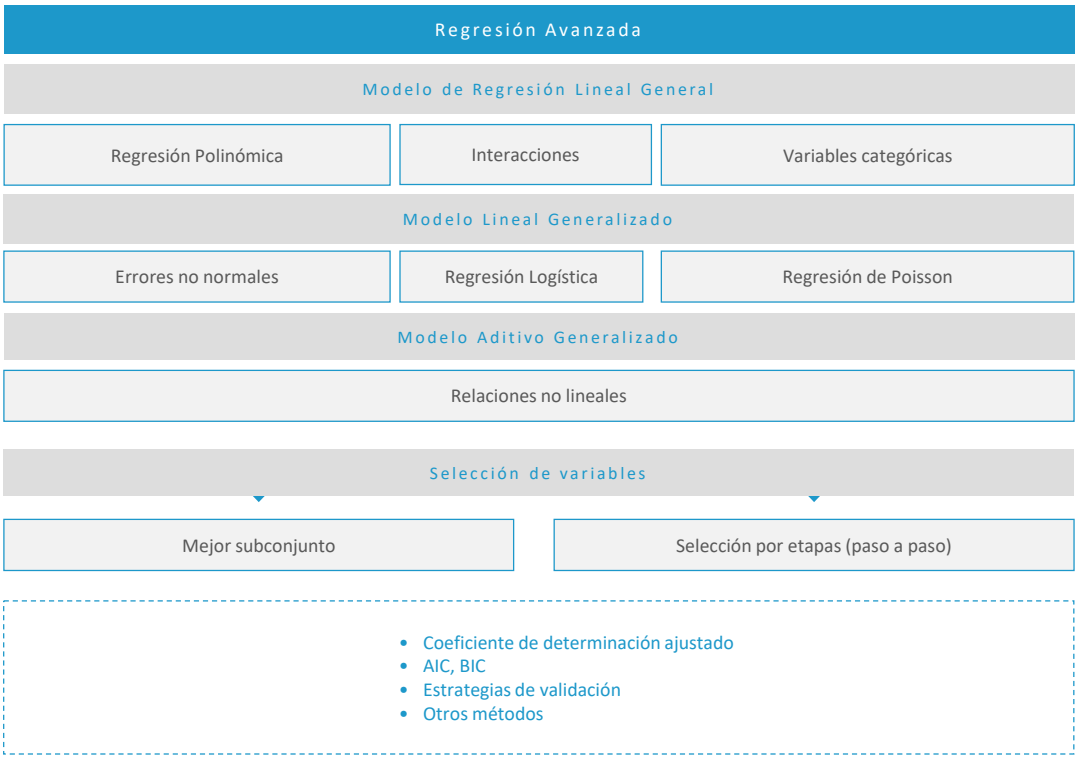
Técnicas Multivariantes

Técnicas de regresión avanzadas I

Índice

Esquema



Ideas clave

5.1 Introducción y objetivos

En el tema anterior se detallaron los conceptos de la regresión lineal simple. En este tema, se tratarán los conceptos asociados a la regresión lineal múltiple y se describirá la regresión lineal general. Además, se abordará la metodología para la selección de las variables predictoras en un modelo. Por último, se presentarán los modelos lineales generalizados y los modelos aditivos generalizados. Todo este tema se presentará bajo el enfoque de la estadística clásica. Posteriormente, en el Tema 6 se abordarán sus aplicaciones en el aprendizaje estadístico (o machine learning).

5.2 Regresión lineal múltiple

El modelo de regresión lineal múltiple es una extensión del modelo de regresión lineal simple, en el cual, la regresión se realiza con diversas variables predictoras $X_j; \quad j = 1, ..., m$. Este modelo es uno de los más populares y que más se aplican en la estadística. En el caso de que el modelo este compuesto por únicamente 2 variables predictoras, se obtendría la Ecuación del modelo de regresión lineal múltiple tal y como se formula en la Ecuación 1.

$$Y = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2. \tag{1}$$

Éste modelo, todavía se puede representar de manera gráfica. En este caso, en lugar de tener una recta de regresión para representar el ajuste del modelo como sucedía

el caso de la regresión lineal simple, se tendrá un plano de regresión, tal y como se muestra, a modo de ejemplo en la Figura 1.

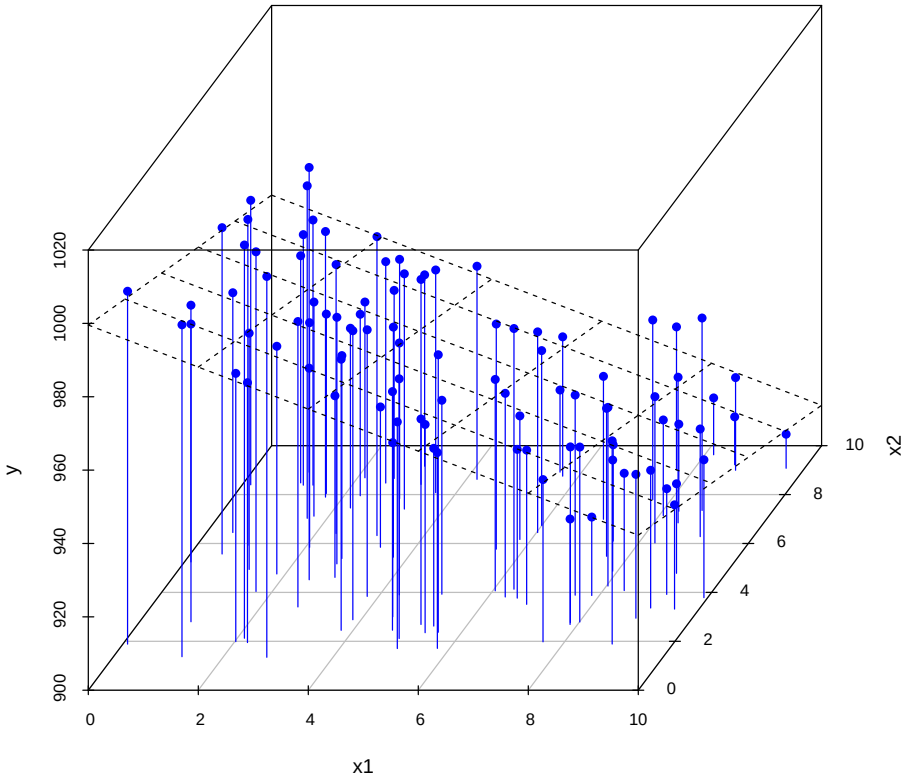


Figura 1: Regresión lineal múltiple con 2 variables predictoras

De manera general, se puede representar la Ecuación del modelo de regresión lineal múltiple como se formula en la Ecuación 2:

$$Y = \beta_0 + \beta_j \cdot X_j; \quad j = 1, ..., m, \tag{2}$$

donde Y se corresponde con la variable dependiente o variable respuesta y X_j se corresponde con las m variables independientes o predictoras. β_0 es la interceptación (o intercepto) del modelo y las $\beta_j; \quad j = 1, ..., m$ son los parámetros de la regresión del modelo, que se deben de estimar a partir de los datos.

Si se quiere particularizar para las diferentes observaciones de las poblaciones de X_j e Y, se puede modificar la formulación de la Ecuación 2 y reformularla tal y como se detalla en la Ecuación 3.

$$Y_i = \beta_0 + \beta_j \cdot X_{ij} + \varepsilon_i; \quad j = 1, ..., m; \quad i = 1, ..., n, \tag{3}$$

donde se añade la contribución del error a la fórmula. Por último, en el caso de particularizar para las distintas observaciones de una muestra de las poblaciones X e Y, se representaría el modelo de regresión muestral como se muestra en la Ecuación 4.

$$y_i = b_0 + b_j \cdot x_{ij} + e_i, ; \quad j = 1, ..., m; \quad i = 1, ..., n, \tag{4}$$

donde y_i es el valor de la muestra.

Además, se puede definir a $\hat{y}_i$ como el valor del ajuste. Es decir:

$$\hat{y}_i = b_0 + b_j \cdot x_{ij}; \quad j = 1, ..., m; \quad i = 1, ..., n. \tag{5}$$

De las Ecuaciones 4 y 5, es inmediato obtener el error e_i, que del mismo modo que sucedía en el modelo de regresión lineal simple, es la diferencia entre el valor real y el ajustado de la observación, tal y como se muestra en la Ecuación 6.

$$e_i = \hat{y}_i - y_i; \quad i = 1, ..., n. \tag{6}$$

Parámetros del modelo

En el modelo de regresión lineal simple se obtenían tres parámetros del modelo: los dos parámetros de regresión: β_0 y β_1; y la varianza del modelo: σ^2. En un modelo de regresión lineal general con m variables, de manera similar, se obtendrán $m+2$ parámetros del modelo. Los $m+1$ parámetros asociados con los coeficientes de regresión $\beta_0, \beta_1, ..., \beta_m$ y la varianza del modelo.

Los coeficientes de regresión

En este caso, Los coeficientes de regresión, $\beta_j,$ con$j = 1, ..., m$, indican el cambio en la esperanza de y cuando se produce un incremento unitario en la variable x_j, ceteris paribus, es decir manteniendo el resto de variables $x_{k \neq j}$ invariables. En la Ecuación 7 se muestra el significado de un β_j del modelo de regresión.

$$\beta_j = E(y|x_j + 1, \forall x_{k \neq j}) - E(y|x_j, \forall x_{k \neq j}) \tag{7}$$

La varianza del modelo

La estimación s^2 de la varianza σ^2 en un modelo de regresión lineal general se puede obtener realizando la descomposición en suma de cuadrados de la varianza de la respuesta y realizando el cociente representado en la Ecuación 8.

$$s^2 = \frac{SSE}{(n - m - 1)} \tag{8}$$

Para realizar la descomposición en suma de cuadrados en el caso de la regresión general

es necesario realizar operaciones matriciales.

Atribución

De manera similar a lo que sucedía en la regresión lineal simple, se pueden estimar (aunque la fórmula no es inmediata y es necesario realizar el desarrollo matricial) los errores estándar (y los intervalos de confianza) de los coeficientes de regresión del modelo. Estos intervalos (o su p. valor asociado) marcarán si esa variable tiene una asociación significativa con la variable respuesta y puede definir uno de los criterios a la hora de seleccionar variables en el modelo. La mayoría de los programas estadísticos ofrecen información de los errores estándar de las variables predictoras en un modelo de regresión lineal general, y suelen venir marcadas con un asterisco aquellas que son significativas en el modelo.

5.3 La descomposición en suma de cuadrados

En el caso de la regresión lineal general también se puede realizar la descomposición en suma de cuadrados de la variable respuesta Y, donde SSR ahora se corresponde con la varianza que explican todas las variables predictores del modelo tal y como se representa mediante la Ecuación 9.

$$SST = SSE + SSR, \tag{9}$$

donde:

- **SST** es la suma de cuadrados total y es una medida de la variabilidad de la muestra de Y respecto a la media muestral.

- **SSE** es la suma de cuadrados residual, o suma de cuadrados del error, y es una medida de la variabilidad de los datos de Y respecto a los valores ajustados.

- **SSR** es la suma de cuadrados de la regresión y es la parte de la varianza que explican todas las variables predictoras $X_1, ..., X_j$ del modelo.

El coeficiente de determinación del modelo

En el tema anterior se explicó el coeficiente de determinación simple, que marcaba la relación entre la variable predictora y la variable respuesta y a su vez daba una medida de la bondad del ajuste del modelo. En un modelo de regresión lineal general se puede definir el coeficiente de determinación del modelo o coeficiente de determinación general, que también denotaremos con R^2 y se calcula como el cociente entre la suma de cuadrados relativa a la regresión y la suma de cuadrados de la varianza de la respuesta total, tal y como se formula en la Ecuación 10.

$$R^2 = \frac{SSR(X_1, ..., X_j)}{SST} = \frac{SST - SSE}{SST} = 1 - \frac{SSE}{SST}. \quad (10)$$

Coeficiente de determinación general ajustado

El coeficiente de determinación general nos proporciona una información sobre la bondad del ajuste del modelo. Sin embargo, no resulta útil para poder elegir entre diferentes modelos con un número de variables predictoras distintas. Por la propia definición del coeficiente de determinación, a medida que se van añadiendo más variables en el modelo, el coeficiente de determinación siempre va a aumentar, aunque las variables añadidas no aporten mucha información. Para tener una medida que penalice los modelos complejos existe el coeficiente de determinación general ajustado,

que se define tal y como se muestra en la Ecuación 11.

$$R^2_{adj} = 1 - \frac{(n-1)\cdot SSE}{(n-m-1)\cdot SST}. \quad (11)$$

Se observa que a medida que aumenta el número de variables m en el modelo el cociente $\frac{n-1}{n-m-1}$ será mayor y por tanto, el valor de R^2_{adj} decrece respecto al que se obtendría del coeficiente de determinación sin ajustar. Es interesante comentar, aunque no se va a profundizar que existen otros estadísticos que permiten evaluar la bondad del ajuste de un modelo comparando el número de variables que presentan. Dos de los más conocidos son el índice de AIC (Akaike information criterion) y el BIC (Bayesian information criterion), donde cuanto menor sea su valor, mejor ajuste presentará el modelo.

En los modelos de regresión lineal múltiple, además de comprobar las hipótesis que aplicaban a la regresión lineal simple es necesario comprobar que no existe colinealidad entre las variables predictoras. La colinealidad significa que una variable está linealmente relacionada o es combinación lineal de una o varias de las otras variables del modelo. Si esto sucede, tendremos problemas de identificabilidad, ya que va a ser muy difícil determinar a qué es debido los cambios sobre la variable respuesta. Además, frente a pequeños cambios en la selección de la muestra se pueden obtener grandes cambios en la estimación de los parámetros del modelo. Para solventar estos problemas es conveniente realizar primero un análisis sobre lo correlacionadas que están las variables predictoras entre sí mediante una matriz de correlación y posteriormente, aplicar alguna técnica de selección de variables predictoras en el modelo.

Para poder detectar si existe colinealidad entre los distintos predictores de un modelo se puede emplear el factor de inflación de la varianza, más conocido como VIF, por sus siglas en inglés. El VIF analiza para cada variable su coeficiente de determinación frente

a un modelo ajustado por el resto de variables del modelo según la Ecuación 12.

$$VIF = \frac{1}{1 - R^2}. \tag{12}$$

Se observa que cuanto mayor sea el coeficiente de determinación, mayor será el valor del VIF y por tanto mayor será la colinealidad de las variables predictoras. A partir de un VIF de 10 se considera que puede existir un problema de colinealidad entre las variables predictoras. Este análisis se puede realizar antes de realizar la regresión lineal o, si se hace alguna selección de variables, puede realizarse a posteriori con las variables que finalmente se han utilizado para ajustar el modelo. En Python, se puede calcular de manera rápida empleando la función *variance_inflation_factor* de la librería *statsmodels*.

5.4 Selección de variables

En este apartado, se desarrollan algunos de los métodos que más se han empleado, y se siguen empleando, en la selección de variables. La selección de variables para ajustar un modelo de regresión es un tema sobre el cual se ha discutido en profundidad en los últimos tiempos y cuyo debate sigue todavía vigente.

En algunos modelos, algunas variables predictoras pueden no aportan ninguna información a la variable respuesta y se deberían de excluir del modelo. En otras situaciones, alguas de las variables predictoras pueden estar muy relacionadas entre sí y aportar prácticamente la misma información a la variable respuesta, pudiendo causar, como se ha comentado, problemas de colinealidad.

En estas situaciones, sería conveniente aplicar el principio de parsimonia o *navaja de Ockham*, que afirma que ante dos modelos con una capacidad predictiva similar deberíamos escoger siempre el modelo más sencillo.

Para realizar la selección de variables de un modelo de regresión lineal general existen distintas estrategias.

El mejor subconjunto de variables (best subset)

Uno de los métodos más habituales de selección de variables es el método de selección del mejor modelo ("best subset"), en el cual, fijado el número de variables, se analizan todos los modelos posibles y se obtiene el mejor modelo según sus capacidades predictivas, por ejemplo, atendiendo al coeficiente de determinación del modelo R^2.

Para desarrollar esta estrategia se debe estimar y evaluar el modelo de regresión para cada una de las posibles combinaciones de las m variables predictoras. Una vez se tiene al mejor modelo de cada dimensión, se selecciona al mejor modelo de entre las distintas dimensiones atendiendo a algún criterio que hayamos prefijado de antemano, como por ejemplo el R^2_{adj}, o algún otro estadístico de la bondad del ajuste del modelo como el AIC, el BIC, o por validación cruzada.

Esta estrategia puede realizarse cuando el número de variables m no es muy elevado, ya que a partir de un cierto número de variables predictoras el número de posibles combinaciones de modelos se hace computacionalmente muy costoso de realizar. En concreto, se deberían ajustar m modelos de una variable, $\binom{m}{2}$ modelos con 2 variables, $\binom{m}{3}$ modelos con 3 variables, etc., con lo que el número total de modelos a ajustar es 2^m.

Selección de variables por etapas (stepwise selection)

Una alternativa computacionalmente más económica se encuentra en los métodos de selección de variables por etapas ("paso a paso" o *stepwise*), que a diferencia del método de selección del mejor modelo, son métodos que intentan buscar el mejor modelo de forma secuencial. Dentro de la selección de variables "paso a paso", existe

la selección hacia adelante y la selección hacia atrás.

En la selección hacia adelante se parte del modelo nulo (M_0) y se añade a cada paso la variable que añada más información al modelo. El proceso termina cuando el añadir una variable no mejora el ajuste del modelo.

En la selección hacia atrás se actúa de manera similar, pero se parte del modelo con todas las variables (M_m) y se elimina en cada paso a la variable con la que se consigue mejorar más el ajuste del modelo. El proceso termina cuando eliminar una variable no mejora el ajuste del modelo. Es interesante hacer notar que si $m > n$ sólo se puede utilizar la selección hacia adelante (y terminar en el paso n), ya que los modelos con más parámetros que observaciones no son identificables.

5.5 Extensiones de los Modelos lineales

Hasta ahora, se ha visto el modelo lineal de regresión, con una o más de una variables predictoras, siendo todas ellas numéricas y continuas. Sin embargo, existen muchas variantes de este tipo de modelo que conforman lo que se conoce como los modelos lineales generales, y donde, aplican las mismas definiciones que se acaban de presentar.

Modelo lineal general

Los modelos lineales generales abarcan distintos modelos de regresión lineal con lo cual se permite analizar una gran variedad de situaciones. De hecho, el modelo de regresión lineal múltiple, en el que todas las variables predictoras son cuantitativas y no se expresa ningún efecto ni ninguna interacción entre las variables predictoras se puede considerar como un caso particular de un modelo lineal general. Aparte del modelo de regresión lineal múltiple, se puede encontrar distintos tipos de modelos que pertenecen a los modelos lineales generales. A continuación, sin entrar a profundizar,

se introducen los modelos más importantes:

- **Modelos de regresión polinómica**. Los modelos de regresión polinómica son modelos lineales generales que contienen potencias de las variables predictoras y, por tanto, producen funciones de respuesta no lineales. En este tipo de modelos es necesario ser cautelosos a la hora de escoger los grados de la regresión del modelo, ya que se puede producir un sobreajuste del modelo. A partir de un cierto grado de la regresión, se producen modelos que se ajustan muy bien a los datos con los que se estiman los coeficientes de regresión del modelo, pero sus capacidades predictivas son muy limitadas. En la Ecuación 13 se presenta un modelo de regresión polinómica cúbico de una variable.

$$Y = \beta_0 + \beta_1 \cdot X + \beta_2 \cdot X^2 + \beta_3 \cdot X^3 \tag{13}$$

Se puede entender el modelo de regresión polinómica de grado 3 como un modelo de regresión lineal múltiple con 3 variables predictoras en el que se podría definir $X_1 = X;\ \ X_2 = X^2$ y $X_3 = X^3$.

- **Modelos con interacciones**. Otra clase de modelos lineales generales son aquellos en los que se incluyen los efectos de las interacciones entre todas o algunas de las distintas variables predictoras sobre la variable respuesta. Este tipo de modelos se pueden tratar también como un caso particular dentro de los modelos de regresión lineal múltiple. Por ejemplo, el modelo presentado en la Ecuación 14 se puede expresar como un modelo de regresión lineal múltiple si sustituimos la interacción $X_1 \cdot X_2$ por una variable X_3 tal y como se muestra en la Ecuación 15.

$$Y = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2 + \beta_3 \cdot X_1 \cdot X_2, \quad \text{es equivalente a:} \tag{14}$$

$Y = \beta_0 + \beta_1 \cdot X + \beta_2 \cdot X_2 + \beta_3 \cdot X_3,$ considerando la interacción $X_1 \cdot X_2$ como X_3. (15)

- **Modelos con variables predictoras categóricas**. Hasta ahora se ha supuesto que todas las variables predictoras eran numéricas y continuas, pero también se pueden introducir variables predictoras cualitativas en los modelos de regresión. La manera en la cual se introducen estas variables categóricas en el modelo de regresión es mediante variables indicadoras (o variables *dummy*). Para observar el modo en el que funcionan estas variables indicadoras se va a trabajar con un posible modelo. Por ejemplo, se quiere estimar la altura de los niños entre 8 y 10 años mediante un modelo de regresión lineal, parece adecuado introducir como variables predictoras la edad y el sexo. En este caso, la formulación del modelo que se podría realizar, de modo similar a las ecuaciones anteriores sería la que se indica en la Ecuación 16:

$$Altura = \beta_0 + \beta_1 \cdot Edad + \beta_2 \cdot Sexo. \quad (16)$$

Sin embargo, con esta formulación no queda claro que significa β_2 ni cómo se va a introducir el valor hombre o mujer en el modelo. La solución para conseguir formular adecuadamente este modelo es realizarlo a través de las variables indicadoras. Veamos la Ecuación 17:

$$Altura = \beta_0 + \beta_1 \cdot Edad + \beta_2 \cdot I(Sexo), \quad (17)$$

donde la variable *I(Sexo)* es una variable indicadora, que valdrá 0 cuando *Sexo = hombre* y 1 cuando *Sexo = Mujer*. Esto quiere decir que:

$$HombreE(Y|X_1, X_2) = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot 0 = \beta_0 + \beta_1 \cdot X_1$$

$$MujerE(Y|X_1, X_2) = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot 1 = (\beta_0 + \beta_2) + \beta_1 \cdot X_1$$

y ambas funciones respuesta son rectas paralelas con diferentes interceptos. En este modelo, el significado de β_2 se corresponde con el efecto de que el sexo sea mujer en vez de hombre (y el efecto de que el sexo sea hombre viene recogido en la interceptación del modelo, β_0).

Si se tienen más de dos niveles, (en concreto, n niveles), de la variable categórica, habrá que introducir $n-1$ variables indicadoras. Por ejemplo, si se está realizando un modelo de regresión sobre la respuesta de un paciente respecto a su edad y al tratamiento empleado (con 3 posibles tratamientos), se formularía el modelo del modo que se presenta en la Ecuación 18.

$$Altura = \beta_0 + \beta_1 \cdot Edad + \beta_2 \cdot \text{Tratamiento B} + \beta_3 \cdot \text{Tratamiento C}, \qquad (18)$$

donde el efecto del tratamiento A viene recogido en la interceptación.

- **Modelos con la variable respuesta transformada**. Otra familia de modelos de regresión lineal generales son los modelos en los que se realiza una transformación a la variable respuesta. Estos modelos pueden considerarse también como casos particulares del modelo de regresión lineal múltiple. Por ejemplo, el modelo en el que se ajusta un regresión lineal múltiple a una respuesta logarítmica, tal y como se muestra en la Ecuación 19.

$$log(Y) = \beta_0 + \beta_j \cdot X_j; \quad j = 1, ..., m. \qquad (19)$$

En esta situción, se puede proceder de manera habitual a la hora de estimar los coeficientes de regresión del modelo, pero teniendo en cuenta que el resultado está

afectando al logaritmo de Y (en lugar de a Y).

Modelo lineal generalizado (GLM)

En todos los casos anteriores, los errores de los ajustes de los modelos presentaban un error con distribución normal, pero cuando esto no sucede, es posible acudir a un modelo de regresión lineal mediante un modelo lineal generalizado (GLM, por sus siglas en inglés) a través de una función de enlace.

Así pues, el GLM es una generalización flexible de la regresión lineal ordinaria que permite operar con variables de respuesta que tienen modelos de distribución de errores distintos a los de una distribución normal. El GLM generaliza la regresión lineal al permitir que el modelo lineal esté relacionado con la variable de respuesta a través de la función de enlace.

Entre las aplicaciones más importantes de los GLM, podemos encontrar la aplicación de modelos de regresión lineal generalizado cuando la variable de respuesta es **binomial**, habitualmente mediante la regresión logística (aunque existen otras regresiones cuando se emplean otras funciones de enlace distintas al *logit*); o cuando la variable respuesta es de tipo ***poisson***, mediante la regresión de *poisson* (utilizando como función de enlace el logaritmo).

Modelo aditivo generalizado (GAM)

Por último, también se pueden aplicar modelos de regresión en los que no es posible generalizarlos mediante un modelo lineal. Estos modelos son conocidos como modelos aditivos generalizados (o GAM, por sus siglas en inglés). Entre los modelos GAM, uno de los más empleados es el que está basado en *splines* cúbicos, y se ajustan los valores de Y mediante regresiones polinómicas de grado 3, pero a trozos, esto es, dividiendo el espacio de la variable o variables predictoras X_j en distintos nudos. En la Figura 2

se muestra el ajuste mediante un modelo aditivo generalizado con base de Splines cúbicos para el ejemplo de las horas trabajadas vs las ganancias que vimos en el Tema 4.

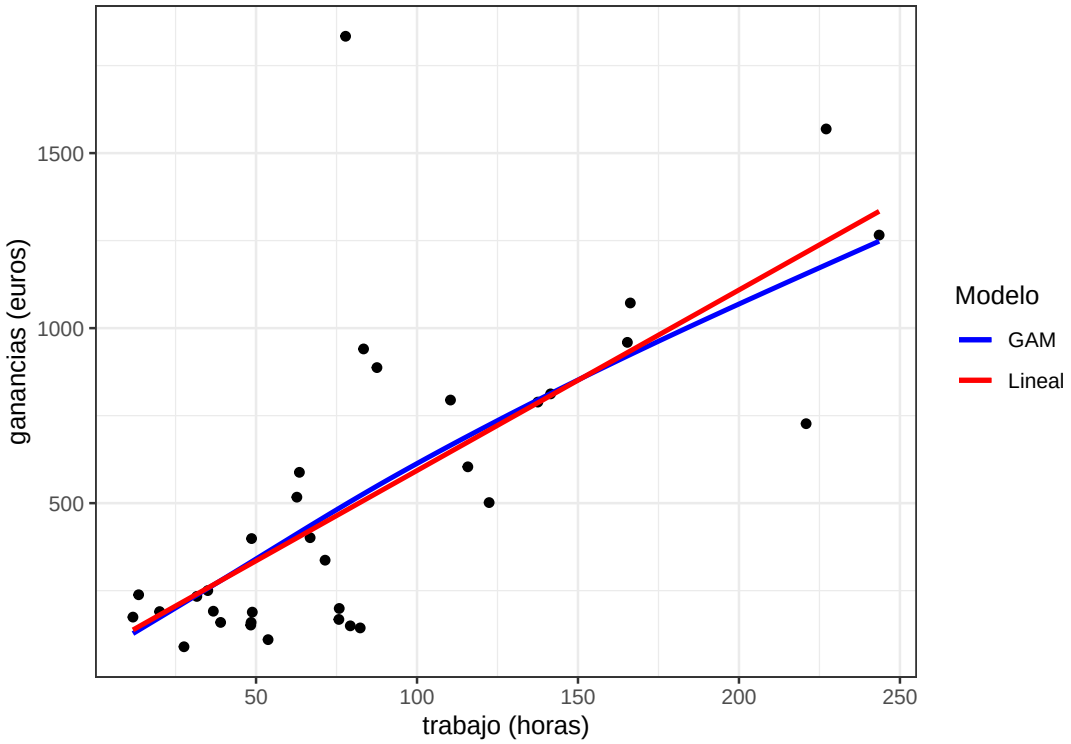


Figura 2: Ajuste ganancias-trabajo mediante un GAM de splines cúbicos

En este caso, el ajuste obtenido mediante el modelo aditivo generalizado no es muy diferente al que se obtuvo con el modelo de regresión lineal simple, pero ante otro tipo de problemas puede funcionar mejor.

Los modelos aditivos generalizados pueden capturar relaciones no lineales entre las variables y suelen ajustar muy bien sobre los datos que se emplean en el entrenamiento. Sus capacidades predictivas van a depender del tipo de problema en el que nos encontremos, y tienen como mayor inconveniente en que son muy difíciles de interpretar.

5.6 Regresión lineal múltiple en la práctica

Una vez que se han abordado los conceptos teóricos sobre la regresión lineal múltiple, es necesario tratar los problemas que vamos a encontrar en la práctica cuando apliquemos estos métodos sobre un conjunto de datos real.

Datos faltantes

En los conjuntos de datos reales hay datos faltantes: existen datos que se han perdido, medidas que no se han tomado, otras que se han tomado mal y la lista podría continuar. Para poder abordar esta problemática se hace necesario escoger estrategias para abordar los datos faltantes. La estrategia más conservadora consiste en eliminar todas las observaciones que tenga algún dato faltante en algunas de sus variables. Esta estrategia se puede hacer cuando o bien tenemos una muestra muy grande y pocos datos faltantes o cuando la información de los datos faltantes es de crucial importancia y es difícil de estimar. Sin embargo, es una estrategia en la cual estamos perdiendo el resto de información de la observación. Para no perder toda esta información existe otra estrategia consistente en imputar los valores faltantes. Esta imputación se puede realizar con técnicas de regresión donde se trata el dato faltante como una variable respuesta y se utilizan el resto de variables como predictores o, de manera más sencilla, se puede imputar por la mediana (ya hemos visto que es más robusto que la media) para las variables numéricas y por la moda para las variables categóricas.

Preprocesar las variables

Otra de las operaciones habituales en las técnicas de regresión es preprocesar las variables predictoras para que tengan una escala comparable. Este preprocesado puede consistir en;

- **Centrar** las variables (restarle a cada valor su media).

- **Estandarizar** (restarle la media y dividir por su desviación estandar).

- **Escalar** la variable al rango [0,1].

En el caso de la regresión lineal múltiple por mínimos cuadrados no es necesario realizar ninguna de estas transformaciones de preprocesado, aunque tampoco van a variar los valores de los parámetros del modelo (exceptuando la intercerptación) si se realizan. Por este motivo, cuando se van a aplicar diferentes técnicas de regresión, además de la regresión lineal por múltiples cuadrados, se suele estandarizar las variables predictoras antes.

Material audiovisual

Accede al vídeo: Regresión Avanzada

5.7 Referencias bibliográficas

Geron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* 2 edition.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition.

Pat Fernández, L. A. (2013). *Introducción a los modelos de regresión*. Plaza y Valdés, México, D.F.

Peña, D. (1987). *Regresión y diseño de experimentos*.

5.8 Ejercicios resueltos

Ejercicio 1

En este ejercicio profundizarás en los modelos de regresión lineal general. Carga el dataset **California housing**, y realiza:

a) Un modelo de regresión lineal general de la mediana de los valores de las casas (variable respuesta Y) con todas las variables numéricas del dataset.

b) La librería *Scikit-Learn* no proporciona los p. valores o intervalos de confianza de la estimación de los coeficientes de regresión del modelo. Obten estos valores mediante la librería *statsmodels*. ¿Qué variables son estadísticamente significativas en el modelo de regresión general? Además, representa los valores predichos del modelo de regresión lineal general frente a los reales. ¿Ajusta bien el modelo?

c) ¿Cuál sería la predicción de un barrio donde todas sus variables numéricas estuvieran en el valor de la mediana y su variable categórica tuviera el valor de la moda?

Solución

El primer paso, como viene siendo habitual es importar las librerías que se van a emplear en la resolución de los ejercicios:

```
# cargar librerias-----------------------------------------------
import pandas as pd
import numpy as np
from pandas.core.common import flatten
from plotnine import *
from array import *
import scipy.stats as stats
import math
import matplotlib as mpl
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
```

Una vez que se han importado las librerías, se carga el dataset **California housing** (en el Tema 3 viene una explicación de como descargarlo de la web desde Python) en un dataframe.

```
# path que se va a crear en nuestro sistema----------------------
HOUSING_PATH = os.path.join("datasets", "housing")

# definir una funcion que cargue el csv en un dataframe----------
def load_housing_data(housing_path=HOUSING_PATH):
    csv_path = os.path.join(housing_path, "housing.csv")
    return pd.read_csv(csv_path)

housing = load_housing_data()
```

Como se va a realizar el ajuste de un modelo de regresión lineal general de la variable respuesta *median_house_value* con el resto de variables del dataset como variables

predictoras es necesario separar en la respuesta Y, y el resto de variables del dataset.

```
# separar variable respuesta del dataset-------------------------
respuesta = housing["median_house_value"].copy()
housing = housing.drop("median_house_value", axis=1)
```

Recordando la información sobre las variables que conforman el dataset nos damos cuenta que es necesario salvar un obstáculo para poder ajustar el modelo de regresión lineal: la variable total_bedrooms tiene menos valores que el resto de variables, esto quiere decir que tiene datos faltantes.

```
# recordatorio variables numericas-------------------------------
print(housing.info())
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 20640 entries, 0 to 20639
## Data columns (total 9 columns):
##  #   Column              Non-Null Count  Dtype
## ---  ------              --------------  -----
##  0   longitude           20640 non-null  float64
##  1   latitude            20640 non-null  float64
##  2   housing_median_age  20640 non-null  float64
##  3   total_rooms         20640 non-null  float64
##  4   total_bedrooms      20433 non-null  float64
##  5   population          20640 non-null  float64
##  6   households          20640 non-null  float64
##  7   median_income       20640 non-null  float64
##  8   ocean_proximity     20640 non-null  object
## dtypes: float64(8), object(1)
## memory usage: 1.4+ MB
## None
```

Es necesario hacer algo con los datos faltantes que se encuentran en la variable *total*

bedrooms. En el tema anterior se eliminaron esas observaciones, pero para no perder la información que sí proporcionan el resto de variables en este caso se va a imputar el valor faltante, por el valor de la mediana de la variable. Para lograr la imputación se va a hacer uso del objeto *SimpleImputer()* en el cual definiremos como estrategia de imputación la mediana.

Este imputador va a tratar de imputar todas las variables del dataset. Si se prueba a imputar la base de datos original obtendremos un error relacionado con que no se puede imputar la mediana en una variable categórica. Así pues, quitaremos la variable categórica *ocean_proximity* y se imputará el valor de los datos faltantes.

Para quitar una variable empleamos el método *.drop()* donde en el argumento *axis* definimos que se refiere a columnas utilizando *axis=1*. Hay que tener en cuenta que al realizar la imputación mediante el método *.transform()* se obtiene un *array* y es necesario volver a definirlo como un dataframe con la función *pd.DataFrame()*. Por último, se vuelve a aplicar el método *.info()* sobre el dataframe imputado para comprobar que ya no existen datos faltantes.

```
# imputar NAs-----------------------------------------------------
from sklearn.impute import SimpleImputer
imputador = SimpleImputer(strategy="median")
# imputador.fit(housing) # da error
# se quita la variable categorica---------------------------------
housing_num = housing.drop("ocean_proximity", axis=1)
# se calcula la imputacion----------------------------------------
imputador.fit(housing_num)
# los valores de las medianas-------------------------------------
```

```
## SimpleImputer(strategy='median')
```

```
print(imputador.statistics_)
# se aplica la imputacion-----------------------------------------
```

```
## [-118.49     34.26     29.     2127.      435.     1166.      409.
##     3.5348]
```

```
housing_num_i = imputador.transform(housing_num)
# volver a data frame-----------------------------------------------
housing_i = pd.DataFrame(housing_num_i,
columns=housing_num.columns, index=housing.index)
# Comprobar df imputado---------------------------------------------
print(housing_i.info())
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 20640 entries, 0 to 20639
## Data columns (total 8 columns):
##  #   Column              Non-Null Count  Dtype
## ---  ------              --------------  -----
##  0   longitude           20640 non-null  float64
##  1   latitude            20640 non-null  float64
##  2   housing_median_age  20640 non-null  float64
##  3   total_rooms         20640 non-null  float64
##  4   total_bedrooms      20640 non-null  float64
##  5   population          20640 non-null  float64
##  6   households          20640 non-null  float64
##  7   median_income       20640 non-null  float64
## dtypes: float64(8)
## memory usage: 1.3 MB
## None
```

Una vez que se ha comprobado que ya no existen valores faltantes se procede a preprocesar las variables predictoras. En muchos de los métodos de aprendizaje automático es necesario que el rango de las variables predictoras sea similar (la variable respuesta no suele hacer falta escalarla). Para ello existen varias estrategias, tal y como se ha visto en el último apartado del tema.

En este ejercicio se va a optar por estandarizar las variables predictoras. Esta estandarización se debe realizar después de imputar. Para automatizar estos procesos para futuros problemas se puede definir la clase *pipeline* de *Scikit-Learn*, la cual nos permite realizar todas estas operaciones en el orden correcto. En este caso, se crea un *Pipeline* con 2 pasos: imputar y estandarizar, que aplicaremos sobre las variables numéricas del dataset.

```
# importar el "estandarizador"-----------------------------------
from sklearn.preprocessing import StandardScaler
# importar la clase pipeline"------------------------------------
from sklearn.pipeline import Pipeline
# definir el pipeline--------------------------------------------
num_pipeline = Pipeline([
        ("imputador", SimpleImputer(strategy="median")),
        ("std_scaler", StandardScaler()),
    ])

# aplicar el pipeline--------------------------------------------
housing_num_tr = num_pipeline.fit_transform(housing_num)
```

```
# importar clase-------------------------------------------------
from sklearn.linear_model import LinearRegression
# ajustar el modelo----------------------------------------------
lm1 = LinearRegression()
lm1.fit(housing_num_tr, respuesta)
# obtener coeficientes del modelo--------------------------------
# intercepto
```

```
## LinearRegression()
```

```
print(lm1.intercept_)
# coeficientes de regresion
```

```
## 206855.81690891393
```

```
print(lm1.coef_)
```

```
## [-85369.22518    -90723.40175504  14403.20315262 -14443.94445799
##   34037.42560482 -45153.79498679  30319.8204304   75520.30834439]
```

Se obtiene en *lm1.intercept_* la interceptación del modelo y en *lm1.coef_* los betas del modelo.

b)

Para ajustar el modelo de regresión lineal mediante statmodels se va a aprovechar todo el preprocesado empleado en el apartado a). Se importa la librería *statsmodels.api* y la clase stats de la librería *scipy*. Se define la matriz X como el conjunto de valores de housing_prepared y como y la variable respuesta. Es necesario, además, definir explícitamente que el modelo tiene interceptación mediante la función *sm.add_constant()* a la matriz X. El ajuste por minimos cuadrados se construye con la función *sm.OLS()*. Por último, podemos representar el sumario de la regresión con la función *est2.summary()*.

```
# cargar funciones-----------------------------------------------
import statsmodels.api as sm
from scipy import stats
# definir matriz de disenyo y variable respuesta-----------------
X = housing_num_tr
y = respuesta
# anyadir intercepto---------------------------------------------
X2 = sm.add_constant(X)
# ajustar el modelo----------------------------------------------
est = sm.OLS(y, X2)
# ver ajuste-----------------------------------------------------
est2 = est.fit()
print(est2.summary())
```

Dep. Variable:	median_house_value	**R-squared:**	0.636
Model:	OLS	**Adj. R-squared:**	0.635
Method:	Least Squares	**F-statistic:**	4498.
Date:	Sat, 09 Jan 2021	**Prob (F-statistic):**	0.00
Time:	02:36:21	**Log-Likelihood:**	-2.5945e+05
No. Observations:	20640	**AIC:**	5.189e+05
Df Residuals:	20631	**BIC:**	5.190e+05
Df Model:	8		

	coef	**std err**	**t**	**P$>\|$t$\|$**	**[0.025**	**0.975]**
const	2.069e+05	484.967	426.536	0.000	2.06e+05	2.08e+05
x1	-8.537e+04	1430.745	-59.668	0.000	-8.82e+04	-8.26e+04
x2	-9.072e+04	1440.817	-62.967	0.000	-9.35e+04	-8.79e+04
x3	1.44e+04	544.172	26.468	0.000	1.33e+04	1.55e+04
x4	-1.444e+04	1688.798	-8.553	0.000	-1.78e+04	-1.11e+04
x5	3.404e+04	2514.453	13.537	0.000	2.91e+04	3.9e+04
x6	-4.515e+04	1213.549	-37.208	0.000	-4.75e+04	-4.28e+04
x7	3.032e+04	2579.191	11.756	0.000	2.53e+04	3.54e+04
x8	7.552e+04	630.255	119.825	0.000	7.43e+04	7.68e+04

Omnibus:	5044.234	**Durbin-Watson:**	0.964
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	18974.225
Skew:	1.186	**Prob(JB):**	0.00
Kurtosis:	7.055	**Cond. No.**	14.1.

Se observa que todas las variables son estadísticamente significativas ya que el p.valor obtenido es menor de 0.05 para todas las variables. Además de por el p. valor, se puede comprobar la significatividad observando el intervalo de confianza al nivel $1-\alpha$ $[0.025-0.975]$ donde ninguna variable (quitando del mismo nivel donde el p. valor era mayor que 0.05) contiene al 0.

Se puede observar que el coeficiente de determinación del modelo es igual a 0.636, y que el coeficiente de determinación ajustado es 0.635. Que la diferencia entre uno

y otro sea tan pequeña es debido a que el número de observaciones es mucho más grande que el número de variables del modelo.

Aparte del coeficiente de determinación y de la varianza, se puede realizar un gráfico donde se comparan los valores reales de la muestra frente a los valores del ajuste (ver Figura 3. Cuanto más se aproximen los puntos del gráfico de dispersión a la bisectriz del primer cuadrante, mejor será el ajuste del modelo. Aunque este método es útil gráficamente, para decidir si un modelo ajusta bien o no es preferible basarse en los p.valores y en el coeficiente de determinación obtenido.

```
# pintar reales vs predichos
fig = plt.figure(figsize=(8, 8))
y_pred = lm1.predict(housing_num_tr)

plt.scatter(y, y_pred)
plt.xlabel("puntos reales")
plt.ylabel("puntos ajuste")
plt.plot([0, max(y)], [0, max(y)], color = 'red', linewidth = 3)
plt.show()
```

En términos generales el modelo ajusta bien, concentrándose la mayoría de las predicciones alrededor de la bisectriz del primer cuadrante, aunque existen algunas predicciones que arrojan valores negativos, por lo que es necesario trabajar en mejorar el modelo.

c) Para obtener el valor de la predicción de un barrio donde todas las variables numéricas tienen el valor de la mediana, lo primero es obtener los valores de esa observación y después emplear el método *.predict()* de dicha observación. Las medianas se obtienen de aplicar la función *median()* al *array* que contiene a las variables numéricas. Es interesante recordar que las medianas es necesario extraerlas de las variables numéricas una vez han pasado por el *pipeline()* del tratamiento numérico para introducirlas en la predicción con sus valores escalados.

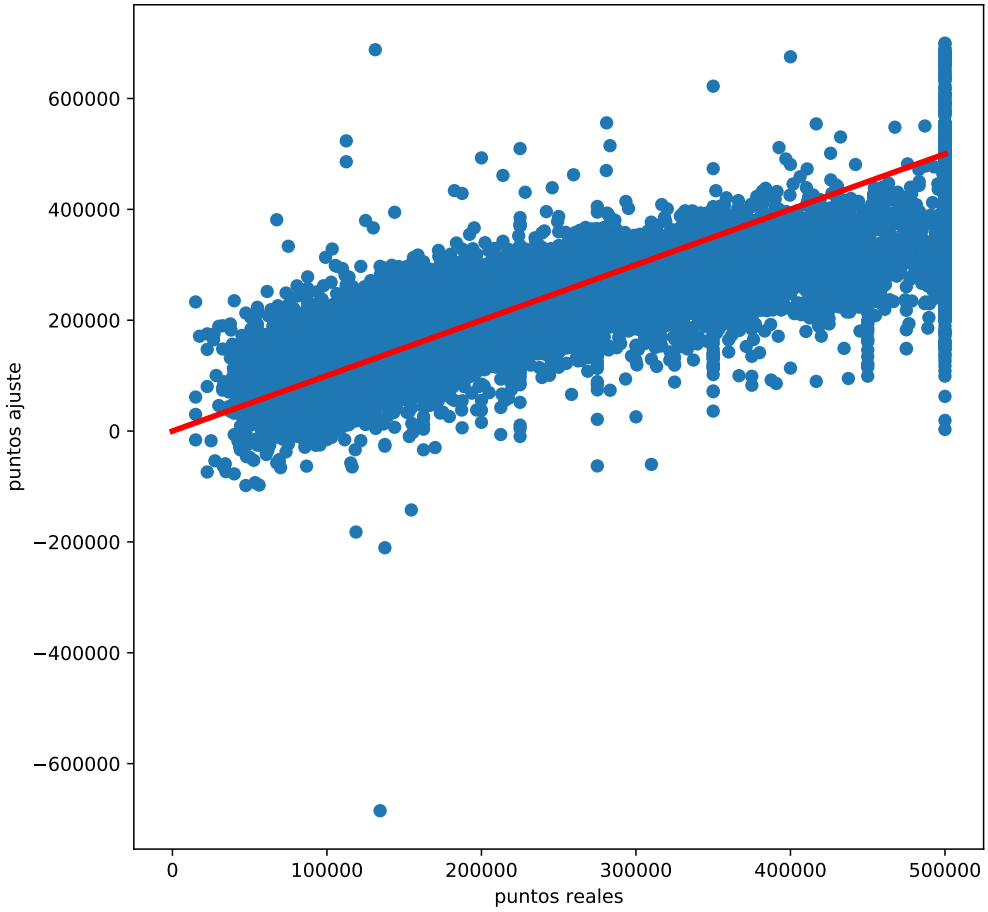


Figura 3: Valor real vs Valor ajustado

```
# obtener observacion medianas
medianas = np.median(housing_num_tr, axis=0)
print(medianas)
```

```
## [ 0.53891366 -0.6422871   0.02864572 -0.23321042 -0.24283094 -
0.22913175
##  -0.23681619 -0.17679508]
```

Se observa que el nivel más repetido es el primero, que es el que se incluye en la interceptación. Por lo tanto, hay que definir a los coeficientes de regresión relativos a esa variable (β_9 a β_12) como 0.

```
# obtener observacion para predecir
x_nueva = np.array(medianas)
x2 = x_nueva.reshape(1, -1)
y_nueva = lm1.predict(x2)
print("el valor de y predicho es: ", y_nueva, "dólares")
```

```
## el valor de y predicho es:  [204449.69888918] dólares
```

Ejercicio 2.

Repite el ejercicio 2 añadiendo la variable categórica a la regresión.

Solución

a) Ahora, es necesario definir las variables indicadoras correspondientes a la variable categórica. En primer lugar, se guarda la variable categórica del dataframe, *ocean_proximity*, en una nueva variable, *ocean_cat*. Después se crea el indicador de variables empleando el objeto *OneHotEncoder* de la librería *Scikit-Learn** y

se cambia el argumento *sparse = False* para que cree una matriz densa (con todos los 0 y 1) correspondiente a las diferentes variables indicadoras que contienen la información relativa a la variable categórica. Por último, se crea la matriz (el array) de variables indicadoras *ocean_indicadora*.

```
# Transformar variable categorica en variables indicadoras-------
# guardar variable-----------------------------------------------
ocean_cat = housing[["ocean_proximity"]]
# crear categorias para los encabezados-------------------------
header_ocean_cat = housing["ocean_proximity"].unique()
from sklearn.preprocessing import OneHotEncoder
indicar_var = OneHotEncoder(sparse=False)
ocean_indicadora = indicar_var.fit_transform(ocean_cat)
print(ocean_indicadora)
# volver a data frame--------------------------------------------
```

```
## [[0. 0. 0. 1. 0.]
##  [0. 0. 0. 1. 0.]
##  [0. 0. 0. 1. 0.]
##  ...
##  [0. 1. 0. 0. 0.]
##  [0. 1. 0. 0. 0.]
##  [0. 1. 0. 0. 0.]]
```

```
ocean_df = pd.DataFrame(ocean_indicadora,
columns = header_ocean_cat, index = housing.index)
# Comprobar ocean_df---------------------------------------------
print(ocean_df.info())
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 20640 entries, 0 to 20639
## Data columns (total 5 columns):
##  #   Column      Non-Null Count  Dtype
```

```
## ---  ------      --------------  -----
##  0   NEAR BAY    20640 non-null  float64
##  1   <1H OCEAN   20640 non-null  float64
##  2   INLAND      20640 non-null  float64
##  3   NEAR OCEAN  20640 non-null  float64
##  4   ISLAND      20640 non-null  float64
## dtypes: float64(5)
## memory usage: 806.4 KB
## None
```

```
print(ocean_df.head())
```

```
##    NEAR BAY  <1H OCEAN  INLAND  NEAR OCEAN  ISLAND
## 0       0.0        0.0     0.0         1.0     0.0
## 1       0.0        0.0     0.0         1.0     0.0
## 2       0.0        0.0     0.0         1.0     0.0
## 3       0.0        0.0     0.0         1.0     0.0
## 4       0.0        0.0     0.0         1.0     0.0
```

También, podemos añadir el paso de transformar la variable categórica a la *Pipeline* mediante la clase *ColumnTransformer* de la siguiente manera:

```
# importar clase-------------------------------------------------
from sklearn.compose import ColumnTransformer
# separar dataset en variables numericas y variable categorica---
num_attribs = list(housing_num)
cat_attribs = ["ocean_proximity"]
# definir full pipeline------------------------------------------
full_pipeline = ColumnTransformer([
        ("num", num_pipeline, num_attribs),
        ("cat", OneHotEncoder(), cat_attribs),
    ])
```

```
housing_prepared = full_pipeline.fit_transform(housing)
```

Se ha dividido el dataset entre las variables numéricas y la categórica y para cada trozo se introduce en el pipeline las transformaciones necesarias. Como resultado se obtiene el dataset housing_prepared. Al fin, ya se puede ajustar el modelo de regresión lineal.

```
# importar clase----------------------------------------------------
from sklearn.linear_model import LinearRegression
# ajustar el modelo-------------------------------------------------
lm2 = LinearRegression()
lm2.fit(housing_prepared, respuesta)
# obtener coeficientes del modelo-----------------------------------
# intercepto
```

```
## LinearRegression()
```

```
print(lm2.intercept_)
# coeficientes de regresion
```

```
## 242709.13524913642
```

```
print(lm2.coef_)
```

```
## [-52952.95152846 -53767.62485624  13312.88334575 -10320.06092603
##   29920.76507621 -44490.47744263  29746.22226671  73636.15586366
##  -23472.13460582 -63238.53335023 132593.58521653 -27169.53626692
##  -18713.38099356]
```

Se obtiene en *lm2.intercept_* la interceptación del modelo y en *lm2.coef_* los betas del modelo. Hay que tener en cuenta que los últimos cinco β corresponden con los niveles de la variable categórica (en este caso, no se han incluido ninguno de los niveles en el intercepto). Para incluir uno de los niveles en el intercepto y reducir el número de parámetros del modelo debemos definir el argumento *(drop = first)* en el transformador

de las variables categóricas de la Pipeline.

```
# importar clase----------------------------------------------
from sklearn.compose import ColumnTransformer
# separar dataset en variables numericas y variable categorica---
num_attribs = list(housing_num)
cat_attribs = ["ocean_proximity"]
# definir full pipeline---------------------------------------
full_pipeline = ColumnTransformer([
        ("num", num_pipeline, num_attribs),
        ("cat", OneHotEncoder(drop = "first"), cat_attribs),
    ])
housing_prepared = full_pipeline.fit_transform(housing)
# importar clase----------------------------------------------
from sklearn.linear_model import LinearRegression
# ajustar el modelo-------------------------------------------
lm2 = LinearRegression()
lm2.fit(housing_prepared, respuesta)
# obtener coeficientes del modelo-----------------------------
# intercepto
```

```
## LinearRegression()
```

```
print(lm2.intercept_)
# coeficientes de regresion
```

```
## 219237.0006433122
```

```
print(lm2.coef_)
```

```
## [-52952.95152846 -53767.62485624  13312.88334575 -10320.06092603
##   29920.76507621 -44490.47744263  29746.22226671  73636.15586366
##  -39766.3987444  156065.71982235  -3697.40166109   4758.75361226]
```

Los dos modelos son equivalentes, pero el segundo es más sencillo al haber eliminado un parámetro. Además, podemos observar cómo los parámetros relativos a los coeficientes de regresión de las variables numéricas se mantienen iguales.

b)

Para ajustar el modelo de regresión lineal mediante statmodels se va a aprovechar todo el preprocesado empleado en el apartado a). Se importa la librería *statsmodels.api* y la clase stats de la librería *scipy*. Se define la matriz X como el conjunto de valores de housing_prepared y como y la variable respuesta. Es necesario, además, definir explicitamente que el modelo tiene interceptación mediante la función *sm.add_constant()* a la matriz X. El ajuste por minimos cuadrados se construye con la función *sm.OLS()*. Por último, podemos representar el sumario de la regresión con la función *est2.summary()*.

```
# cargar funciones-----------------------------------------------
import statsmodels.api as sm
from scipy import stats
# definir matriz de disenyo y variable respuesta-----------------
X = housing_prepared
y = respuesta
# anyadir intercepto---------------------------------------------
X2 = sm.add_constant(X)
# ajustar el modelo----------------------------------------------
est = sm.OLS(y, X2)
# ver ajuste-----------------------------------------------------
est2 = est.fit()
print(est2.summary())
```

Dep. Variable:	median_house_value	**R-squared:**	0.645
Model:	OLS	**Adj. R-squared:**	0.645
Method:	Least Squares	**F-statistic:**	3129.
Date:	do., 18 oct. 2020	**Prob (F-statistic):**	0.00
Time:	02:36:21	**Log-Likelihood:**	-2.5917e+05
No. Observations:	20640	**AIC:**	5.184e+05
Df Residuals:	20627	**BIC:**	5.185e+05
Df Model:	12		

	coef	std err	t	P> |t|	[0.025	0.975]
const	2.192e+05	836.139	262.202	0.000	2.18e+05	2.21e+05
x1	-5.295e+04	2031.313	-26.068	0.000	-5.69e+04	-4.9e+04
x2	-5.377e+04	2135.527	-25.178	0.000	-5.8e+04	-4.96e+04
x3	1.331e+04	550.049	24.203	0.000	1.22e+04	1.44e+04
x4	-1.032e+04	1681.185	-6.139	0.000	-1.36e+04	-7024.805
x5	2.992e+04	2487.720	12.027	0.000	2.5e+04	3.48e+04
x6	-4.449e+04	1204.800	-36.928	0.000	-4.69e+04	-4.21e+04
x7	2.975e+04	2545.710	11.685	0.000	2.48e+04	3.47e+04
x8	7.364e+04	631.147	116.670	0.000	7.24e+04	7.49e+04
x9	-3.977e+04	1736.250	-22.904	0.000	-4.32e+04	-3.64e+04
x10	1.561e+05	3.08e+04	5.072	0.000	9.57e+04	2.16e+05
x11	-3697.4017	1906.041	-1.940	0.052	-7433.393	38.589
x12	4758.7536	1562.723	3.045	0.002	1695.692	7821.815

Omnibus:	5177.939	**Durbin-Watson:**	0.968
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	19747.113
Skew:	1.213	**Prob(JB):**	0.00
Kurtosis:	7.132	**Cond. No.**	127.

Se observa que todas las variables son estadísticamente significativas ya que el p.valor obtenido es menor de 0.05 para todas las variables. Hay una variable indicadora que tiene un p. valor > 0.05, pero es necesario que recordar que la significatividad de una variable se mide en su conjunto, y si un nivel sale significativo es necesario

mantener a esa variable en el modelo. Además de por el p. valor, se puede comprobar la significatividad observando el intervalo de confianza al nivel $1-\alpha\,[0.025-0.975]$ donde ninguna variable (quitando del mismo nivel donde el p. valor era mayor que 0.05) contiene al 0.

Se puede observar que el coeficiente de determinación del modelo es igual a 0.645, y que el coeficiente de determinación ajustado, con 3 decimales de precisión tiene también el mismo valor. Este hecho es debido a que el número de observaciónes es mucho más grande que el número de variables del modelo.

Aparte del coeficiente de determinación y de la varianza, se puede realizar un gráfico donde se comparan los valores reales de la muestra frente a los valores del ajuste (ver Figura 4). Cuanto más se aproximen los puntos del gráfico de dispersión a la bisectriz del primer cuadrante, mejor será el ajuste del modelo. Aunque este método es útil gráficamente, para decidir si un modelo ajusta bien o no es preferible basarse en los p.valores y en el coeficiente de determinación obtenido.

```
# pintar reales vs predichos
fig = plt.figure(figsize=(8, 8))
y_pred = lm2.predict(housing_prepared)

plt.scatter(y, y_pred)
plt.xlabel("puntos reales")
plt.ylabel("puntos ajuste")
plt.plot([0, max(y)], [0, max(y)], color = 'red', linewidth = 3)
plt.show()
```

En términos generales el modelo ajusta bien, concentrándose la mayoría de las predicciones alrededor de la bisectriz del primer cuadrante, aunque existen algunas predicciones que arrojan valores negativos, por lo que es necesario trabajar en mejorar el modelo.

c) Para obtener el valor de la predicción de un barrio donde todas las variables numé-

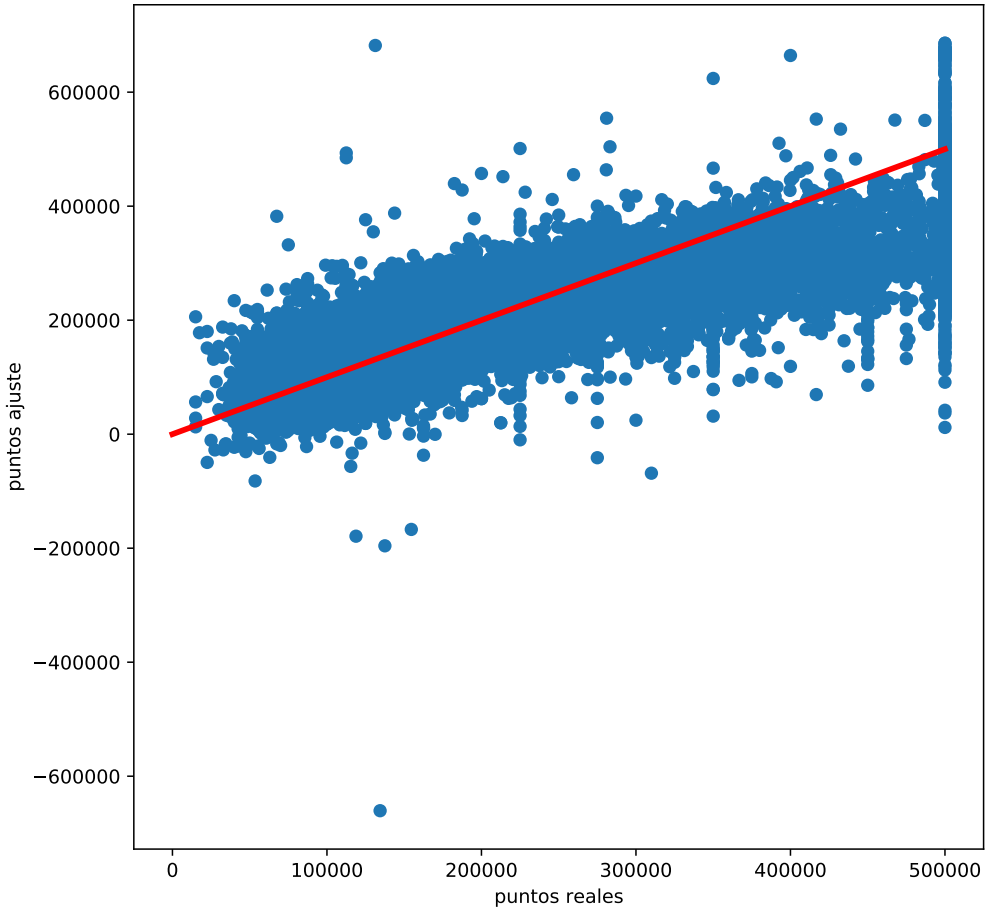


Figura 4: Valor real vs Valor ajustado

ricas tienen el valor de la mediana y la variable categórica tiene el valor de la moda, lo primero es obtener los valores de esa observación y después emplear el método *.predict()* de dicha observación. Las medianas se obtienen de aplicar la función *median()* al *array* que contiene a las variables numéricas, mientras que la moda de la variable categórica se extrae a partir del método *.value_conts()* de la variable categórica. Es importante recordar que las medianas es necesario extraerlas de las variables numéricas una vez han pasado por el *pipeline()* del tratamiento numérico para introducirlas en la predicción con sus valores escalados.

```
# obtener observacion medianas
medianas = np.median(housing_num_tr, axis=0)
print(medianas)
# la moda de
```

```
## [ 0.53891366 -0.6422871   0.02864572 -0.23321042 -0.24283094 -
0.22913175
##  -0.23681619 -0.17679508]
```

```
print(housing["ocean_proximity"].value_counts())
```

```
## <1H OCEAN     9136
## INLAND        6551
## NEAR OCEAN    2658
## NEAR BAY      2290
## ISLAND           5
## Name: ocean_proximity, dtype: int64
```

Se observa que el nivel más repetido es el primero, que es el que se incluye en la interceptación. Por lo tanto, hay que definir a los coeficientes de regresión relativos a esa variable (β_9 a β_12) como 0.

```
# obtener observacion para predecir
array_medianas = np.array(medianas)
x_nueva = np.append(array_medianas, [0, 0, 0, 0])
```

```
x2 = x_nueva.reshape(1, -1)
y_nueva = lm2.predict(x2)

print("el valor de y predicho es: ", y_nueva, "dólares.")
```

```
## el valor de y predicho es:  [210887.88263401] dólares.
```

Ejercicio 3. Analizar los resultados obtenidos de los ejercicios 1 y 2:

a) por separado, comprobar si existe colinealidad entre las variables predictoras.

b) compararlos entre sí y compararlos también con la regresión lineal simple del tema anterior.

Solución

a)

En el primer paso de este ejercicio se va a realizar es evaluar la posible colinealidad entre las variables predictoras de los modelos de regresión lineal múltiple de los Ejercicios 1 y 2. Para ello, se emplea la función *variance_inflation_factor* de la librería *statsmodels* sobre un objeto de tipo *pandas*. En este caso, se tiene que el *dataframe* del Ejercicio 1 con las variables numéricas después de realizar la imputación es *housing_i*.

```
# cargar funcion-----------------------------------------------
from statsmodels.stats.outliers_influence import
 variance_inflation_factor
# predictores ejercicio 1
housing_i.info()
# crear dataframe para aplicar VIF
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 20640 entries, 0 to 20639
## Data columns (total 8 columns):
##  #   Column              Non-Null Count  Dtype
## ---  ------              --------------  -----
##  0   longitude           20640 non-null  float64
##  1   latitude            20640 non-null  float64
##  2   housing_median_age  20640 non-null  float64
##  3   total_rooms         20640 non-null  float64
##  4   total_bedrooms      20640 non-null  float64
##  5   population          20640 non-null  float64
##  6   households          20640 non-null  float64
##  7   median_income       20640 non-null  float64
## dtypes: float64(8)
## memory usage: 1.3 MB
```

```
ej1_VIF =  pd.DataFrame()
ej1_VIF["variable"] = housing_i.columns
# aplicar el metodo
ej1_VIF["VIF"] = [variance_inflation_factor(housing_i.values, i)
                          for i in range(len(housing_i.columns))]
# resultados
print(ej1_VIF)
```

```
##              variable         VIF
## 0           longitude  614.390295
## 1            latitude  548.954416
## 2  housing_median_age    7.281102
## 3         total_rooms   29.032423
## 4      total_bedrooms   70.525566
## 5          population   16.173089
## 6          households   73.547059
## 7       median_income    8.003945
```

Se observa que la longitud y la latitud presentan unos valores de VIF muy elevados y podemos estar teniendo un problema de colinealidad.

De modo similar, se procede para obtener el VIF de las variables del modelo del Ejercicio 2, utilizando en este caso un dataframe de tipo *pandas* a partir de los dataframe *housing_i* y que vamos a llamar *housing_t*.

```
# cargar funcion-------------------------------------------------
from statsmodels.stats.outliers_influence import
 variance_inflation_factor
# predictores ejercicio 2
housing.info()
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 20640 entries, 0 to 20639
## Data columns (total 9 columns):
##  #   Column              Non-Null Count  Dtype
## ---  ------              --------------  -----
##  0   longitude           20640 non-null  float64
##  1   latitude            20640 non-null  float64
##  2   housing_median_age  20640 non-null  float64
##  3   total_rooms         20640 non-null  float64
##  4   total_bedrooms      20433 non-null  float64
##  5   population          20640 non-null  float64
##  6   households          20640 non-null  float64
##  7   median_income       20640 non-null  float64
##  8   ocean_proximity     20640 non-null  object
## dtypes: float64(8), object(1)
## memory usage: 1.4+ MB
```

```
housing_t = pd.merge(housing_i, ocean_df, left_index = True
, right_index=True)
# es necesario quitar una de las variables categoricas-----------
```

```
housing_t = housing_t.drop("<1H OCEAN", axis=1)
# Comprobar df creado-------------------------------------------
print(housing_t.info())
# crear dataframe para aplicar VIF
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 20640 entries, 0 to 20639
## Data columns (total 12 columns):
##  #   Column              Non-Null Count  Dtype
## ---  ------              --------------  -----
##  0   longitude           20640 non-null  float64
##  1   latitude            20640 non-null  float64
##  2   housing_median_age  20640 non-null  float64
##  3   total_rooms         20640 non-null  float64
##  4   total_bedrooms      20640 non-null  float64
##  5   population          20640 non-null  float64
##  6   households          20640 non-null  float64
##  7   median_income       20640 non-null  float64
##  8   NEAR BAY            20640 non-null  float64
##  9   INLAND              20640 non-null  float64
##  10  NEAR OCEAN          20640 non-null  float64
##  11  ISLAND              20640 non-null  float64
## dtypes: float64(12)
## memory usage: 1.9 MB
## None
```

```
ej2_VIF =  pd.DataFrame()
ej2_VIF["variable"] = housing_t.columns
# aplicar el metodo
ej2_VIF["VIF"] = [variance_inflation_factor(housing_t.values, i)
                          for i in range(len(housing_t.columns))]
# resultados
```

```
print(ej2_VIF)
```

```
##                variable         VIF
## 0             longitude  942.605262
## 1              latitude  844.284917
## 2    housing_median_age    8.116462
## 3           total_rooms   30.259091
## 4        total_bedrooms   70.550282
## 5            population   16.381178
## 6            households   75.251325
## 7         median_income    8.822349
## 8              NEAR BAY    3.881851
## 9                INLAND    1.003045
## 10           NEAR OCEAN    1.624071
## 11               ISLAND    1.798857
```

Se observa que los valores del VIF para la *longitude* y la *latitude* son muy elevados, además de otras variables como *total_bedrooms* y *households*, que también presentan valores elevados. Lo adecuado, en este caso es acudir a otro tipo de métodos que puedan adaptarse mejor a la colinieal idad.

b)

Una buena manera de comparar modelos con diferente número de variables predictoras es a través del valor del coeficiente de determinación ajustada. Se tiene, que:

- El modelo de regresión lineal simple obtenía un valor de $R^2 = 0.473$.
- El modelo con las variables numéricas un coeficiente de determinación ajustado de $R^2_{adj} = 0.636$.
- El modelo con todas las variables un coeficiente de determinación ajustado de

$R^2_{adj} = 0.645$, que esl mayor valor de los 3. Por lo tanto este es el mejor de los tres modelos.

En cualquier caso, como se ha visto que presenta problemas de colinealidad, lo adecuado sería buscar otro tipo de modelo de regresión, como por ejemplo alguno de los que veremos en el Tema 6.