Asignatura	Datos del alumno	Fecha
Técnicas Multivariantes	Apellidos:	
	Nombre:	

Laboratorio: Resolver un problema de regresión

- **Objetivos** de la actividad.

En esta actividad vas a profundizar en las distintas técnicas que se pueden aplicar para abordar un problema de regresión. Además, profundizarás en tus conocimientos sobre las librerías *statsmodels* y *scikit-learn* de Python.

- **Descripción** de la actividad.

El primer paso consiste en la creación de un conjunto de datos ficticio. Para garantizar que cada alumno obtiene un conjunto de datos distinto se va a emplear el documento de identidad de cada alumno para crear el conjunto de datos. Para que los números sean comparables entre todos los alumnos, si el número de identidad tiene menos de 8 cifras, replicaremos las primeras cifras hasta obtener exactamente 8 cifras. Además, para evitar los dígitos 0 y 1, si alguna de las cifras es menor que 2, sustituiremos esa cifra por 2. Aplicando estos cambios tendremos el **número del documento de identidad preparado** para la resolución de la actividad. Veamos algunos ejemplos:

- Ejemplo 1:

  12345678 -> 22345678
- Ejemplo 2:

  304156 -> 30414563**0** -> 3242563**2**

Una vez tenemos el número del documento de identidad preparado vamos a crear el conjunto de datos para el problema de clasificación con la función *sklearn.datasets.make_regression* de la librería scikit-learn (más información en:

Asignatura	Datos del alumno	Fecha
Técnicas Multivariantes	Apellidos:	
	Nombre:	

(https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_regression.html).

- Crea el conjunto de datos empleando los siguientes argumentos:
    - n_muestra = 200 + 10 · primer dígito dni
    - n_predictores = 10 + segundo dígito dni + tercer dígito dni
    - n_informativas = 10 + segundo dígito dni
    - sesgo = 2
    - ruido = 10 * cuarto dígito dni
    - semilla = numero_dni
    - shuffle = False

Este conjunto de datos tendrá al menos 220 observaciones y al menos 14 variables predictoras de las cuales al menos 2 no estarán relacionadas con la variable respuesta. Se añade el sesgo y el ruido para que la variable respuesta no sea una combinación lineal exacta de las variables informativas. Al definir *shuffle = False* estamos forzando a que las variables no informativas del conjunto de datos aparezcan al final.

- Divide el conjunto de datos en 200 observaciones para el entrenamiento y el resto para realizar la validación de los distintos métodos de regresión aplicados.
- Describe tu conjunto de datos (transformalo en un data.frame, aplica los métodos .info(), .describe() y obtén el histograma de todas las variables (predictoras y la variable respuesta).
- Obtén un modelo de regresión lineal múltiple. ¿Son todas las variables predictoras significativas? Utiliza la librería *statsmodels*.
- Realiza una selección de variables mediante un algoritmo de tipo step-wise donde en cada paso elimines la variable predictora menos significativa atendiendo a su p.valor hasta que todas las variables del modelo sean significativas (p.valor < 0.05).

Asignatura	Datos del alumno	Fecha
Técnicas Multivariantes	Apellidos:	
	Nombre:	

- Realiza una regresión mediante la red elástica. Prueba distintos valores de r y obtén el valor óptimo de r y de α mediante validación cruzada.
- Comprueba con la muestra de validación con cuál de los 3 modelos se obtiene un menor error cuadrático medio.
- No olvides añadir las referencias utilizadas para la elaboración del trabajo al final de la memoria.

- **Rúbrica**

Título de la actividad (valor real: 3 puntos)	Descripción	Puntuación máxima (puntos)	Peso %
Criterio 1	El conjunto de datos y su descriptivo son correctos	2	20%
Criterio 2	El modelo de regresión lineal múltiple y la selección de variables se realiza de manera adecuada	3	30%
Criterio 3	Se aplica correctamente el modelo de regresión de red elástica	3	30%
Criterio 4	La validación se ha realizado correctamente.	1	10%
Criterio 5	Presentación y formato de la memoria de la práctica. Bibliografía.	1	10%
		10	**100 %**

- **Extensión** máxima de la actividad (20 páginas).
- **SOLUCIÓN.** La solución dependerá del documento de identidad del alumno. A continuación, se resuelve, a modo de ejemplo, para el número: 12345678.

Asignatura	Datos del alumno	Fecha
Técnicas Multivariantes	Apellidos:	
	Nombre:	

Solución

Se resuelve la actividad con el DNI de ejemplo 12345678

Construir el conjunto de datos

- DNI = 12345678
- DNI_Preparado = 22345678

```
# 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

# guardar dni preparado------------------------------------------
numero_dni = 22345678
lista_dni = [2, 2, 3, 4, 5, 6, 7, 8]

# crear conjunto de datos----------------------------------------
n_muestra = 200 + 10 * lista_dni[0]
n_predictores = 10 + lista_dni[1] + lista_dni[2]
n_informativas = 10 + lista_dni[1]
sesgo = 2
ruido = 10 * lista_dni[3]
semilla = numero_dni

from sklearn.datasets import make_regression


X_0, y_0 = make_regression(n_samples = n_muestra, n_features = n_predictores, shuff
le = False,
                           random_state = semilla, noise = ruido, bias = sesgo)

# muestra de entrenamiento
X = X_0[0:200, :]
y = y_0[0:200]
# muestra de validacion
X_val = X_0[200:len(X_0)+1, :]
y_val = y_0[200:len(y_0)+1]

# comprobacion particion
X_0.shape

## (220, 15)
```

```
X.shape

## (200, 15)

X_val.shape

## (20, 15)

y_0.shape

## (220,)

y.shape

## (200,)

y_val.shape

## (20,)
```

Descriptivo

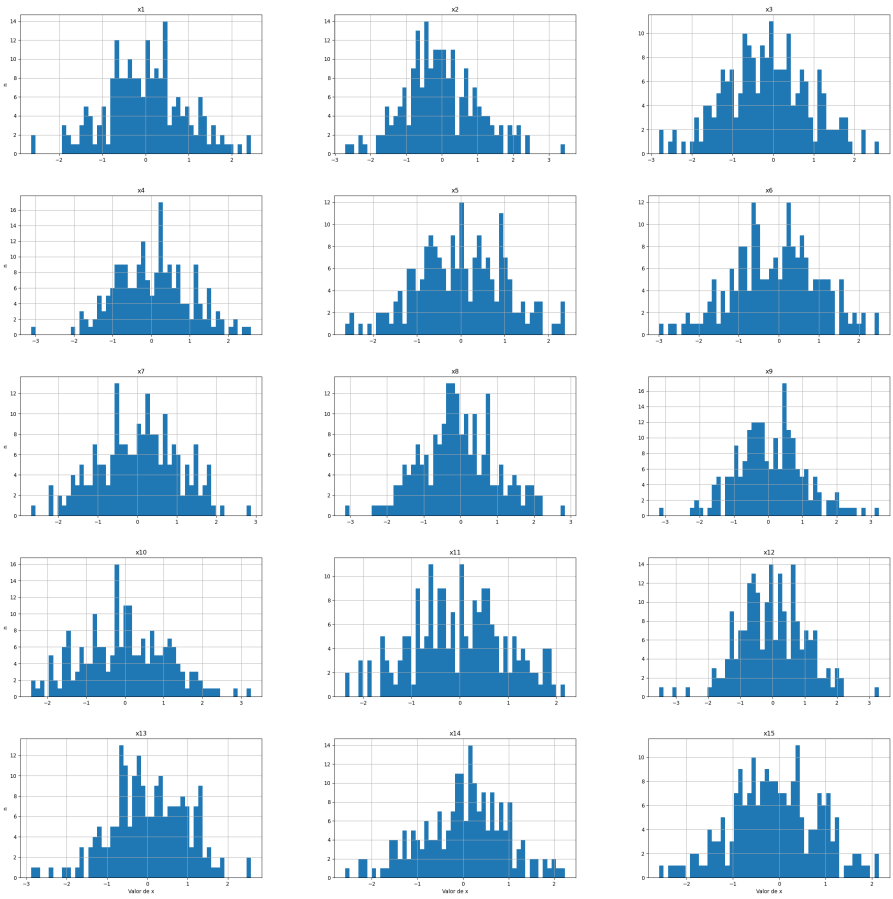
Una vez que se ha construido el conjunto de datos se pasa a representar el histograma de cada una de las variables predictoras, las cuales se muestran en la Figura 1. También se muestra la distribución de la variable respuesta y en la Figura 2.

```
# descriptivo de la base de datos--------------------------------
import matplotlib
import matplotlib.pyplot as plt

nombre_variables = ["x%d" % i for i in range(1, n_predictores + 1)]

df_X = pd.DataFrame(data=X[0:,0:], columns = nombre_variables)

df_X.info()

## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 200 entries, 0 to 199
## Data columns (total 15 columns):
##  #   Column  Non-Null Count  Dtype
## ---  ------  --------------  -----
##  0   x1      200 non-null    float64
##  1   x2      200 non-null    float64
##  2   x3      200 non-null    float64
##  3   x4      200 non-null    float64
##  4   x5      200 non-null    float64
##  5   x6      200 non-null    float64
##  6   x7      200 non-null    float64
##  7   x8      200 non-null    float64
##  8   x9      200 non-null    float64
##  9   x10     200 non-null    float64
##  10  x11     200 non-null    float64
##  11  x12     200 non-null    float64
##  12  x13     200 non-null    float64
##  13  x14     200 non-null    float64
##  14  x15     200 non-null    float64
## dtypes: float64(15)
## memory usage: 23.6 KB
```

Asignatura	Datos del alumno	Fecha
Técnicas Multivariantes	Apellidos:	
	Nombre:	

```
df_X.describe()

##                x1          x2          x3  ...         x13         x14         x
15
## count  200.000000  200.000000  200.000000  ...  200.000000  200.000000  200.0000
00
## mean    -0.005311   -0.031662   -0.159079  ...    0.046293    0.007734   -0.0999
67
## std      0.920068    1.029345    1.038639  ...    0.936165    0.931905    0.9252
66
## min     -2.643569   -2.712311   -2.803755  ...   -2.881037   -2.582123   -2.5892
02
## 25%     -0.615077   -0.674200   -0.863176  ...   -0.590079   -0.578988   -0.7255
77
## 50%      0.013161   -0.113682   -0.146445  ...    0.027768    0.093073   -0.1284
97
## 75%      0.511208    0.628305    0.506068  ...    0.764401    0.662945    0.5103
73
## max      2.435933    3.444393    2.599237  ...    2.572050    2.231854    2.1490
84
##
## [8 rows x 15 columns]

fig, axes = plt.subplots(nrows = 5, ncols = 3, figsize = (30,30))


df_X.hist(bins = 50, ax = axes);
plt.setp(axes[-1, :], xlabel = 'Valor de x');
plt.setp(axes[:, 0], ylabel = 'n');

plt.show();
```

Asignatura	Datos del alumno	Fecha
Técnicas Multivariantes	Apellidos:	
	Nombre:	



Figrua 1. Histogramas de las variables predictoras.

Se observa que todas las variables predictoras generadas son tienen una forma similar a la normal con una media cercana a 0 y una desviación estándar en torno a 1. La variable respuesta, en cambio, tiene una media de -10 y una desviación estándar de 183.

```
# descriptivo de la base de datos--------------------------------
name_y = ["y"]
df_y = pd.DataFrame(data = y.flatten(), columns = name_y)


df_y.info()
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 200 entries, 0 to 199
## Data columns (total 1 columns):
##  #   Column  Non-Null Count  Dtype
## ---  ------  --------------  -----
##  0   y       200 non-null    float64
## dtypes: float64(1)
## memory usage: 1.7 KB

df_y.describe()

##                 y
## count  200.000000
## mean   -10.087191
## std    183.666551
## min   -503.513817
## 25%   -130.022635
## 50%    -21.630332
## 75%    122.833237
## max    513.254929

df_y.hist(bins = 50, color = "red");
plt.xlabel("Valor de y");
plt.ylabel("n");

plt.show();
```

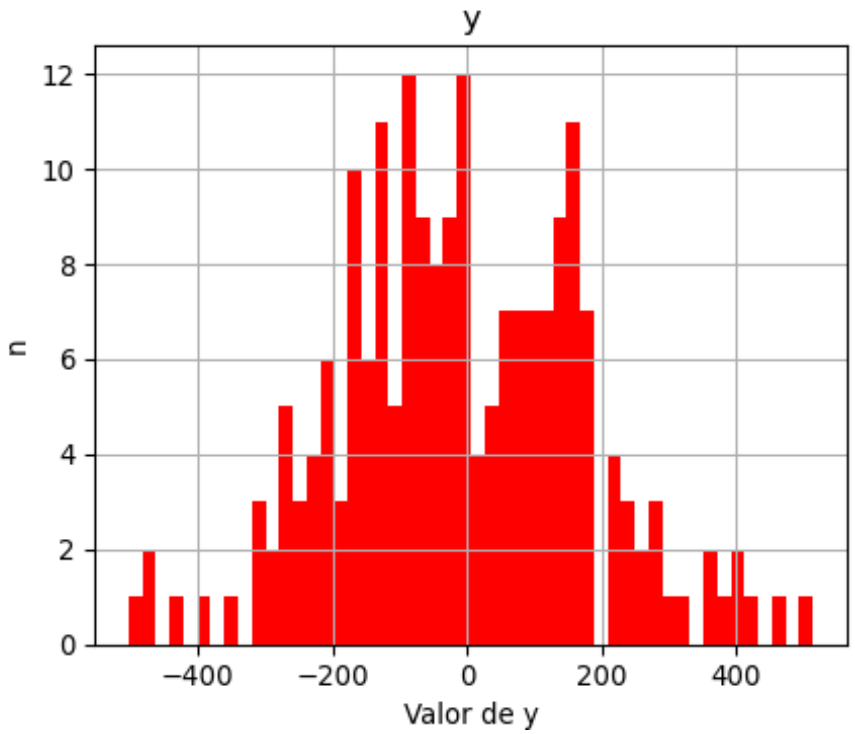


Figura 2. Histogramas de la variable respuesta y.

Asignatura	Datos del alumno	Fecha
Técnicas Multivariantes	Apellidos:	
	Nombre:	

Regresión lineal por mínimos cuadrados ordinarios

Una vez que ya se ha definido el conjunto de datos y se ha analizado descriptivamente se procede a ajustar por regresión lineal múltiple por mínimos cuadrados ordinarios. Se observa que algunas variables son no significativas. Podemos proceder mediante una metodología de selección de variables tipo step-wise a ir eliminando la variable menos significativa en el modelo y volviendo a ajustar cada vez con las variables restantes hasta que todas las variables sean significativas.

Step 1

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

## LinearRegression()

print(lm1.intercept_)
# coeficientes de regresion

## -3.055622988244945

print(lm1.coef_)

## [ 81.49196286   9.75892834  12.71182107   8.05715627  80.73115702
##    2.95401511  77.84715934 103.59119166  20.4889533   68.4641411
##    1.13352272   4.68912874   0.87363828   1.23843259   3.02177697]
```

La librería *scikit-learn* no aporta toda la información sobre el modelo de regresión lineal múltiple, por lo que se va a complementar el análisis con la librería *statsmodels*.

```
# cargar funciones---------------------------------------------
import statsmodels.api as sm
# anyadir intercepto-------------------------------------------
X1 = sm.add_constant(df_X)
# comprobar df-------------------------------------------------
X1.head()
# ajustar el modelo--------------------------------------------

##    const        x1        x2        x3  ...       x12       x13       x14
x15
## 0    1.0 -0.152005 -0.703594  0.422149  ...  2.000984  0.951938  0.013655  0.976
570
## 1    1.0 -1.853600  1.099557 -0.969300  ... -0.197315  0.190277  0.156317  0.079
980
## 2    1.0 -0.293498 -0.439834  0.595284  ... -0.877035  0.950507  1.055299  0.802
885
## 3    1.0 -1.685395  0.632809 -2.379930  ... -3.007046 -0.169368  0.596482  0.549
016
```

```
## 4    1.0  0.261581 -0.349490 -0.391385  ... -1.354455  0.286155 -1.058892  0.102
902
##
## [5 rows x 16 columns]

est_X1 = sm.OLS(y, X1)
# ver ajuste----------------------------------------------------
est1 = est_X1.fit()
print(est1.summary())

##                             OLS Regression Results
## ==============================================================================
## Dep. Variable:                      y   R-squared:                       0.953
## Model:                            OLS   Adj. R-squared:                  0.949
## Method:                 Least Squares   F-statistic:                     249.7
## Date:                Sun, 07 Feb 2021   Prob (F-statistic):          1.45e-113
## Time:                        23:44:00   Log-Likelihood:                -1019.8
## No. Observations:                 200   AIC:                             2072.
## Df Residuals:                     184   BIC:                             2124.
## Df Model:                          15
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          t      P>|t|      [0.025      0.975]
## ------------------------------------------------------------------------------
## const         -3.0556      3.004     -1.017      0.310      -8.983       2.872
## x1            81.4920      3.288     24.784      0.000      75.005      87.979
## x2             9.7589      2.993      3.261      0.001       3.854      15.663
## x3            12.7118      2.902      4.380      0.000       6.986      18.437
## x4             8.0572      3.216      2.505      0.013       1.712      14.403
## x5            80.7312      2.957     27.303      0.000      74.898      86.565
## x6             2.9540      2.854      1.035      0.302      -2.676       8.584
## x7            77.8472      2.992     26.015      0.000      71.943      83.751
## x8           103.5912      2.986     34.692      0.000      97.700     109.482
## x9            20.4890      2.980      6.876      0.000      14.610      26.368
## x10           68.4641      2.763     24.782      0.000      63.014      73.915
## x11            1.1335      3.101      0.366      0.715      -4.985       7.252
## x12            4.6891      2.981      1.573      0.117      -1.192      10.571
## x13            0.8736      3.212      0.272      0.786      -5.464       7.212
## x14            1.2384      3.300      0.375      0.708      -5.273       7.750
## x15            3.0218      3.234      0.934      0.351      -3.360       9.403
## ==============================================================================
## Omnibus:                        6.353   Durbin-Watson:                   2.064
## Prob(Omnibus):                  0.042   Jarque-Bera (JB):                6.564
## Skew:                          -0.313   Prob(JB):                       0.0375
## Kurtosis:                       3.630   Cond. No.                         1.62
## ==============================================================================
##
## Warnings:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Quitamos la variable x13.

Step 2

```
# eliminar var 13 ----------------------------------------------
X2 = X1.drop("x13", axis=1)
# comprobar df-------------------------------------------------
X2.head()
# ajustar el modelo--------------------------------------------
```

```
##    const        x1        x2        x3  ...       x11       x12       x14
x15
## 0    1.0 -0.152005 -0.703594  0.422149  ... -0.289348  2.000984  0.013655  0.976
570
## 1    1.0 -1.853600  1.099557 -0.969300  ...  0.214637 -0.197315  0.156317  0.079
980
## 2    1.0 -0.293498 -0.439834  0.595284  ...  0.028472 -0.877035  1.055299  0.802
885
## 3    1.0 -1.685395  0.632809 -2.379930  ...  0.415923 -3.007046  0.596482  0.549
016
## 4    1.0  0.261581 -0.349490 -0.391385  ...  0.785493 -1.354455 -1.058892  0.102
902
##
## [5 rows x 15 columns]

est_X2 = sm.OLS(y, X2)
# ver ajuste----------------------------------------------------------
est2 = est_X2.fit()
print(est2.summary())

##                             OLS Regression Results
## ==============================================================================
## Dep. Variable:                      y   R-squared:                       0.953
## Model:                            OLS   Adj. R-squared:                  0.950
## Method:                 Least Squares   F-statistic:                     268.8
## Date:                Sun, 07 Feb 2021   Prob (F-statistic):          8.99e-115
## Time:                        23:44:00   Log-Likelihood:                -1019.8
## No. Observations:                 200   AIC:                             2070.
## Df Residuals:                     185   BIC:                             2119.
## Df Model:                          14
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          t      P>|t|      [0.025      0.975]
## ------------------------------------------------------------------------------
## const         -3.0366      2.996     -1.014      0.312      -8.947       2.874
## x1            81.5178      3.278     24.865      0.000      75.050      87.986
## x2             9.7750      2.985      3.275      0.001       3.887      15.663
## x3            12.6706      2.891      4.383      0.000       6.968      18.374
## x4             8.1200      3.200      2.538      0.012       1.807      14.433
## x5            80.7016      2.947     27.380      0.000      74.887      86.516
## x6             2.9699      2.846      1.044      0.298      -2.645       8.584
## x7            77.9024      2.978     26.159      0.000      72.027      83.778
## x8           103.5102      2.964     34.926      0.000      97.663     109.357
## x9            20.5559      2.962      6.940      0.000      14.712      26.400
## x10           68.3944      2.744     24.926      0.000      62.981      73.808
## x11            1.0815      3.088      0.350      0.727      -5.010       7.173
## x12            4.6544      2.971      1.567      0.119      -1.207      10.515
## x14            1.2360      3.292      0.375      0.708      -5.259       7.731
## x15            3.0095      3.226      0.933      0.352      -3.355       9.374
## ==============================================================================
## Omnibus:                        6.897   Durbin-Watson:                   2.065
## Prob(Omnibus):                  0.032   Jarque-Bera (JB):                7.270
## Skew:                          -0.327   Prob(JB):                       0.0264
## Kurtosis:                       3.667   Cond. No.                         1.60
## ==============================================================================
##
## Warnings:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Quitamos la variable x11.

Step 3

```
# eliminar var 14 ---------------------------------------------
X3 = X2.drop("x11", axis=1)
# comprobar df------------------------------------------------
X3.head()
# ajustar el modelo-------------------------------------------

##    const        x1        x2        x3  ...       x10       x12       x14
x15
## 0    1.0 -0.152005 -0.703594  0.422149  ...  1.010352  2.000984  0.013655  0.976
570
## 1    1.0 -1.853600  1.099557 -0.969300  ...  0.062979 -0.197315  0.156317  0.079
980
## 2    1.0 -0.293498 -0.439834  0.595284  ... -1.149425 -0.877035  1.055299  0.802
885
## 3    1.0 -1.685395  0.632809 -2.379930  ...  0.122320 -3.007046  0.596482  0.549
016
## 4    1.0  0.261581 -0.349490 -0.391385  ...  0.419085 -1.354455 -1.058892  0.102
902
##
## [5 rows x 14 columns]

est_X3 = sm.OLS(y, X3)
# ver ajuste--------------------------------------------------
est3 = est_X3.fit()
print(est3.summary())

##                             OLS Regression Results
## ==============================================================================
## Dep. Variable:                      y   R-squared:                       0.953
## Model:                            OLS   Adj. R-squared:                  0.950
## Method:                 Least Squares   F-statistic:                     290.9
## Date:                Sun, 07 Feb 2021   Prob (F-statistic):          5.49e-116
## Time:                        23:44:00   Log-Likelihood:                -1019.9
## No. Observations:                 200   AIC:                             2068.
## Df Residuals:                     186   BIC:                             2114.
## Df Model:                          13
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          t      P>|t|      [0.025      0.975]
## ------------------------------------------------------------------------------
## const         -3.0434      2.989     -1.018      0.310      -8.940       2.853
## x1            81.4356      3.262     24.962      0.000      75.000      87.872
## x2             9.6642      2.961      3.264      0.001       3.823      15.505
## x3            12.7711      2.870      4.450      0.000       7.110      18.432
## x4             8.2066      3.183      2.578      0.011       1.928      14.486
## x5            80.6634      2.938     27.451      0.000      74.866      86.460
## x6             3.0287      2.834      1.069      0.287      -2.563       8.620
## x7            77.7993      2.957     26.315      0.000      71.967      83.632
## x8           103.4904      2.956     35.008      0.000      97.658     109.322
## x9            20.5330      2.954      6.950      0.000      14.705      26.361
## x10           68.3972      2.737     24.986      0.000      62.997      73.798
## x12            4.6090      2.961      1.557      0.121      -1.233      10.451
## x14            1.3387      3.271      0.409      0.683      -5.114       7.792
## x15            2.9127      3.207      0.908      0.365      -3.413       9.239
## ==============================================================================
## Omnibus:                        6.371   Durbin-Watson:                   2.065
## Prob(Omnibus):                  0.041   Jarque-Bera (JB):                6.566
## Skew:                          -0.315   Prob(JB):                       0.0375
## Kurtosis:                       3.625   Cond. No.                         1.60
```

```
## ==============================================================================
##
## Warnings:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Quitamos la variable x14.

Step 4

```
# eliminar var 13 ---------------------------------------------
X4 = X3.drop("x14", axis=1)
# comprobar df------------------------------------------------
X4.head()
# ajustar el modelo-------------------------------------------

##    const        x1        x2        x3  ...        x9       x10       x12
x15
## 0    1.0 -0.152005 -0.703594  0.422149  ...  0.812167  1.010352  2.000984  0.976
570
## 1    1.0 -1.853600  1.099557 -0.969300  ... -1.204713  0.062979 -0.197315  0.079
980
## 2    1.0 -0.293498 -0.439834  0.595284  ... -1.460141 -1.149425 -0.877035  0.802
885
## 3    1.0 -1.685395  0.632809 -2.379930  ... -0.959316  0.122320 -3.007046  0.549
016
## 4    1.0  0.261581 -0.349490 -0.391385  ...  1.808751  0.419085 -1.354455  0.102
902
##
## [5 rows x 13 columns]

est_X4 = sm.OLS(y, X4)
# ver ajuste--------------------------------------------------
est4 = est_X4.fit()
print(est4.summary())

##                             OLS Regression Results
## ==============================================================================
## Dep. Variable:                      y   R-squared:                       0.953
## Model:                            OLS   Adj. R-squared:                  0.950
## Method:                 Least Squares   F-statistic:                     316.5
## Date:                Sun, 07 Feb 2021   Prob (F-statistic):          3.28e-117
## Time:                        23:44:00   Log-Likelihood:                -1020.0
## No. Observations:                 200   AIC:                             2066.
## Df Residuals:                     187   BIC:                             2109.
## Df Model:                          12
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          t      P>|t|      [0.025      0.975]
## ------------------------------------------------------------------------------
## const         -3.0646      2.982     -1.028      0.305      -8.947       2.817
## x1            81.2639      3.228     25.174      0.000      74.896      87.632
## x2             9.8174      2.931      3.350      0.001       4.036      15.599
## x3            12.7225      2.861      4.447      0.000       7.079      18.366
## x4             8.1725      3.175      2.574      0.011       1.910      14.435
## x5            80.5778      2.924     27.553      0.000      74.809      86.347
## x6             2.9036      2.811      1.033      0.303      -2.643       8.450
## x7            77.8636      2.946     26.433      0.000      72.052      83.675
## x8           103.4029      2.942     35.149      0.000      97.599     109.206
## x9            20.4475      2.940      6.954      0.000      14.647      26.248
## x10           68.3663      2.730     25.040      0.000      62.980      73.752
```

```
## x12            4.5898      2.954      1.554      0.122      -1.238      10.417
## x15            2.7652      3.179      0.870      0.386      -3.506       9.037
## ==============================================================================
## Omnibus:                        6.353   Durbin-Watson:                   2.058
## Prob(Omnibus):                  0.042   Jarque-Bera (JB):                6.554
## Skew:                          -0.314   Prob(JB):                       0.0377
## Kurtosis:                       3.627   Cond. No.                         1.55
## ==============================================================================
##
## Warnings:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Quitamos la variable x15.

Step 5

```
# eliminar var 13 ---------------------------------------------
X5 = X4.drop("x15", axis=1)
# comprobar df-------------------------------------------------
X5.head()
# ajustar el modelo-------------------------------------------

##    const        x1        x2        x3  ...        x8        x9       x10
x12
## 0    1.0 -0.152005 -0.703594  0.422149  ...  0.371789  0.812167  1.010352  2.000
984
## 1    1.0 -1.853600  1.099557 -0.969300  ... -1.038767 -1.204713  0.062979 -0.197
315
## 2    1.0 -0.293498 -0.439834  0.595284  ... -0.251607 -1.460141 -1.149425 -0.877
035
## 3    1.0 -1.685395  0.632809 -2.379930  ... -0.421021 -0.959316  0.122320 -3.007
046
## 4    1.0  0.261581 -0.349490 -0.391385  ...  0.212894  1.808751  0.419085 -1.354
455
##
## [5 rows x 12 columns]

est_X5 = sm.OLS(y, X5)
# ver ajuste---------------------------------------------------
est5 = est_X5.fit()
print(est5.summary())

##                             OLS Regression Results
## ==============================================================================
## Dep. Variable:                      y   R-squared:                       0.953
## Model:                            OLS   Adj. R-squared:                  0.950
## Method:                 Least Squares   F-statistic:                     345.7
## Date:                Sun, 07 Feb 2021   Prob (F-statistic):          2.51e-118
## Time:                        23:44:00   Log-Likelihood:                -1020.4
## No. Observations:                 200   AIC:                             2065.
## Df Residuals:                     188   BIC:                             2104.
## Df Model:                          11
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          t      P>|t|      [0.025      0.975]
## ------------------------------------------------------------------------------
## const         -3.3394      2.963     -1.127      0.261      -9.184       2.505
## x1            81.1974      3.225     25.177      0.000      74.835      87.559
## x2             9.6784      2.924      3.310      0.001       3.910      15.447
## x3            12.7339      2.859      4.454      0.000       7.094      18.374
```

```
## x4             7.9906      3.166      2.524      0.012       1.746      14.235
## x5            80.4024      2.916     27.576      0.000      74.651      86.154
## x6             3.0313      2.806      1.080      0.281      -2.504       8.566
## x7            77.8203      2.943     26.439      0.000      72.014      83.627
## x8           103.3584      2.939     35.162      0.000      97.560     109.157
## x9            20.5961      2.933      7.021      0.000      14.809      26.383
## x10           68.3421      2.728     25.049      0.000      62.960      73.724
## x12            4.6381      2.952      1.571      0.118      -1.184      10.461
## ==============================================================================
## Omnibus:                        5.761   Durbin-Watson:                   2.036
## Prob(Omnibus):                  0.056   Jarque-Bera (JB):                5.774
## Skew:                          -0.301   Prob(JB):                       0.0558
## Kurtosis:                       3.575   Cond. No.                         1.53
## ==============================================================================
##
## Warnings:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Quitamos la variable x6.

Step 6

```
# eliminar var 13 ---------------------------------------------
X6 = X5.drop("x6", axis=1)
# comprobar df-------------------------------------------------
X6.head()
# ajustar el modelo-------------------------------------------

##    const        x1        x2        x3  ...        x8        x9       x10
x12
## 0    1.0 -0.152005 -0.703594  0.422149  ...  0.371789  0.812167  1.010352  2.000
984
## 1    1.0 -1.853600  1.099557 -0.969300  ... -1.038767 -1.204713  0.062979 -0.197
315
## 2    1.0 -0.293498 -0.439834  0.595284  ... -0.251607 -1.460141 -1.149425 -0.877
035
## 3    1.0 -1.685395  0.632809 -2.379930  ... -0.421021 -0.959316  0.122320 -3.007
046
## 4    1.0  0.261581 -0.349490 -0.391385  ...  0.212894  1.808751  0.419085 -1.354
455
##
## [5 rows x 11 columns]

est_X6 = sm.OLS(y, X6)
# ver ajuste---------------------------------------------------
est6 = est_X6.fit()
print(est6.summary())

##                             OLS Regression Results
## ==============================================================================
## Dep. Variable:                      y   R-squared:                       0.953
## Model:                            OLS   Adj. R-squared:                  0.950
## Method:                 Least Squares   F-statistic:                     379.8
## Date:                Sun, 07 Feb 2021   Prob (F-statistic):          2.24e-119
## Time:                        23:44:00   Log-Likelihood:                -1021.0
## No. Observations:                 200   AIC:                             2064.
## Df Residuals:                     189   BIC:                             2100.
## Df Model:                          10
## Covariance Type:            nonrobust
## ==============================================================================
```

```
##                  coef    std err          t      P>|t|      [0.025      0.975]
## ------------------------------------------------------------------------------
## const         -3.6189      2.953     -1.226      0.222      -9.444       2.206
## x1            81.0663      3.224     25.143      0.000      74.706      87.426
## x2             9.3344      2.908      3.210      0.002       3.598      15.071
## x3            12.6837      2.860      4.435      0.000       7.042      18.325
## x4             8.0506      3.167      2.542      0.012       1.804      14.297
## x5            80.7182      2.902     27.813      0.000      74.993      86.443
## x7            77.5085      2.931     26.449      0.000      71.728      83.289
## x8           103.0888      2.930     35.182      0.000      97.309     108.869
## x9            20.5709      2.935      7.010      0.000      14.782      26.360
## x10           68.1791      2.725     25.016      0.000      62.803      73.555
## x12            5.1480      2.915      1.766      0.079      -0.602      10.898
## ==============================================================================
## Omnibus:                        5.130   Durbin-Watson:                   2.031
## Prob(Omnibus):                  0.077   Jarque-Bera (JB):                4.877
## Skew:                          -0.301   Prob(JB):                       0.0873
## Kurtosis:                       3.472   Cond. No.                         1.47
## ==============================================================================
##
## Warnings:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Quitamos la variable x12.

Step 7

```
# eliminar var 13 ---------------------------------------------
X7 = X6.drop("x12", axis=1)
# comprobar df-------------------------------------------------
X7.head()
# ajustar el modelo-------------------------------------------

##    const        x1        x2        x3  ...        x7        x8        x9
x10
## 0    1.0 -0.152005 -0.703594  0.422149  ... -1.453966  0.371789  0.812167  1.010
352
## 1    1.0 -1.853600  1.099557 -0.969300  ... -1.490572 -1.038767 -1.204713  0.062
979
## 2    1.0 -0.293498 -0.439834  0.595284  ... -0.658011 -0.251607 -1.460141 -1.149
425
## 3    1.0 -1.685395  0.632809 -2.379930  ...  0.748315 -0.421021 -0.959316  0.122
320
## 4    1.0  0.261581 -0.349490 -0.391385  ... -0.302579  0.212894  1.808751  0.419
085
##
## [5 rows x 10 columns]

est_X7 = sm.OLS(y, X7)
# ver ajuste---------------------------------------------------
est7 = est_X7.fit()
print(est7.summary())

##                             OLS Regression Results
## ==============================================================================
## Dep. Variable:                      y   R-squared:                       0.952
## Model:                            OLS   Adj. R-squared:                  0.950
## Method:                 Least Squares   F-statistic:                     417.0
## Date:                Sun, 07 Feb 2021   Prob (F-statistic):          4.99e-120
## Time:                        23:44:01   Log-Likelihood:                -1022.6
```

```
## No. Observations:                  200   AIC:                              2065.
## Df Residuals:                      190   BIC:                              2098.
## Df Model:                            9
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          t      P>|t|      [0.025      0.975]
## ------------------------------------------------------------------------------
## const         -3.6257      2.969     -1.221      0.224      -9.483       2.231
## x1            80.9884      3.242     24.982      0.000      74.594      87.383
## x2             9.0338      2.919      3.094      0.002       3.275      14.792
## x3            13.0184      2.869      4.537      0.000       7.358      18.678
## x4             7.1836      3.146      2.284      0.023       0.979      13.388
## x5            80.4715      2.915     27.606      0.000      74.722      86.221
## x7            77.6352      2.946     26.353      0.000      71.824      83.446
## x8           103.0937      2.946     34.989      0.000      97.282     108.906
## x9            20.5702      2.951      6.971      0.000      14.749      26.391
## x10           68.4910      2.735     25.045      0.000      63.097      73.885
## ==============================================================================
## Omnibus:                        6.544   Durbin-Watson:                   1.962
## Prob(Omnibus):                  0.038   Jarque-Bera (JB):                6.470
## Skew:                          -0.349   Prob(JB):                       0.0394
## Kurtosis:                       3.539   Cond. No.                         1.46
## ==============================================================================
##
## Warnings:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Ya sólo quedan variables predictoras significativas. Se ha obtenido un $R^2 = 0.952$ y un $R^2_{ajustado} = 0.950$. Además, se puede comprobar que se obtienen los mismos coeficientes con la librería *scikit-learn*.

```
# importar clase-----------------------------------------------
from sklearn.linear_model import LinearRegression
# quitar intercepto (scikit lo pone automaticamente)-----------
Xlm = X7.drop("const", axis=1)
# ajustar el modelo--------------------------------------------
lm7 = LinearRegression()
lm7.fit(Xlm, y)
# obtener coeficientes del modelo------------------------------
# intercepto

## LinearRegression()

print(lm7.intercept_)
# coeficientes de regresion

## -3.625664050917978

print(lm7.coef_)

## [ 80.98844016   9.03377116  13.01839235   7.18355283  80.47147282
##   77.63522062 103.09370968  20.57019231  68.49098464]
```

Red elástica

Se ajusta un modelo mediante la red elástica. Para escoger el mejor modelo se empleará una rejilla de valores entre 0.1 y 1 (cada 0.1) para el hiperparámetro r. El valor de α se obtendrá por validación cruzada (Usando $k = 10$ particiones).

El modelo de tipo red elástica se ajusta realizando validación cruzada utilizando la clase *ElasticNetCV* del paquete *sklearn.linear_model*. En los argumentos de la clase definiremos las particiones de la CV con el argumento *cv*, y el hiperparámetro *r* se fija con el argumento *l1_ratio: (cv = 10 y l1_ratio = valores_r)*.

```
# semilla para que los resultados sean los mismos----------------
np.random.seed(semilla)
# importar clase-------------------------------------------------
from sklearn.linear_model import ElasticNetCV
# definir valores de r
valores_r = np.linspace(0.1, 1, 10)
# ver valores de r
print(valores_r)
# ajustar el modelo----------------------------------------------

## [0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1. ]

e_net = ElasticNetCV(cv = 10, l1_ratio = valores_r)
e_net.fit(X, y)
# obtener coeficientes del modelo--------------------------------

## ElasticNetCV(cv=10,
##              l1_ratio=array([0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]))

print(e_net.l1_ratio_)
# se puede comprobar que el error de test es mayor con r = 0.1---
# error = e_net_c.mse_path_
# np.mean(error, axis = 2)
# intercepto

## 1.0

print(e_net.intercept_)
# coeficientes de regresion

## -3.5811120064353688

print(e_net.coef_)

## [ 79.42647644   7.89462035  11.39074347   6.64225337  79.46721521
##    1.72585848  76.60688605 101.47639739  19.14905415  67.20109574
##    0.           3.47451458   0.           0.           1.17895844]
```

Se observa que se ha escogido el valor de 1, por lo que es el valor que minimiza el error de test por CV (el cual se puede analizar en el atributo *.mse_path_*). Además, la red elástica (realmente regresión LASSO al escogerse $r = 1$) encoge los valores de x11, x13 y x14 a 0.

Asignatura	Datos del alumno	Fecha
Técnicas Multivariantes	Apellidos:	
	Nombre:	

Validación

Se va a comprobar mediante la muestra de validación cuál de los 3 métodos ajusta mejor en términos del error cuadrático medio.

```
# definir funcion para calcular error cuad medio muestra val-----
def calc_val_error(X_val, y_val, modelo):
    pred = modelo.predict(X_val)
    # error cuadratico medio
    mse = mean_squared_error(y_val, pred)
    # raiz del error cuadratico medio (por si acaso)
    rmse = np.sqrt(mse)
    # se devuelve el error cuadratico medio
    return mse

# regresion lineal multiple
calc_val_error(X_val, y_val, modelo = lm1)
# seleccion de variables

## 1217.5526921453497

calc_val_error(X_val[:, [0, 1, 2, 3, 4, 6, 7, 8, 9] ], y_val, modelo = lm7)
# red elastica

## 1231.593381438371

calc_val_error(X_val, y_val, modelo = e_net)


## 1181.2017832600473
```

Conclusiones

Con la red elástica se obtiene el menor error cuadrático medio en la muestra de validación.