http://www.flightstats.com/go/Home/home.do

Fecha de consulta: 27 de Noviembre 2015

29. **FlightStats - Status & Track**

https://developer.flightstats.com/api-docs/flightstatus/v2

Fecha de consulta: 27 de Noviembre 2015

30. **FlighStats - Schedules API**

https://developer.flightstats.com/api-docs/scheduledFlights/v1

Fecha de consulta: 28 de Noviembre 2015

31. **Accelinn - Data Processing**

http://www.accelinn.com/de/unified-efficiency/27-unified-efficiency/data-processing-structuration/50-data-processing-structuration-a

Fecha de consulta: 28 de Noviembre 2015

32. **SlideShare - Big Data Visual Analytics and Predictive Analytics Tools**

http://es.slideshare.net/idigdata/big-data-visual-analytics-and-predictive-analytics-tools

Fecha de consulta: 28 de Noviembre 2015

33. **NSA - Visual Analytics**

https://www.nsa.gov/research/tnw/tnw204/article4.shtml

Fecha de consulta: 28 de Noviembre 2015

34. **Wikipedia - Skewness**

https://en.wikipedia.org/wiki/Skewness

Fecha de consulta: 28 de Noviembre 2015

35. **ITL - Measures of Skewness and Kurtosis**

http://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm

Fecha de consulta: 29 de Noviembre 2015

36. **MathWorld - Skewness**

http://mathworld.wolfram.com/Skewness.html

Fecha de consulta: 29 de Noviembre 2015

37. **GraphPad - Interpreting Results**

http://www.graphpad.com/guides/prism/6/statistics/index.htm?stat_skewness_and_kurtosis.htm

Fecha de consulta: 29 de Noviembre 2015

38. **SCIKIT Learn - Preprocessing Data**

http://scikit-learn.org/stable/modules/preprocessing.html

Fecha de consulta: 29 de Noviembre 2015

39. **Satnford University - Data Preprocessing**

http://ufldl.stanford.edu/wiki/index.php/Data_Preprocessing

Fecha de consulta: 29 de Noviembre 2015

40. **Wikipedia Root-Mean-Square Deviation**

https://en.wikipedia.org/wiki/Root-mean-square_deviation_of_atomic_positions

Fecha de consulta: 29 de Noviembre 2015

41. **SCIKIT Learn - Feature Selection**

http://scikit-learn.org/stable/modules/feature_selection.html

Fecha de consulta: 29 de Noviembre 2015

42. **Machine Learning Mastery - Feature Selection**

http://machinelearningmastery.com/an-introduction-to-feature-selection/

Fecha de consulta: 4 de Diciembre 2015

43. **Wikipedia - Asimetría Estadística**

https://es.wikipedia.org/wiki/Asimetr%C3%ADa_estad%C3%ADstica

Fecha de consulta: 5 de Diciembre 2015

44. **Universidad Jaime I - Coeficientes de Asimetria de Fisher y Pearson**

www3.uji.es/~mateu/tema4-d37.doc

Fecha de consulta: 11 de Diciembre 2015

45. **SlideShare - Pearson**

http://es.slideshare.net/carlosrv0/coeficiente-de-asimetra-de-pearson

Fecha de consulta: 11 de Diciembre 2015

46. **Tareas Plus - Bowley**

https://aula.tareasplus.com/Marcela-Gomez/Probabilidad-y-Estadistica/Asimetria-Coeficiente-de-Bowley

Fecha de consulta: 13 de Diciembre 2015

47. **UGRAD - Random Forest Predict**

http://ugrad.stat.ubc.ca/R/library/randomForest/html/predict.randomForest.html

Fecha de consulta: 13 de Diciembre 2015

48. **Leo Breiman - Random Forests**

https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

Fecha de consulta: 13 de Diciembre 2015

49. **Matplotlib - Plotting Tutorial**

http://matplotlib.org/

Fecha de consulta: 14 de Diciembre 2015

50. **Python Wiki - Plotting**

https://wiki.python.org/moin/NumericAndScientific/Plotting

Fecha de consulta: 14 de Diciembre 2015

Edición Impresa

Geisser, Seymour (1993)
Predictive Inference: An Introduction. New York: Chapman & Hall. p. [page needed].
ISBN 0-412-03471-9.

Domingos, Pedro (12 Noviembre 2015)
"A Few Useful Things to Know about Machine Learning".

Dhar, V. (2013).
"Data science and prediction". Communications of the ACM 56 (12): 64.
doi:10.1145/2500499.

Brieman, L (June 1997)
"Arcing The Edge"

Schaffer, C. (Septiembre 2011)
Data Structures and Algorithm Analysis

Joanes, D. N.; Gill, C. A. (1998).
"Comparing measures of sample skewness and kurtosis".
Journal of the Royal Statistical Society (Series D): The Statistician 47
(1): 183–189. doi:10.1111/1467-9884.00122.

Fco. Javier Martín-Pliego López (2007)
'Introducción a la Estadística Económica y Empresarial. Teoría y Práctica.'
Editorial Thomson, (Madrid).

ANEXO 1 (DataExplotiveAnalysis.py)

```
import pandas as pd
import numpy as np
from sklearn.neighbors import NearestNeighbors
from urllib2 import Request, urlopen
import json
from pandas.io.json import json_normalize
import gc
import seaborn as sbn
import matplotlib.pyplot as plt
from numpy import arange
import bisect

sbn.set_style("whitegrid")

def find_nearest(group, match, groupname):
    match = match[match[groupname] ==
group['flightEquipment.tailNumber']]
    if len(match)>0:
        nbrs = NearestNeighbors(n_neighbors=len(match),
algorithm='auto').fit(match['operationalTimes.actualDateTime'].values[:,N
one])
        dist, ind =
nbrs.kneighbors(match['operationalTimes.actualDateTime'].values[:,None])
        mindiff = 99999999
        diff = -111
        idx = -1
        for (x,y), i in np.ndenumerate(ind):
            diff = match['operationalTimes.actualDateTime'].values[i]-
group['operationalTimes.actualDateTime']
            if diff>0 and diff<mindiff:
                idx = i
                mindiff = diff
        if diff<0:
            return pd.DataFrame(),pd.DataFrame()
        else:
            return group.values,match.iloc[idx].values
    else:
        return pd.DataFrame(),pd.DataFrame()

def scatterplot(x,y,x_title,y_title):
    plt.plot(x,y,'b.')
    plt.xlabel(x_title)
    plt.ylabel(y_title)
    plt.xlim(min(x)-1,max(x)+1)
    plt.ylim(min(y)-1,max(y)+1)
    plt.show()

def barplot(labels,data,x_title,y_title):
    pos=arange(len(data))
    plt.xlabel(x_title)
    plt.ylabel(y_title)
    plt.xticks(pos+0.4,labels)
    plt.bar(pos,data)
    plt.show()

def histplot(data,x_title,y_title,bins=None,nbins=10):
    minx,maxx=min(data),max(data)
    space=(maxx-minx)/float(nbins)
```

```
    if not bins: bins=arange(minx,maxx,space)
    binned=[bisect.bisect(bins,x) for x in data]
    l=['%.1f'%x for x in list(bins)+[maxx]] if space<1 else [str(int(x))
for
    x in list(bins)+[maxx]]
    displab=[x+'-'+y for x,y in zip(l[:-1],l[1:])]
    barplot(displab,[binned.count(x+1) for x in
range(len(bins))],x_title,y_title)

def barchart(x,y,x_title,y_title,numbins=10):
    datarange=max(x)-min(x)
    bin_width=float(datarange)/numbins
    pos=min(x)
    bins=[0 for i in range(numbins+1)]

    for i in range(numbins):
        bins[i]=pos
        pos+=bin_width
    bins[numbins]=max(x)+1
    binsum=[0 for i in range(numbins)]
    bincount=[0 for i in range(numbins)]
    binaverage=[0 for i in range(numbins)]

    for i in range(numbins):
        for j in range(len(x)):
            if x[j]>=bins[i] and x[j]<bins[i+1]:
                bincount[i]+=1
                binsum[i]+=y[j]

    for i in range(numbins):
        if (bincount[i]>0):
            binaverage[i]=float(binsum[i])/bincount[i]
        else:
            binaverage[i]=0
    barplot(range(numbins),binaverage,x_title,y_title)

appID = '******'
appKey = '************************'

flights_arrivals = pd.read_csv("flights_arrivals.csv")
flights_departures = pd.read_csv("flights_departures.csv")

# ---------- Getting data from API ----------
# flights_arrivals = pd.DataFrame()
# flights_departures = pd.DataFrame()
# for day in range(28,30):
#     for hour in [6,18]:
#
request=Request('https://api.flightstats.com/flex/flightstatus/rest/v2/js
on/airport/status/LEBL/arr/2015/10/'+str(day)+'/'+str(hour)+'?appId='+app
ID+'&appKey='+appKey+'&utc=false&numHours=6')
#         response_arrivals = urlopen(request)
#         arrivals = response_arrivals.read()
#         data_arr = json.loads(arrivals)
#         df_arr =
pd.DataFrame(json_normalize(data_arr['flightStatuses']))
#         flights_arrivals = flights_arrivals.append(df_arr)
#
#
request=Request('https://api.flightstats.com/flex/flightstatus/rest/v2/js
on/airport/status/LEBL/dep/2015/10/'+str(day)+'/'+str(hour)+'?appId='+app
```

```
ID+'&appKey='+appKey+'&utc=false&numHours=6')
#          response_departures = urlopen(request)
#          departures = response_departures.read()
#          data_dep = json.loads(departures)
#          df_dep =
pd.DataFrame(json_normalize(data_dep['flightStatuses']))
#          flights_departures = flights_departures.append(df_dep)

column_names_arrivals = [    'airportResources.baggage',
                             'airportResources.arrivalTerminal',
                             'arrivalAirportFsCode',
                             'arrivalDate.dateLocal',
                             'carrierFsCode',

'delays.arrivalGateDelayMinutes','delays.arrivalRunwayDelayMinutes',

'flightDurations.blockMinutes','flightDurations.scheduledBlockMinutes',

'flightDurations.scheduledTaxiInMinutes','flightDurations.taxiInMinutes',

'flightEquipment.scheduledEquipmentIataCode','flightEquipment.tailNumber'
,

'flightId','flightNumber','operationalTimes.actualGateArrival.dateLocal',

'operationalTimes.actualRunwayArrival.dateLocal',

'operationalTimes.estimatedGateArrival.dateLocal',

'operationalTimes.estimatedRunwayArrival.dateLocal',

'operationalTimes.publishedArrival.dateLocal',

'operationalTimes.scheduledGateArrival.dateLocal',

'schedule.flightType','schedule.serviceClasses','status']

column_names_departures = [ 'airportResources.baggage',
                            'airportResources.departureGate',
                            'airportResources.departureTerminal',
                            'departureAirportFsCode',
                            'departureDate.dateLocal',
                            'carrierFsCode',

'delays.departureGateDelayMinutes','delays.departureRunwayDelayMinutes',

'flightDurations.blockMinutes','flightDurations.scheduledBlockMinutes',

'flightDurations.scheduledTaxiOutMinutes','flightDurations.taxiOutMinutes
',

'flightEquipment.scheduledEquipmentIataCode','flightEquipment.tailNumber'
,

'flightId','flightNumber','operationalTimes.actualGateDeparture.dateLocal
',

'operationalTimes.actualRunwayDeparture.dateLocal',

'operationalTimes.estimatedGateDeparture.dateLocal',
```

```
'operationalTimes.estimatedRunwayDeparture.dateLocal',

'operationalTimes.publishedDeparture.dateLocal',

'operationalTimes.scheduledGateDeparture.dateLocal',

'schedule.flightType','schedule.serviceClasses','status']

flights_arrivals = flights_arrivals[column_names_arrivals]
flights_departures = flights_departures[column_names_departures]

flights_arrivals =
flights_arrivals.dropna(subset=['flightEquipment.tailNumber'])
flights_departures =
flights_departures.dropna(subset=['flightEquipment.tailNumber'])

# ---------------------------------------------------------
flights_arrivals['operationalTimes.actualDateTime'] =
flights_arrivals['operationalTimes.actualGateArrival.dateLocal']
flights_departures['operationalTimes.actualDateTime'] =
flights_departures['operationalTimes.actualGateDeparture.dateLocal']
# ---------------------------------------------------------

flights_arrivals =
flights_arrivals.dropna(subset=['operationalTimes.actualDateTime'])
flights_departures =
flights_departures.dropna(subset=['operationalTimes.actualDateTime'])

flights_arrivals['operationalTimes.actualDateTime'] =
1440*pd.to_datetime(flights_arrivals['operationalTimes.actualDateTime']).
dt.dayofyear + \

60*pd.to_datetime(flights_arrivals['operationalTimes.actualDateTime']).dt
.hour + \

pd.to_datetime(flights_arrivals['operationalTimes.actualDateTime']).dt.mi
nute
flights_departures['operationalTimes.actualDateTime'] =
1440*pd.to_datetime(flights_departures['operationalTimes.actualDateTime']
).dt.dayofyear + \

60*pd.to_datetime(flights_departures['operationalTimes.actualDateTime']).
dt.hour + \

pd.to_datetime(flights_departures['operationalTimes.actualDateTime']).dt.
minute

gc.collect()
flights1 = []
flights2 = []
for index, row in flights_arrivals.iterrows():
    result1,result2 =
find_nearest(row,flights_departures,'flightEquipment.tailNumber')
    if len(result1) > 0:
        flights1.append(result1)
        flights2.append(result2)


flights = pd.DataFrame()
featuresArr = [str(col) + '_x' for col in flights_arrivals.columns]
featuresDep = [str(col) + '_y' for col in flights_departures.columns]
```

```
flights[featuresArr] = pd.DataFrame(flights1, columns=featuresArr)
flights[featuresDep] = pd.DataFrame(flights2, columns=featuresDep)

delay_columns = [x for x in flights.columns.values.tolist() if
x.startswith("delays") or x.startswith("airportResources")]
flight_durations = [x for x in flights.columns.values.tolist() if
x.startswith("flightDurations")]
flights[delay_columns] = flights[delay_columns].fillna(0)
flights[flight_durations] =
flights[flight_durations].interpolate(method="linear")
flights = flights.dropna()
print flights.isnull().any()
print len(flights)
flights.to_csv("flights.csv", index=False)
```

ANEXO 2 (FeaturesSelectionAndProcessing.py)

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn import preprocessing
import matplotlib.pyplot as plt
from sklearn import ensemble
import random
import scipy.stats as stats
import seaborn as sbn

sbn.set_style("whitegrid")

def build_features(features, data):

    # First we add numeric variables
    features.append('airportResources.arrivalTerminal_x')

    # Secondly we add categorical variables and transform them into the
numerical ones

features.extend(['BaggageClaimCarrousel','CarrierFsCode','DepartureGate']
)
    le = LabelEncoder().fit(data['airportResources.baggage_x'])

data['BaggageClaimCarrousel']=le.transform(data['airportResources.baggage
_x'])
    le = LabelEncoder().fit(data['carrierFsCode_x'])
    data['CarrierFsCode']=le.transform(data['carrierFsCode_x'])
    le = LabelEncoder().fit(data['airportResources.departureGate_y'])

data['DepartureGate']=le.transform(data['airportResources.departureGate_y
'])


    # Thirdly we decompose date and time into separate columns
    features.extend(['Month', 'Day', 'DayOfWeek', 'WeekOfYear',
'ArrivalTimeMinutes'])
    data['Month'] =
pd.to_datetime(data['arrivalDate.dateLocal_x']).dt.month
    data['Day'] = pd.to_datetime(data['arrivalDate.dateLocal_x']).dt.day
    data['DayOfWeek'] =
pd.to_datetime(data['arrivalDate.dateLocal_x']).dt.dayofweek
    data['WeekOfYear'] =
pd.to_datetime(data['arrivalDate.dateLocal_x']).dt.weekofyear
    data['ArrivalTimeMinutes'] =
60*pd.to_datetime(data['arrivalDate.dateLocal_x']).dt.hour + \

pd.to_datetime(data['arrivalDate.dateLocal_x']).dt.minute

    return data



df = pd.read_csv("flights.csv")

features = []
build_features(features, df)
```

```
print df[features].head()

# for x in range(0, len(features)):
#       plt.figure(x)
#       plt.hist(df[features].values[:,x])
#       plt.title(features[x])
#       plt.xlabel("Value")
#       plt.ylabel("Frequency")
#       plt.show()
#
# for x in range(0, len(features)):
#     print "%s Mean:" %features[x]
#     print df[features].values[:,x].mean(axis=0)
#     print "%s Std.Dev.:" %features[x]
#     print df[features].values[:,x].std(axis=0)

df_scaled = preprocessing.scale(df[features])

print "\nAfter processing:\n"

for x in range(0, len(features)):
    print "%s Mean:" %features[x]
    #print stats.skew(df_scaled[:,x])
    print df_scaled[:,x].mean(axis=0)
    print "%s Std.Dev.:" %features[x]
    print df_scaled[:,x].std(axis=0)

# # Scaled data has zero mean and unit variance
# #print df_scaled.mean(axis=0)
# #print df_scaled.std(axis=0)

turnaroundtime_minutes =
(60*pd.to_datetime(df['operationalTimes.actualGateDeparture.dateLocal_y']
).dt.hour + \

pd.to_datetime(df['operationalTimes.actualGateDeparture.dateLocal_y']).dt
.minute) - \

(60*pd.to_datetime(df['operationalTimes.actualGateArrival.dateLocal_x']).
dt.hour + \

pd.to_datetime(df['operationalTimes.actualGateArrival.dateLocal_x']).dt.m
inute)


# Fit a random forest with (mostly) default parameters to determine
feature importance
random.seed(111)
forest = ensemble.RandomForestRegressor(n_estimators=10,
min_samples_split=2, n_jobs=-1, random_state=111)
forest.fit(df_scaled, turnaroundtime_minutes)
feature_importance = forest.feature_importances_

# Make importances relative to max importance
feature_importance = 100.0 * (feature_importance /
feature_importance.max())

# A threshold below which to drop features from the final data set.
Specifically, this number represents
# the percentage of the most important feature's importance value
fi_threshold = 2
```

```
# Get the indexes of all features over the importance threshold
important_idx = np.where(feature_importance > fi_threshold)[0]

# Create a list of all the feature names above the importance threshold
features = np.array(features)
important_features = features[important_idx]
print "\n", important_features.shape[0], "Important features(>",
fi_threshold, "% of max importance):\n", important_features

# Get the sorted indexes of important features
sorted_idx = np.argsort(feature_importance[important_idx])
print "\nFeatures sorted by importance (ASC):\n",
important_features[sorted_idx]

# Adapted from http://scikit-
learn.org/stable/auto examples/ensemble/plot gradient boosting regression
.html
pos = np.arange(sorted_idx.shape[0]) + .5
plt.subplot(1, 2, 2)
plt.barh(pos, feature_importance[important_idx][sorted_idx],
align='center')
plt.yticks(pos, important_features[sorted_idx])
plt.xlabel('Relative Importance')
plt.title('Variable Importance')
plt.draw()
plt.show()

# Remove non-important features from the feature set, and reorder those
remaining

df_scaled = df_scaled[:, important_idx][:, sorted_idx]

df_scaled = pd.DataFrame(df_scaled)
df_scaled['turnaround_time'] = turnaroundtime_minutes
df_scaled = df_scaled[~df_scaled['turnaround_time']<0]
important_features = np.append(important_features,['turnaround_time'])
sorted_idx = np.append(sorted_idx,len(important_features)-1)
df_scaled.to_csv("flights_transformed.csv",
header=important_features[sorted_idx],index=False)
```

ANEXO 3 (Plotting.py)

```
import pandas as pd
import numpy as np
from sklearn.neighbors import NearestNeighbors

from urllib2 import Request, urlopen
import json
from pandas.io.json import json_normalize

import seaborn as sbn
import matplotlib.pyplot as plt
from numpy import arange
import bisect

sbn.set_style("whitegrid")

def find_nearest(group, match, groupname):
    match = match[match[groupname] ==
group['flightEquipment.tailNumber']]

    if len(match)>0:
        nbrs =
NearestNeighbors(1).fit(match['operationalTimes.actualDateTime'].values[:
, None])
        dist, ind =
nbrs.kneighbors(group['operationalTimes.actualDateTime'])
        group['operationalTimes.actualDateTime1'] =
group['operationalTimes.actualDateTime']
        group['operationalTimes.actualDateTime'] =
float(match['operationalTimes.actualDateTime'].values[ind.ravel()])
        return group
    else:
        return pd.DataFrame()


flights_arrivals = pd.read_csv("flights_arrivals.csv")
flights_departures = pd.read_csv("flights_departures.csv")


column_names_arrivals = [   'airportResources.baggage',
                            'airportResources.arrivalTerminal',
                            'arrivalAirportFsCode',
                            'arrivalDate.dateLocal',
                            'carrierFsCode',

'delays.arrivalGateDelayMinutes','delays.arrivalRunwayDelayMinutes',

'flightDurations.blockMinutes','flightDurations.scheduledBlockMinutes',

'flightDurations.scheduledTaxiInMinutes','flightDurations.taxiInMinutes',

'flightEquipment.scheduledEquipmentIataCode','flightEquipment.tailNumber'
,

'flightId','flightNumber','operationalTimes.actualGateArrival.dateLocal',

'operationalTimes.actualRunwayArrival.dateLocal',
```

```
'operationalTimes.estimatedGateArrival.dateLocal',

'operationalTimes.estimatedRunwayArrival.dateLocal',

'operationalTimes.publishedArrival.dateLocal',

'operationalTimes.scheduledGateArrival.dateLocal',

'schedule.flightType','schedule.serviceClasses','status']

column_names_departures = [ 'airportResources.baggage',
                            'airportResources.departureGate',
                            'airportResources.departureTerminal',
                            'departureAirportFsCode',
                            'departureDate.dateLocal',
                            'carrierFsCode',

'delays.departureGateDelayMinutes','delays.departureRunwayDelayMinutes',

'flightDurations.blockMinutes','flightDurations.scheduledBlockMinutes',

'flightDurations.scheduledTaxiOutMinutes','flightDurations.taxiOutMinutes
',

'flightEquipment.scheduledEquipmentIataCode','flightEquipment.tailNumber'
,

'flightId','flightNumber','operationalTimes.actualGateDeparture.dateLocal
',

'operationalTimes.actualRunwayDeparture.dateLocal',

'operationalTimes.estimatedGateDeparture.dateLocal',

'operationalTimes.estimatedRunwayDeparture.dateLocal',

'operationalTimes.publishedDeparture.dateLocal',

'operationalTimes.scheduledGateDeparture.dateLocal',

'schedule.flightType','schedule.serviceClasses','status']

flights_arrivals = flights_arrivals[column_names_arrivals]
flights_departures = flights_departures[column_names_departures]

flights_arrivals =
flights_arrivals.dropna(subset=['flightEquipment.tailNumber'])
flights_departures =
flights_departures.dropna(subset=['flightEquipment.tailNumber'])

flights_arrivals[column_names_arrivals].to_csv("flights_arrivals.csv")
flights_departures[column_names_departures].to_csv("flights_departures.cs
v")


flights =
flights_arrivals[column_names_arrivals].merge(flights_departures[column_n
ames_departures], on='flightEquipment.tailNumber')


#flights =
```

```
flights.dropna(subset=['flightEquipment.tailNumber','airportResources.dep
artureGate'])
#flights =
flights.drop_duplicates(['flightEquipment.tailNumber','flightDurations.ta
xiInMinutes'],take_last=True)

delay_column_names = [ 'delays.arrivalGateDelayMinutes',
                       'delays.arrivalRunwayDelayMinutes',
                       'airportResources.departureTerminal',
                       'delays.departureGateDelayMinutes',
                       'delays.departureRunwayDelayMinutes']


flights[delay_column_names] = flights[delay_column_names].fillna(0)

times_column_names = [ 'flightDurations.blockMinutes_x',
                       'flightDurations.scheduledBlockMinutes_x',
                       'flightDurations.scheduledTaxiInMinutes',
                       'flightDurations.taxiInMinutes',
                       'flightDurations.blockMinutes_y',
                       'flightDurations.scheduledTaxiOutMinutes',
                       'flightDurations.taxiOutMinutes']

flights[times_column_names] =
flights[times_column_names].interpolate(method="linear")

flights = flights.dropna()


new_columns = flights_arrivals.columns.values
new_columns[15] = 'operationalTimes.actualDateTime'
flights_arrivals.columns = new_columns

new_columns = flights_departures.columns.values
new_columns[16] = 'operationalTimes.actualDateTime'
flights_departures.columns = new_columns

flights_arrivals =
flights_arrivals.dropna(subset=['operationalTimes.actualDateTime'])
flights_departures =
flights_departures.dropna(subset=['operationalTimes.actualDateTime'])

flights_arrivals['operationalTimes.actualDateTime'] =
pd.to_datetime(flights_arrivals['operationalTimes.actualDateTime'])
flights_departures['operationalTimes.actualDateTime'] =
pd.to_datetime(flights_departures['operationalTimes.actualDateTime'])

flights_arrivals['operationalTimes.actualDateTime'] =
(flights_arrivals['operationalTimes.actualDateTime'] -
flights_arrivals['operationalTimes.actualDateTime'].min()) /
np.timedelta64(1,'D')
flights_departures['operationalTimes.actualDateTime'] =
(flights_departures['operationalTimes.actualDateTime'] -
flights_departures['operationalTimes.actualDateTime'].min()) /
np.timedelta64(1,'D')

flights_arrs = []
for index, row in flights_arrivals.iterrows():
    result =
find_nearest(row,flights_departures,'flightEquipment.tailNumber')
    if len(result) > 0:
```

```
            flights_arrs.append(result)

    flights_arrs = pd.DataFrame(flights_arrs)

    flights = pd.merge(flights_arrs, flights_departures,
    on=['flightEquipment.tailNumber', 'operationalTimes.actualDateTime'])

    #print flights.head()

    flights =
    flights.drop_duplicates(['flightId_x','flightId_y'],take_last=True)

    print len(flights)

    #flights[delay_column_names] = flights[delay_column_names].fillna(0)
    #flights[times_column_names] =
    flights[times column names].interpolate(method="linear")

    flights.to_csv("flights.csv")

    def scatterplot(x,y,x_title,y_title):
        plt.plot(x,y,'b.')
        plt.xlabel(x_title)
        plt.ylabel(y_title)
        plt.xlim(min(x)-1,max(x)+1)
        plt.ylim(min(y)-1,max(y)+1)
        plt.show()

    def barplot(labels,data,x_title,y_title):
        pos=arange(len(data))
        plt.xlabel(x_title)
        plt.ylabel(y_title)
        plt.xticks(pos+0.4,labels)
        plt.bar(pos,data)
        plt.show()

    def histplot(data,x_title,y_title,bins=None,nbins=10):
        minx,maxx=min(data),max(data)
        space=(maxx-minx)/float(nbins)
        if not bins: bins=arange(minx,maxx,space)
        binned=[bisect.bisect(bins,x) for x in data]
        l=['%.1f'%x for x in list(bins)+[maxx]] if space<1 else [str(int(x))
    for
        x in list(bins)+[maxx]]
        displab=[x+'-'+y for x,y in zip(l[:-1],l[1:])]
        barplot(displab,[binned.count(x+1) for x in
    range(len(bins))],x_title,y_title)

    def barchart(x,y,x_title,y_title,numbins=10):
        datarange=max(x)-min(x)
        bin_width=float(datarange)/numbins
        pos=min(x)
        bins=[0 for i in range(numbins+1)]

        for i in range(numbins):
            bins[i]=pos
            pos+=bin_width
        bins[numbins]=max(x)+1
        binsum=[0 for i in range(numbins)]
        bincount=[0 for i in range(numbins)]
        binaverage=[0 for i in range(numbins)]
```

```
    for i in range(numbins):
        for j in range(len(x)):
            if x[j]>=bins[i] and x[j]<bins[i+1]:
                bincount[i]+=1
                binsum[i]+=y[j]

    for i in range(numbins):
        if (bincount[i]>0):
            binaverage[i]=float(binsum[i])/bincount[i]
        else:
            binaverage[i]=0
    barplot(range(numbins),binaverage,x_title,y_title)

#barchart(flights['carrierFsCode_x'].values,
 #        flights['delays.departureGateDelayMinutes'].values,
  #       "CARRIER", "ARRIVAL DELAY",
len(flights['carrierFsCode_x'].unique()))

histplot(flights['flightDurations.scheduledBlockMinutes_x'],'DURACION
VUELOS','NUMERO DE VUELOS')


#----------FLIGHT EQUIPMENT CHART-------------------------------------

temp_flights_equipment = pd.DataFrame()
temp_flights_equipment['flightEquipment.scheduledEquipmentIataCode_x'] =
flights['flightEquipment.scheduledEquipmentIataCode_x']
temp_flights_equipment['delays.departureGateDelayMinutes'] =
flights['delays.departureGateDelayMinutes']
temp_flights_equipment = temp_flights_equipment.dropna()

data_equipment = pd.DataFrame()
data_equipment['flightEquipment.scheduledEquipmentIataCode_x'] =
temp_flights_equipment['flightEquipment.scheduledEquipmentIataCode_x']
data_equipment['delays.departureGateDelayMinutes'] =
temp_flights_equipment['delays.departureGateDelayMinutes']
equipment_group =
data_equipment.groupby('flightEquipment.scheduledEquipmentIataCode_x')
delays_totals = equipment_group.mean()
delays_totals.sort(columns='delays.departureGateDelayMinutes')
ax = delays_totals.plot(kind='bar', title ="Restrasos por Modelos de
Avion",legend=False)
ax.set_xlabel("Modelos de Avion")
ax.set_ylabel("Retraso Medio, m.")
plt.show()

#----------AIRLINE CHART----------------------------------------

temp_airline = pd.DataFrame()
temp_airline['carrierFsCode_x'] = flights['carrierFsCode_x']
temp_airline['delays.departureGateDelayMinutes'] =
flights['delays.departureGateDelayMinutes']
temp_airline = temp_airline.dropna()

data_airline = pd.DataFrame()
data_airline['carrierFsCode_x'] = temp_airline['carrierFsCode_x']
data_airline['delays.departureGateDelayMinutes'] =
temp_airline['delays.departureGateDelayMinutes']
airline_group = data_airline.groupby('carrierFsCode_x')
delays_totals_airline = airline_group.mean()
```

```
delays_totals_airline.sort(columns='delays.departureGateDelayMinutes')
ax = delays_totals_airline.plot(kind='bar', title ="Retrasos por
Companias",legend=False)
ax.set_xlabel("Companias")
ax.set_ylabel("Retraso Medio, m.")
plt.show()

#----------RESOURCE BAGGAGE CHART-----------------------------------

temp_baggage = pd.DataFrame()
temp_baggage['airportResources.baggage_x'] =
flights['airportResources.baggage_x']
temp_baggage['delays.departureGateDelayMinutes'] =
flights['delays.departureGateDelayMinutes']
temp_baggage = temp_baggage.dropna()

data_baggage = pd.DataFrame()
data_baggage['airportResources.baggage_x'] =
temp_baggage['airportResources.baggage_x']
data_baggage['delays.departureGateDelayMinutes'] =
temp_baggage['delays.departureGateDelayMinutes']
baggage_group = data_baggage.groupby('airportResources.baggage_x')
delays_totals_baggage = baggage_group.mean()
delays_totals_baggage.sort(columns='delays.departureGateDelayMinutes')
ax = delays_totals_baggage.plot(kind='bar', title ="Restrasos por Cinta
Recogida Equipaje",legend=False)
ax.set_xlabel("Cinta Recoida Equipaje")
ax.set_ylabel("Retraso Medio, m.")
plt.show()

#----------DEPARTURE GATE CHART-------------------------------------

temp_gate = pd.DataFrame()
temp_gate['airportResources.departureGate'] =
flights['airportResources.departureGate']
temp_gate['delays.departureGateDelayMinutes'] =
flights['delays.departureGateDelayMinutes']
temp_gate = temp_gate.dropna()

data_gate = pd.DataFrame()
data_gate['airportResources.departureGate'] =
temp_gate['airportResources.departureGate']
data_gate['delays.departureGateDelayMinutes'] =
temp_gate['delays.departureGateDelayMinutes']
gate_group = data_gate.groupby('airportResources.departureGate')
delays_totals_gate = gate_group.mean()
delays_totals_gate.sort(columns='delays.departureGateDelayMinutes')
ax = delays_totals_gate.plot(kind='bar', title ="Restrasos por Puerta
Embarque",legend=False)
ax.set_xlabel("Puerta Embarque")
ax.set_ylabel("Retraso Medio, m.")
plt.show()

#----------FLIGHT DURATION CHART------------------------------------

temp_duration = pd.DataFrame()
temp_duration['flightDurations.blockMinutes_x'] =
flights['flightDurations.blockMinutes_x']
temp_duration['delays.departureGateDelayMinutes'] =
flights['delays.departureGateDelayMinutes']
temp_duration = temp_duration.dropna()
```

```
barchart(temp_duration['flightDurations.blockMinutes_x'].values,temp_dura
tion['delays.departureGateDelayMinutes'].values,"Duracion Vuelo, h.",
"Retraso Medio, m.")

#----------TYPES OF CLASSES CHART----------------------------------

temp_class = pd.DataFrame()
temp_class['schedule.serviceClasses_x'] =
flights['schedule.serviceClasses_x']
temp_class['delays.departureGateDelayMinutes'] =
flights['delays.departureGateDelayMinutes']
temp_class = temp_class.dropna()

data_class = pd.DataFrame()
data_class['schedule.serviceClasses_x'] =
temp_class['schedule.serviceClasses_x']
data_class['delays.departureGateDelayMinutes'] =
temp_class['delays.departureGateDelayMinutes']
class_group = data_class.groupby('schedule.serviceClasses_x')
delays_totals_class = class_group.mean()
delays_totals_class.sort(columns='delays.departureGateDelayMinutes')
ax = delays_totals_class.plot(kind='bar', title ="Restrasos por Tipos de
Clases",legend=False)
ax.set_xlabel("Clases")
ax.set_ylabel("Retraso Medio, m.")
plt.show()

#----------TERMINAL CHART-------------------------------------

temp_terminal = pd.DataFrame()
temp_terminal['airportResources.arrivalTerminal'] =
flights['airportResources.arrivalTerminal']
temp_terminal['delays.departureGateDelayMinutes'] =
flights['delays.departureGateDelayMinutes']
temp_terminal = temp_terminal.dropna()

data_terminal = pd.DataFrame()
data_terminal['airportResources.arrivalTerminal'] =
temp_terminal['airportResources.arrivalTerminal']
data_terminal['delays.departureGateDelayMinutes'] =
temp_terminal['delays.departureGateDelayMinutes']
terminal_group =
data_terminal.groupby('airportResources.arrivalTerminal')
delays_totals_terminal = terminal_group.mean()
delays_totals_terminal.sort(columns='delays.departureGateDelayMinutes')
ax = delays_totals_terminal.plot(kind='bar', title ="Restrasos por
Terminal",legend=False)
ax.set_xlabel("Terminal")
ax.set_ylabel("Retraso Medio, m.")
plt.show()
```