Practica 2 Exploración de datos con Pandas

February 9, 2021

PRACTICA 2: EXPLORACIÓN DE DATOS CON PANDAS

0.0.1 Ejercicio 1: Obtenga tres ejemplos de ficheros de datos en formato CSV, ARFF u otro cualquiera de Weka datasets o UCI

0.0.2 Ejercicio 2: Usando Pandas cargue los ficheros y evalúe qué información puede obtener del histograma de atributos

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.io import arff

iris = arff.loadarff('./Datos/iris.arff')
diabetes = arff.loadarff('./Datos/diabetes.arff')
df_iris = pd.DataFrame(iris[0])
df_diabetes = pd.DataFrame(diabetes[0])

# print(vote)
# print(iris)
# print(diabetes)

#Graficamos Histogramas
#Ejercico 2
df_iris.plot.hist(bins=12, alpha=0.4)
plt.title('Histograma IRIS DataSet')

df_diabetes.plot.hist(bins=12, alpha=0.4)
plt.title('Histograma DIABETES DataSet')
```

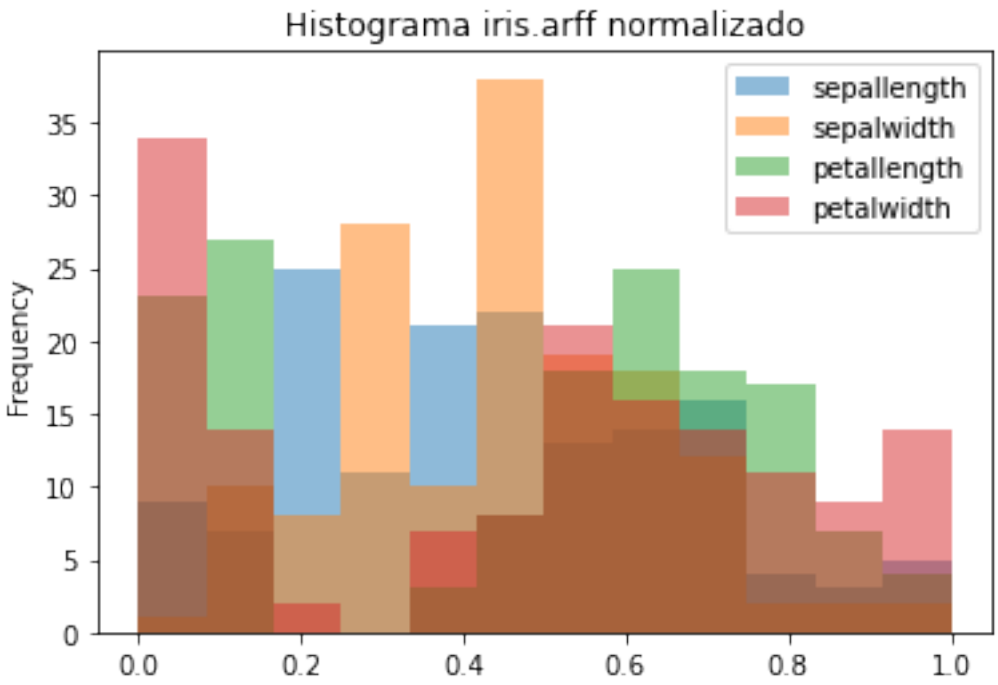
En un Histograma se agrupan los datos en 'clases o atributos', y se cuenta cuántas observaciones hay en cada una de ellas.

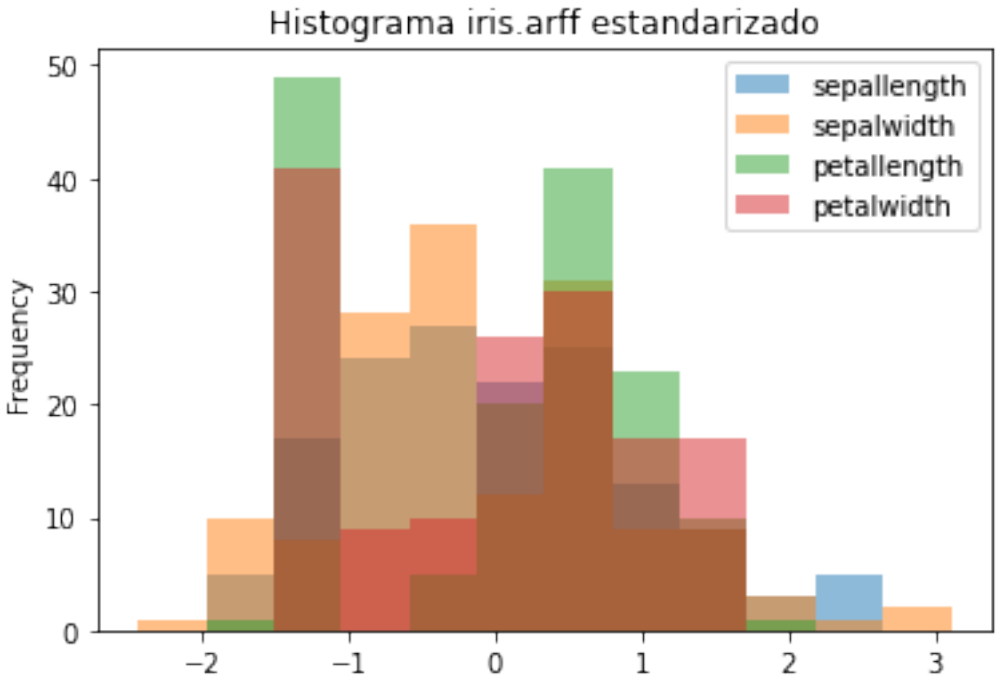
En estos dos histogramas podemos ver en la leyenda los diferentes atributos con los colores que los representan, y en la columnas comprobamos la frecuencia de aparición de estos mismos.

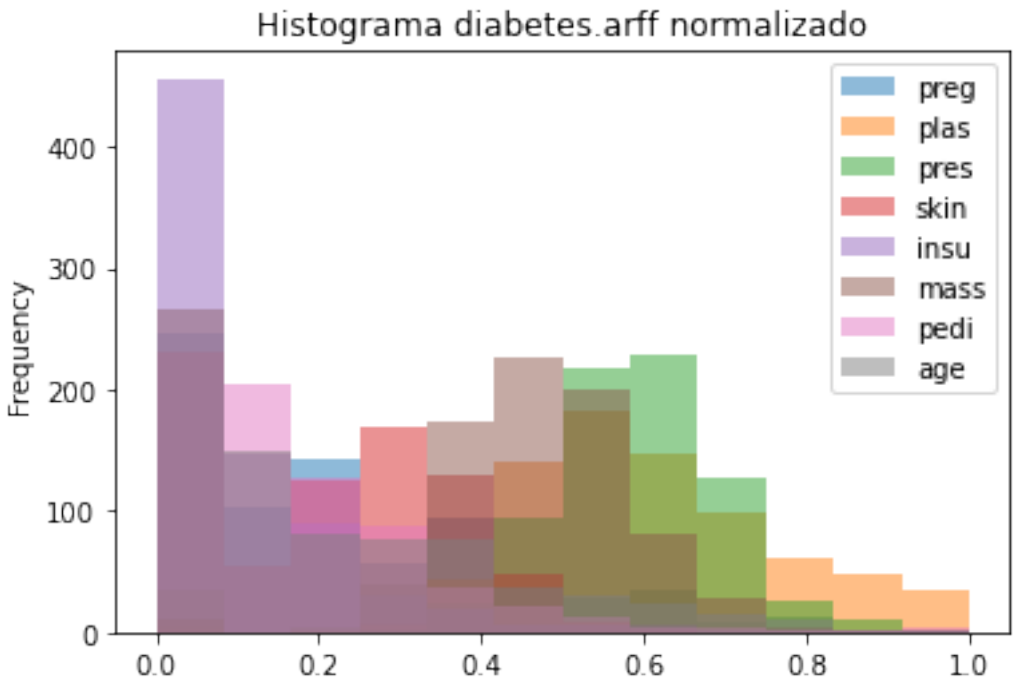
0.0.3 Ejercicio 3: Estudie el efecto de la normalización (reescalar en el intervalo [0,1] y la estandarización ($\mu = 0, \sigma = 1$) sobre el histograma

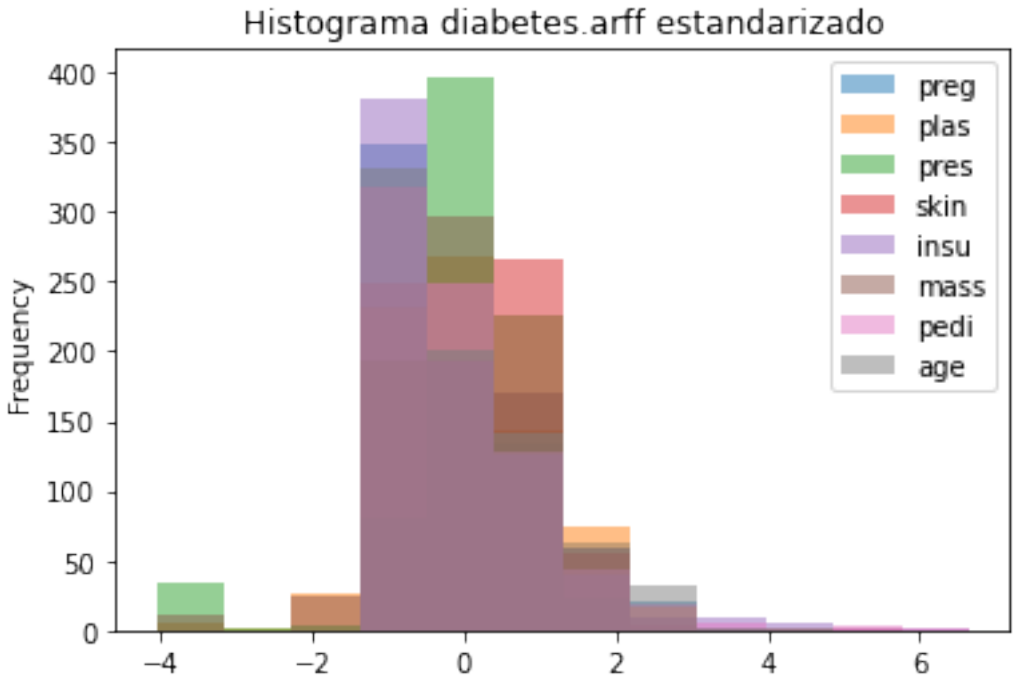
[71]:
```
#Ejercicio 3: NORMALIZACIÓN
def normalizar(dataframe):
    dataframe = dataframe.iloc[:, :-1]
    df_normalizado = (dataframe - dataframe.min()) / (dataframe.max() -␣
 ↪dataframe.min())
    return df_normalizado

def estandarizar(dataframe, , ):
    dataframe = dataframe.iloc[:, :-1]
    df_estandarizado = (dataframe -  ) / ( )
    return df_estandarizado

#Graficamos IRIS Normalizado
df_iris_normalizado = normalizar(df_iris)
#print(df_iris_normalizado)
df_iris_normalizado.plot.hist(bins=12, alpha=0.5)
plt.title('Histograma iris.arff normalizado')

#Graficamos IRIS Estandarizado
df_iris_estandarizado = estandarizar(df_iris,df_iris.mean(),df_iris.std(axis=0))
df_iris_estandarizado.plot.hist(bins=12, alpha=0.5)
plt.title('Histograma iris.arff estandarizado')

#Graficamos DIABETES Normalizado
df_diabetes_normalizado = normalizar(df_diabetes)
#print(df_diabetes_normalizado)
df_diabetes_normalizado.plot.hist(bins=12, alpha=0.5)
plt.title('Histograma diabetes.arff normalizado')

#Graficamos DIABETES Estandarizado
df_diabetes_estandarizado = estandarizar(df_diabetes,df_diabetes.
 ↪mean(),df_diabetes.std(axis=0))
df_diabetes_estandarizado.plot.hist(bins=12, alpha=0.5)
plt.title('Histograma diabetes.arff estandarizado')
```

[71]:
```
Text(0.5, 1.0, 'Histograma diabetes.arff estandarizado')
```


Histograma iris.arff normalizado
sepallength
sepalwidth
petallength
petalwidth
Frequency
0
5
10
15
20
25
30
35
0.0
0.2
0.4
0.6
0.8
1.0


Histograma iris.arff estandarizado
sepallength
sepalwidth
petallength
petalwidth
Frequency
0
10
20
30
40
50
−2
−1
0
1
2
3



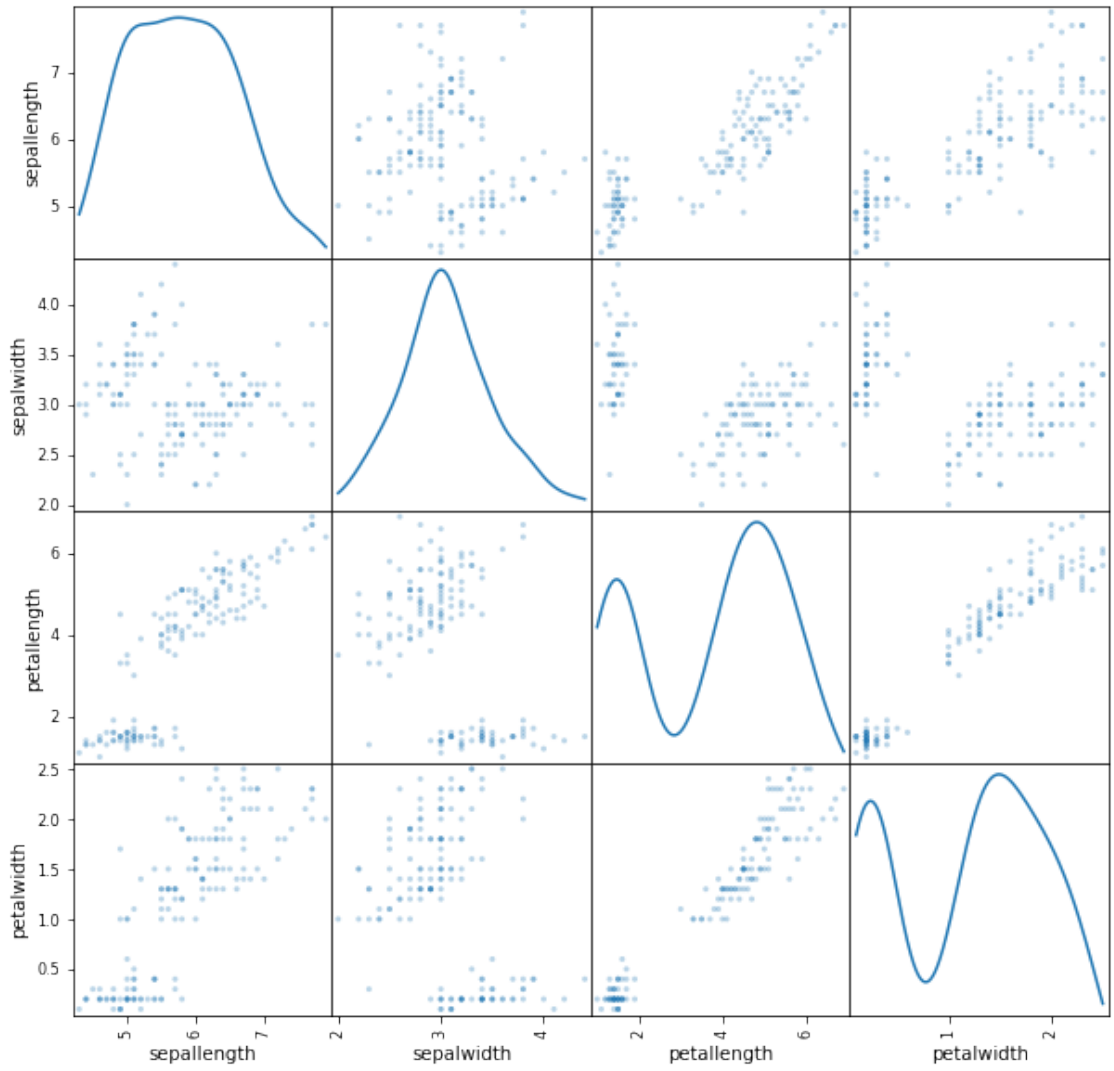


Para los histogramas de normalización los intervalos están entre [0,1], mientras que para los histogramas de estadarización tenemos la típica campana.

0.0.4 Ejercicio 5: Usando la visualización del histograma de dispersión (scatter plot) estuide qué información puede obtener de dicha representación gráfica.

[70]:
```
#Pintamos Matrix Iris
pd.plotting.scatter_matrix(df_iris, alpha = 0.3, figsize = (10, 10),␣
 ↪diagonal='kde')
```

[70]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001E19906DC40>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19A1BFC40>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19A1F3340>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19A21BAF0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001E19A251280>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19A278940>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19A278A30>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19A2B0250>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001E19A303160>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19A338880>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19A3630A0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19A3987C0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001E19A3C0F40>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19A3F7700>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19A421E80>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19A457640>]],
      dtype=object)
```



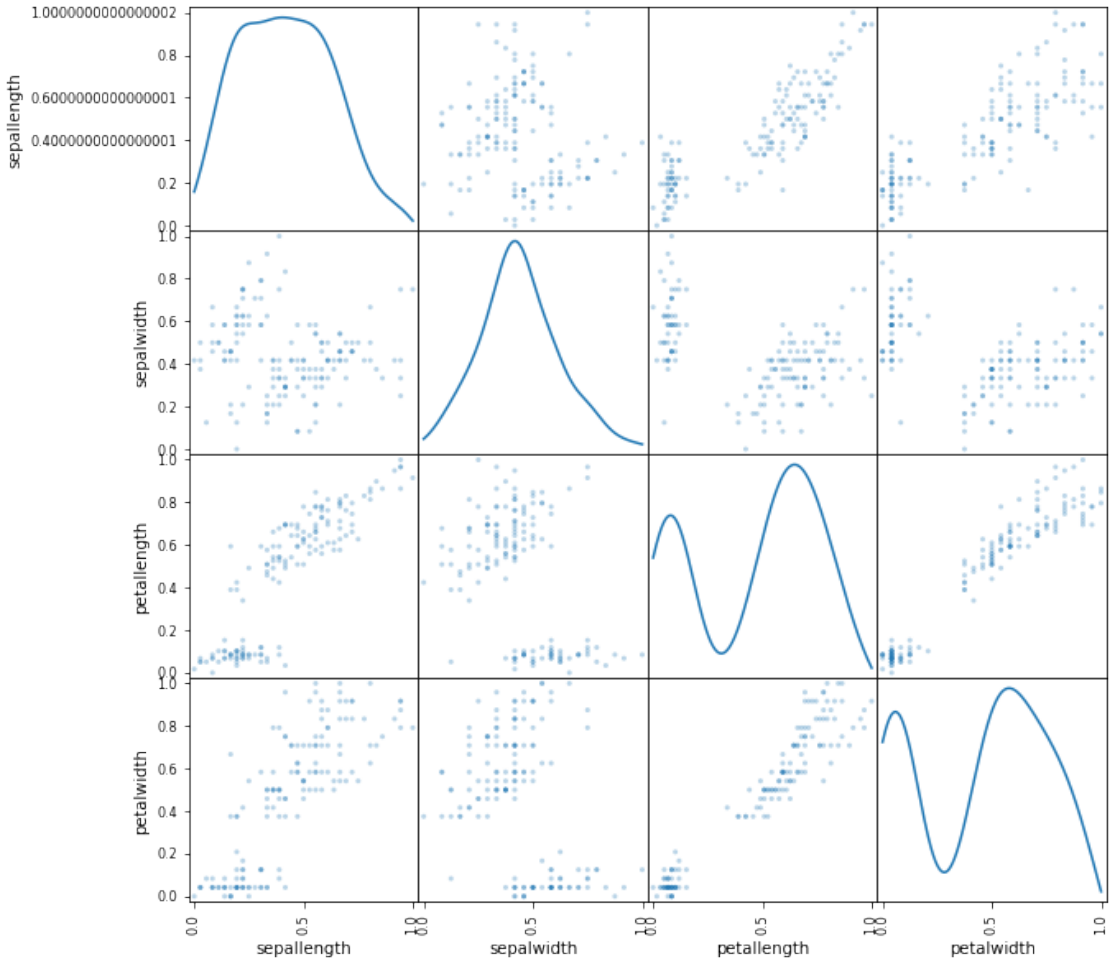
Pandas usa matplotlib para mostrar matrices de dispersión. Con el parámetro 'diagonal' podemos elegir entre 'hist' 'kde'. Elija entre 'kde' e 'hist' para la estimación de densidad de kernel o la gráfica de histograma en la diagonal.

0.0.5 Ejercicio 6: Estudie el efecto de la normalización y la estandarización sobre el diagrama de dispersión.

[75]:
```
#Pintamos Matrix Iris Normalizada
pd.plotting.scatter_matrix(df_iris_normalizado, alpha = 0.3, figsize = (10,␣
↪10), diagonal='kde')
```

[75]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001E19AA96E20>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19AB968B0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19ABBEF10>,
```

```
 <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19ABF46A0>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x000001E19AC1FE20>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19AC55520>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19AC55610>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19AC7DDF0>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x000001E19ACDDCA0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19AD11460>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19AD3ABE0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19AD6E3A0>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x000001E19AD98B20>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19ADD0310>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19ADF7A90>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x000001E19AE2E250>]],
dtype=object)
```

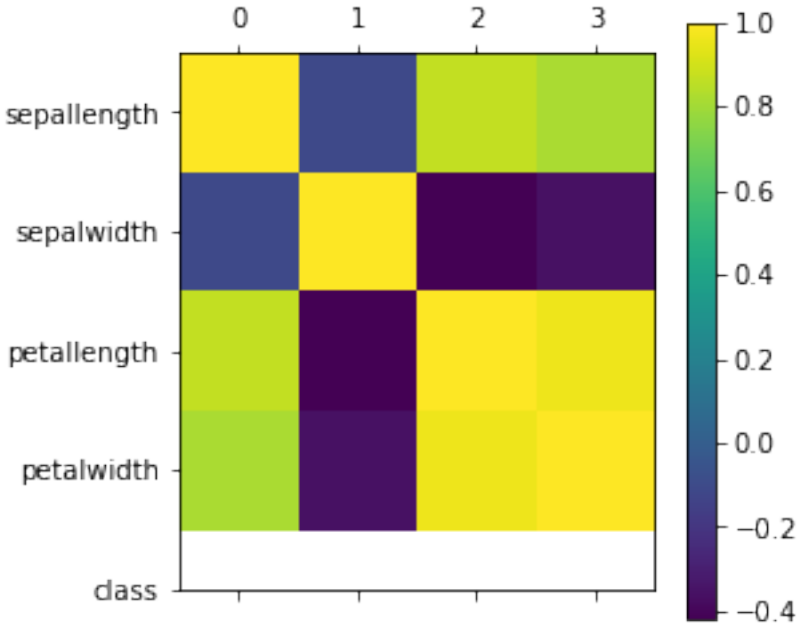


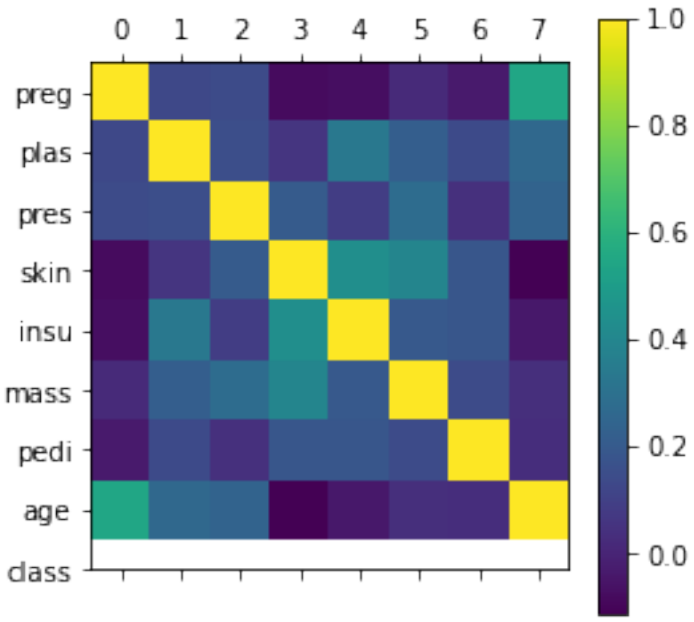
Tanto la normalización como la estandarización no causan ningún efecto, tenemos el mismo diagrama solo que está a otra escala.

0.0.6 Ejercicio 8: Estudie el diagrama de correlaciones de los tres conjuntos e indique qué información relatica a las diferentes clases puede obtener.

[84]:
```
#Correlación IRIS
plt.matshow(df_iris.corr())
plt.yticks(range(len(df_iris.columns)), df_iris.columns)
plt.colorbar()

#Correlación DIABETES
plt.matshow(df_diabetes.corr())
plt.yticks(range(len(df_diabetes.columns)), df_diabetes.columns)
plt.colorbar()
```

[84]:
```
<matplotlib.colorbar.Colorbar at 0x1e19c703df0>
```

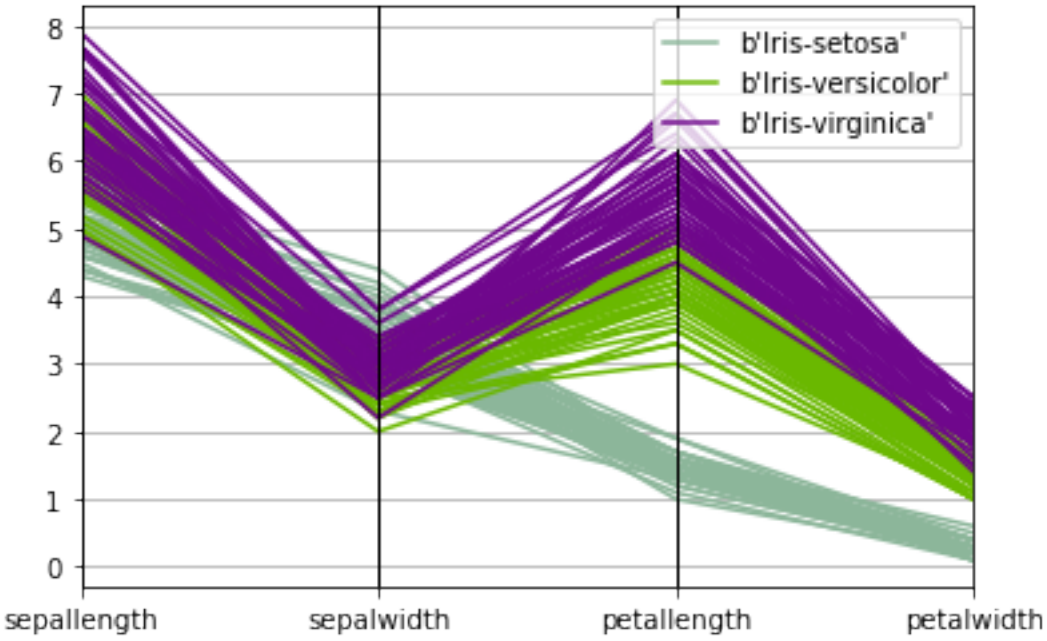


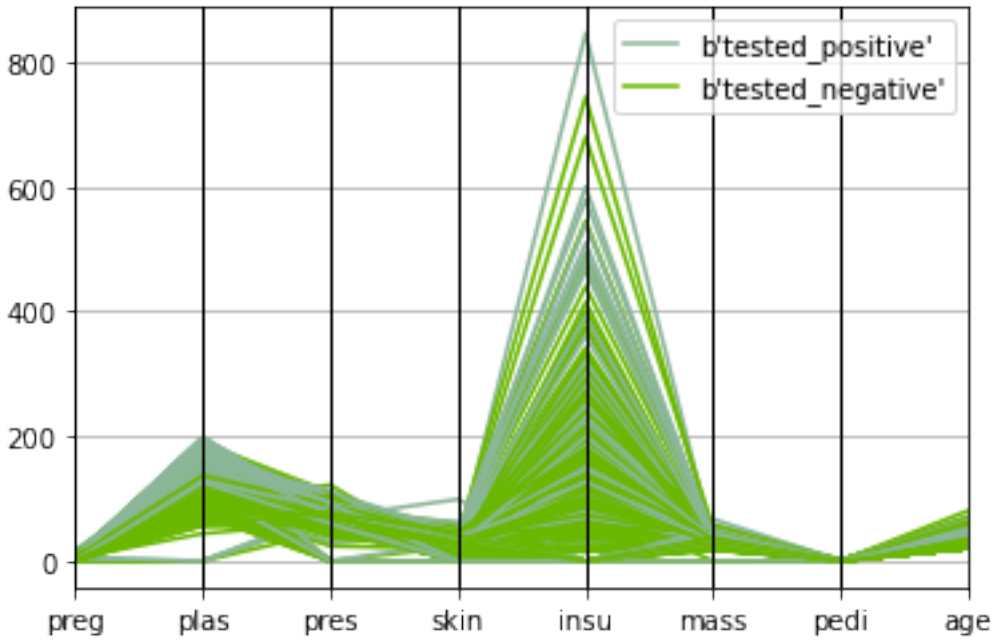


Se entiende por correlación el grado de relación entre dos variables. En el gráfico de IRIS podemos comprobar como las variables petalwidth y petallength están muy correlacionadas.

0.0.7 Ejercicio 9: Estudie la representación en coordenadas paralelas de los tres conjuntos e indique qué información relativa a las diferentes clases puede obtener.

[85]:
```
pd.plotting.parallel_coordinates(df_iris,'class')
plt.show()
pd.plotting.parallel_coordinates(df_diabetes,'class')
plt.show()
```





Las coordenadas paralelas son una manera común de visualizar y analizar conjuntos de datos n-dimensionales.