Modelización y mantenimiento predictivo

PID_00264732

Xavi Font

Tiempo mínimo de dedicación recomendado: 4 horas

Universitat Oberta de Catalunya

Xavi Font

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por el profesor: Jordi Ayza Graells (2019)

Primera edición: marzo 2019

Av. Tibidabo, 39-43, 08035 Barcelona
Diseño: Manel Andreu
Realización editorial: Oberta UOC Publishing, SL

Índice

Introducción

La importancia de los modelos predictivos se basa en la capacidad de estos para predecir. Obviamente, el estudio del pasado puede abrir las puertas a entender y predecir el futuro. Entendemos que los modelos que se basan en el concepto de *predictive data analytics* justamente se caracterizan por intentar entender el pasado (datos históricos) para hacer predicciones.

En la mayoría de las ocasiones se busca la capacidad para predecir el futuro. Sería el caso de modelado de series temporales, donde el interés está en saber la repuesta en el instante siguiente (a los datos actuales) para así actuar de modo que nos adelantamos al futuro. En otras ocasiones el estudio del pasado puede ayudarnos a entender qué clientes se darán de baja de nuestros servicios. Conocer a estos clientes puede ser interesante para realizar actuaciones que permitan revertir la situación.

El conjunto de los sectores donde podemos encontrar aplicaciones de modelización y mantenimiento predictivo es numeroso. Por citar unos cuantos ejemplos:

- **Predicción de cotizaciones**: la utilización de modelos con la finalidad de ser capaces de predecir qué tendencia seguirá una acción es en ocasiones el grial de muchos econometristas e inversores. Si el modelo fuera correcto (mínimimente), tendríamos ante nosotros la gallina de los huevos de oro. La inversión sería siempre favorable (por desgracia, no es fácil encontrar un modelo de este tipo).

- **Predicción de rendimiento de motores**: evaluar y estudiar los parámetros de los motores puede ayudar a identificar posibles problemas y evitar la aparición de estos realizando alguna acción correctiva.

- **Predicción de dosis**: en estudios farmacológicos y médicos la cuestión que en ocasiones resulta crítica radica en la dosis que prescribir. El uso de modelos que tengan en consideración resultados pasados puede ayudar a identificar patrones para la correcta estimación de la dosis para un determinado paciente.

- **Clasificación de documentos**: de nuevo los datos pasados nos pueden ayudar a clasificar y categorizar correctamente documentos o correo electrónico, a realizar minería de opiniones (*sentiment analysis*).

Se puede apreciar que algunas de las situaciones descritas las podríamos fácilmente colocar en algunos de los métodos de ML que se han visto en los módulos anteriores. Las dos características subyacentes a los modelos predictivos son que el modelo se construye para hacer predicciones y que el modelo es entrenado con datos históricos.

La deducción de qué tipo de procedimiento de aprendizaje se utiliza en esta tipología de problemas es clara: métodos supervisados.

La parte más interesante de este módulo es que actúa como repaso de algunos de los puntos ya vistos e introduce un capítulo muy importante asociado a los modelos de procesos estocásticos. El objetivo es poner en evidencia la frase célebre del matemático y estadístico George E. P. Box:

> «all models are wrong, but some are useful».

De nuevo se repetirá el proceso típico donde debemos automatizar el aprendizaje del modelo a partir de un conjunto de datos históricos para revelar las posibles relaciones entre las características del conjunto de datos históricos y nuestra variable *target*.

Se presentará la metodología Cross Industry Standard Process for Data Mining (CRISP-DM). Este fue un proyecto de la Unión Europea que involucró a las empresas SPSS, Teradata, Daimler AG, NCR y Ohra y que se caracteriza por ser no propietario.

Las herramientas que permiten resolver este tipo de situaciones las podemos dividir en dos tipos, como ya se comentó anteriormente:

- **Soluciones basadas en aplicaciones**: donde podemos encontrar opciones de pago, como IBM SPSS, SAS interprise miner o Analytics platform, o también soluciones tipo *open source*: weka, knime, RapidMiner Studio.

- **Soluciones basadas en lenguajes de programación**: como R, Python o Julia.

El enfoque presentado será sobre RStudio y R.

Objectivos

Los objetivos principales que persigue este módulo son diversos:

1. Ser capaz de utilizar datos históricos para generar un modelo predictivo que resuelva de manera satisfactoria nuestra necesidad.

2. Identificar correctamente la tipología del problema y qué solución damos. En este sentido, debemos diagnosticar correctamente para proponer la técnica adecuada. Será importante conocer las diferentes estrategias para modelar procesos estocásticos, al menos los más conocidos:
   - AR
   - MA
   - ARMA
   - ARIMA
   - SARIMA

3. Entender y aplicar la metodología CRISP-DM o una similar para resolver problemas complejos de modelización predictiva.

4. Conocer estrategias dentro de algunos de estos enfoques:
   - Aprendizaje basado en criterios de información.
   - Aprendizaje basado en similitud.
   - Aprendizaje basado en probabilidad.
   - Aprendizaje basado en error.

1. Introducción a la ingeniería de la predicción

1.1. Introducción

El uso de los modelos analíticos de predicción resulta de gran utilidad para toda la problemática asociada con el mantenimiento de equipamientos industriales. La problemática de no detectar fallos o roturas inesperadas tiene un coste enorme, tanto para el cliente como para la imagen de la empresa.

Desde la perspectiva del negocio, se desea resolver problemas asociados a la:

- Detección de anomalías en el rendimiento o funcionamiento del equipo o sistema industrial.
- Estimación de la probabilidad de fallo antes de que ocurra.
- Estimación de la vida útil restante de un equipo o sistema.
- Identificación de las causas principales de una rotura o fallo de un equipo o sistema.

En todas estas situaciones es crítico disponer de un conjunto de datos históricos y de suficiente tamaño para que se pueda resolver con los procedimientos que describiremos en el módulo.

A diferencia del mantenimiento preventivo, el mantenimiento predictivo se basa en datos objetivos para determinar la probabilidad de rotura de una sistema antes de que ocurra. Gracias al uso de herramientas de análisis predictivo, podremos realizar predicciones sobre resultados futuros.

1.1.1. Casos de negocio

Hay muchos ejemplos en el sector industrial que requieren soluciones predictivas.

- **Rotura de circuitos eléctricos de potencia**: se requiere un sistema capaz de predecir la rotura antes de que suceda. O qué situación provoca este final para tomar actuaciones que eviten que suceda.

- **ATM sin *stock***: se requiere un sistema que indique la cantidad de dinero que debe contener para que no haya ningún cliente desatendido. Disponer de datos históricos de funcionamiento de diferentes dispositivos ATM pue-

de facilitar la construcción de un modelo predictivo útil y eficiente para la entidad financiera que los utiliza.

- **Motor en turbinas**: especialmente crítico en turbinas aerogeneradoras por la dificultad y los costes asociados. Es de vital importancia evitar problemas en este tipo de sistemas.

Existen diferentes propuestas comerciales que dan solución a este tipo de situaciones.

Por ejemplo, la solución de SAP (SAP Leonardo IoT for Asset Management) tiene como módulos asociados a la ingeniería predictiva los que permiten simular el equipo o sistema. Esto es: FEDEM (Finite Element Dynamics in Elastic Mechanisms) básicamente para calcular fuerzas, resistencias y zonas de estrés en las estructuras del equipamiento. Añade herramientas de comportamiento de estructuras, aerodinámica y vibraciones. Evidentemente, dispone del módulo de ML, que es sobre el que se basan la mayoría de los modelos predictivos analíticos.

Los casos de uso más comunes para el mantenimiento predictivo se encuentran en alguna de las siguientes industrias:

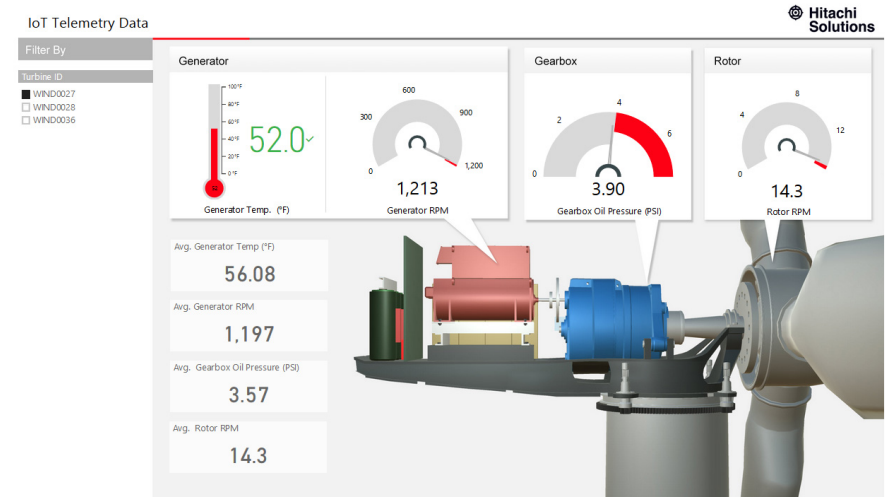
- Aeroespacial
- Generación de energía
- Fabricación
- ATM
- Ascensores
- Sistemas HVAC (*heating ventilation air conditioning*)

¿Cuál es el principal motivo?

Hay dos razones principales. La primera tiene que ver con la calidad de los datos disponibles en estas industrias. No solo se dispone de medidas contrastadas y de calidad, sino de un historial de años de funcionamiento. Estos datos históricos son clave para desarrollar buenos modelos predictivos. El segundo aspecto tiene que ver con costes y ROI. El uso de ingeniería predictiva permite ahorrar grandes sumas de dinero e incrementar la calidad de los servicios a los clientes.

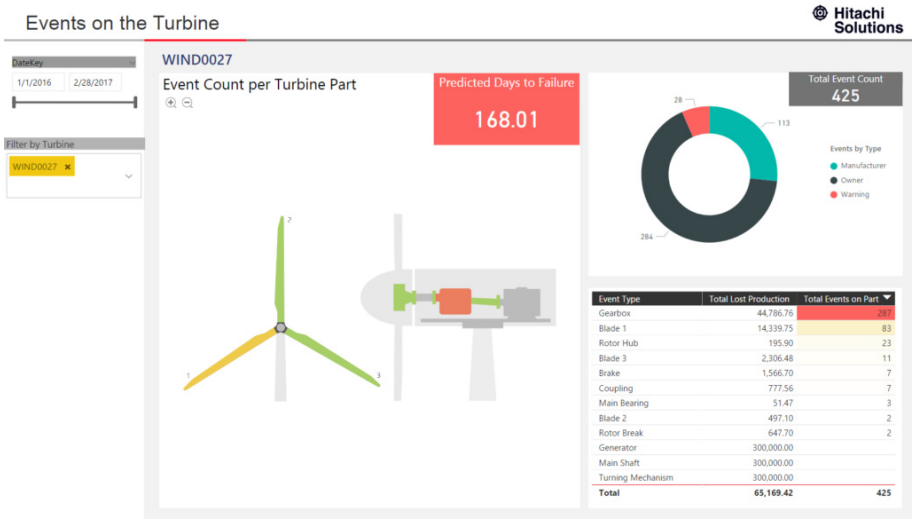
Otro ejemplo interesante es el de la empresa Hitachi con las turbinas aerogeneradoras. En la figura 1 se puede apreciar un cuadro de mando con las métricas más relevantes de la turbina. Hitachi dispone de otra vista (ver) donde el modelo predictivo hace una estimación de los días restantes hasta la próxima rotura.

Figura 1. Motor de turbina en aerogenerador



Fuente: https://us.hitachi-solutions.com/blog/6-tools-for-a-successful-predictive-maintenance-program/

Figura 2. Representación del cuadro de mando con la estimación de próxima rotura en días



Fuente: https://us.hitachi-solutions.com/blog/6-tools-for-a-successful-predictive-maintenance-program/

Aclaración

Algunos de los ejemplos no reflejan necesariamente las mejores prácticas y deben verse solo con fines ilustrativos. Todos los análisis también se realizarán sin ningún procesamiento previo de los datos.

Otra empresa del sector industrial, Siemens, señala que:

> «Hoy gracias al avance de la digitalización, la recopilación continua y el análisis inteligente de los datos operativos y de proceso podemos predecir el momento óptimo para el mantenimiento de los componentes de la máquina o sistema. La evaluación de estos datos también ayuda a aumentar la transparencia y se beneficia de un aumento sustancial de la productividad y la eficiencia, gracias a nuestros servicios predictivos.»

Enlace de interés

Podéis encontrar información adicional en el siguiente enlace: https://pwc.to/2NgQGV3

Este es el denominador común de la utilización de este tipo de soluciones.

1.2. Cross Industry Standard Process for Data Mining

Tener un proceso o estándar, no propietario, que ha sido verificado y aplicado en la industria permite desarrollar proyectos del ámbito de la ingeniería predictiva con más garantías.

La metodología se caracteriza por pasar por seis fases, que son las que están representadas en la figura 3 y que a continuación detallamos.

1) Objetivos de negocio

2) Comprensión de datos

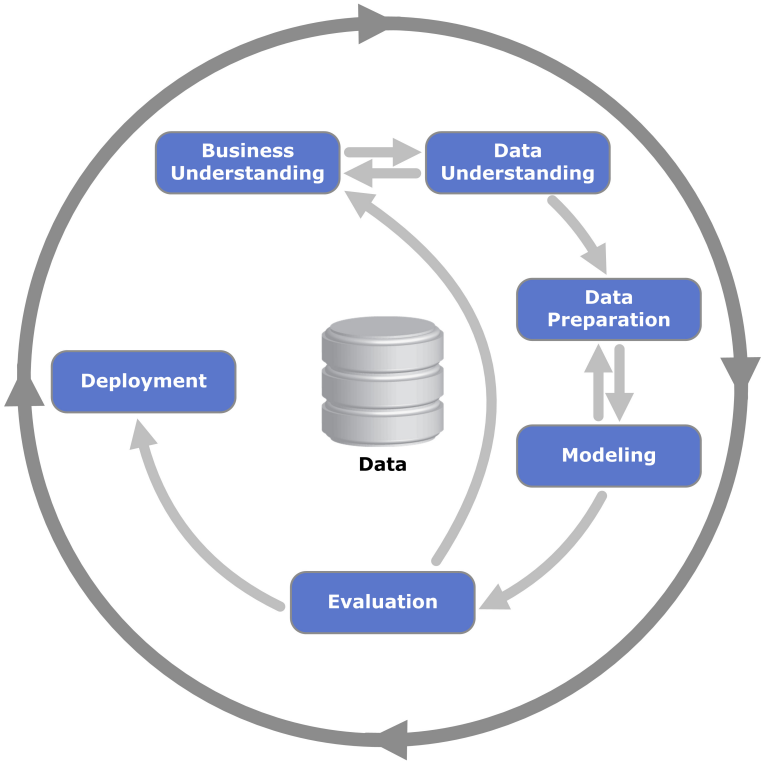
3) Preparación de los datos

4) Modelización

5) Evaluación

6) Despliegue

Figura 3. Representación de la metodología CRISP - DM



1.2.1. Etapa 1: Determinar los objetivos del negocio

En todo proyecto hay un inicio. En nuestro caso, el primer paso es la comprensión del negocio donde está el problema definido.

Implica analizar y evaluar todos los elemento relacionados con el problema para evitar situaciones que puedan producir las respuestas correctas a las preguntas incorrectas.

Por tanto, se deben realizar las siguientes tareas:

En primer lugar, especificar las salidas que debe generar el proyecto.

1) Definir el conjunto de objetivos.

2) Describir la planificación del proyecto.

3) Definir métricas de seguimiento y/o éxito.

En segundo lugar, evaluar la situación actual. Esto implica identificar recursos, restricciones, suposiciones, actores y otros factores que el análisis de datos y plan de proyecto deben considerar.

1) Inventario de recursos.

2) Requisitos, suposiciones y restricciones.

3) Riesgos y contingencias.

4) Costes y beneficios.

En tercer lugar, identificación y descripción de los objetivos de análisis del proyecto. Esto quiere decir establecer los objetivos del proyecto en términos técnicos.

En cuarto lugar, la planificación del proyecto. Se deben especificar los pasos que hay que realizar y la selección de herramientas y técnicas.

Debemos resaltar que en la finalización de esta etapa está ya diseñada la propuesta de solución analítica.

1.2.2. Etapa 2: Comprensión de datos

Se debe resaltar la importancia de comprender la naturaleza de los datos disponibles, así como su procedencia y tipología. Esta etapa puede requerir la carga de datos en nuestra plataforma de análisis o en nuestro entorno de desarrollo.

El conjunto de acciones que podemos realizar en esta fase es:

1) Descripción de los datos.

2) EDA.

3) Verificación de la calidad de los datos.

4) Información de calidad de los datos.

1.2.3. Etapa 3: Preparación de los datos

La construcción de los modelos predictivos requiere unas necesidades específicas de los datos. En el caso ya comentado sobre *data carpentry* se describió el concepto de *tidy*; en el caso de la metodología CRISP hay un concepto similar, llamado *analytics base table* (ABT), que es la estructura en la que organizamos los datos para poder ser correctamente utilizados en las técnicas de ML.

¿Qué operaciones pueden desarrollarse en esta etapa?:

1) Identificar y tratar errores o valores *missing*.

2) Construcción e integración de nuestros datos ABT.

1.2.4. Etapa 4: Modelización de los datos

Si bien es posible que ya hayamos seleccionado una tipología de solución y propuesta de modelo durante la fase de comprensión del negocio, en esta etapa se seleccionará la técnica de modelado específica. Es decir, se entra en detalle, de modo que en caso de tener más propuestas de modelado se tendrán en cuenta igualmente.

Cada una de estas técnicas puede ser descrita y se puede añadir el conjunto de suposiciones que deben garantizarse. También se pueden generar los mecanismos para testear la validez del modelo.

Esta etapa, que en ocasiones es la más esperada, debe:

1) Identificar la lista de parámetros del modelo.

2) Ajustar el modelo.

3) Interpretar el modelo.

4) Verificar el modelo.

5) Revisar el modelo y/o reajustar parámetros.

Esta fase puede finalizar con una lista de las preferencias sobre los modelos ajustados.

1.2.5. Etapa 5: Evaluación de los datos

En la etapa anterior nos ocupamos de la precisión y la generalidad del modelo. Durante este etapa, se debe evaluar el grado en el que el modelo cumple con los objetivos del negocio.

Hay que determinar si existe alguna razón comercial por la que este modelo no es de interés o probar si los modelos dan problemas en entornos reales de producción en cuanto a tiempo de respuesta u otros aspectos.

1.2.6. Etapa 6: Despliegue del modelo

Esta etapa implica desarrollar un plan de implementación donde se incluyen los pasos necesarios y realizarlos para su correcta utilización.

Entre algunas de las tareas que se deben realizar encontramos las siguientes:

1) Mantenimiento y gestión.

2) Generación de un informe final del proyecto.

3) Revisar el proyecto para posibles mejoras.

Enlaces de interés

Para información adicional sobre la metodología podéis consultar los siguientes enlaces:
https://bit.ly/2zWL92p
https://bit.ly/2srmK1m

1.3. *Predictive analytics* bajo criterios de información

La idea principal de los métodos predictivos basados en la teoría de la información es la de utilizar conceptos de la teoría de la información desarrollados por C.E. Shannon. La mayoría de dichos métodos utilizan estos criterios para construir un árbol de decisión.

La construcción se basa en el índice de Gini:

$$G_m = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

donde K es el número de clases del nodo m.

Otra de las medidas utilizadas es la ganancia de la información:

$$D = -\sum_{k=1}^{K} \hat{p}_{mk}\ log(\hat{p}_{mk})$$

R dispone de un conjunto de librerías que permiten ajustar árboles de decisión. Podemos recomendar las siguientes:

- **rpart**: uso de árboles de regresión (y del índice de Gini para decidir si divide).

- **randomForest (*bagging*)**: algoritmos para modelos de tipo *random forest*. Con la idea de ajustar diversos árboles para formar un bosque, todos los

árboles participan en la generación de su predicción y es el sistema el que finalmente promedia las predicciones.

- **gbm**: para algoritmos de tipo *gradient boosting*. La idea es ajustar secuencialmente múltiples modelos sencillos, llamados *weak learners* (idea del algoritmo de Viola Jones), de modo que cada modelo aprende de los errores del anterior. Al final se ajusta la media de todas las predicciones.

- **xgboost**: uso de *extreme gradient boosting*. Algoritmo más exitoso en la actualidad en las competiciones de Kaggle junto a los modelos de *deep learning*.

Algunos de los métodos presentados tienen mejor rendimiento cuando se utilizan técnicas de ensamblaje (*ensemble*). En general, intentan combinar múltiples modelos para mejorar los problemas en el *bias* y varianza de los árboles de decisión. Entre estas técnicas encontramos el *bagging* y el *boosting*.

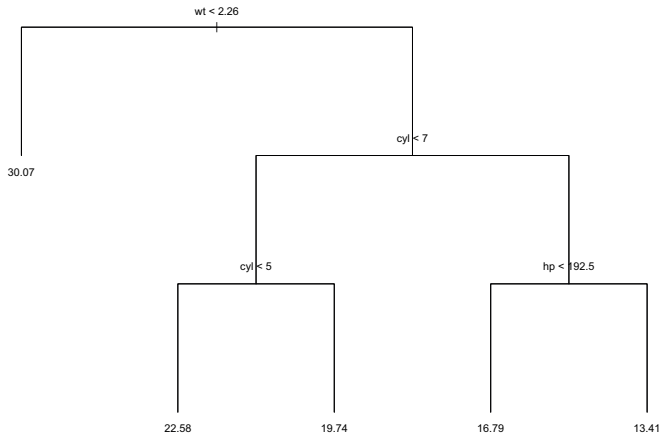
1.3.1. Ejemplo de árbol de decisión

Si nos situamos en el problema de decidir según el consumo de los vehículos del fichero mtcars, podemos realizar las siguientes operaciones:

```
library(tree)
data(mtcars)
tr <- tree(mpg ~ ., mtcars)
tr
node), split, n, deviance, yval
      * denotes terminal node

 1) root 32 1126.000 20.09
   2) wt < 2.26 6   44.550 30.07 *
   3) wt > 2.26 26  346.600 17.79
     6) cyl < 7 12   42.120 20.92
      12) cyl < 5 5    5.968 22.58 *
      13) cyl > 5 7   12.680 19.74 *
     7) cyl > 7 14   85.200 15.10
      14) hp < 192.5 7   16.590 16.79 *
      15) hp > 192.5 7   28.830 13.41 *
plot(tr,type="uniform"); text(tr,pretty=0)
```

Que gráficamente es más fácil de interpretar:

Figura 4. Árbol de decisión para el consumo



1.3.2. Ejemplo de método *bagging*

Una de las características interesantes de este tipo de procedimiento es que nos da una lista de importancia de las variables explicativas. Se puede obtener una tabla con los valores o una representación gráfica de esta.

```
library(randomForest)

mtcars.rf <- randomForest(mpg ~ ., data=mtcars, ntree=1000, keep.forest=FALSE,
                          importance=TRUE)
summary(mtcars.rf)
plot(mtcars.rf, log="y")
varImpPlot(mtcars.rf)
mtcars.rf$importance
        %IncMSE IncNodePurity
cyl   6.7426631     158.39810
disp 10.7436916     252.80123
hp    8.5107641     197.89095
drat  1.3033640      76.32746
wt    8.9464055     245.19991
qsec  0.7441381      37.07004
vs    0.6992265      28.16353
am    0.2259983      14.07587
gear  0.4134914      16.31108
carb  1.1002136      29.54994
```

1.3.3. Ejemplo de método *boosting*

De nuevo podemos analizar este ejemplo con los datos del fichero de datos mtcars:

```
library(gbm)
data(mtcars)
mtcars.gmb <- gbm(mpg~.,
         data=mtcars,
         distribution = "gaussian",
         interaction.depth=4,
         bag.fraction=0.7,
         n.trees = 10000,
         shrinkage = 0.01)

summary(mtcars.gmb)
```

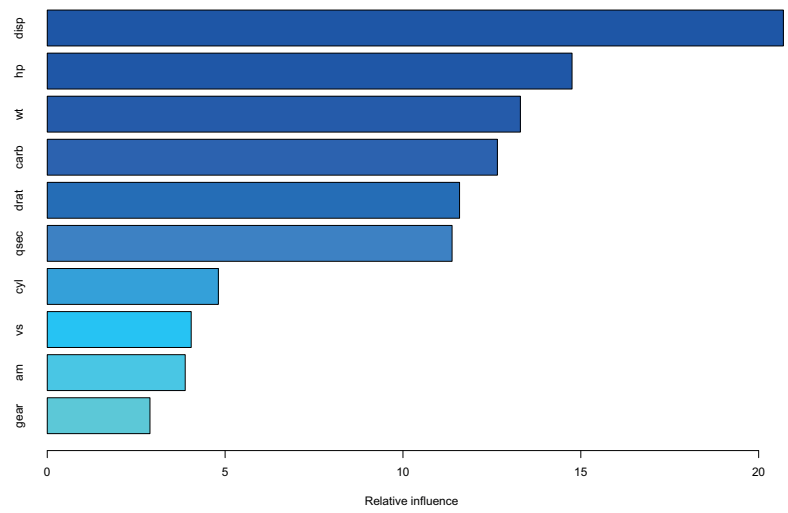
```
      var   rel.inf
disp disp 20.694756
hp     hp 14.753831
wt     wt 13.300953
carb carb 12.654673
drat drat 11.591178
qsec qsec 11.381032
cyl   cyl  4.810315
vs     vs  4.047032
am     am  3.878635
gear gear  2.887595
```

Nuevamente obtenemos una lista de la importancia de los predictores. Atención: sin haber realizado evaluación del modelo alguna.

Enlace de interés

Para complementar la información podéis consultar el siguiente enlace: http://uc-r.github.io/gbm_regression

Figura 5. Importancia relativa de las variables explicativas



1.4. *Predictive analytics* bajo criterios de similitud

Se basan en el algoritmo denominado *k-d tree nearest neighbor retrieval algorithm*.

Enlace de interés

Para complementar la información podéis consultar el siguiente enlace: http://theory.stanford.edu/ rinap/papers/kdtreelatin.pdf

Esta aproximación no se detalla, aunque se indican algunas referencias de interés.

1.5. *Predictive analytics* bajo criterios de probabilidad

La aproximación basada en conceptos relativos a la probabilidad tiene su fundamentación en el teorema de Bayes. Básicamente, nos ofrece una formulación de la probabilidad condicionada reformulada:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

A esta probabilidad también se le denomina *posterior probability*.

Tenemos dos algoritmos dentro de este grupo: Naive Bayes y las redes neuronales.

Naive Bayes

Como su nombre indica, se basa en el concepto bayesiano de la probabilidad. Parte de una suposición que en pocas ocasiones se da y que es la de que el conjunto de variables predictoras son independientes entre sí. Veamos una breve formulación del problema.

Dado un conjunto de factores o variables : X_1,X_2,$X_3 \ldots$,X_n y dado un conjunto de categorías: C_1,C_2,$C_3 \ldots$,C_k que son las que puede tomar la variable target y, una vez observada una instancia: $X_1 = x_1$, $X_2 = x_2$, $X_3 = x_3, \ldots, X_n = x_n$, podemos preguntarnos por la probabilidad de que la variable repuesta sea C_1 sabiendo que se ha observado esta instancia. Es decir:

$$P(C_1|x_1,x_2,x_3,\ldots,x_n) = \frac{P(C_1)P(x_1,x_2,\ldots,x_n|C_1)}{P(x_1,x_2,\ldots,x_n)}$$

Si observamos el conjunto total de instancias, al final la expresión resultante será:

$$P(C_k|x_1,x_2,x_3,\ldots,x_n) = \frac{P(C_k)\prod_{j=1}^{n} P(x_j|C_k)}{P(x_1,x_2,\ldots,x_n)}$$

Que podemos maximizar. Es decir, para cada valor de C_k evaluamos esta expresión y nos quedamos con el valor de probabilidad más alto. Dentro de la expresión anterior hay dos interpretaciones interesantes:

- $P(C_k)$ es lo que se denomina *prior probability*.
- $\prod_{j=1}^{n} P(x_j|C_k)$ es lo que se denomina verosimilitud (*likelihood*).

La librería que permite aplicar un modelo Naive Bayes está en e1071:

```
#Loading the library
library(e1071)
?naiveBayes
```

Redes bayesianas

Las redes bayesianas son el segundo ejemplo de uso de aprendizaje probabilístico.

R tiene diversas librerías para implementar modelos bayesianos. Recomendamos:

- bnlearn: para la modelización de redes bayesianas.
- graph y Rgraphviz para la visualización.
- gRain para la inferencia.

Sin entrar en profundidad, la idea que encontramos tras este tipo de modelos es representar una distribución de probabilidad como un grafo probabilístico acíclico dirigido (DAG) con las siguientes propiedades:

- **Grafo**: nodos y bordes (arcos) denotan variables y dependencias, respectivamente.

- **Dirigido**: las flechas representan las direcciones de las relaciones entre los nodos.

- **Acíclico**: si trazas flechas con un lápiz, no puedes volver al mismo nodo sin levantar tu lápiz.

- **Probabilístico**: cada nodo tiene una probabilidad asociada que puede ser influenciada.

1.6. *Predictive analytics* bajo criterios de error

En este grupo de algoritmos predictivos, que basan su proceso de aprendizaje en la minimización de una función asociada al error, podemos encontrar básicamente tres grandes grupos de métodos.

- Modelos de regresión lineal.
- Modelos de regresión logística.
- Modelos SVM (*support vector machines*).

Los modelos de regresión lineal y logística ya han sido vistos. Nos centramos brevemente en los modelos SVM, que están implementados en la librería e1071 ya presentada para el caso de modelos Naive Bayes.

1.6.1. *Suport vector machines*

Los modelos SVM inicialmente se desarrollaron como método de clasificación binaria. Tenemos dos grupos y el objetivo de este tipo de método es encontrar un hiperplano que permita separar los dos grupos haciendo máximo el margen entre el hiperplano y los puntos.

Si tenemos un conjunto de puntos (observación multidimensional):

$$\mathbf{x} = (x_1, x_2, ..., x_p)$$

Podemos definir un hiperplano:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p = 0$$

que para los puntos del plano da exactamente 0 (cumplen la ecuación del hiperplano), pero para cualquier punto que consideremos nos puede dar una inecuación en sentido positivo o negativo. Es decir:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p < 0$$

O bien:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p > 0$$

Si disponemos de un conjunto de n observaciones, y de p predictores con una variable *target* binaria (dos niveles), entonces podemos emplear hiperplanos para construir un clasificador.

Podemos imaginar que el *target* se codifica como +1 (valor positivo) o –1 (valor negativo). Entonces,

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p > 0, \; si \; y_i = 1$$

o

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p < 0, \; si \; y_i = -1$$

Observad que las dos condiciones anteriores pueden simplificarse en una única:

$$y_i(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p) > 0, \;\; para \; i = 1...n$$

Para una observación cualquiera x^h, si es positiva la observación, se asigna a la clase +1 y si es negativa, a la clase –1.

La solución a este problema consiste en seleccionar al que se conoce como *maximal margin hyperplane* (hiperplano óptimo de separación), que se corresponde con el hiperplano que se encuentra más separado de todas las observaciones de *training*. Una de la librerías que permite aplicar SVM es e1071.

Ejemplo de uso

```
library(e1071)

# la variable respuesta a factor
datos$y <- as.factor(datos$y)

modeloSvm <- svm(formula = y ~ X1 + X2, data = datos, kernel = "linear", cost = 10,
                 scale = FALSE)

summary(modeloSvm)
```

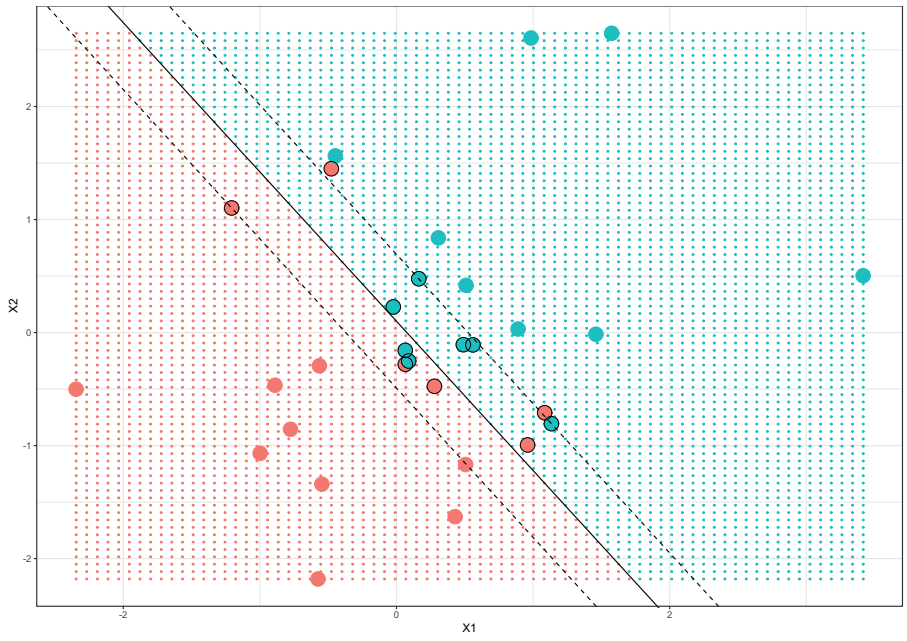
La función summary nos proporciona información relativa a la configuración de la solución. Tenemos 13 vectores soporte de los cuales 6 son para la clase –1 y 7 para la clase +1:

```
Call:
svm(formula = y ~ X1 + X2, data = datos, kernel = "linear", cost = 10, scale = FALSE)
```

```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  10
      gamma:  0.5

Number of Support Vectors:  13

 ( 6 7 )


Number of Classes:  2

Levels:
 -1 1
```

La idea de este método queda reflejada en la figura 6.

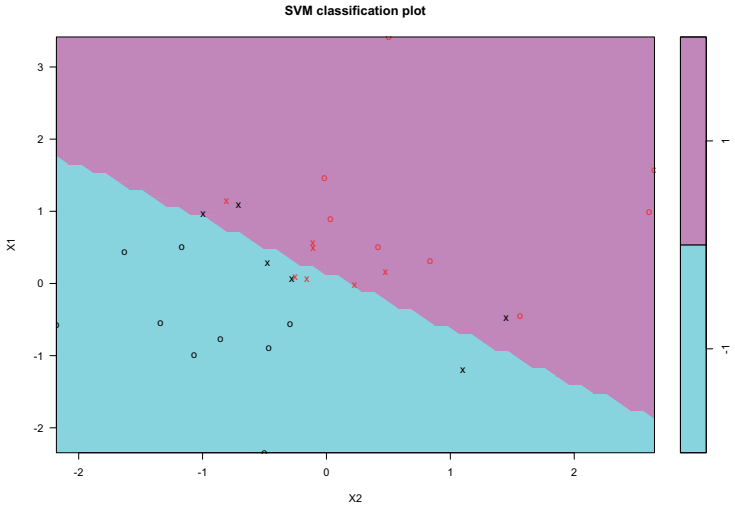
Figura 6. Hiperplanos (líneas) de separación entre los dos grupos



Si se utiliza la función plot, también podemos obtener una representación de las dos clases:

```
plot(modeloSvm, datos)
```

Figura 7. Separación entre los dos grupos con SVM lineal



Una de las grandes ventajas de SVM es la utilización de *kernels*. Las dimensiones de un conjunto de datos pueden transformarse combinando o modificando cualquiera de sus dimensiones. Este proceso permite transformar un espacio de dos dimensiones en uno de tres, cuatro o más aplicando la siguiente función:

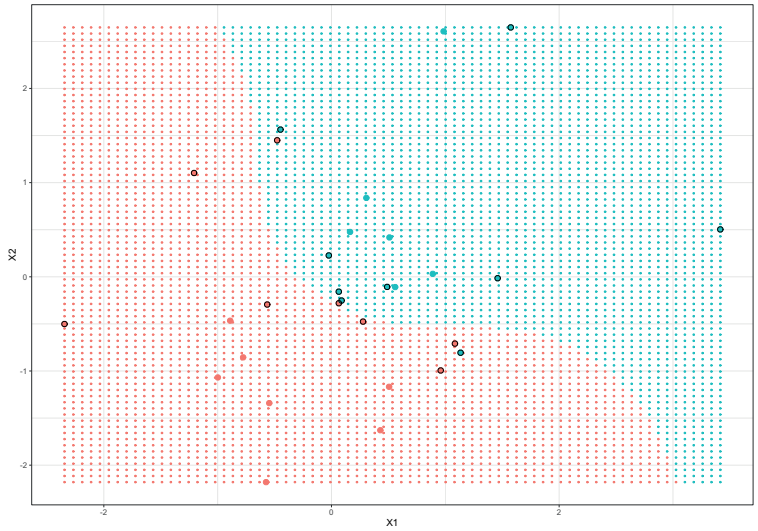
$$f(x_1,x_2) = (x_1^2,\sqrt{2}x_1x_2,x_2^2,log(x_1x_2))$$

Lista de posibles *kernels*:

- *Kernel* lineal: $K(\mathbf{x},\mathbf{x'}) = \mathbf{x} \cdot \mathbf{x'}$
- *Kernel* polinómico: $K(\mathbf{x},\mathbf{x'}) = (\mathbf{x} \cdot \mathbf{x'} + c)^d$
- *Gaussian Kernel* (RBF): $K(\mathbf{x},\mathbf{x'}) = exp(-\gamma||\mathbf{x} - \mathbf{x'}||^2)$

Al utilizar un *kernel* radial la separación entre los grupos cambia (ved figura 8).

Figura 8. Separación entre los dos grupos con SVM *kernel* = *"radial"*

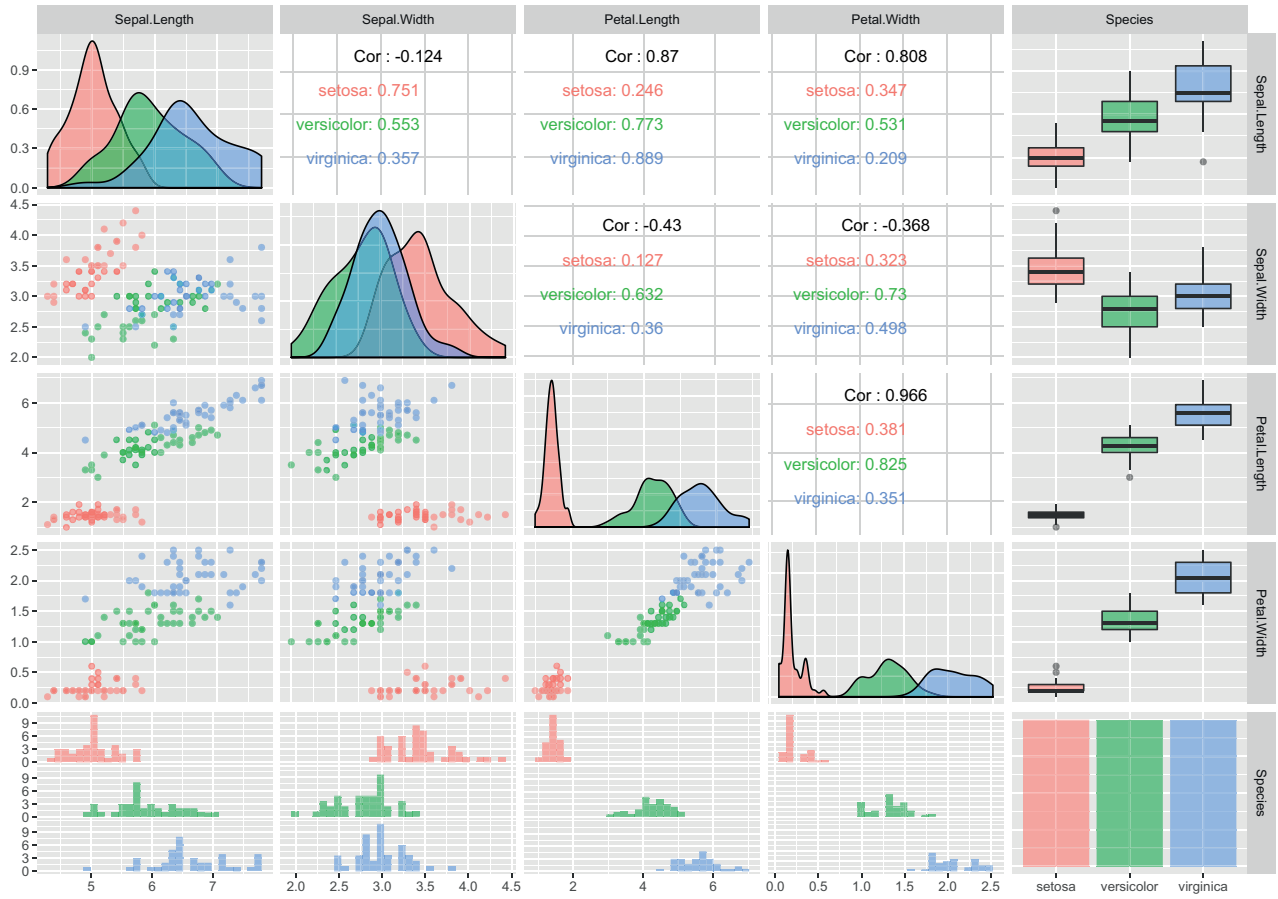


Enlace de interés

Para complementar la información podéis consultar el siguiente enlace: https://bit.ly/2ElVOpz

Intentad aplicar un método SVM sobre el conjunto de datos iris:

Figura 9. EDA del fichero iris (sobre un fichero *training*)



2. Introducción al modelado de procesos estocásticos

2.1. Introducción

La idea principal de modelar procesos estocásticos es su importancia y relevancia en la mayoría de las soluciones reales donde la variable tiempo es imprescindible. Desde una perspectiva y enfoque meramente estadístico, la mayoría de los enfoques se basan en un procedimiento que básicamente se resume en EDA, transformaciones para conseguir estacionariedad, modelado, evaluación y explotación.

2.1.1. Ejemplos de series temporales

Podemos encontrar una gran variedad de situaciones en las que tenemos fenómenos reales vinculados a series temporales.

Datos atmosféricos

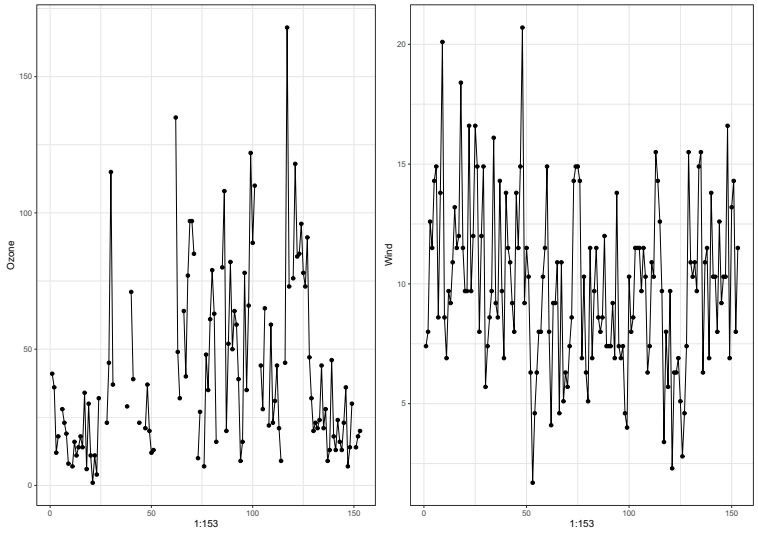
Ejemplo de datos atmosféricos relativos a Nueva York.

```
library(datasets)
data(airquality)
str(airquality)
'data.frame': 153 obs. of  6 variables:
 $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
 $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
 $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
 $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
 $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
 $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...
```

Este conjunto de datos tiene mediciones diarias de la calidad del aire en Nueva York de mayo a septiembre de 1973 durante un periodo de cinco meses. Las medidas de ozono corresponden al promedio de concentración en partes por billon (USA). Las medidas de Wind son en mph (*miles per hour*).

La visualización de la figura 10 permite observar la distribución de puntos a lo largo del eje temporal x. ¿Podemos predecir los valores de Ozono y de Wind? A primera vista parece complicado por la falta de datos (se debería corregir) y por la gran variedad en la distribución de los puntos.

Figura 10. Visualización de la serie temporal Ozone (con valores NA) y la serie temporal Wind



La visualización se ha realizado de forma directa con la librería ggplot:

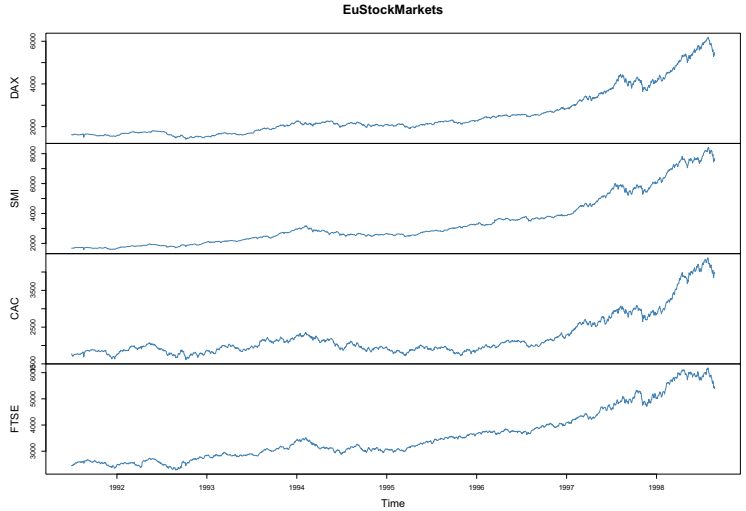
```
ggplot(airquality,aes(x=1:153,y=Ozone))+geom_point()+geom_line()
ggplot(airquality,aes(x=1:153,y=Wind))+geom_point()+geom_line()
```

Datos de mercados: índices y valores

El estudio de la cotización y de los índices asociados a diferentes mercados es de interés por parte de diversos actores: *brokers*, inversores, bancos, etc. Un modelo para estos ejemplos nos haría ricos (si funcionara).

En la figura 11 observamos cierta tendencia y una correlación entre los diferentes índices.

Figura 11. Visualización de las cuatro series temporales relativas a los índices: Germany DAX (Ibis), Switzerland SMI, France CAC y UK FTSE. Los datos corresponden a días laborables



En la figura 12 observamos el fuerte incremento en la cotización de Amazon y su posterior estabilización en el precio.

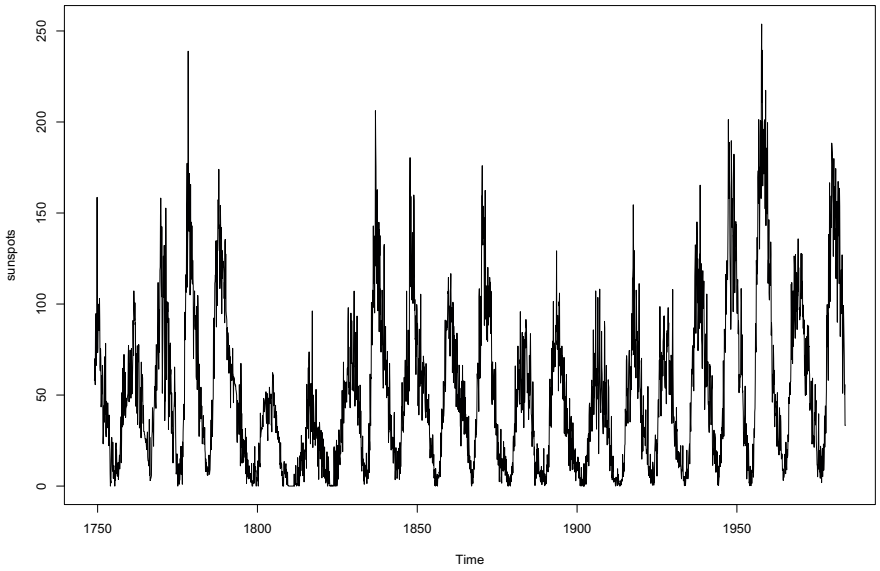
Figura 12. Visualización de la cotización de Amazon


```
library(quantmod)
library(forecast)
library(tseries)

amzn.tkr = getSymbols(.^MZN",from = as.Date("2017-01-04"),
to = as.Date("2017-10-27"),auto.assign = F)
chartSeries(amzn.tkr)


data(EuStockMarkets)
plot.ts(EuStockMarkets)
```

Datos físicos

Serie temporal de las explosiones solares. Esta serie temporal tiene mucha relevancia para diferentes usos. En primer lugar, para identificar el funcionamiento y la naturaleza de las explosiones (fenómeno físico) y, en segundo lugar, para predecir las explosiones para evitar o reducir incidencias en las comunicaciones vía satélite.

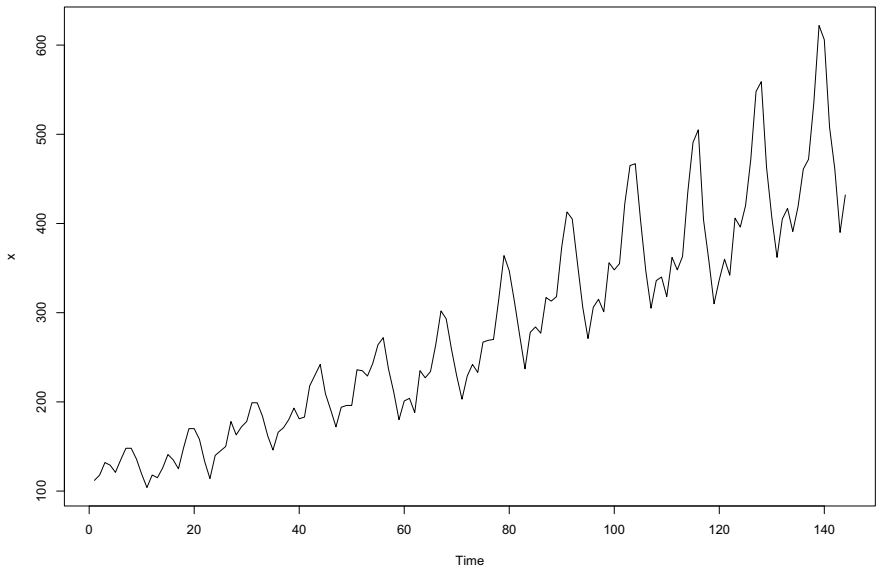
Figura 13. Registro de las explosiones solares



Datos comerciales

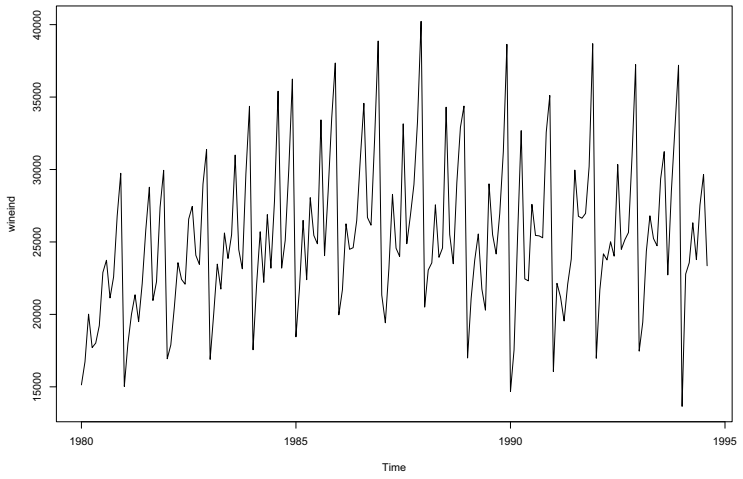
Para mejorar la planificación de nuestras acciones y operativa, el estudio de series temporales de carácter comercial como la mostrada en la figura 14 ayuda significativamente.

Figura 14. Pasajeros totales mensuales de línea aérea de 1949 a 1960



Ventas de vino en Australia totales de enero de 1980 a agosto de 1994 (ved la figura 15).

Figura 15. Ventas totales mensuales de línea aérea de 1949 a 1960




```
data(wineind)
str(wineind)
plot.ts(wineind)
?wineind
```

2.1.2. Características de las series temporales

Las características que definirán una serie temporal son:

- **Tendencia**: se identifica una línea ascendente o descendente de la serie de forma constante.
- **Estacionalidad**: se identifica un patrón estacional que se repite. Este patrón puede ser mensual (cada final de mes tenemos incremento de ventas), anual (cada mes de diciembre bajan las ventas) o semanal (los sábados hay siempre incremento de unidades vendidas).
- **Ciclo**: se identifica un patrón de repetición, pero que no podemos asociar a una estacionalidad conocida. Se repite el proceso térmico cada 77 días.
- **Ruido**: aquella variación que no podemos explicar por el modelo.

2.2. Modelos autorregresivos

Los modelos autorregresivos son los que tienen una formulación más comprensible.

Para el caso de un modelo AR(1), su formulación es:

$$Y_t = \mu + \phi_1 y_{t-1} + e_t$$

En realidad, la formulación corresponde a una ecuación en diferencias (caso discreto de una ecuación diferencial), donde se modeliza la variable de interés en el instante t como un modelo lineal respecto a la misma variable en el instante anterior. La semejanza con un modelo de regresión lineal es evidente.

La generalización de este modelo es:

$$Y_t = \mu + \sum_{i=1}^{p} \phi_i y_{t-i} + e_t$$

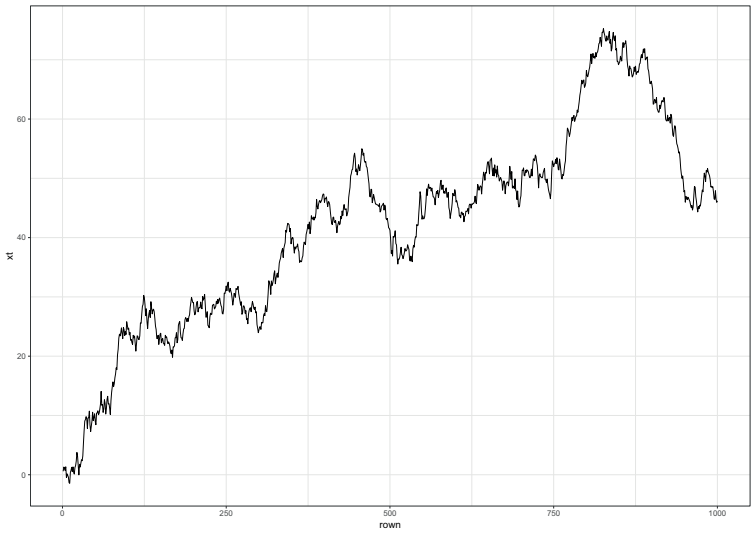
¿Cuántos parámetros están definidos bajo la notación AR(p)?

- p parámetros de los coeficientes asociados a los regresores: $\phi_1, \phi_2, \cdots, \phi_p$.
- σ_e asociado a la desviación del error.

Un modelo muy conocido autorregresivo es el que se denomina random walk. La figura 16 no es más que un proceso estocástico generado aleatoriamente.

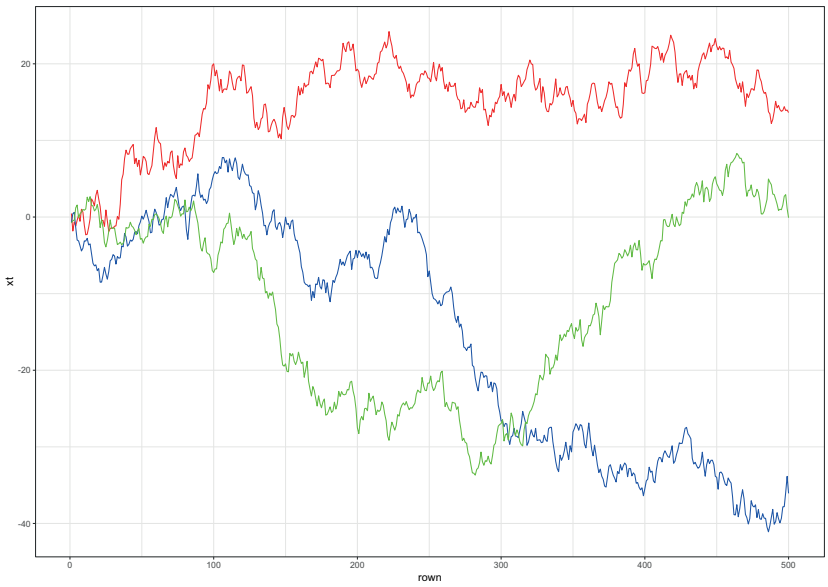
```
set.seed(12345)
randomWalk <- function(n){
  data.frame(x = rnorm(n),
             rown = c(1:n)) %>%
    mutate(xt = cumsum(x))
}

p <- ggplot() + aes(x = rown, y = xt)
p + geom_line(data = randomWalk(500))+theme_bw()
```

Figura 16. ¿Qué acción representa esta serie temporal?



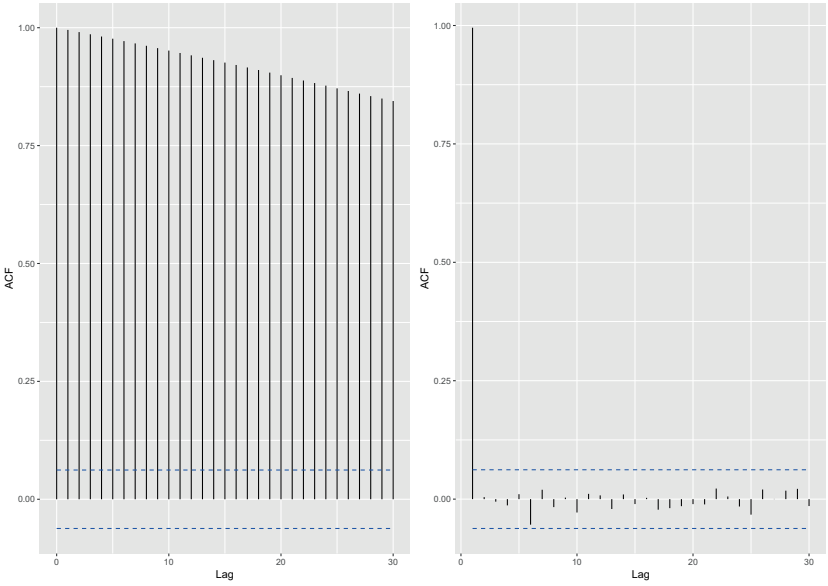
Podemos visualizar diferentes representaciones gráficas utilizando este tipo de modelo (ved figura 17).

Figura 17. ¿Qué acciones representan estas series temporales?



La relación existente entre los retardos puede facilitar la identificación del modelo. El uso de la función de autocorrelación (acf) y el de la autocorrelación parcial (pacf) (figura 18) permite establecer un primer diagnóstico sobre el tipo de modelo que tenemos:

Figura 18. ACF y PACF serie temporal de la figura 16



Como regla que hemos de aplicar:

- Si el modelo es autorregresivo AR(p), entonces la función PACF debe ser igual a cero después del retraso p (orden del modelo autorregresivo).

- Si el modelo es de medias móviles MA(q), entonces la función ACF debe ser igual a cero después del retraso q (orden del modelo de medias móviles).

- Si el modelo es ARMA(p,q), entonces las funciones ACF y PACF oscilarán alrededor de cero y no será fácil encontrar los valores p y q.

2.3. Modelos MA

Otro tipo de modelos proviene de una formulación relativa a los errores de predicción pasados. En el caso más simple, la formulación del modelo queda como:

$$Y_t = \mu + \theta_1 e_{t-1} + e_t$$

La formulación general de un modelo MA(q) es:

$$Y_t = \mu + \sum_{i=1}^{q} \theta_t e_{t-i} + e_t$$

Los parámetros que están definidos bajo la notación MA(q) son:

- q parámetros de los coeficientes asociados a los regresores: $\theta_1, \theta_2, \cdots, \theta_q$.
- σ_e asociado a la desviación del error.

Un ejemplo de serie MA(1) se puede ver en la figura 19.

Figura 19. Serie temporal simulada MA(1)

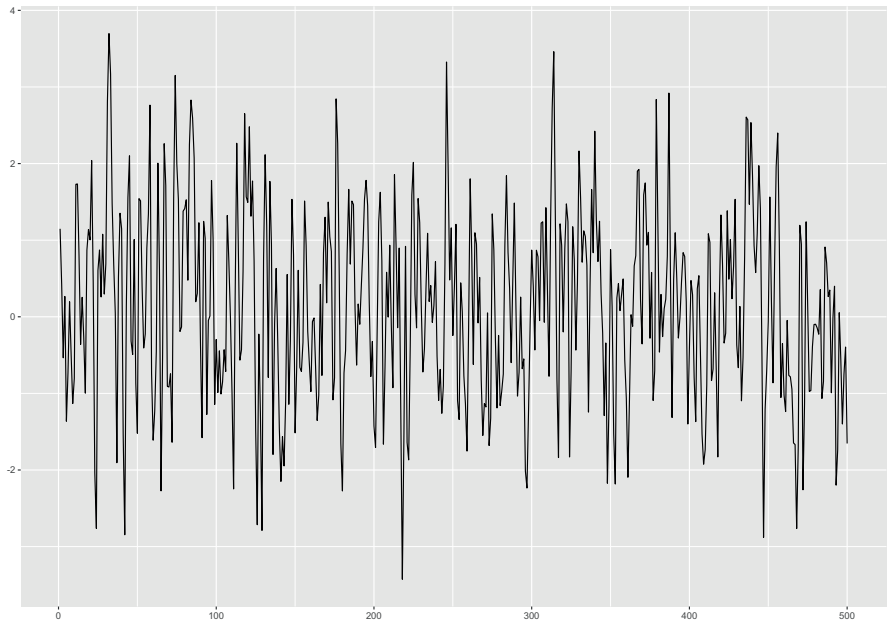
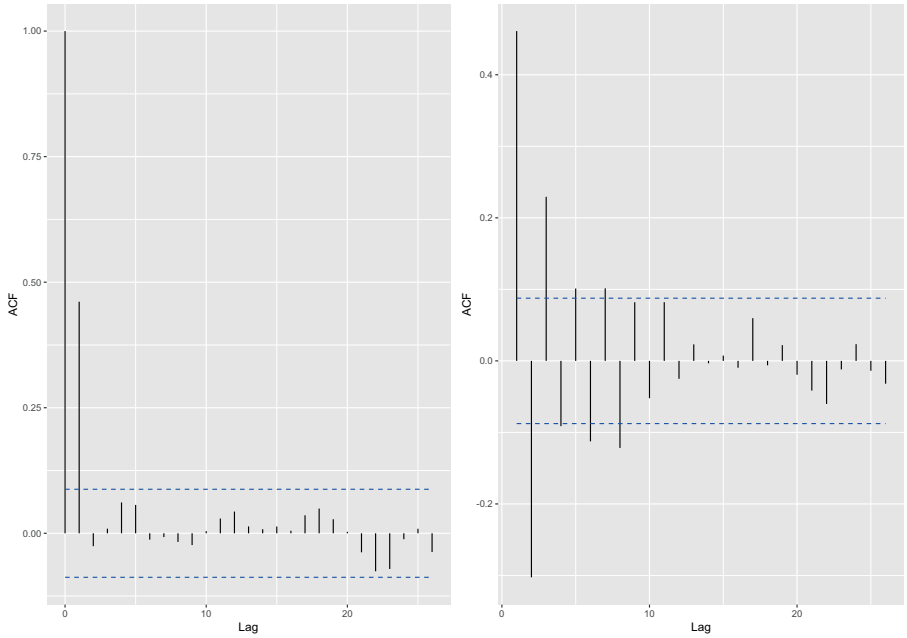


Figura 20. ACF y PACF serie temporal simulada MA(1)




```
set.seed(12345)
ma1<-arima.sim(list(order=c(0,0,1), ma=c(0.75)), n=500)
plot(ma1)
```

2.4. Modelos ARMA

La idea de un modelo ARMA es combinar ambas aproximaciones. Es decir, unir el modelo AR(p) con el MA(q). La formulación matemática para un modelo ARMA(p,q) es la siguiente:

$$Y_t = \mu + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_t e_{t-i} + e_t$$

2.5. Modelos ARIMA

Como en numerosas situaciones, el modelo ajustado a la serie no es correcto dado que la serie no es estacionaria. En estos casos se introduce el modelo con integración (para permitir que la serie sea estacionaria).

2.6. Modelos SARIMA

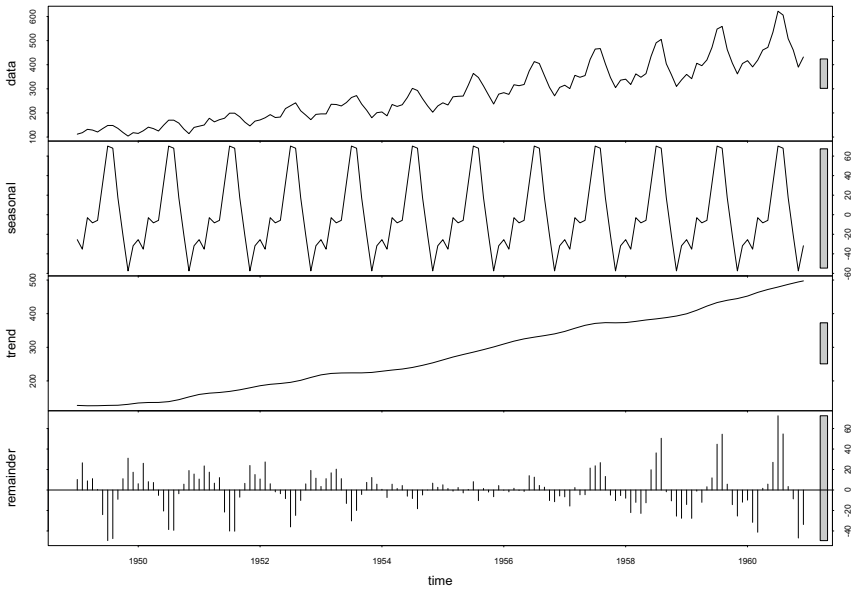
La última extensión del modelo inicial es la incorporación de una componente estacional, situación que podemos observar en las figuras 14 y 15.

2.7. Ejemplo de análisis

El objetivo es predecir el número de pasajeros (del ejemplo mostrado anteriormente).

Una manera sencilla y rápida de identificar las diferentes componentes de nuestra serie es utilizando el método *stl*, que permite descomponer la serie original en una parte estacional (se repite cada periodo estacional, en nuestro caso, cada año), una parte de tendencia y finalmente una parte de error:

Figura 21. Descomposición en diferentes componentes de las ventas totales mensuales de línea aérea de 1949 a 1960




```
data(AirPassengers)
plot.ts(AirPassengers)

decomp = stl(AirPassengers, s.window="periodic")
plot(decomp)
```

No es necesaria la pregunta de si la serie es estacionaria.

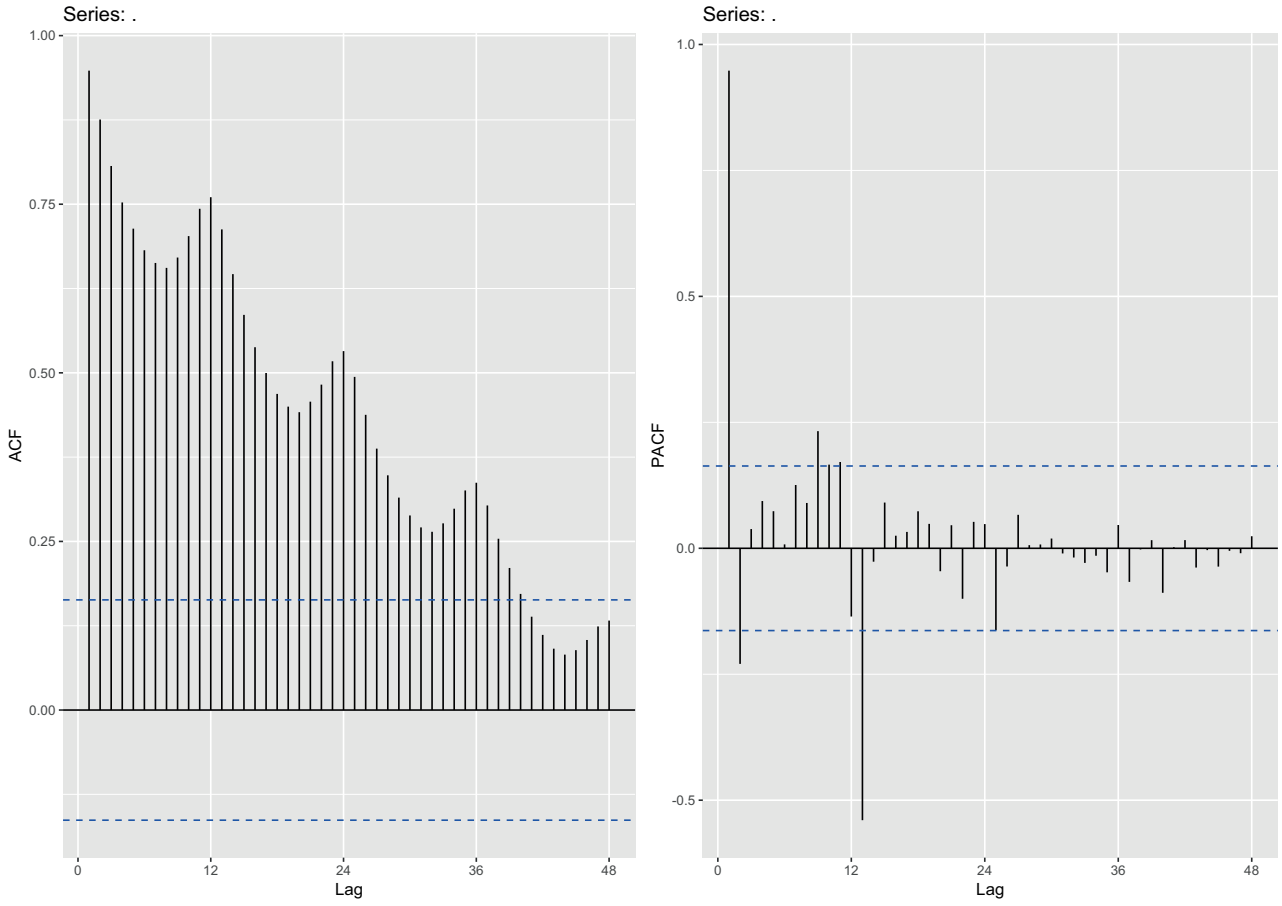
Si deseamos verificar la estacionariedad de la serie, el contraste parte con la hipótesis nula de serie no estacionaria (la media no se mantiene constante).

```
library(tseries)

adf.test(AirPassengers, alternative = "stationary")

Augmented Dickey-Fuller Test
```

```
data:  AirPassengers
Dickey-Fuller = -7.3186, Lag order = 5, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(AirPassengers, alternative = "stationary") :
  p-value smaller than printed p-value
```

Otro de los aspectos relevantes corresponde a la estructura interna de autocorrelaciones y autocorrelaciones parciales. Los gráficos permiten obtener una idea de qué tipo de modelo podemos tener:

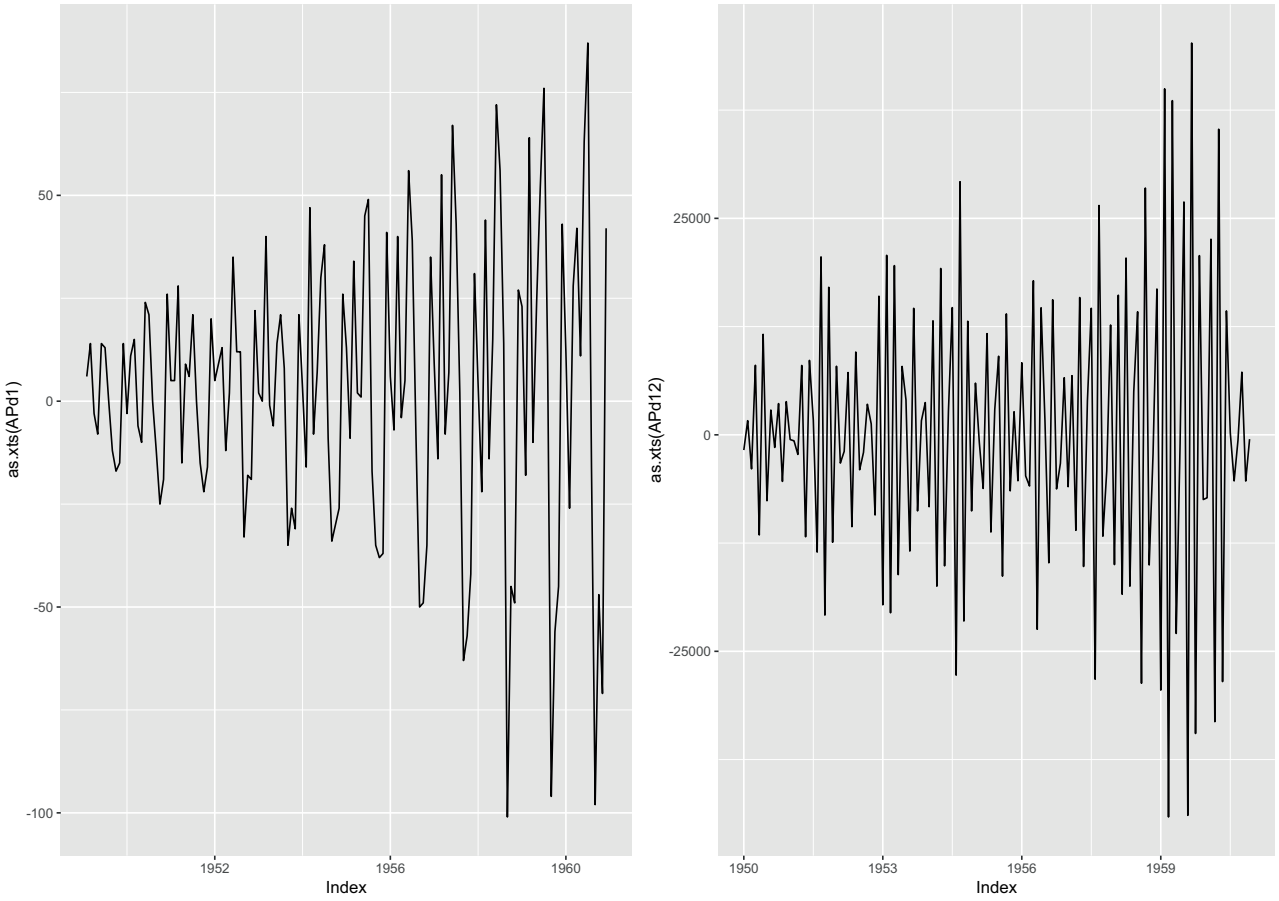
Figura 22. Autocorrelaciones (acf) y autocorrelaciones parciales (pacf) del conjunto de datos AirPassengers




```
AirPassengers %>% Acf(plot=FALSE,48) %>% autoplot
AirPassengers %>% Pacf(plot=FALSE,48) %>% autoplot
```

Operaciones de diferenciación:

Figura 23. Resultado de aplicar una diferenciación $\nabla_1 = x_t - x_{t-1}$ sobre AirPassengers y $\nabla_1 2 = x_t - x_{t-12}$



Estimación del modelo:

Tenemos un método que permite estimar de manera automática el modelo. *auto.arima* de la librería forecast devuelve el mejor modelo ARIMA según el valor AIC, AICc o BIC. La función realiza una búsqueda sobre el modelo indicado (SARIMA o ARIMA) y ofrece el óptimo según estos indicadores.

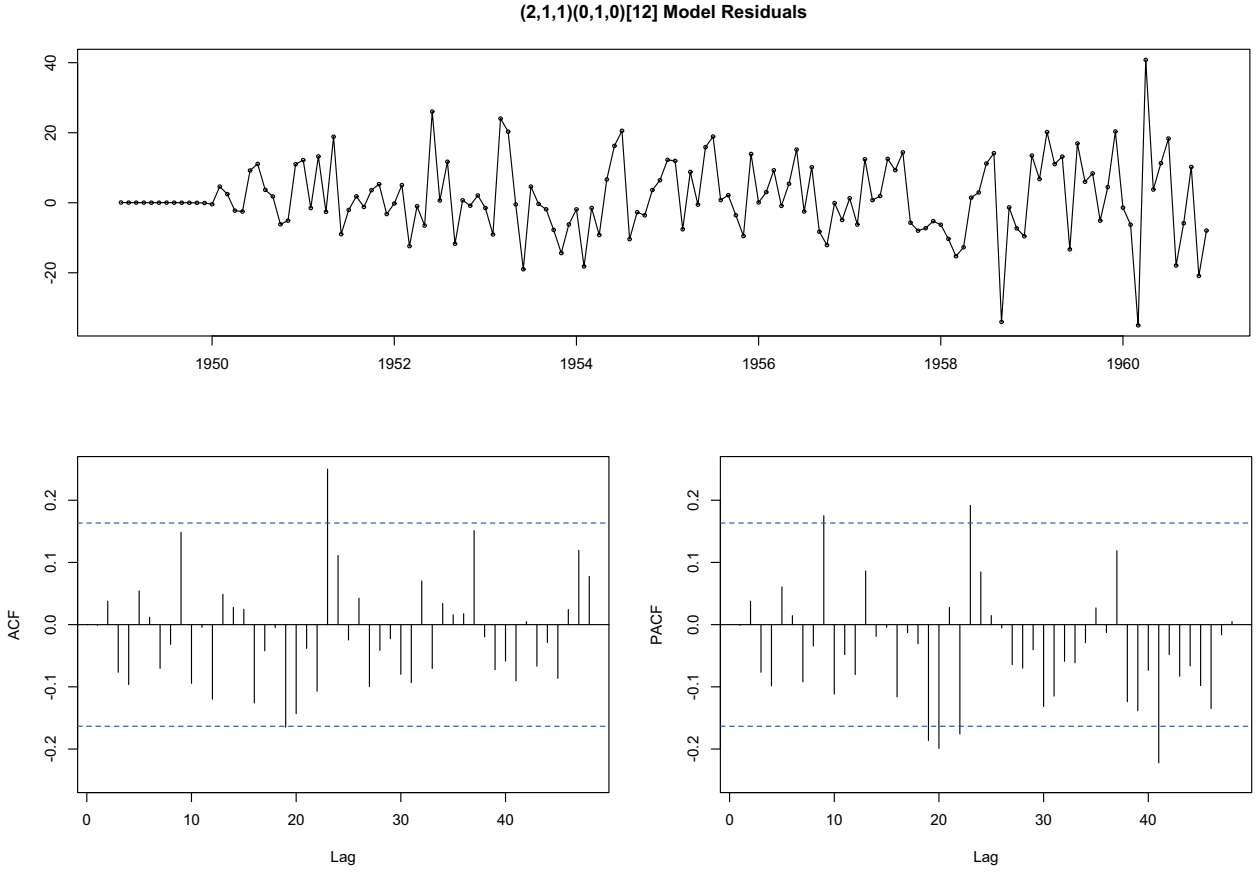
```
auto.arima(AirPassengers, seasonal=TRUE)

Series: AirPassengers
ARIMA(2,1,1)(0,1,0)[12]

Coefficients:
         ar1     ar2      ma1
      0.5960  0.2143  -0.9819
s.e.  0.0888  0.0880   0.0292

sigma^2 estimated as 132.3:  log likelihood=-504.92
AIC=1017.85   AICc=1018.17   BIC=1029.35
```

Análisis de los residuos:

Una vez terminada la etapa de estimación, se puede analizar el comportamiento de los residuos.

```
modAP <- auto.arima(AirPassengers, seasonal=TRUE)
tsdisplay(residuals(modAP), lag.max=48, main='(2,1,1)(0,1,0)[12] Model Residuals')
```

Figura 24. Representación de los residuos del modelo estimado

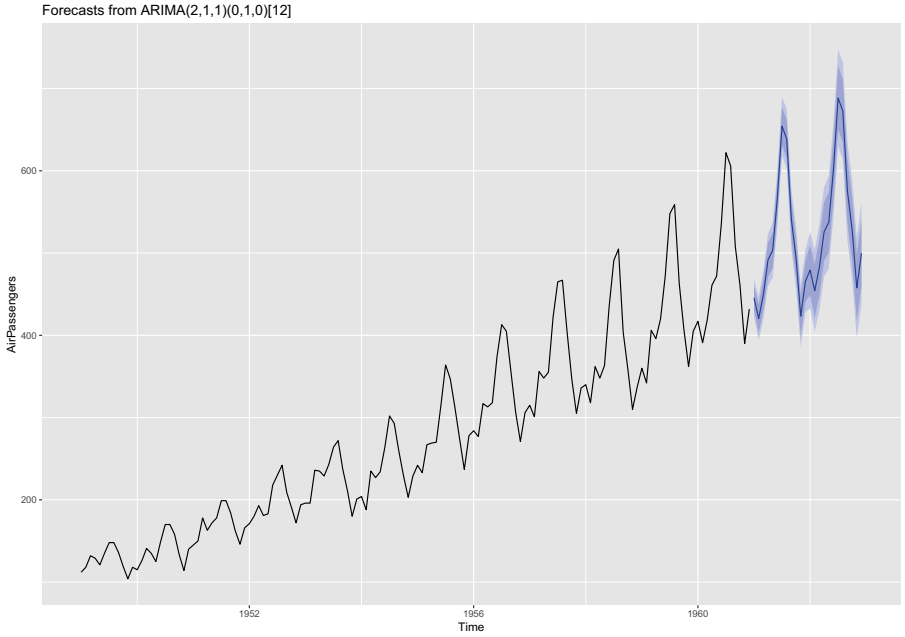


Previsión de resultados:

La previsión, una vez que tenemos el modelo estimado y verificado, es directa utilizando la función *fit*.

```
fcast <- forecast(modAP, h=24)
plot(fcast)
```

Figura 25. Estimación de la previsión de pasajeros para 24 meses (dos años vista)



Enlace de interés

Para complementar la información podéis consultar los siguientes enlaces:
https://bit.ly/2STYxQf
https://bit.ly/2z6iB46

3. Evaluación de modelos predictivos

3.1. Introducción

La evaluación de los modelos predictivos tiene igual peso y relevancia que la descrita en los modelos de aprendizaje supervisado.

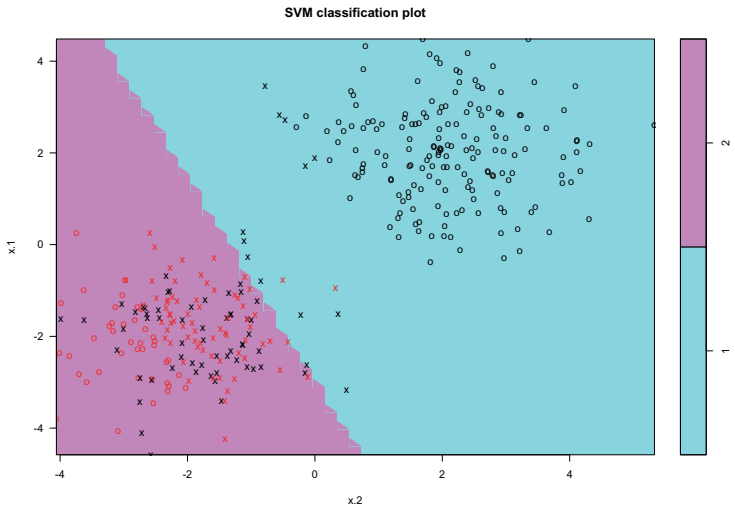
3.2. Ejemplo modelo SVM

La aplicación es sobre un conjunto simulado:

```
library(caret)
library(e1071)
set.seed (12345)
x <- matrix(rnorm(500*2), ncol=2)
x[1:250,]=x[1:250,]+2
x[251:500,]=x[251:500,]-2

y <- c(rep(1,350),rep(2,150))
dat <- data.frame(x=x,y=as.factor(y))

plot(x, col=y)
```

El ajuste de un modelo SVM nos da:

Figura 26. Separación de los dos grupos simulados con SVM lineal



El ajuste del modelo es:

```
# indices para el training set
train <- sample(500, 350)

# modelo ajustado sobre el training
svm.fit <- svm(y ~., data=dat[train,], kernel='linear', gamma=1, cost=1)

# Visualización
plot(svm.fit, dat[train,])
```

El rendimiento del sistema sobre el juego de datos test nos da:

```
Confusion Matrix and Statistics

          Reference
Prediction  1  2
         1 75  3
         2 35 37

               Accuracy : 0.7467
                 95% CI : (0.6693, 0.8141)
    No Information Rate : 0.7333
    P-Value [Acc > NIR] : 0.3961

                  Kappa : 0.4837
 Mcnemar's Test P-Value : 4.934e-07

            Sensitivity : 0.6818
            Specificity : 0.9250
         Pos Pred Value : 0.9615
         Neg Pred Value : 0.5139
             Prevalence : 0.7333
         Detection Rate : 0.5000
   Detection Prevalence : 0.5200
      Balanced Accuracy : 0.8034

       'Positive' Class : 1
```

Este resultado es directo con dos sentencias:

```
# uso de la función de predicción
yhat <- predict(svm.fit, dat[-train,])
```

```
# uso de la función Matriz de confusión
confusionMatrix(yhat, dat[-train,'y'])
```

La librería caret permite realizar diferentes evaluaciones con distintos parámetros para elegir el mejor:

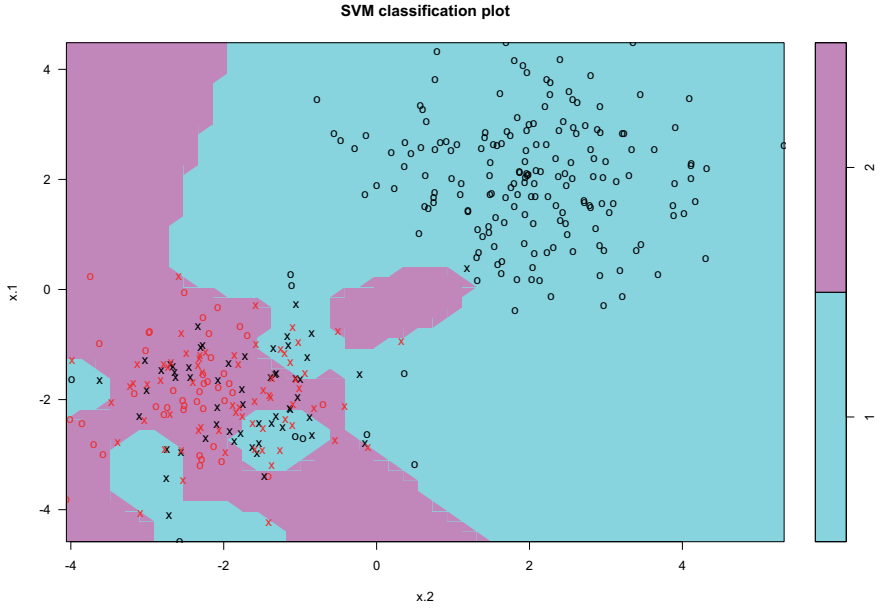
```
set.seed(12345)
tune.out <- tune(svm, y ~., data=dat[train,],
                 kernel='radial',
                 ranges = list(cost=c(0.1,1,10,100,1000),
                               gamma=c(0.5, 1,2,3,4)))

summary(tune.out)

yhat <- predict(tune.out$best.model, dat[-train,])
confusionMatrix(yhat, dat[-train, 'y'])
```

Que nos daría un incremento en el rendimiento del algoritmo:

```
Confusion Matrix and Statistics

          Reference
Prediction  1  2
         1 74  0
         2 36 40

               Accuracy : 0.76
                 95% CI : (0.6835, 0.8259)
    No Information Rate : 0.7333
    P-Value [Acc > NIR] : 0.2619

                  Kappa : 0.523
 Mcnemar's Test P-Value : 5.433e-09

            Sensitivity : 0.6727
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.5263
             Prevalence : 0.7333
         Detection Rate : 0.4933
   Detection Prevalence : 0.4933
      Balanced Accuracy : 0.8364

       'Positive' Class : 1
```

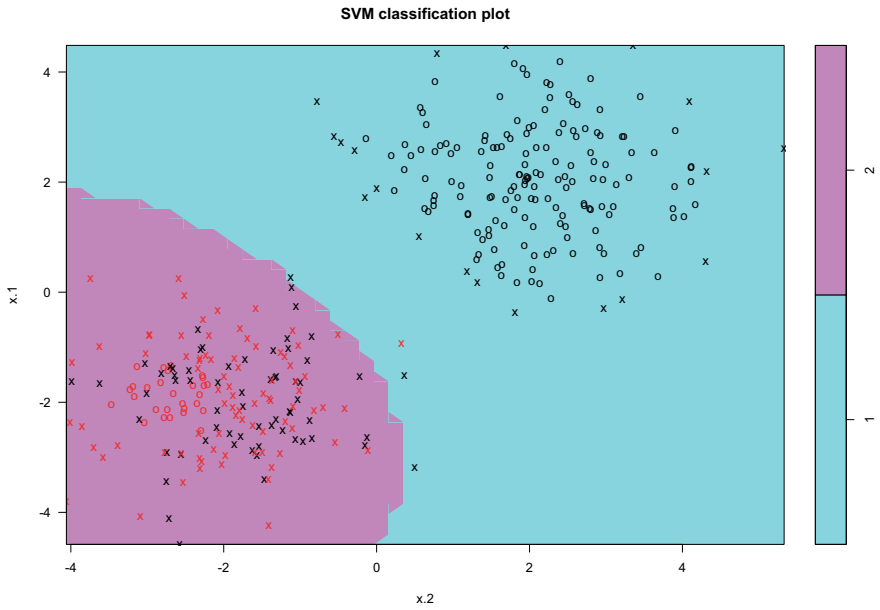
El resultado gráficamente de la situación inicial:

Figura 27. Separación de los dos grupos simulados con SVM radial



y el del resultado óptimo:

Figura 28. Separación de los dos grupos simulados con SVM radial con elección óptima de los parámetros



Resumen

Los principales puntos que debemos recordar, en cuanto a temas predictivos, es que nos encontramos ante problemas de tipo supervisado.

Aquellas situaciones donde tenemos una variable objetivo (*target*) nos permiten identificar un problema de tipo de clasificación. En caso de que tengamos una variable temporal, la aproximación será a través de modelos de series temporales.