| DATOS PERSONALES | FIRMA |
| --- | --- |
| |  |  | | --- | --- | | Nombre: Maria Cristina | DNI:06288784D | |  |
| Apellidos: Rodriguez de la Guia |

| ESTUDIO | ASIGNATURA | CONVOCATORIA |
| --- | --- | --- |
| MÁSTER UNIVERSITARIO EN INGENIERÍA MATEMÁTICA Y COMPUTACIÓN (PLAN 2016) | 4391020006.- TÉCNICAS MULTIVARIANTES | Ordinaria Número periodo 1823 |

| FECHA | MODELO | CIUDAD DEL EXAMEN |
| --- | --- | --- |
| 14-16/01/2022 | Modelo - A |  |

| Etiqueta identificativa |
| --- |
|  |

**INSTRUCCIONES GENERALES**

1. Ten disponible tu documentación oficial para identificarte, en el caso de que se te solicite.
2. Si tu examen consta de una parte tipo test, indica las respuestas en la plantilla según las características de este.
3. Debes contestar en el documento adjunto, respetando en todo momento el espaciado indicado para cada pregunta. Si este es en formato digital, los márgenes, el interlineado, fuente y tamaño de letra vienen dados por defecto y no deben modificarse. En cualquier caso, asegúrate de que la presentación es suficientemente clara y legible. Entrega toda la documentación relativa al examen, revisando con detenimiento que los archivos o documentos son los correctos. El envío de archivos erróneos o un envío incompleto supondrá una calificación de “no presentado”.
4. Durante el examen y en la corrección por parte del docente, se aplicará el Reglamento de Evaluación Académica de UNIR que regula las consecuencias derivadas de las posibles irregularidades y prácticas académicas incorrectas con relación al plagio y uso inadecuado de materiales y recursos.
5. No está permitido el uso de Internet ni ningún tipo de comunicación con otra persona.Durante todo el examen tu teléfono móvil debe estar en modo avión.
6. La parte principal de cada pregunta consiste en interpretar y comentar los resultados obtenidos. Si te limitas a hacer los cálculos no vas a poder superar el examen.
7. Es fundamental que las respuestas estén debidamente redactadas, de forma clara y precisa y sin faltas de ortografía.
8. Para hacer el examen puedes utilizar los apuntes del curso y los scripts que hayas preparado y Python para hacer los cálculos.

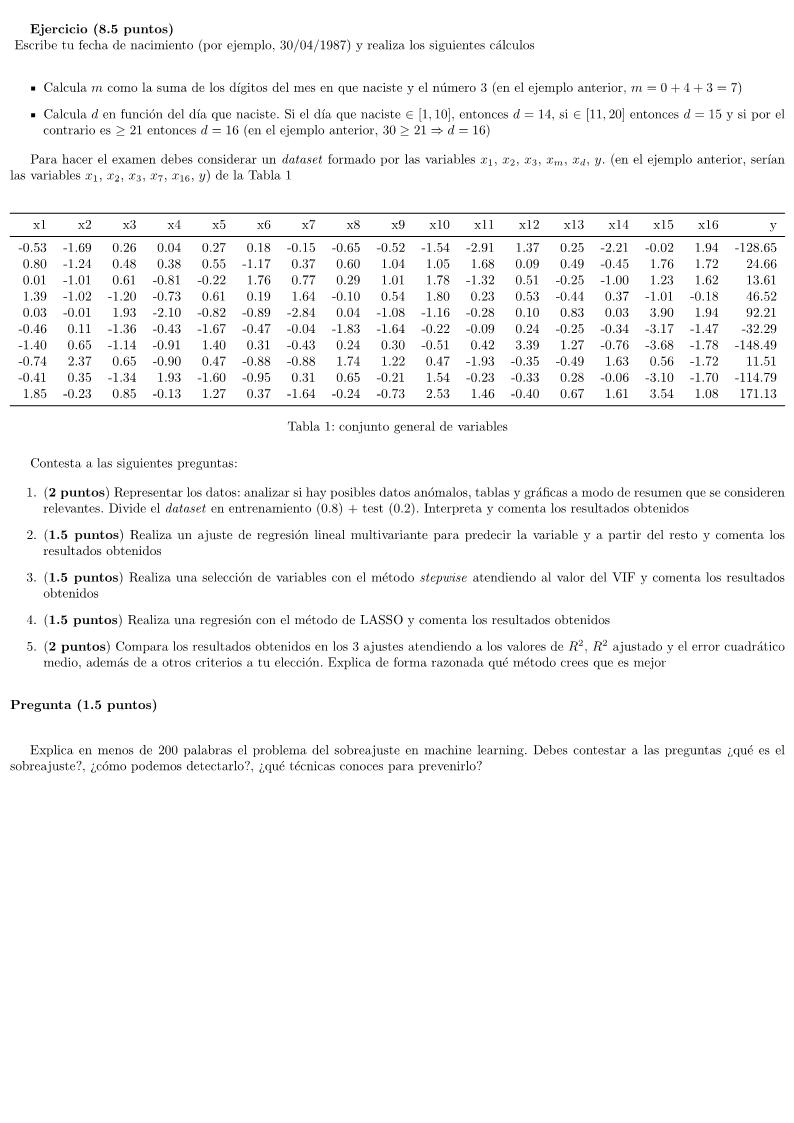
**Puntuación**

**Preguntas**

* Puntuación máxima 10.00 puntos

El examen constará de un ejercicio práctico (8,5 puntos) y una pregunta teórica (1,5 puntos). Los enunciados están en la página 14 y el espacio para responder el examen está entre las práginas 4 y 13.  
  
**1.** Pregunta

 (Responder en 10 caras)



EJERCICIO 2:

El problema de sobreajuste se obtiene cuando el metodo se ajusta demasiado para un set de datos pero si se cambia el data set se obteien errores muy grandes (varianza grande). Por ese motivo se trabaja con el conjunto de entramiento para obtener el metodo y con el conjunto test para validarlo y evitar sobreajuste

EJERCICIO 1

Apartado 1

# -\*- coding: utf-8 -\*-

"""

Created on Fri Jan 14 14:30:00 2022

EXAMEN METODOS MULTIVARIANTES

@author: Maria Cristina Rodriguez de la Guia

"""

# EJERCICIO 1

# apartado 1)----------------------------------------------------

fecha='22/11/1996'

m=1+1+3

d=16

# Definimos variables predictoras xi y variable respuesta y------

x1=[-0.53,0.8,0.01,1.39,0.03,-0.46,-1.40,-0.74,-0.41,1.85]

print(len(x1))

x2=[-1.69,-1.24,-1.01,-1.02,-0.01,0.11,0.65,2.37,0.35,-0.23]

print(len(x2))

x3=[0.26,0.48,0.61,-1.20,1.96,-1.36,-1.14,0.65,-1.34,0.85]

print(len(x3))

x5=[0.27,0.55,-0.22,0.61,-0.82,-1.67,1.40,0.47,-1.60,1.27]

print(len(x5))

x16=[1.94,1.72,1.62,-0.18,1.94,-1.47,-1.78,-1.72,-1.70,1.08]

print(len(x16))

y=[-128.65,24.66,13.61,46.52,92.21,-32.29,-148.49,11.51,-114.79,171.33]

print(len(y))

# metemos todo en un dataFrame llamado datos---------------------

d={"x1":x1,"x2":x2,"x3":x3,"x5":x5,"x16":x16,"y":y}

import pandas as pd

datos=pd.DataFrame(d)

# Comprobamos nombres, tipo y cantidad de las variables---------

datos.info()

# Comprobamos estadisticos de nuestro set------------------------

datos.describe()

# representamos los datos en histogramas-------------------------

import matplotlib.pyplot as plt

fig, axes = plt.subplots(nrows = 3, ncols = 2, figsize = (12,12))

# definir nombres ejes--------------------------------------------

# primera fila

axes[0,0].set\_xlabel("x1")

axes[0,0].set\_ylabel("n")

axes[0,1].set\_xlabel("x2")

axes[0,1].set\_ylabel("n")

# segunda fila

axes[1,0].set\_xlabel("x3")

axes[1,0].set\_ylabel("n")

axes[1,1].set\_xlabel("x5")

axes[1,1].set\_ylabel("n")

# tercera fila

axes[2,0].set\_xlabel("x16")

axes[2,0].set\_ylabel("n")

axes[2,1].set\_xlabel("y")

axes[2,1].set\_ylabel("n")

datos.hist(bins = 50, ax = axes); #bins indica el numero de barras del histograma

# partimos los datos --------------------------------------------

# Los definimos en dos conjuntos: entrenamiento y validadcion

import numpy as np

def particiones(dataset, test\_part):

test\_part\_size = int(len(dataset) \* test\_part)

mezclar\_indices = np.random.permutation(len(dataset)) # mezclarmos los indices

test\_indices = mezclar\_indices[:test\_part\_size]

train\_indices = mezclar\_indices[test\_part\_size:]

return dataset.iloc[train\_indices], dataset.iloc[test\_indices]

train\_set, test\_set = particiones(datos, 0.2)

# comprobar longitudes de los set de entrenamiento y de test-----

print(round(len(datos) \* 0.8, 1)) # Sale 8

print(len(train\_set)) # Sale 8

print(len(datos) \* 0.2) # Sale 2

print(len(test\_set)) # Sale 2

La figura con los 6 histogramas de las variables predictoras y la respuesta salen a continuación. A simple vista podemos observar en los histogramas de la figura 1 que no hay datos atípicos o outliers. Los valores de las variables predictoras oscilan entre [-2,2] aproximadamente y el valor de la variable respuesta oscila [-150,150]. Quizas se pueda destacar como outlier el valor 2.37 de la variable x2 ya que los demás estan en el intervalo [-1.69, 0.65]. Tambien se puede considar outlier el valor 1.96 de la variable x3 ya que los demás estan en el intervalo [-1.36, 0.61]. Hemos creado dos set, el de entrenaimiento “train” y el de validación “test”.

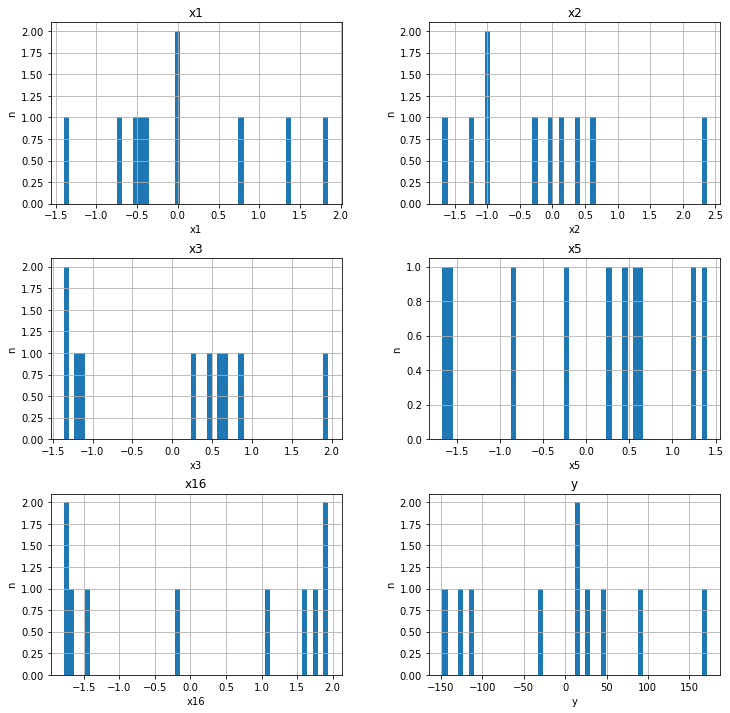


Figura1. Histograma con variables respuesta y predictoras

EJERCICIO 1

Apartado 2

# apartado 2)----------------------------------------------------

# Ajuste regresion lineal multiple-------------------------------

# No hay varaibles categoricas ni datos faltantes. Pero si hay que

# estandarizar todo.

# Definimos variables predictoras x

X\_train=train\_set.drop("y", axis=1)

y\_train=train\_set["y"].copy()

X\_test=test\_set.drop("y", axis=1)

y\_test=test\_set["y"].copy()

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import StandardScaler

num\_pipeline = Pipeline([

("std\_scaler", StandardScaler()), # estandarizamos los valores numericos

])

# Aplicamos pipeline con la estandarización-----------------------

X\_train\_tr = num\_pipeline.fit\_transform(X\_train)

# ajustar el modelo----------------------------------------------

from sklearn.linear\_model import LinearRegression

lm1 = LinearRegression()

lm1.fit(X\_train\_tr, y\_train) # .fit(x,y)

# obtener coeficientes del modelo--------------------------------

# Y = beta0 + beta1\*X1 + beta2\*X2 + beta3\*X3 + beta5\*X5 + beta16\*X16

print('coeficiente modelo:')

print(lm1.intercept\_) # beta0

# coeficientes de regresion

print('coeficientes regresion')

print(lm1.coef\_) # [beta1 beta2 ... beta16]

# miramos p.valores en liberia statsmodels----------------------

import statsmodels.api as sm

# ajustar el modelo----------------------------------------------

# OLS: ordinary least squares model -> Minimos cuadrados

est = sm.OLS(y\_train, sm.add\_constant(X\_train)) # .OLS(y, x)

# ver ajuste-----------------------------------------------------

est2 = est.fit()

print(est2.summary())

y\_pred = lm1.predict(X\_test)

from sklearn.metrics import mean\_squared\_error

print("Error cuadrático medio 1: ", str(mean\_squared\_error(y\_true = y\_test, y\_pred = y\_pred)))

Del apartado 2 hemos obtenido los siguienets outputs. Podemos comprobar que se tiene un coeficiente de correción bastante bueno R-squared = 0.967. El coeficiente de correlación ajustado es menor Adj. R-squared: 0.884, esto puede deberse a que hay colinealidad.

coeficiente modelo:

-7.093750000000011

coeficientes regresion

[ 90.92421751 813.32526345 -684.32631788 -2.49666013 1052.91278727]

OLS Regression Results

==============================================================================

Dep. Variable: y R-squared: 0.967

Model: OLS Adj. R-squared: 0.884

Method: Least Squares F-statistic: 11.72

Date: Fri, 14 Jan 2022 Prob (F-statistic): 0.0805

Time: 16:16:38 Log-Likelihood: -35.095

No. Observations: 8 AIC: 82.19

Df Residuals: 2 BIC: 82.67

Df Model: 5

Covariance Type: nonrobust

==============================================================================

coef std err t P>|t| [0.025 0.975]

------------------------------------------------------------------------------

const -7.3867 20.368 -0.363 0.752 -95.024 80.250

x1 89.2272 15.345 5.815 0.028 23.202 155.253

x2 690.8486 2539.441 0.272 0.811 -1.02e+04 1.16e+04

x3 -614.8939 2529.552 -0.243 0.831 -1.15e+04 1.03e+04

x5 -2.6169 24.418 -0.107 0.924 -107.680 102.446

x16 662.8257 2542.815 0.261 0.819 -1.03e+04 1.16e+04

==============================================================================

Omnibus: 2.571 Durbin-Watson: 2.348

Prob(Omnibus): 0.277 Jarque-Bera (JB): 0.891

Skew: -0.257 Prob(JB): 0.641

Kurtosis: 1.448 Cond. No. 644.

==============================================================================

EJERCICIO 1

Apartado 3

# apartado 3)----------------------------------------------------

# Stepwise

# En la selección hacia atrás se parte del modelo con todas las

# variables (xm) y van eliminando en cada paso la variable con la

# que se consigue mejorar más el ajuste del modelo.

# Sacamos los p valores de los resultados anteriores.

# Definimos funcion stepwise hacia atras:

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

def stepwise(datos, respuesta, var\_elim):

independientes\_nueva = datos.drop(var\_elim, axis=1)

lm = LinearRegression()

res = lm.fit(independientes\_nueva, respuesta)

independientes\_nueva = sm.add\_constant(independientes\_nueva)

modelo = sm.OLS(respuesta, independientes\_nueva)

resultados = modelo.fit()

print(resultados.summary())

VIF=pd.DataFrame()

# añadimos columna "variable" con el nombre de las columnas

VIF["variable"] = independientes\_nueva.columns

# aplicar el metodo y creamos nueva columna "VIF":

VIF["VIF"] = [variance\_inflation\_factor(independientes\_nueva.values, i)

for i in range(len(independientes\_nueva.columns))]

print(VIF)

# Colinealidad: relacion lineal fuerte entre variables predictoras

# VIF: variance\_inflation\_factor para comprobar la existencia de colinealidad.

# Analiza para cada variable su coef de determinacion R2 frente a un modelo ajustado

# por el resto de variables del modelo.

# VIF = 1/(1-R2). Cuando mayor sea R2 mayor será VIF. A partir de un VIF=10

# se afirma que hay colinealidad y con VIF =[4-5] se sospecha.

VIF=pd.DataFrame()

# añadimos columna "variable" con el nombre de las columnas

VIF["variable"] = X\_train.columns

# aplicar el metodo y creamos nueva columna "VIF":

VIF["VIF"] = [variance\_inflation\_factor(X\_train.values, i)

for i in range(len(X\_train.columns))]

print(VIF)

# Paso 1: Eliminamos variable predictora x16

stepwise(X\_train, y\_train, "x16")

X\_train\_new=X\_train.drop('x16', axis=1)

X\_test\_new=X\_test.drop('x16', axis=1)

# Calculamos error

lm2 = LinearRegression()

lm2.fit(X\_train\_new, y\_train) # .fit(x,y)

y\_pred = lm2.predict(X\_test\_new)

print("Error cuadrático medio de stepwise: ", str(mean\_squared\_error(y\_true = y\_test, y\_pred = y\_pred)))# Parecen que todas tiene aproximadamente el mismo VIF al eliminar x16

# Paso 2: Eliminamos variable predictora x1

stepwise(X\_train\_new, y\_train, "x1")

Los ouputs que se obtienen en el apartado 3. Hemos eliminado la variable x16 ya que su VIF era el mayor de todos lo que confirmaba la presencia de colinealidad del apartado anterior. Una vez eliminada la variable X16 se ha rehecho los calculos obteniendose un coeficiente R-squared =0.966 bastante alto y mejor que en el caso anterior. El coeficiente ajustado Adj. R-squared=0.920 ha aumentado y es similar a R-squared, lo que indica que hemos eliminado la colinealidad existente. Una vez eleiminada la variable x16, hemos vuelto a recalcular los VIF dandonos resultados más bajos.

riable VIF

0 x1 1.287348

1 x2 22668.057955

2 x3 20126.704909

3 x5 1.692837

4 x16 41624.358779

OLS Regression Results

==============================================================================

Dep. Variable: y R-squared: 0.966

Model: OLS Adj. R-squared: 0.920

Method: Least Squares F-statistic: 21.23

Date: Fri, 14 Jan 2022 Prob (F-statistic): 0.0154

Time: 16:19:53 Log-Likelihood: -35.228

No. Observations: 8 AIC: 80.46

Df Residuals: 3 BIC: 80.85

Df Model: 4

Covariance Type: nonrobust

==============================================================================

coef std err t P>|t| [0.025 0.975]

------------------------------------------------------------------------------

const -11.2213 11.696 -0.959 0.408 -48.443 26.000

x1 89.0707 12.731 6.997 0.006 48.556 129.585

x2 28.9110 10.699 2.702 0.074 -5.139 62.961

x3 44.4663 10.527 4.224 0.024 10.965 77.968

x5 2.4353 12.331 0.197 0.856 -36.807 41.678

==============================================================================

Omnibus: 1.128 Durbin-Watson: 2.387

Prob(Omnibus): 0.569 Jarque-Bera (JB): 0.656

Skew: -0.251 Prob(JB): 0.721

Kurtosis: 1.690 Cond. No. 1.79

==============================================================================

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

variable VIF

0 const 1.049010

1 x1 1.290568

2 x2 1.216732

3 x3 1.052548

4 x5 1.061295

EJERCICIO 1

Apartado 4

# apartado 4)----------------------------------------------------

from sklearn.linear\_model import Lasso

# ajustar el modelo

# Modelo de LASSO: minimiza el error cuadrático medio cumpliendo con la restrccion

# sum(|bi|)<s siendo s=parámetro. Se consigue eliminar variables predictoras al contrario que

# Ridge.

# Probamos con dos alphas: alphas=1-------------------------------

lasso\_reg = Lasso(alpha = 1)

lasso\_reg.fit(X\_train, y\_train) # fit(x,y)

# obtener coeficientes del modelo

# intercepto

print(lasso\_reg.intercept\_)

# coeficientes de regresion

print(lasso\_reg.coef\_)

# otro alpha alphas=10--------------------------------------------

lasso\_reg\_b = Lasso(alpha = 10)

lasso\_reg\_b.fit(X\_train, y\_train) # fit(x,y)

# obtener coeficientes del modelo

# intercepto

print(lasso\_reg\_b.intercept\_)

# coeficientes de regresion

print(lasso\_reg\_b.coef\_)

La salida del apartado 4 se muestra a continuación. Se ha realizado el ajuste LASSO para dos coeficientes alpha distintos. Con alpha=10 se consigue eliminar los coeficientes asociados a las variables x16 y x5 obteniendose un modelo más sencillo que en los casos anteriores.

-11.068347769675695

[87.88093374 27.83978391 43.82136599 1.56679712 0. ]

-10.83873875155824

[75.69425052 17.43943645 38.63127294 0. 0. ]

EJERCICIO 1

Apartado 5

Se concluye que el metodo de stepwise es mejor que el metodo lineal multiple normal. Con el metodo de LASSO se consigue eliminar variables obteniendose un metodo mucho más sencillo. Habria que estudiar la bodad de los metodos mediante el error cuadrático medio de y\_test frente al y\_pred obtenido con los 3 métodos. Al ser muy pocos valores de test 2, el ajuste lienal no tiene mucho sentido. Se deberia trabajar con más valores.