MODELOS DE MACHINE LEARNING DE VARIABLE EXPLICADA CONTINUA

ÍNDICE:

[**INTRODUCCIÓN** 2](#_Toc87603541)

[**1.- PLANTEAMIENTO DEL CASO GENERAL** 2](#_Toc87603542)

[**2.- UN MODELO BÁSICO: REGRESIÓN LINEAL** 4](#_Toc87603543)

[**3.- VARIANTES ML DEL MODELO DE REGRESIÓN LINEAL: RIDGE – LASSO Y ELASTIC NET** 6](#_Toc87603544)

[3.1 Variantes “Clásicas” 6](#_Toc87603545)

[3.2 Regularización: extensiones tipo ML a la regresión lineal 8](#_Toc87603546)

[Regresión Ridge 9](#_Toc87603547)

[Regresión Lasso 10](#_Toc87603548)

[Regresión Elastic-Net 11](#_Toc87603549)

[**4.- MEDIDAS DE FIABILIDAD Y COMPARACIÓN DE MODELOS DE VARIABLE CONTINUA** 11](#_Toc87603550)

[4.1.- Introducción a los Criterios de Información 13](#_Toc87603551)

[4.2.- Cross Validation 16](#_Toc87603552)

[**5.- REGRESSION TREES** 18](#_Toc87603553)

[**6.- MODELOS CON NOCIÓN DE DISTANCIAS: EL KNN** 21](#_Toc87603554)

[**7.- MODELOS DE TARGET CENSURADA** 24](#_Toc87603555)

[7.1.- Regresión de Cox 27](#_Toc87603556)

[7.2.- Regresión de Cox con regularización 30](#_Toc87603557)

[**8.- MODELOS KRIGING: UNA APLICACIÓN PARA ESTIMACIÓN DEL PRECIO DE LA VIVIENDA A NIVEL REGIONAL** 30](#_Toc87603558)

[8.1.- Conceptos básicos de geoestadística 31](#_Toc87603559)

[8.2.- Un poco de matemática: El modelo de estimación kriging 32](#_Toc87603560)

[8.3.- Aplicación del Kriging con R, un ejemplo sencillo 33](#_Toc87603561)

[8.4.- Un caso de uso de krigeado: Medición del precio de la vivienda mediante técnica kriging 34](#_Toc87603562)

[**BIBLIOGRAFÍA** 35](#_Toc87603563)

# **INTRODUCCIÓN**

Bajo el paradigma de modelos supervisados cabe distinguir 2 grandes familias de modelos según sean las variables a considerar de tipo continuo o discreto.

Se comienza por los *modelos con variable explicada de tipo continuo* sobre los que se van a analizar una serie de familias algorítmicas, algunas de las cuáles se han visto ya, pero aquí se trata de dar un detalle adicional de profundidad, mientras otras se van a presentar por primera vez en el curso.

También acompañando a la presentación de los distintos modelos – algoritmos se ofrecerán las distintas medidas de eficacia sobre datos reales que deben ser utilizadas y que como se verá en los *modelos de variable explicada de tipo discreto*, cambiarán algunas cosas que resultarán importantes.

Así pues, esta sesión tiene como punto de partida el algoritmo en sí y se supone en muchos casos que parte del camino que vas desde la toma del dato hasta el momento de la creación de un primer tablón analítico se tiene resuelto y por tanto, se va tener que construir nuevas variables, imputarse valores, seleccionar variables adecuadas, etc para posteriormente, dado ya un determinado dataset más limpio, éste se subdividirá en 2 conjuntos, el de training y el de validación, siendo importante cómo funciona el algoritmo y cómo ajustar los parámetros necesarios t, para después pasar a comprobar y por tanto validar que ek ajuste ha sido correcto.

# **1.- PLANTEAMIENTO DEL CASO GENERAL**

Cuando se trata de explicar un determinado hecho entre los elementos de una población será habitual medir sobre parte de dichos elementos una serie de variables o de característica en un número *p*, denominándose a estas variables: *variables explicativas*. Estas variables van a tratar de explicar, en función de sus valores un determinado hecho que es el que se desea predecir cómo puede ser por ejemplo un nivel de ingresos, un porcentaje de recuperación, el valor de un objeto como pueda ser una vivienda, etc. siguiendo por tanto esta lógica cabe llamar a esta variable, un tanto especial como *variable explicada*. Así pues, en los modelos que se estudian en esta sesión, lo que se tendrá por tanto son elementos de una población sobre los que se les calcula unas *variables explicativas* en un determinado momento del tiempo y se trata de predecir, en función de los valores que el modelo tome sobre estas variables, el correspondiente valor de la *variable explicada* creando una función general tal como sigue:

En ocasiones a la expresión anterior se le puede añadir una dimensión temporal, pero en esta sesión esto no se hará, postergándose esto para el momento en el que se traten las *Series Temporales*. Así pues, en los modelos y técnicas estadísticas que se describen a continuación será factible el hecho de poder recoger las mencionadas *variables explicativas* para poder explicar o bien las causas de un hecho, o bien el valor que dicho hecho tomen tras un horizonte temporal fijo representado bajo una *variable explicada*, por lo que el orden de las observaciones de la tabla de la cual se deducirá la función *F* va a ser irrelevante a efectos de entrenamiento.

Así pues, en general se dispondrá de una muestra de la población cuya función se quiere deducir dada mediante una tabla tal como sigue:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | var1 | … | varp | target |
| individuo1 | v11 | … | v1p | t1 |
| … | … | … | … | … |
| individuom | vm1 | … | vmp | tm |

La variable denominada aquí *id* obviamente no intervendrá en el modelo y solamente está añadida por motivos de claridad y por lo general va a tener un valor distinto por individuo de la muestra, de modo que cuando llegue un nuevo individuo, se le podría dar el valor de identificación individuom+1 del cual se va a conocer a priori los valores de las variables explicativas y por tanto lo que realmente se quiere es que con la función *F* se tuviera:

El problema es que con la información con la que se construye *F* en general la igualdad anterior no se va a cumplir y va a existir un error, a niver de observación del tipo:

Para que dicha igualdad tenga lugar, error que va a ser una variable aleatoria y que tendrá por lo general media nula, siendo independiente de las observaciones, por tanto, lo que realmente se plantea es que, en términos medios, el valor de la función *F* condicionados a los valores de las variables explicativas, coincida con la media del valor a predecir.

Por tanto, en función del problema que se tenga, siempre y cuando se esté ante un problema con variable *target* continua se va a realizar una aproximación donde cabe suponer distintos supuestos, estos supuestos afectan fundamentalmente a:

* La propia forma funcional *F*
* La distribución de las propias variables (explicativas y explicadas)
* La distribución de las componentes de error
* Etc

El considerar los anteriores supuestos hace que se esté ante un modelo u otro, modelo bajo el cual existirá un *algoritmo* que tratará de hacer que la función *F* se ajuste lo más posible a los datos existente de un modo claro y concreto bajo los supuestos considerados.

Por lo que en cada caso cabe suponer un determinado tipo de *funciones de coste* que medirán la separación entre la realidad y el modelo supuesto.

# **2.- UN MODELO BÁSICO: REGRESIÓN LINEAL**

Por motivos de exhaustividad y también de sencillez se comienza con el modelo más sencillo que se puede plantear con targets de carácter continuo.

En el caso particular de un modelo de regresión lineal, la forma funcional se traduce simplemente a una combinación lineal de las variables explicativas cuyos coeficientes deben ser estimados en función a la información disponible:

Lógicamente la anterior formulación no es realista y es de esperar que cuando se pretende realizar estimaciones se cometa un error, así pues, la formulación queda completada añadiendo un vector de errores tal como sigue:

Es aquí donde se aplica las ideas del Método de Mínimos Cuadrados, cuando se pasa de las anteriores fórmulas teóricas a datos reales donde existe un número *m* de observaciones, se tendrá un total de *m* relaciones lineales con *m* en general >> *p* donde se estimarán coeficientes y por tanto los errores también tal y como se observa a continuación:

Así pues, si se tuviesen estimaciones de los coeficientes, automáticamente se tendría estimaciones de los errores, por tanto la idea del método de mínimos cuadrados es el intento de hacer una búsqueda en el espacio de los coeficientes de modo que los errores a estimar tengan el menor valor posible, por tanto el valor que tomarían los errores estimados, dados los datos y dadas las “supuestas” estimaciones de los coeficientes sería:

Y por tanto lo que se pretende es hacer mínima la siguiente función:

¿Cómo se logra lo anterior? Pues bien, como ya se ha visto hay 2 modos, uno mediante una conocida fórmula cerrada, que nos dice que los parámetros de la regresión lineal múltiples vienen dados por la siguiente expresión:

El otro modo de conseguirlo es aplicando el a. g. d. a la expresión de la suma cuadrática y seguir el ya conocido proceso iterativo de tipo:

Donde en este caso la función de coste adopta un modo muy sencillo con derivadas que podrían fácilmente escribirse.

Ejemplo 1: Generalización del concepto de gradiente de la función de coste para el modelo lineal múltiple

En este ejemplo se considera la función de coste siguiente:

Una función adicional que se supone, interna a la anterior función de coste es la función:

Con lo que la anterior función queda del siguiente modo (parecido al que se observa en la literatura, sólo que aquí se insiste en la naturaliza vectorial de *p* componentes de las variables que se miden sobre el individuo *i* de la muestra )

Nótese que si se considera el vector ampliado siguiente: entonces no es más que el producto escalar del vector paramétrico por el de observaciones, un número al que se le aplica la función, en este caso la identidad y se queda exactamente igual, pero claro, está, la función puede adoptar muchas formas como por ejemplo:

Y claramente se podría comparar si una función, tras aplicar un determinado número de iteraciones se ajusta mejor o peor a unos datos dados.

\_\_\_\_\_

# **3.- VARIANTES ML DEL MODELO DE REGRESIÓN LINEAL: RIDGE – LASSO Y ELASTIC NET**

Si se siguiese el camino de la teoría clásica que en general se estudia en las universidades, debería seguirse a continuación con los problemas clásicos del modelo como son:

* La multicolinealidad (ya mencionado y sobre el que no se insiste más)
* La hipótesis de exogeneidad que viene a decir que:
* La heterocedasticidad, que prácticamente se va a encontrar en la gran mayoría de las aplicaciones y que, aunque existen extensiones para poder controlarla, es un problema de difícil solución
* La autocorrelación residual que tiene un efecto muy importante y que se tratará en el capítulo dedicado a las series temporales
* Sobre e infra parametrización que es un problema que afecta a la eficiencia de los estimadores paramétricos y donde se preferirá modelos con cierto sobre-ajuste pero sin que éste sea excesivo

Para todo lo anterior hay distintas técnicas que al final lo que pretenden es hacer que el modelo cumpla las condiciones de un Teorema que lleva ya en boga varios siglos.

En un entorno aplicado más tipo “Machine Learning”, como se irá descubriendo, se va a tener en cuenta algunos de los problemas anteriormente citados, pero se va a cambiar el enfoque. En este enfoque no se va a hacer uso en general de supuestos de distribución de los parámetros, observaciones, y errores ya que en determinados casos no van a ser conocidos y ni tan siquiera planteables, en cambio se va a observar la actuación de un modelo que se construye sobre una base de datos sobre uno o varios conjuntos de datos test, así pues, en Machine Learning, se va a aplicar ideas que consumen recursos de cómputo como son la técnica de *cross-validation* o la de *bootstrapping* que van a cubrir parte de esa falta de información que se deriva de no existir un teorema del tipo *Gauss-Markov* para por ejemplo un *knn* o un *svm*.

## 3.1 Variantes “Clásicas”

A pesar de la aparente simplicidad en la forma funcional del modelo de regresión lineal múltiple, éste admite formas algo más completas que las aquí mostradas.

Así pues, existen los conocidos como modelos linealizables, que son aquellos, donde con una transformación más o menos sencilla en sus variables explicativas, se transforman en un modelo de carácter lineal. Así pues, si se tiene un modelo sencillo como este:

¿Qué impide no construir uno más complicado como el siguiente?

con:

La anterior transformación de variables permite que se pueda aplicar las fórmulas cerradas anterirormente comentadas sobre los datos, además se permite la introducción de iteraciones (o de productos entre variables) de modo bastante natural y por tanto, como cabe imaginar, ahora el juego de posibles modelos aumenta bastante. En ocasiones estas transformaciones permiten mejorar previsiones o corregir conceptos como la autocorrelación que se trata más adelante, sin embargo, el introducir términos “alegremente” va a hacer que la fiabilidad del modelo aumente pero este aumento será cada vez menor y se corre el riesgo de sobre-parametrizar el modelo.

Ejemplo 2: El modelo de Cobb-Douglals

Se está ante un modelo que aparentemente propone una relación entre el factor capital y trabajo de la siguiente forma:

Estrictamente hablando no sería un modelo lineal, sin embargo una transformación logarítmica lo convertiría en lineal:

Observación: En la anterior transformación habría que obligar a que el error del modelo esté presente en el original de modo multiplicativo

Ejemplo 3: Sobreparametrización de un modelo de regresión lineal

En el código *data/programa0203.R* Se ofrecen unos datos en el fichero en un data.frame denominado A y se realizan las siguientes acciones:

- Estimar los parámetros de un modelo de regresión lineal simple donde la variable dependiente es *Count* y la independiente *Time*

- Estimar los parámetros del mismo modelo pero con 2 independientes: la misma variable *Time* y dicha variable al cuadrado

- Estimar un modelo con un polinomio de grado 10

|  |  |
| --- | --- |
|  |  |

¿Qué ocurre conforme se aumenta el grado polinómico? ¿Dónde convendría parar?

\_\_\_\_\_

Para finalizar esta parte mencionar que en la teoría “clásica” que se estudia en las universidades existen 2 extensiones al modelo de regresión lineal y son las siguientes:

* El modelo lineal generalizado que tiende a tratar de corregir o al menos de suavizar los efectos de la existencia de heterocedasticidad o de autocorrelación residual
* Los modelos de variables instrumentales, que surgen cuando no se cumple la hipótesis de exogeneidad de las variables con la componente residual y en este sentido lo que se hace es utilizar variables que tengan algún sentido de negocio, pero sin que participen en primera instancia en el problema, dichas variables deben ser independientes con respecto a los errores y a estas variables se las denominará *variables instrumentales*

No obstante, estos 2 últimos métodos requieren de un estudio particular de hipótesis “muy pegadas al problema” y actualmente se encuentran fuera el paradigma del Machine Learning, prefiriéndose técnicas que tenga un carácter más escalable y generalizable para una gran variedad de problemas, técnicas que lo único que haya que hacer de modo externo sean elegir funciones y parámetros y donde exista un algoritmo que trate de extraer la información existente tal y como se desarrolla a continuación.

## 3.2 Regularización: extensiones tipo ML a la regresión lineal

Desde el marco del ML, se observan los siguientes problemas en el modelo de regresión lineal que se pretenden solucionar:

* Problemas en el modelo cuando se incorpora predictores correlacionados.
* No se realiza una selección de predictores en función de la relevancia de la información que poseen.
* No puede ajustarse un modelo cuando el número de predictores es superior al de observaciones

Para ello se pueden utilizar las estrategias de regularización que incorporan penalizaciones en el ajuste por *mínimos cuadrados ordinarios* (*OLS*) con el objetivo de evitar *overfitting*, reducir varianza, atenuar el efecto de la correlación entre predictores y minimizar la influencia en el modelo de los predictores menos relevantes. Por lo general, aplicando regularización se consigue modelos con mayor poder predictivo (generalización).

Dado que estos métodos de regularización actúan sobre la magnitud de los coeficientes del modelo, todos deben estar en la misma escala, por esta razón una transformación habitual a realizar en las variables, previa a la estimación de los coeficientes es la estandarización la normalización de los predictores previa al entrenamiento**.**

### Regresión Ridge

También denominada regularización l2 y tratada anteriormente, se tiene que la función de coste a utilizar es la siguiente:

Siguiendo la línea desarrollada en:

<https://www.cienciadedatos.net/documentos/py14-ridge-lasso-elastic-net-python.html>

Se tiene que La principal ventaja de aplicar *ridge* frente al ajuste por mínimos cuadrados ordinarios (*OLS*) es la reducción de varianza. Por lo general, en situaciones en las que la relación entre la variable respuesta y los predictores es aproximadamente lineal, las estimaciones por mínimos cuadrados tienen poco *sesgo* pero aún pueden sufrir alta varianza (pequeños cambios en los datos de entrenamiento tienen mucho impacto en el modelo resultante). Este problema se acentúa conforme el número de predictores introducido en el modelo se aproxima al número de observaciones de entrenamiento o incluso llega a superarlo, lo cual no es posible ajustar el modelo por mínimos cuadrados ordinarios. Empleando un valor adecuado de λ, el método de *ridge* es capaz de reducir varianza sin apenas aumentar el *bias*, consiguiendo así un menor error total. La desventaja del método *ridge* es que, el modelo final, incluye todos los predictores. Esto es así porque, si bien la penalización fuerza a que los coeficientes tiendan a cero, nunca llegan a ser exactamente cero (solo si λ=∞). Este método consigue minimizar la influencia sobre el modelo de los predictores menos relacionados con la variable respuesta, pero en el modelo final, van a seguir apareciendo. Aunque esto no supone un problema para la precisión del modelo, sí lo es para su interpretación.

### Regresión Lasso

También denominada regularización l1 y tratada anteriormente, se tiene que la función de coste a utilizar es la siguiente:

Si se atiende a *Gareth, J.; Witten, D.; Hastie, T. y Tibshirani R. 2013* lo que geométricamente permite que una regresión como la *Lasso* anule los parámetros frente a una *Ridge* es que los términos de penalización no son más que bolas centradas en el origen, sólo que en el caso de la *Ridge* la distancia subyacente utilizada es la euclídea y por tanto los entornos son círculos (o hiper-esferas en dimensiones superiores), mientras que en caso de la *Lasso* la distancia subyacente es la denominada “distancia taxi” cuyos entornos respecto al cero tienen una geometría diferencia siendo de cuadrados (o hiper-cubos en dimensiones superiores), por tanto, cuando se trata de minimizar las funciones de coste anteriores, existe una alta probabilidad de que los parámetros toquen a los ejes cartesianos en caso de la *Lasso* frente al caso *Ridge*, que al tener curvatura suave e igual por toda la superficie de la bola centrada en el origen, difícilmente se anularía el coeficiente, así pues la idea geométrica, en el caso de 2 dimensiones que subyace en la regresión *Ridge-Lasso* resulta ser la que se expresa en la siguiente figura:

Diagrama

Descripción generada automáticamente

**Figura 1 Entornos de significatividad paramétrica en las regresiones con regularización *l1* (regresión lasso) y *l2* (regresión ridge)**

**Fuente: Gareth, J.; Witten, D.; Hastie, T. y Tibshirani R. 2013**

Así pues, la principal diferencia práctica entre *lasso* y *ridge* es que el primero consigue que algunos coeficientes sean exactamente cero, por lo que realiza selección de predictores, mientras que el segundo no llega a excluir ninguno. Esto supone una ventaja notable de *lasso* en escenarios donde no todos los predictores son importantes para el modelo y se desea que los menos influyentes queden excluidos. Por otro lado, cuando existen predictores altamente correlacionados (linealmente), *ridge* reduce la influencia de todos ellos a la vez y de forma proporcional, mientras que *lasso* tiende a seleccionar uno de ellos, dándole todo el peso y excluyendo al resto. En presencia de correlaciones, esta selección varía mucho con pequeñas perturbaciones (cambios en los datos de entrenamiento), por lo que, las soluciones de *lasso*, son muy inestables si los predictores están altamente correlacionados.

Para conseguir un equilibrio óptimo entre estas dos propiedades, se puede emplear lo que se conoce como penalización *elastic net*, que combina ambas estrategias.

### Regresión Elastic-Net

Bajo este modelo se trata de aunar lo mejor de los 2 mundos mediante una función de coste que adopta la siguiente forma:

Ejemplo 04: Comparación de métodos y detalle en la aplicación de modelos lasso, ridge y elastic net

En el código *data/programa0204.py* se discute ampliamente cada uno de los 3 métodos desarrollados en la aplicación de un caso sencillo y se hace una comparación de métodos, donde la *ridge* aparece como ganadora sobre un conjunto test, aunque la regresión *lasso* se queda al mismo nivel que la de mínimos cuadrados, pero con un número relativamente bajo de predictores. A lo la largo del ejemplo se destacan distintos aspectos, como la visión de la evolución de los coeficientes en la medida que se alteran los hiperparámetros o la búsqueda de óptimo que se realiza con técnicas como la de cross-validation y medidas de error cuadrático medio, que aún no se han visto pero que se analizan a continuación.

Gráfico, Gráfico de barras

Descripción generada automáticamente

\_\_\_\_\_

# **4.- MEDIDAS DE FIABILIDAD Y COMPARACIÓN DE MODELOS DE VARIABLE CONTINUA**

Como se ha visto, cuando se trabaja con variable continua varias son las cosas que cambian frente al uso de la discreta. En variable continua se trata de estimar un valor (o más bien una media condicionada), mientras que en variable discreta lo que se estima es una probabilidad de ocurrencia de un evento frente a otro (si es binaria) o frente a otros (si la variable es múltiple).

Una primera medida de fiabilidad y que es justamente la que se ha utilizado para comparar modelos en el ejemplo anterior es la siguiente (siguiendo la notación de *Gareth, J.; Witten, D.; Hastie, T. y Tibshirani R. 2013*):

donde para cada observación *i*, se toma un vector *p-variante* que se evalúa a través de la función estimada a partir de los datos

Esta medida, como también se ha visto en el ejemplo anterior se aplica tanto para ajustar el training (con o sin regularización), como para medir la eficacia en el test que es el dataset que se usa realmente para saber si el modelo generaliza bien o mal. De hecho podrá ocurrir que se pueda, añadiendo suficientes variables, ajustar muy bien el dataset training minimizando costes de la citada función pero que cuando se aplique lo mismo al test, el ajuste se dispare, de hecho lo deseable, es que se tuvieran muchísimas observaciones test lo que debería ser pequeño es el valor medio para los pares test :

Desarrollando la anterior expresión se llega a que:

La minimización de la función anterior es equivalente a conseguir una baja *varianza* en los datos y un bajo *sesgo*, a la vez que habrá un error irreductible sobre el que no se va a poder actuar . Así pues, el primer término se refiere a la cantidad de cambio atribuida a por el uso de un dataset training distinto de modo que:

* Si un método tiene alta *varianza*, pequeños cambios en los datos pueden dar lugar a grandes variaciones
* Mientras que el segundo término se refiere al error introducido por las aproximaciones teóricas que se hace al problema y cómo de alejadas están éstas de la realidad, de modo que por lo general, métodos flexibles tendrán poco sesgo y métodos más rígidos, lo tendrán mayor

El *tradeoff sesgo-varianza* tiene lugar porque a menor flexibilidad del método (como por ejemplo ocurre en una *regresión lineal*), se va a producir un mayor *sesgo*, pero también una menor *varianza*, frente a lo que tendría lugar si se usa un método de mayor flexibilidad, donde el mayor ajuste a los datos produciría bajos *sesgos*, pero elevadísimas *varianzas*. Por tanto, uno de los objetivos del DS debe ser dar con la modelización adecuada que permita simultáneamente el menor valor *sesgo-varianza*

Ejemplo 05: Descomposición *sesgo-varianza* en Python

En el *data/programa0205.py* se hace una comparación de una regresión por defecto Python de sklearn (que es de tipo l2 si no se indica nada) frente a una de tipo ridge y se aplica una función como *bias\_variance\_decomp()* que sería capaz de extraer los 3 elementos de la descomposición del *MSE* para su posterior análisis. ¿Qué ocurre con el bias en caso de modelos con muchos parámetros y con el término *variance*?

\_\_\_\_\_

Otra medida de uso muy habitual es el *coeficiente R-Cuadrado*, esta medida, aunque aplicable a otros modelos, surge de descomponer la varianza de la variable observada, según el modelo formulado, de modo que:

por lo que:

El *coeficiente r-cuadrado* también denominado *coeficiente de determinación*, no es más que la proporción de varianza que explica el modelo, frente a la variabilidad total de los datos, es decir, sería:

Por tanto, se observa que, en modelos de variable continua, la anterior expresión estará entre 0 y 1 y resulta siempre aplicable, aunque en el contexto de la regresión lineal, este coeficiente tiene interesantes propiedades que se suponen conocidas y que no se analizan por estar dentro del contexto de la estadística clásica.

## 4.1.- Introducción a los Criterios de Información

Como elemento a destacar está el hecho de que añadir variables a un modelo implica reducciones sistemáticas (más o menos intensas) del anterior coeficiente, sobre todo mientras se entrena dicho modelo (recordar el trade-off expuesto anteriormente). En este sentido surge un nuevo coeficiente que se denomina *r-cuadrado ajustado* que trata de introducir una penalización por introducción de variables que no aporten suficiente mejora en el modelo, en este sentido, surge un ajuste que tiene en cuenta el número *m* de observaciones y el de parámetros *p*:

Por lo que, en el momento de comparar 2 modelos sobre una misma muestra de datos, y también para decidir si introducir o no una nueva variable, este coeficiente ayuda, aunque resultan más efectivos otros como los que surgen a partir de la teoría de la información:

* Criterio de Información de Akaike (AIC): En su versión más primaria se presenta como:

Con *p* el número de parámetros y *L* el máximo valor de la función de máxima verosimilitud. En el caso de modelos de regresión con errores de tipo gaussiano acorde a *Gareth, J.; Witten, D.; Hastie, T. y Tibshirani R. 2013* Se llega a que el *AIC* tiene una forma aproximadamente tal como sigue:

con: y una estimación de la varianza de los residuos del modelo

Nótese que con este criterio va a ocurrir que:

* + A mejor ajuste, menor valor
  + A mayor número de parámetros, mayor valor
  + A mayor número de observaciones, menos valor
  + A mayor varianza residual mayor valor

Claramente, se puede usar para comparar 2 modelos, donde un modelo con buen ajuste, muchas observaciones, pocos parámetros y errores estables y pequeños en torno a la media serán preferibles

* No obstante, existen otros criterios de selección como el *BIC* que, aunque haciendo uso de la misma idea, el modo de ponderar cambia, penalizándose un poco más el uso de variables adicionales:

Las formulaciones aquí mostradas, para el caso de variables continuas, son extensibles prácticamente a cualquier tipo de modelos de variables continua, siempre y cuando sean deducibles sus grados de libertad o/y número de variables lo cual, en muchos modelos, como se observará no es posible.

Ejemplo 06: Detalle del cálculo de los criterios de información.

En el sencillo *data/programa06.R* se muestra la típica salida del modelo de varios modelos de regresión lineal realizados sobre el mismo dataset y cómo se calculan algunos de los principales valores de su típica salida

|  |  |
| --- | --- |
|  |  |

Pero el cálculo exacto del *AIC* que da R, no es el de ninguno de las fórmulas que se ha visto, ya que intervienen constantes en su construcción tal como sigue:

El caso del *BIC* sale también directamente a partir de una ligera modificación de la fórmula anterior:

Cuestión: ¿Se puede calcular un AIC o un BIC en un modelo de regresión tipo Lasso o Ridge? ¿Qué problemas tendría su cálculo? ¿Qué opciones serían válidas para una adaptación plausible? ¿Qué tratamiento se le daría al hiperparámetro lambda? ¿Esta este parámetro considerado en la teoría?

<https://stats.stackexchange.com/questions/25817/is-it-possible-to-calculate-aic-and-bic-for-lasso-regression-models>

<https://stackoverflow.com/questions/63171921/is-there-a-way-in-r-to-determine-aic-from-cv-glmnet>

\_\_\_\_\_

## 4.2.- Cross Validation

Los criterios de información como los anteriores tienen su utilidad cuando es factible medir el número de parámetros o de grados de libertad que tiene un determinado modelo o grupos de modelos, así pues, como se sabe, es factible medir el número de parámetros en el caso de los modelos de *regresión lineal* al igual que en las *redes neuronales* o incluso modelos más evolucionados tipo *deep learning*, en todos, ellos de un modo más o menos complejo, es factible deducir el número de parámetros y llevar a cabo la comparación con alguno de los criterios anteriores, sin embargo el problema surge con los modelos tipo ensemble como los *random forest* o los de tipo secuencial como los *XGBosst*, en todos ellos, determinar un concepto que tenga que ver con la “complejidad” no resulta ya tan directo como se analiza más adelante.

En base a esto cabe utilizar un método que, aunque intensivo en cálculo, es cierto que a día de hoy permite dar más generalidad al hecho de poder comparar distintas arquitecturas de modelos entre sí.

En muchas librerías tanto de *python* como de *R*, el método ya viene implementado y listo para poder ser utilizado, en general lo que hay que indicar al método es el número de subdivisiones a realizar sobre el conjunto *training*. Así pues, un *cross-validation* 10-fold, indica que lo que se hace es subdividir el dataset training en 10 subconjunto con datos distribuidos uniformemente y con aproximadamente la misma cantidad de observaciones, dicho lo cual, se consigue crear 10 escenarios en los que en cada uno se hace uso de 9 de las 10 partes creadas y se deja una de dichas partes para realizar la predicción, por tanto se generarían 10 predicciones distintas tras haber entrenado 10 modelos distintos, eso sí bajo una serie de restricciones.

Diagrama, Gráfico de barras

Descripción generada automáticamente con confianza media

**Figura 2 Esquema *cross validation* 10*-fold***

**Fuente:** [**https://programmerclick.com/article/8295250019/**](https://programmerclick.com/article/8295250019/)

Hay varios tipos de *cross-validation* como son:

* *Leave-one-out*
* *K-fold cross validation*
* *K-fold* *estratificadaa*
* *Time Series cross validation*

Usos válidos que cabe dar a esta técnica, algunos de los cuáles se desarrollan más a lo largo del curso, serían los siguientes:

* Selección entre una familia de modelos tipo regresión lineal y un random forest
* Selección de hiperparámetros en regresiones ridge-lasso, svm, knn
* Intervalos de variabilidad máximo y mínimos de predicciones finales
* Selección de un número adecuado de variables
* Estimación de intervalos de variabilidad de las accuracies del modelo

Para introducir el método, en el siguiente ejemplo, se analiza cómo esta técnica se puede utilizar para elegir un número adecuado de variables, por lo que podría haber sido también añadido dentro del apartado dedicado a selección de variables ya analizado.

Ejemplo 07: Cross Validation y Feature Engineering

El código de *ejemplos/programa0207.py* proviene de un análisis más amplio de un conjunto de datos de una muestra de 1728 viviendas de Saratoga County, en él se repasan algunos aspectos y se plantean 2 tipos de *cross-validation* sin y con 5-repetición 5-fold. Los resultados en un y otro lado son semejantes:

|  |  |
| --- | --- |
|  |  |

Finalmente se hace una medida sobre el conjunto de validación que no intervino para nada en la construcción, haciendo uso de la misma métrica:



\_\_\_\_\_

Como se observa, con la validación cruzada es posible replicar muchas veces una medida final con el dataset training, de modo que se tenga seguridad de los máximos y los mínimos de eficiencia que cabe alcanzar con un determinado modelo, ya que recuérdese no estamos ante problemas de variables deterministas, sino más bien ante problemas de variables de tipo estocástico.

Repaso de lo llevado hasta el momento: <https://www.kaggle.com/code/jnikhilsai/cross-validation-with-linear-regression>

# **5.- REGRESSION TREES**

Los modelos de árboles son modelos de tipo no lineal que lo que hacen es particionar el espacio de entrada en regiones tipo hiper-cubos. Mientras que en los denominados *decision trees*, la variable a predecir es de tipo binario, en los *regression trees* la variable es continua, e interesará por tanto en cada una de las regiones *J* constituidas cuál es el valor medio de la variable, de modo que, es de esperar que las diferencias sean lo mayores posibles. No obstante, lo que se busca para las regiones creadas es la minimización de la función siguiente:

Con:

Con este método conviene hacer varios comentarios. Bajo un dataset training, será posible construir muchísimas regiones hasta llegar al punto de hacer las sumas cuadráticas anteriores nulas, sin embargo, en estos casos extremos, se estará bajo lo que se conoce como un over-fitting y lo que sucederá es que el poder de generalización será muy reducido ante nuevos datos, por otro lado, el número de regiones a constituir y de posibilidades es inmenso y por tanto hay que considerar alguna estrategia de construcción de dichas regiones, las cuáles suelen partir de ciertos criterios heurísticos tal y como se hacía con los métodos de selección de variables en las regresiones.

Una posible estrategia, es aquella que se realizaría de modo top-down de tipo voraz (greedy) y que en *Gareth, J.; Witten, D.; Hastie, T. y Tibshirani R. 2013* se denomina *recursive binary splitting*. Ante variables de tipo continuo (ya que en las discretas se usarían cada una de las categorías que las componen como separadoras de regiones, tratándose el problema de modo análogo), se trata de buscar un valor *s* tal que se crean 2 regiones por ejemplo para la variable de entre las *p* variables posibles:

Por tanto, se trata de encontrar los valores *j* y *s* que minimizan la siguiente expresión:

Una vez hecha la subdivisión 1, como es la anterior, cada una de las anteriores regiones es susceptible de ser subdividida y por tanto se puede dejar que el proceso avance indefinidamente o se asigna un nivel máximo de profundidad al árbol, a partir del cual, ya no se subdivide más

Ejemplo 08: Entendiendo un *regression tree*

El código *ejemplos/programa0208.py* se observa cómo puede hacerse uso de un *regression tree* para segmentar una variable continua. Construyendo un árbol de profundidad 1 sobre una única variable, es posible “dibujar el árbol” y extraer de modo automático el punto de corte realizando diversas manipulaciones con éste tal y como se observa

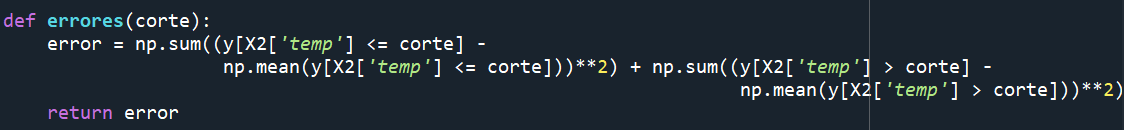
Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Si se dibuja la función de errores (con aproximación hasta las milésimas), se tiene que, abstrayendo a través de la siguiente función (análoga a la función de errores presentada anteriormente):



Se llega, generando el correspondiente bucle a la siguiente gráfica:

Gráfico

Descripción generada automáticamente

Que muestra que el algoritmo se ha comportado adecuadamente tomando el punto mínimo.

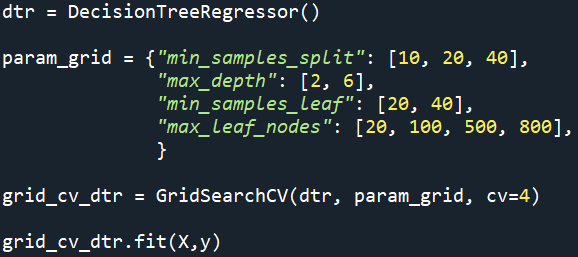
\_\_\_\_\_

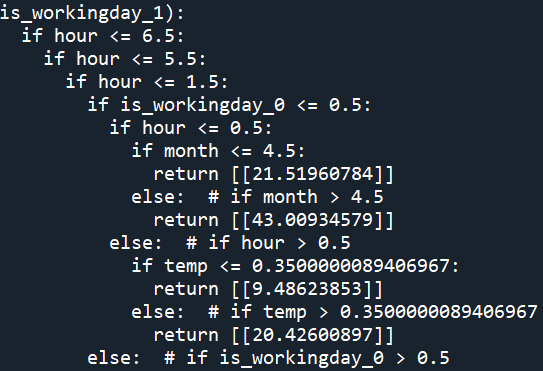
Entendiendo por tanto cómo funciona el algoritmo de construcción de un Regression Tree y que el resultado final va a ser de tipo discreto, conviene destacar los principales parámetros a tener en cuenta cuando se construye este tipo de modelos:

* La *función de error*: existen varias opciones, en general cuando no se dice nada, la utilizada en *python* a través del parámetro *criterion* es la *mse* que ya se ha utilizado, sin embargo, hay algunas más
* La *profundidad*: Tiene que ver con el número de niveles de mayor orden que se le va a permitir crecer al árbol, en ocasiones, dicha profundidad permite abrir, si así lo requiere el algoritmo una misma variable, en más de un nivel, si en el ejemplo anterior, se permite por ejemplo *max\_depth* = 2, entonces lo que podría ocurrir con la única variable utilizada es que se cree una nueva (o nuevas ramas), bien a la izquierda, bien a la derecha que permitiesen reducir más la función de errores cuadrática empleada
* La cantidad mínima para subdividir: permite a través de *min\_samples\_split* controlar si hay suficiente muestra para hacer una subdivisión o parar el algoritmo si en alguna muestra cae menos de un mínimo. Tanto este parámetro como el siguiente son recomendables su uso y su adaptación según el problema para que el árbol pueda generalizar
* La mínima muestra por hoja: entendiendo por hoja cada uno de los nodos finales, con *min\_samples\_leaf*, se puede controlar si la cantidad de datos que habrá en cada hoja final es suficientemente representativa o no
* Máximo número de nodos: controlable este parámetro mediante *max\_leaf\_nodes*

Ejemplo 08 II: Construyendo un regression tree

Si se continúa con el código anterior, ahora se le deja libertad para hacer una “búsqueda ciega” entre un campo paramétrico amplio, de modo que a través de la técnica *cross-validation* se llegue a un buen árbol





\_\_\_\_\_

Observaciones: No se ha tratado el tema de la poda de los árboles, ya que se analiza más adelante en la parte reservada para Decision Trees, donde es factible llevar a cabo, dado que se tiene un modelo ya seleccionado aplicar un “podado” que simplifique el árbol y mantenga en la medida de lo posible su poder predictivo, no obstante, al final, cuando se aplica un grid-search, intrínsecamente también se está dando opción a que se seleccione algo sencillo en función de la información disponible.

# **6.- MODELOS CON NOCIÓN DE DISTANCIAS: EL KNN**

Una familia de modelos de carácter supervisado que resulta interesante son los que tienen en cuenta las distancias entre todos o parte de los elementos que integran una determinada población. En este sentido aparece el modelo *k-nearest-neighbourg* o en sus siglas más conocido como *knn* que como se analiza, lo que hace es “memorizar” una serie de instancias para después aplicarlas en la predicción.

Las instancias que memoriza el modelo o conjunto de entrenamiento son sólo eso, instancias memorizadas (con variables de entrada y target). Por lo que se tendrán *m* pares memorizados del tipo , que después, cuando se desea aplicar sobre un dataset de predicción, habrá que aplicar el correspondiente algoritmo para que asocie a un nuevo dato, el valor de predicción adecuado a cada uno de los puntos que forman dicho dataset test por tanto, el valor a asociar para tal punto sería:

con los puntos cuya distancia sea la menor posible a y con igual al valor de la variable explicada en dicho punto

Así pues, en este tipo de modelos, lo que va a ser intensivo va a ser el hecho de predecir más que el hecho de entrenar, ya que cuando se entrena, lo que se hace prácticamente es “memorizar” los elementos simplemente y será en la utilización de estas memorias donde estará la parte intensiva del *knn*. Por lo tanto, los pasos básicos de un *knn* cuando se va a predecir sobre un determinado dato que no está en el dataset training, resultan ser los siguientes:

* Cálculo de distancias del punto a cada uno de los datos del training. En este punto nótese que el tipo de distancia es un elemento a elegir de entre los hiper-parámetros
* Elegir, en función del parámetro *k* especificado, los vecinos más cercanos
* Asociar un determinado valor en función de los *k* vecinos del paso anterior elegidos, que en el caso de una variable continua será la media de las observaciones

Se observa que el “paso duro” es justamente el cálculo de las distancias, con lo que a mayor peso del training cabe esperar mejor asociación de valores, pero más lentitud del algoritmo final y quizás mayor overfitting, mientras que a menor valor del training se mejoraría la velocidad, pero la asociación de valores estaría más sometida a la existencia de outliers y se esperaría un underfitting.

También hay que destacar que no suele haber “receta mágica” para la elección del hiperparámetro *k* de modo que un *k* pequeño puede generar problemas de overfitting mientras un *k* demasiado elevado permitiría mejores generalizaciones, pero convertiría el método de predicción en algo más lineal, no cubriendo las no-linealidades.

Ejemplo 09: Comparación de un knn frente a otros modelos

El en el código de *ejemplos/programa0209.py* se presenta una comparativa de los modelos knn frente a la regresión lineal, para ello se ha tomado el conocido conjunto Boston y se han considerado sólo 2 de sus variables importantes como explicativas.

Para llegar a los resultados que se obtienen, se ha procedido a estandarizar los datos, aunque dicha estandarización en términos de regresión lineal no tenía mucho efecto, sin embargo, a nivel de *knn* se consigue notorias mejoras como se muestra a continuación:

* Con estandarización:

Interfaz de usuario gráfica, Texto, Chat o mensaje de texto

Descripción generada automáticamente

* Sin estandarización:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

La distribución de los puntos training vs test resulta según la siguiente representación:

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Nótese cómo el punto más la izquierda de la variable RM en test, que es el que tiene por índice en X\_test\_norm que es el que tiene por índice 25 y que vale (-3.51497; 1.10333), tiene claramente como punto más cercano del training al que tiene por índice 218 y que vale (-3.11862; 1.10333). El valor del training de la variable a predecir en el índice 218 de y\_train vale 11.9 y por tanto el valor que debe haber en el índice 25 del vector predk1 es justamente 11.9, tanto para ese valor como para su compañero, que estaría en el índice 96, con valor el mismo (-3.11862; 1.10333), hay que fijarse que estos puntos están superpuestos tal y como se observa a continuación:

Pantalla de computadora con letras

Descripción generada automáticamente

\_\_\_\_\_

Por lo que en definitiva cabe concluir las siguientes líneas generales:

* Los knn pueden dar lugar a problemas cuando se trata de modelizar datos masivos debido a que el cálculo de distancias es un proceso intensivo en cálculo, además adolecen del problema del curse of dimensionality (que no se ha tratado), con lo que son efectivos si hay pocas variables explicativas que rellenen suficientemente el espacio.
* Como regla general, un pre-procesado de variables es recomendable en estos modelos para que funcionen correctamente, observándose efectos muy negativos cuando las escalas de las variables son dispares.
* Con pocos datos, estos modelos pueden mejorar a las regresiones lineales, pero debe hacerse comprobación con varios órdenes *k*, no hay regla general de elección de dicho parámetro y se recomienda en aplicaciones reales, si es que no hay mucho dato, hacer uso de cross-validaciones (que no se ha tratado aquí).

# **7.- MODELOS DE TARGET CENSURADA**

Un problema habitual es la existencia de datos censurados. Este problema que suele ser muy frecuente en estudios de medicina, también aparece de modo bastante natural en otros tipos de negocios como los de distribución de productos perecederos (probabilidad de que un distribuidor o punto de venta agote o no un producto), donde la venta total de un producto por parte de un distribuidor es una clara censura a la venta real que dicho distribuidor hubiera tenido si hubiera tenido suministro suficiente. También es habitual en estudios de prueba de productos de marketing, relación on-line, etc.

En este caso se pretende estimar una variable continua como es el tiempo de supervivencia en función de otras variables explicativas, pero teniendo en cuenta las citadas censuras.

Para cada individuo se supone que hay un *tiempo de supervivencia* real y una censura asociada a dicho tiempo denominada El tiempo de supervivencia representa el tiempo en el que el evento de interés ocurre como puede ser el tiempo en el que se produce la venta, o el tiempo en el que el paciente muere, o el momento en el que un cliente cancela su suscripción sin embargo, el momento de la censura es el momento en que la censura tiene lugar como por ejemplo cuando el producto en un punto de venta se ha vendido, o cuando el paciente sale del estudio (o este finaliza) o cuando la publicación a la que se está suscrito finaliza. Así pues, lo que se observa realmente para cada individuo es el siguiente dato, suponiendo una censura a la derecha:

A dicho dato cabe asociar el siguiente indicador:

Una suposición habitual es que se asume que el mecanismo de censura es independiente de las variables explicativas y que el evento *t* es independiente de la censura *c*.

Definición: Función de supervivencia

Se define como tal a aquella función que se construye del siguiente modo:

\_\_\_\_\_

La función de supervivencia es decreciente, en el caso de la modelización de la fuga, si lo que se modeliza es el tiempo hasta que un cliente se fuga o cancela una suscripción resulta ser entonces a mayor valor final en señalará a aquellos clientes cuya probabilidad de cancelación sea baja (ya que se supone que sobrevivirá a con una mayor probabilidad).

La estimación de la función de la función es sencilla cuando no existen censuras, pero cuando hay clientes con datos censurados, debe recurrirse a estimadores que no van a estar basados en una simple tasa de ocurrencias.

En la construcción siguiente se distinguen los *K* únicos tipos entre los elementos no censurados y que se denotan, de modo ordenado por:

En cada uno de esos tiempos existen elementos cuyo evento ocurrió en dichos tiempos

Por otro lado, se puede calcular los pacientes que son los pacientes vivos o en riesgo justo antes de que tenga lugar por lo que por el Teorema de la Probabilidad Total:

Como se tiene que:

De modo que iterando se tiene que:

Estimando las fracciones de riesgo se tiene que:

Por lo que se llega al estimador de Kaplan-Meier sería el que cumple la siguiente expresión:

Algunas consideraciones a tener en cuenta sobre el *análisis de Kaplan-Meier* según <https://www.analyticsvidhya.com/blog/2020/12/a-brief-introduction-to-survival-analysis-and-kaplan-meier-estimator/> resultan ser las siguientes:

* Para realizar inferencias es necesario aplicar el test *Log Rank*
* Los resultados KM pueden tener sesgo, ya que este estimador no es más que una aproximación univariante al problema que se desea resolver
* Eliminar datos censurados afectará a la forma de la curva y creará sesgos
* Test estadísticos y observaciones pueden llegar a ser confusos si se aplica dicotomización de variables continuas

Ejemplo 10 Estimador KM para la estimación del mandato de líderes mundiales

El problema que se presenta en programa10.py es un problema con datos censurados ya que un líder político como JFK puede ser asesinado a los 2 años de su mandato y por tanto ahí, hay una censura, o bien puede haber muerto por causas naturales durante su mandato (el cual, si no hubiera tenido lugar la censura, es de esperar que fuese más largo). También se considerará censura, si el líder en 2008 está aún en activo (y por tanto al ser el año en que acaba el estudio, no se sabe cuánto más habría durado su gobierno). Por último, como líder aquí se entiende desde un presidente elegido democráticamente hasta un dictador o un monarca.

Cuando se dibuja la función de supervivencia directamente sobre los anteriores datos se obtiene la siguiente curva:

Gráfico, Histograma

Descripción generada automáticamente

Que viene a indicar, cuál es la probabilidad (eje y) de que un líder se mantenga en el poder más de un número determinado de años (eje x), por lo que se observa que la probabilidad de supervivencia de al menos 10 años estaría en torno al 20% (nótese que en este gráfico se está dando no sólo la función de supervivencia sino sus correspondientes intervalos de supervivencia al 95%) así pues, en general los tiempos medios de supervivencia al 95% estarían en torno a:

Texto

Descripción generada automáticamente

O lo que es lo mismo, más de la mitad de los líderes mundiales finalizarán su mandato al 4º año (por lo que las re-elecciones no suele ser lo más habitual en general).

Sin embargo, lo interesante del análisis de KM es que se puede analizar los efectos de modo univariante de un modo muy sencillo, por lo que es conveniente analizar qué diferencias existen entre los regímenes democráticos y no democráticos, observándose la siguiente conclusión:

Gráfico, Histograma

Descripción generada automáticamente

Se observa como en el caso de los regímenes no democráticos, la probabilidad de supervivencia de más de 10 años está por encima del 40%, mientras que en caso de los democráticos estaría por debajo del 10%. Una analítica se puede llevar a cabo aplicando el test Log-Rank quedando de modo marcada las diferencia que se observan a simple vista con el gráfico:

Texto

Descripción generada automáticamente

En el ejemplo anterior se ha aplicado un test Log-Rank, para verificar la diferencia significativa entre los estados de una variable explicativa, para generar este tipo de test, por cada uno de los puntos lo que se hace es generar sendas tablas 2x2 del tipo:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Grupo 1 | Grupo 2 | Total |
| Evento |  |  |  |
| Sobrevive |  |  |  |
| Total |  |  |  |

La idea general de este test es verificar una H0 de que no existe diferencia significativa entre las medias de las poblaciones, pero teniendo en cuenta la censura de las observaciones.

## 7.1.- Regresión de Cox

No obstante, lo que se pretende en este tipo de modelos es ir más allá y al menos tratar de hacer lo mismo que se hacía en los modelos de regresión lineal, es decir, poner la variable explicada, en este caso el tiempo de supervivencia en función de todas las demás, sin embargo, esto no puede ser realizado de modo directo debido a la censura de las observaciones y se recurre por tanto a aplicar lo que se conoce como *Regresión de Cox*, para lo cual hay que introducir unos conceptos adicionales.

Definición: Función de Hazard

Se define como tal al siguiente límite: o lo que es lo mismo, dicha función es la tasa de muerte inmediatamente después del instante *t*

\_\_\_\_\_

Se observa que la Función de Hazard cumple la siguiente propiedad (por definición de la probabilidad condicionada):

siendo la función de densidad asociada a *T*

Por tanto, dado que la verosimilitud para cada una de las *m* observaciones es:

Se puede construir la siguiente función de verosimilitud asociada al problema:

Para poder hacer uso de covariables asociadas, lo que se supone es una forma funcional concreta sobre la Función Hazard que permitirá aplicando máxima verosimilitud, estimar los parámetros asociados a las variables explicativas, dotándoles de interpretabilidad adecuada, así pues, una suposición natural es la denominada *Hazard Proporcional*:

con

A partir de esta forma y con algunas manipulaciones, se puede llegar a la estimación de los coeficientes beta, independientemente de la forma de y por tanto los procedimientos existentes tanto en Python como en R van a permitir la estimación de los coeficientes y de la Función Hazard base

Ejemplo 11 Regresión de Cox en python

En el programa11.py se muestra lo sencillo que es estimar una *Regresión de Cox* bajo entorno Python, llegándose a obtener la tabla de coeficientes siguientes donde poder decidir qué variables resultan o no significativa bajo un criterio tipo *p*-valor

Tabla

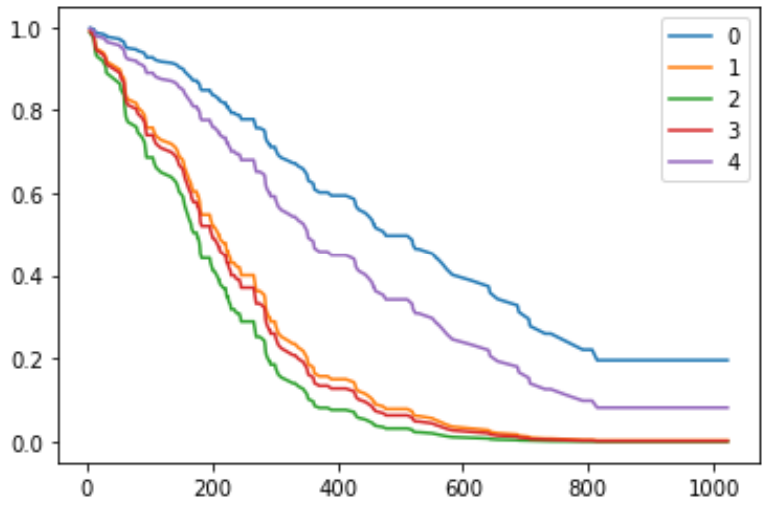
Descripción generada automáticamente

Se observan que las variables que aparecen como significativas en este estudio son la relativa a *sex*, *ph.ecog, ph.karno* y *wt.loss* cuyos respectivos *ratios Hazard* resultan ser 0.57, 2.09, 1.02 y 0.99, en el primero de los casos se indica una asociación fuerte en la caída del evento dependiente de *sex* ocurriendo lo contrario con *ph.ecog*, así pues un valor de *ratio Hazard >* 1 incrementa el riesgo y viceversa si dicho valor es menor que 1, de modo más claro las diferencias entre las variables *sex* y *ph.ecog* que son las que aparecen como más significativas se podría ver con:

Gráfico

Descripción generada automáticamente

Finalmente es posible observar las *funciones de supervivencia* por ejemplo para los 5 primeros individuos de la muestra:



Observándose que el individuo 0 es el que más probabilidad de sobrevivir tiene y el 2 el que menos, en función de las variables consideradas

Para finalizar esta parte comentar algunos comentarios sobre el análisis de supervivencia:

* Es un análisis al que actualmente los paquetes informáticos permiten incorporar variables explicativas que tengan en cuenta la censura de la información, lo que da gran versatilidad para una amplia gama de situaciones
* Los supuestos de censura y de estimación de KM descansan en supuestos de independencia que no siempre se van a cumplir
* La estimación de KM es eso, una estimación debida a no conocerse el tiempo real de vida de una población. Si realmente los datos con censura son una gran parte de la población conviene aplicar el análisis en sí, si son una pequeña parte, quizás la eliminación y el aplicar técnicas de regresión más sencillas sean lo recomendable

## 7.2.- Regresión de Cox con regularización

Al igual que ocurre con la regresión lineal, la *regresión de Cox* también tiene extensiones con regularización tanto de tipo *l1* como *l2*. Así pues, en este caso, la *función de coste* a minimizar es:

con o

La interpretación de esta regularización es análoga a la parte lineal.

Ejemplo 12 Regresión de Cox con regularización en Python

En el programa12.py se muestra una posible continuación del ejemplo 11, donde se preparan los datos de modo adecuado para aplicar una regularización tipo lasso a través de los métodos que existen en la librería *sksurv*, que es una librería que trata de copiar en la medida de lo posible el comportamiento de la librería *sklearn*. El resultado que se obtiene bajo este dataset resulta en cierto modo curioso si se compara con el anterior, sólo una variable resultaría significativa y la elegida en principio no estaría entre las más significativas acorde a los *p-valores* obtenidos anteriormente

# **8.- MODELOS KRIGING: UNA APLICACIÓN PARA ESTIMACIÓN DEL PRECIO DE LA VIVIENDA A NIVEL REGIONAL**

El krigeado o análisis geoespacial es una técnica desarrollada en los 60 que de tener un campo de aplicación muy limitado, se ha acabado aplicando a una gran cantidad de problemas ya que básicamente lo que ofrece es un cálculo medio de un conjunto de medidas finitas a lo largo de un área infinito pero teniendo en cuenta la posición física y real de estas medidas.

Con el krigeado, se trata de ir de unos pocos datos e ir enriqueciendo sus representaciones conforme vaya teniendo más información, por lo que se presenta como una técnica interesante donde a partir de una base, como la aquí desarrollada, se recomienda que al interesado profundice y expanda los conocimientos que se exponen a continuación.

El Kriging es un procedimiento geoestadístico avanzado que genera una superficie estimada a partir de un conjunto de puntos dispersados con valores *z*. A diferencia de otros métodos de interpolación del conjunto de herramientas Interpolación, utilizar la herramienta Kriging de forma efectiva implica una investigación interactiva del comportamiento espacial del fenómeno representado por los valores *z* antes de seleccionar el mejor método de estimación para generar la superficie de salida.

La técnica Kriging o de krigeado, presupone que la distancia o la dirección entre los puntos de muestra reflejan una correlación espacial que puede utilizarse para explicar la variación en la superficie. La herramienta Kriging ajusta una función matemática a un número específico de puntos o a todos los puntos dentro de un radio especificado, para determinar el valor de salida para cada ubicación. Kriging es un proceso que tiene varios pasos, entre los que se incluyen, el análisis estadístico exploratorio de los datos, el modelado de variogramas, la creación de la superficie y (opcionalmente) la exploración de la superficie de varianza. Este método es más adecuado cuando se sabe que hay una influencia direccional o de la distancia correlacionada espacialmente en los datos. Se utiliza a menudo en la ciencia del suelo y la geología.

Así pues, dada una región con un conjunto *n* de puntos geoposicionados y que toman un valor (por ejemplo, el precio medio de la vivienda en ese punto o área pequeño), lo que se pretende con el *kriging* es que cualquier punto de dicha región (que no tiene que está en la lista anteriormente considera) que no tiene por qué estar geoposicionado se le pueda dar un valor en función de los puntos que le rodean del siguiente modo:

Lo que hace el análisis kriging como se analiza a continuación es que estima pesos adaptados por cada punto en concreto cuyo valor se desea estimar. Así pues, el algoritmo a utilizar es intensivo en cálculo, porque realiza para puntos desconocidos (dispuestos en una red más o menos densa) el mismo conjunto de cálculos una y otra vez (con lo que realmente los pesos van a ir adaptándose al entorno de cada punto en función de los puntos previamente prefijados)

## 8.1.- Conceptos básicos de geoestadística

*Geoestadística*: Aplicación de la teoría de variables regionalizadas a la estimación de recursos mineros. Aunque actualmente la aplicación de esta teoría excede al estudio de recursos mineros

*Media Kriging (ponderada)*: Para un punto del espacio *x* es el resultado de realizar la siguiente suma ponderada de valores:

*Variables regionalizadas*: Es una función que representa variación en el espacio. Se designa por *Z*(*x*)

Texto

Descripción generada automáticamente

Una variable regionalizada posee propiedades intermedias entre una variable completamente aleatoria y una completamente determinística. A modo de definición una variable regionalizada es una variable aleatoria cuya realización depende de la posición. Ejemplos típicos de variables regionalizadas son la elevación topográfica de algún terreno, el perfil de pozo registrado con alguna herramienta de sondeo, etc. En contraposición con una variable totalmente aleatoria, una variable regionalizada tiene cierta correlación punto a punto, pero estos cambios son tan complejos que no pueden ser descriptos por una función determinística.

*Variograma* o *semivariograma*: es una función que constituye la herramienta fundamental de la geoestadística. Sean *x* y *x + h* dos puntos en el espacio:

Diagrama

Descripción generada automáticamente

Mientras que lo que se pretende es tener una función teórica (o *semivariograma teórico*) que responde a la siguiente expresión en función de una distancia continua *h*:

Un dibujo de una persona

Descripción generada automáticamente con confianza baja

El primer paso a dar es la determinación de lo que se denomina un *semivariograma empírico* que responde a la siguiente fórmula:

Un reloj de aguja

Descripción generada automáticamente con confianza baja

Ya que su valor depende directamente de los valores de los datos en un determinado entorno *N*(*h*) es el número de pares de elementos separados una distancia *h*.

*Pesos*: que representados por expresan la importancia con la que contribuye cada uno de los puntos base de valores conocidos, en la fórmula de ponderación cuando se desea estimar el valor del sumatorio ponderado de la *media kriging*.

*Anisotropía* vs *Isotropía*: Son conceptos que tienen que ver si la función variograma tiene en cuenta además de la distancia, la dirección u orientación en la que se disponen los puntos (anisótropos). El caso de isotropía se da cuando no hay direcciones privilegiadas en la variabilidad

## 8.2.- Un poco de matemática: El modelo de estimación kriging

Cualquier interpolación kriging, lo que necesita es realizar predicciones de valores de la variable bajo análisis en cualquier punto de un área a partir de unas pocas mediciones tomadas en algunos puntos. Para ello se trata de tener en cuenta tanto la distancia, como la distribución de los valores en la muestra tomadas y para ello es necesario un variograma teórico que ofrezca valores en un continuo. Dicho variograma teórico es el que se pretende estimar a partir del variograma empírico como se ha comentado antes y será lo que nos facilite R de modo sencillo como se estudia a continuación.

Por lo que, si no se quiere entrar en mayor complicación, cabe decir que, si se consigue ese variograma teórico, lo que se tiene realmente es una función, para cualquier punto de un espacio (que puede suponerse bi-dimensional) del tipo:

Un reloj de aguja

Descripción generada automáticamente con confianza baja

Como por ejemplo podría ser la siguiente función de tipo lineal:

Nota: Dejamos esta forma por simplicidad, aunque la fórmula de variograma más utilizada es la de tipo esférico que es algo más compleja, por tanto, con más parámetros a estimar

Lo que impone la matemática es que la media de la variabilidad teórica de los datos asociados sea mínima y que la estimación obtenida sea insesgada, esto da lugar a un sistema de ecuaciones que tiene solución y que procede de resolver la siguiente cuestión:

y que sea mínimo el error de predicción

Es decir, lo que se pretende minimizar es la expresión de la varianza siguiente:



Observación: V es la función de varianza y COV es la función covarianza. Lo que dice la fórmula es que la diferencia entre la varianza del valor estimado y del real les igual a la suma de las varianzas de cada uno de esos valores menos dos veces la covarianza del producto de dichos valores. El detalle matemático es un poco largo y escapa a los objetivos prácticos que se pretende en este curso, por lo que no se desarrolla más

Con lo que al final se obtienen expresiones cerradas para estimar, para cada punto , en cada punto un valor del tipo:

## 8.3.- Aplicación del Kriging con Python. Ejemplos sencillos

Actualmente tanto R como Python, incorporan librerías adecuadas para el tratamiento de datos geo-espaciales. En concreto se implementa en Python la librería *pykrige* que permite la aplicación de distintas técnicas de kriging.

Los elementos básicos en cualquier técnica de krigeado son los siguientes:

* Las coordenadas de las muestras
* Los valores de las muestras
* En modelos más allá del *kriging ordinario*, estarían los valores de las *variables explicativas*

Básicamente lo que va a hacer la técnica aquí tratada es distribuir espacialmente los valores de una muestra a lo largo de una superficie, pero teniendo en cuenta la distancia y cómo dichos valores aumentan o disminuyen acorde al semiseriograma o variograma especial, esta es la principal diferencia de realizar una distribución espacial con un *KNN*, en este modelo y bajo la versión continua, se aplican medias en función de la cercanía o lejanía en el espacio de las variables y dichas asignaciones son sólo medias, mientras que en el krigeado, la estructura espacial va aparte y la distribución superficial implicar re-estimar en cada caso un valor teniendo en cuenta la distancia entre las muestras obtenidas.

Ejemplo 13: Ejemplo de juguete del kriging ordinario. En el fichero programa13.py se distribuye, mediante un “kriging de juguete”, unos 5 puntos ubicados en una rejilla 0 x 10

Imagen que contiene Histograma

Descripción generada automáticamente

## 8.4.- Un caso de uso de krigeado: Medición del precio de la vivienda mediante técnica kriging

Algunos de los usos fundamentales, como se comentará son:

* La estimación de valores ausentes en la distribución territorial de la vivienda
* Analizar su distribución territorial para tener un mejor conocimiento del valor a lo largo de un territorio determinado

# **BIBLIOGRAFÍA**

**Amat Rodrigo J.** *Regularización Ridge, Lasso y Elastic Net con Python, available under a Attribution 4.0 International (CC BY 4.0)* at <https://www.cienciadedatos.net/documentos/py14-ridge-lasso-elastic-net-python.html>

**Matilla García, M; Pérez Pascual, P.; Sanz Carnero, B. (2013)** *Econometría y Predicción* Ed. UNED ISBN 9788448183103

**Navarrete Álvarez M. (2011)** *Modelos geoestadísticos del precio de la vivienda: aproximación al conocimiento intraurbano de la ciudad de Madrid* Tesis Doctoral de la Univervidad Autónoma de Madrid. Instituto Lawrence R. Klein

**Gareth, J.; Witten, D.; Hastie, T. y Tibshirani R. (2013)** *An Introduction to Statistical Learning with Applications in R* Springer Science + Business Media New York ISBN 978-1-4614-7137-0