Sistemas de Información y Telemedicina. *

Marta Girones Sanguesa   Silvia Marset Gomis

Ignacio Amat Hernández   Sofía Gutiérrez Santamaría

November 30, 2019

Contents

List of Figures

Listings

1 Preámbulo

```python
import numpy as np
from   scipy import stats

#   names of variables
labels = ['age', 'leptin', 'bmi', 'adiponectin', 'glucose',
          'resistin', 'insulin', 'MCP1', 'HOMA']

#   loads data
data = np.loadtxt (open (r'../../data.csv', 'rb'), delimiter = ',')

#   rewrites data as all the rows of data w/out nan cells
data = data [~np.isnan (data).any (axis=1)]

#   separates parameters into matrix x
x    = np.array ([list (data [x][:-1])    for x in range (len (data))])

#   and class (1, 2) into vector y
y    = np.array ([int  (data [x][ -1])    for x in range (len (data))])

#   removes outliers
data_no = data [(np.abs (stats.zscore (data)) < 3).all (axis = 1)]
#     ↑ = No Outliers

x_no = np.array ([list (data_no [x][:-1]) for x in range (len (data_no))])
y_no = np.array ([int  (data_no [x][ -1]) for x in range (len (data_no))])
```

Listing 1: Importaciones iniciales y preparacion de datos.

2 Histogramas

En este apartado dibujamos los histogramas comparativos.

Fig. 1: Histogramas para datos con y sin anomalias.

```
1  import matplotlib as mpl
2  import matplotlib.pyplot as plt
3
4  #  load preprocessed data, x and y are raw, x_no and y_no contain no outliers
5  from preprocessing import x, y, x_no, y_no, labels
6
7  #  colours for the histograms
8  fc = [(), (0, 1, 0, 0.6), (0, 0, 1, 0.6)]
9  #       (R, G, B,  α )← transparency
10
11 fig, ax = plt.subplots (nrows = 5, ncols = 2, figsize = (13, 10))
12 ax = ax.flatten ()
13
14 #  draws each of the histograms, two for each variable
15 for i in range (0, 9):
16     for j in [1, 2]:
17         ax[i].hist (x [y == j, i], bins = 15, fc = fc [j], label = labels [i] + str ↘
            (j))
18         ax[i].legend (loc = 1, prop={'size': 15})
19
20 fig.suptitle ('con anómalos', fontsize = 30)
21 fig.savefig  ('../images/hist.pdf', bbox_inches = 'tight', pad_inches = 0)
```

Listing 2: Código generador de los histogramas con datos anómalos.

3 Kernel Density

Fig. 2: Kernel Density para datos con y sin anomalias.

```
1   import matplotlib as mpl

2   import matplotlib.pyplot as plt

3   import numpy as np

4   from scipy.stats import gaussian_kde

5

6   #  load preprocessed data, x and y are raw, x_no and y_no contain no outliers

7   from preprocessing import x, y, x_no, y_no, labels

8

9   #  colours

10  fc = ['', 'green', 'blue']

11

12  fig, ax = plt.subplots (nrows = 5, ncols = 2, figsize = (13, 10))

13  ax = ax.flatten ()

14

15  #  same loop in principle as before

16  for i in range (0, 9):

17      for j in [1, 2]:

18          kde = gaussian_kde (x_ := x [y == j, i])

19          xs = np.linspace(np.min (x_) - 10, np.max (x_), num=len (x_))

20          ax[i].plot (xs, kde(xs), c = fc[j], label = labels [i] + str (j))

21          ax[i].legend (loc = 1, prop={'size': 15})

22

23  fig.suptitle ('con anómalos', fontsize = 30)

24  fig.savefig  ('../images/kden.pdf', bbox_inches = 'tight', pad_inches = 0)
```

Listing 3: Código generador de los kernel density plots con datos anómalos.

4 Boxplot

con anómalos   sin anómalos

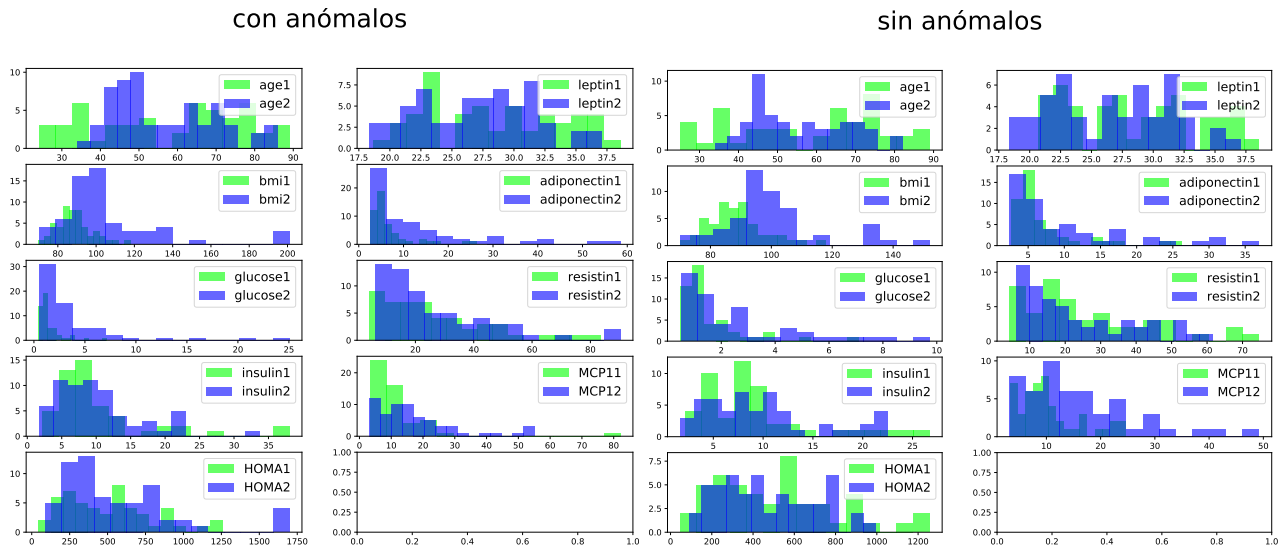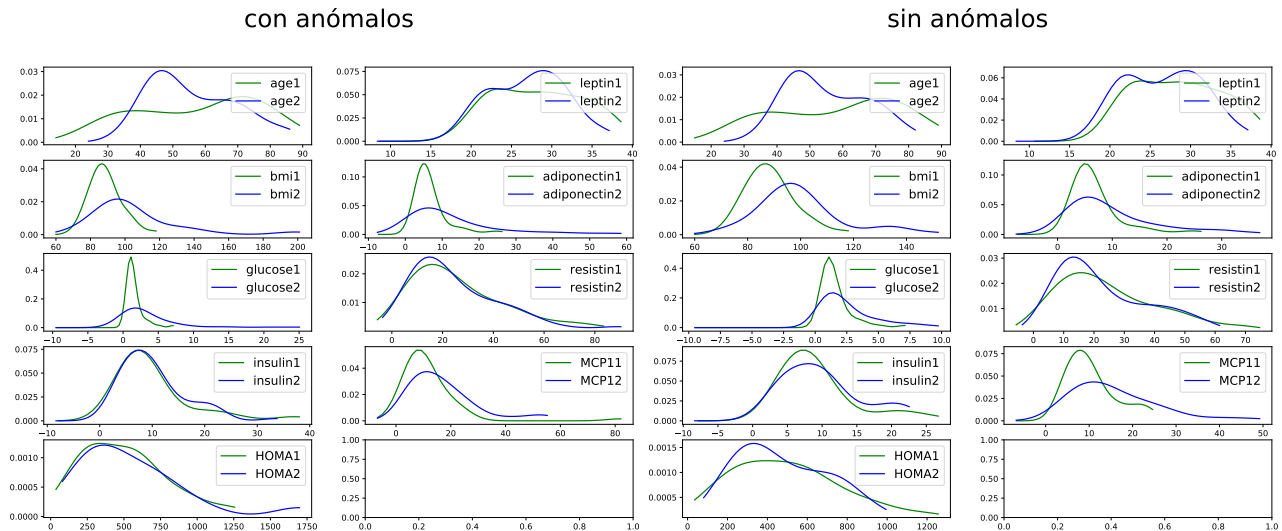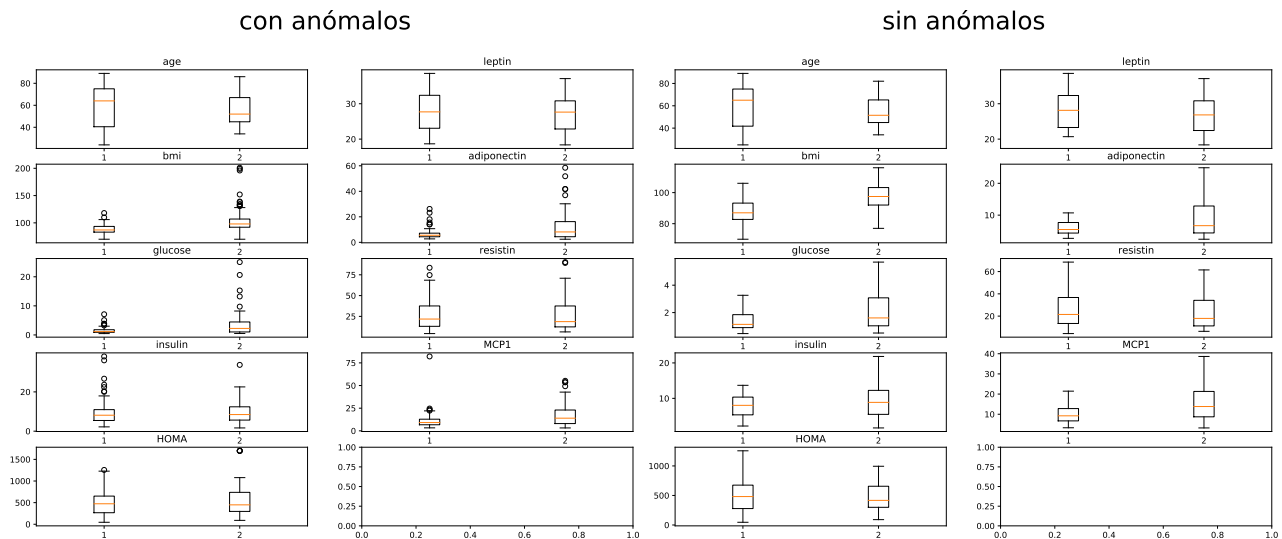Fig. 3: Boxplots para datos con y sin anomalias.

```python
import matplotlib as mpl
import matplotlib.pyplot as plt

# load preprocessed data, x and y are raw, x_no and y_no contain no outliers
from preprocessing import x, y, x_no, y_no, labels

fig, ax = plt.subplots (nrows = 5, ncols = 2, figsize = (13, 10))
ax = ax.flatten ()

for i in range (0, 9):
    ax[i].boxplot ([x [y == 1, i], x [y == 2, i]])
    ax[i].title.set_text (labels [i])

fig.suptitle ('con anómalos', fontsize = 30)
fig.savefig  ('../images/boxp.pdf', bbox_inches = 'tight', pad_inches = 0)
```

Listing 4: Código generador de los boxplots con datos anómalos.

5 QQplot

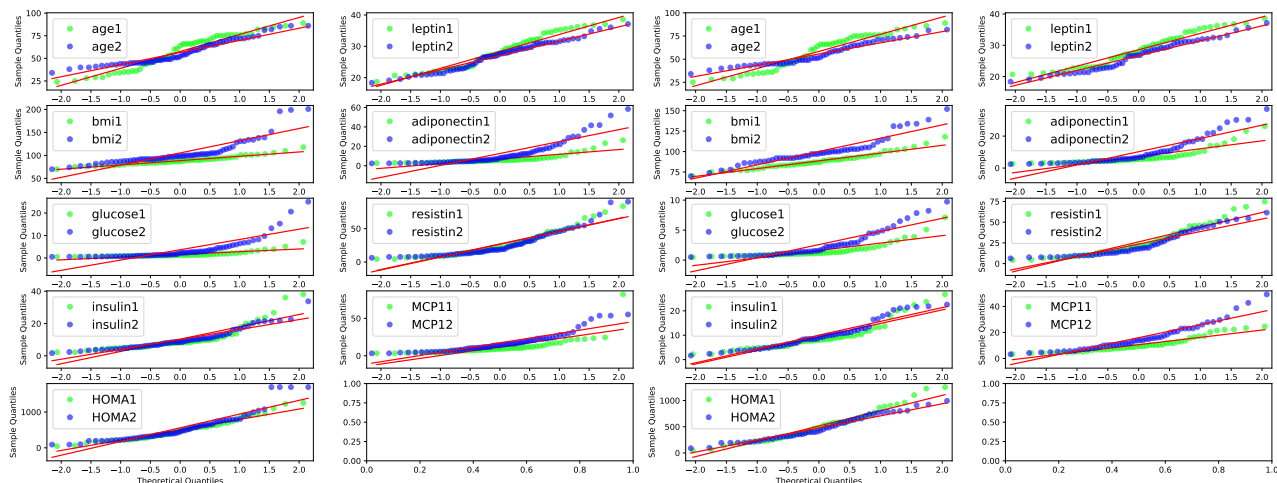con anómalos   sin anómalos

Fig. 4: QQplots para datos con y sin anomalias.

```python
import matplotlib as mpl
import matplotlib.pyplot as plt

#  load preprocessed data, x and y are raw, x_no and y_no contain no outliers
from preprocessing import x, y, x_no, y_no, labels
import statsmodels.api as sm


fc = [(), (0, 1, 0, 0.6), (0, 0, 1, 0.6)]


fig, ax = plt.subplots (nrows = 5, ncols = 2, figsize = (13, 10))
ax = ax.flatten ()


for i in range (0, 9):
    for j in [1, 2]:
        sm.qqplot (x [y == j, i], ax = ax[i], c = fc[j],
                line = 's', label = labels [i] + str (j))
        ax[i].legend (loc = 2, prop={'size': 15})


fig.suptitle ('con anómalos', fontsize = 30)
fig.savefig  ('../images/qqp.pdf', bbox_inches = 'tight', pad_inches = 0)
```

Listing 5: Código generador de los QQplots con datos anómalos.

6 Corrplot

Fig. 5: Corrplot para datos con anomalias.

```python
import pandas as pd
import seaborn as sns
dataframe = pd.DataFrame.from_records(x)
sns.pairplot (dataframe, kind = 'reg')
plt.suptitle ('con anómalos', fontsize = 30)
plt.savefig  ('../images/corrp.pdf', bbox_inches = 'tight', pad_inches = 0)
```

Listing 6: Código generador de los corrplots con datos anómalos.
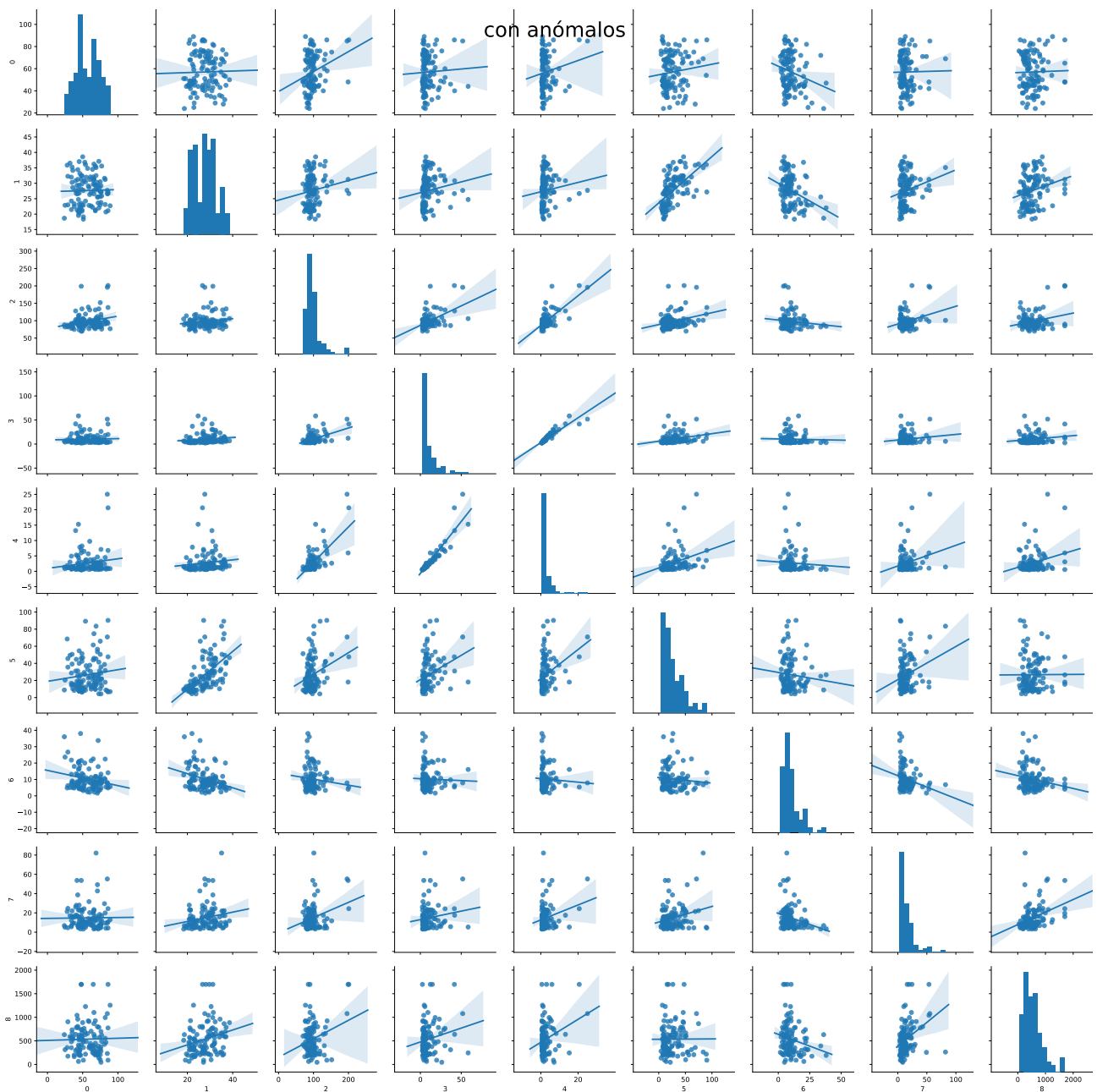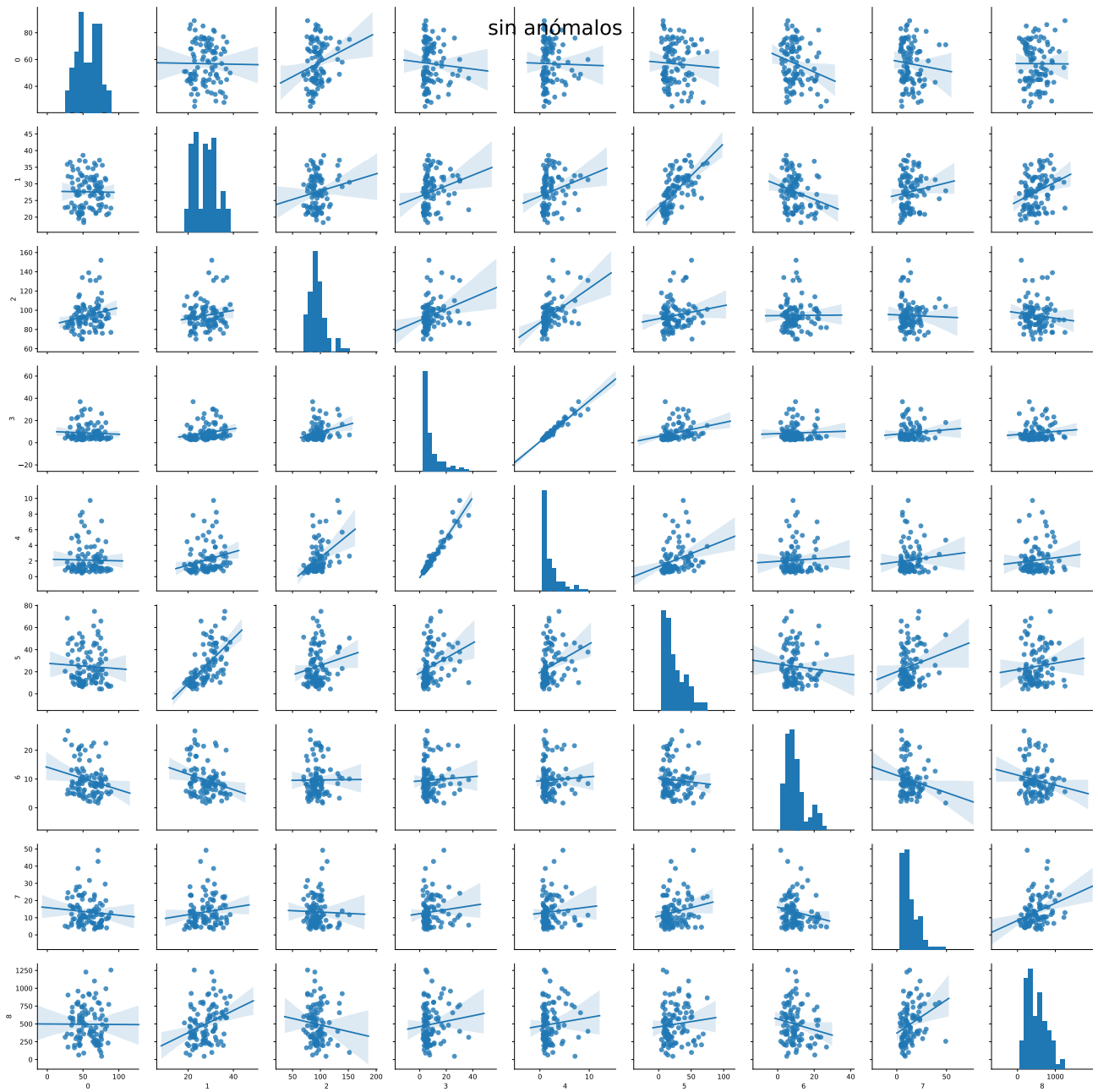
Fig. 6: Corrplot para datos sin anomalias.

7 Filter Methods

```python
#  Filter Methods
import sklearn.feature_selection as sk


Fscore, pval = sk.f_classif (x_no, y_no)
r1 = Fscore.argsort().argsort() #  fscore rank
print (r1+1)


import ReliefF as rl


r2 = rl.ReliefF (n_neighbors = 1) #  relieff rank
r2.fit(x_no, y_no)
r2 = r2.top_features
print (r2+1)


diferencias = abs (r1-r2)
media = np.mean (diferencias)
```

Listing 7: Aplicación métodos *filter* de selección características.

```
[4 5 9 6 7 3 1 8 2] -> fscore
[1 9 8 7 6 5 4 2 3] -> relieff
[3 4 1 1 1 2 3 6 1] -> diferencias
2.4444444444444446  -> media
```

Listing 8: Ranking de variables según los métodos filter.

8   Wrapper Methods

```python
from sklearn.neighbors import KNeighborsClassifier
from mlxtend.feature_selection import SequentialFeatureSelector


knn = KNeighborsClassifier (n_neighbors = 50)


sfs = SequentialFeatureSelector (knn ,
                 k_features = 4,
                 forward = True ,
                 scoring = 'accuracy',
                 cv = 10)


sfs.fit (x_no , y_no , custom_feature_names = labels)
print (sfs.k_score_)
print ('Sequential Forward  Selection', sfs.k_feature_names_ , end = '\n\n')


sfs.forward = False


sfs.fit (x_no , y_no , custom_feature_names = labels)
print (sfs.k_score_)
print ('Sequential Backward Selection', sfs.k_feature_names_ , end = '\n\n')
```

Listing 9: Aplicación métodos *wrapper* de selección características.

```
0.7054545454545454
Sequential Forward  Selection ('leptin', 'bmi', 'glucose', 'MCP1')

0.7094949494949495
Sequential Backward Selection ('leptin', 'bmi', 'glucose', 'insulin')
```

Listing 10: Resultados del filtrado mediante wrappers.

9   PCA

```
from sklearn.preprocessing import StandardScaler
x_no = StandardScaler ().fit_transform (x_no) #  typify
from sklearn.decomposition import PCA
pca = PCA (n_components = 9)
principalComponents = pca.fit_transform(x_no)
evr = pca.explained_variance_ratio_
```

Listing 11: *Principal Component Analysis*

```
[0.29146865 0.18490568 0.14125105 0.11727276 0.08486126 0.07999359
 0.06636991 0.03254865 0.00132847]
[0.29146865 0.47637432 0.61762537 0.73489813 0.81975939 0.89975298
0.96612289 0.99867153 1.         ]
```

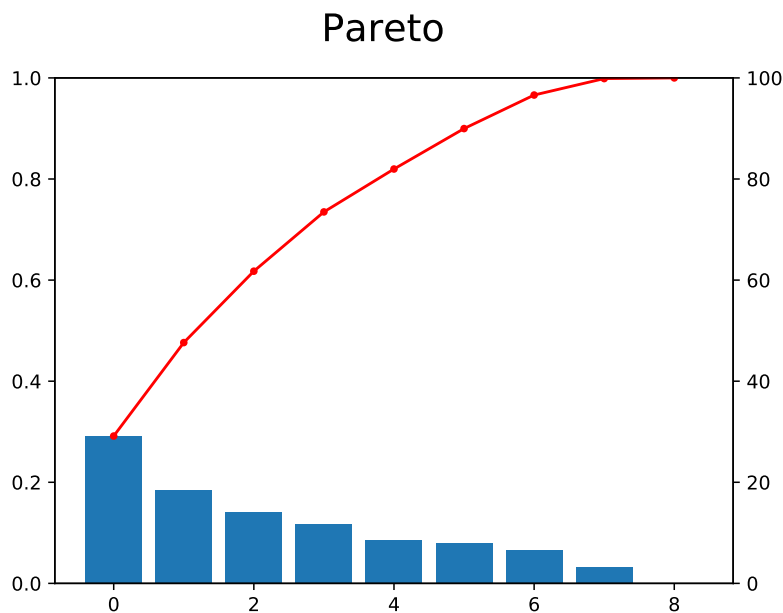Listing 12: Varianza explicada por componente y suma acumulada.

9.1   Pareto

Fig. 7: Diagrama de Pareto.

```
ax.bar (range (len (evr)), evr)
ax.set_ylim (top=1)
ax1 = ax.twinx ()
ax1.set_ylim (top=100)
ax1.plot (range (len (evr)), np.cumsum (evr)*100, marker = '.', color = 'red')
fig.suptitle ('Pareto', fontsize = 20)
fig.savefig  ('../images/pareto.pdf', bbox_inches = 'tight', pad_inches = 0)
```

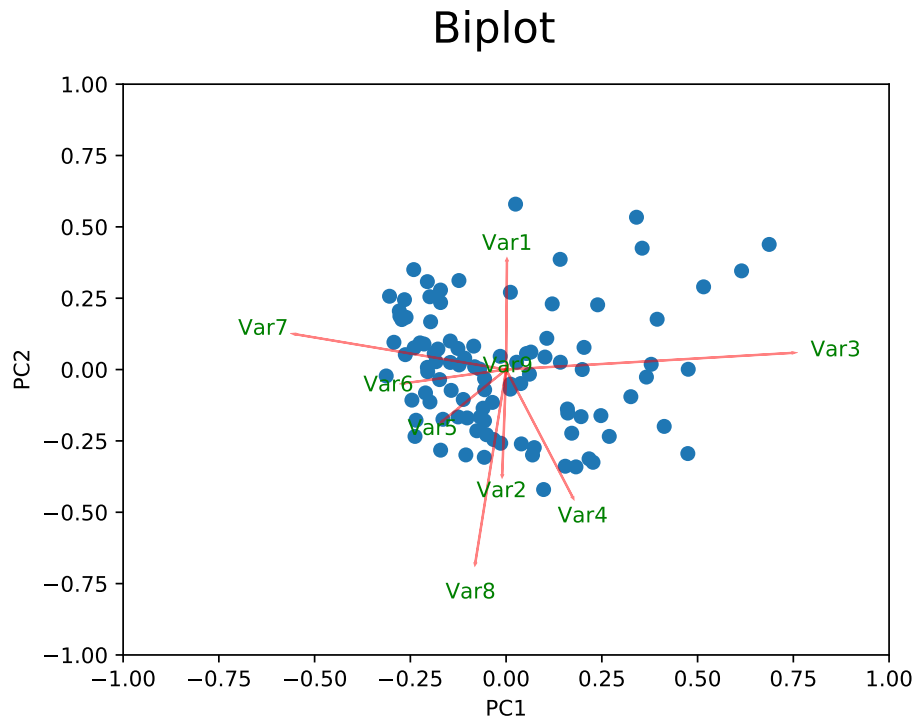Listing 13: Código generador del diagrama de Pareto

9.2 Biplot

Biplot

Fig. 8: Biplot.

```python
def biplot(score,coeff,pcax,pcay,labels=None):
    pca1=pcax-1; pca2=pcay-1
    xs = score[:,pca1]; ys = score[:,pca2]
    n=score.shape[1]
    scalex = 1.0/(xs.max()- xs.min()); scaley = 1.0/(ys.max()- ys.min())
    plt.scatter(xs*scalex,ys*scaley)
    for i in range(n):
        plt.arrow(0, 0, coeff[i,pca1], coeff[i,pca2],color='r',alpha=0.5)
        if labels is None:
            plt.text(coeff[i,pca1]* 1.15, coeff[i,pca2] * 1.15, "Var"+str(i+1), ↘
                color='g', ha='center', va='center')
        else:
            plt.text(coeff[i,pca1]* 1.15, coeff[i,pca2] * 1.15, labels[i], color='g'↘
                , ha='center', va='center')
    plt.xlim(-1,1); plt.ylim(-1,1)
    plt.xlabel("PC{}".format(pcax)); plt.ylabel("PC{}".format(pcay))
    return plt
bp = biplot (pca.fit_transform (x_no), pca.components_,1,2)
bp.suptitle ('Biplot', fontsize = 20)
bp.savefig  ('../images/biplotpca.pdf', bbox_inches = 'tight', pad_inches = 0)
```

Listing 14: Código generador del Biplot.