A2_DataManagmentIris

Pamela Sánchez Arellano A01636995

In [65]:
```
# Import the packages that we will be using
import pandas as pd
```

In [66]:
```
# Define where you are running the code: colab or local
RunInColab          = True     # (False: no  | True: yes)

# If running in colab:
if RunInColab:
    # Mount your google drive in google colab
    from google.colab import drive
    drive.mount('/content/drive')

    # Find location
    #!pwd
    #!ls
    #!ls "/content/drive/My Drive/Colab Notebooks/MachineLearningWithPyt
hon/"

    # Define path del proyecto
    Ruta            = "/content/drive/My Drive/Colab Notebooks/TC1002S/N
otebooksStudents/A01636995"

else:
    # Define path del proyecto
    Ruta            = "/Users/pamelasanchez/Documents/TC1002S/NotebooksS
tudents/A01636995"
```

```
Drive already mounted at /content/drive; to attempt to forcibly remoun
t, call drive.mount("/content/drive", force_remount=True).
```

In [81]:
```
# url string that hosts our .csv file
url = Ruta + "/datasets/iris/iris.csv"
# Read the .csv file and store it as a pandas Data Frame
df = pd.read_csv(url, header = None)

# Column names are added to facilitate the rest of the work
df = df.rename(columns={0: "Largo_Sepalo"})
df = df.rename(columns={1: "Ancho_Sepalo"})
df = df.rename(columns={2: "Largo_Petalo"})
df = df.rename(columns={3: "Ancho_Petalo"})
df = df.rename(columns={4: "Especie"})
```

If we want to print the information about th output object type we would simply type the following: type(df)

In [74]:
```
type(df)
```

Out[74]:
```
pandas.core.frame.DataFrame
```

Exploring the content of the data set

In [75]:
```
df.shape
```

Out[75]:
```
(150, 5)
```

In [76]:
```
df.shape[0]
```

Out[76]:
```
150
```

In [77]:
```
df.shape[1]
```

Out[77]:
```
5
```

If we want to show the entire data frame we would simply write the following:

In [82]:
```
df
```

Out[82]:

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

As you can see, we have a 2-Dimensional object where each row is an independent observation and each coloum is a variable.

Now, use the the `head()` function to show the first 5 rows of our data frame

In [83]:
```
df.head()
```

Out[83]:

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Also, you can use the the `tail()` function to show the last 5 rows of our data frame

In [84]:
```
df.tail()
```

Out[84]:

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

The columns in a Pandas data frame have names, to see the names, use the `columns` method:

To gather more information regarding the data, we can view the column names with the following function:

In [85]:
```
df.columns
```

Out[85]:
```
Index(['Largo_Sepalo', 'Ancho_Sepalo', 'Largo_Petalo', 'Ancho_Petalo',
       'Especie'],
      dtype='object')
```

In [86]:
```
df.dtypes
```

Out[86]:
```
Largo_Sepalo    float64
Ancho_Sepalo    float64
Largo_Petalo    float64
Ancho_Petalo    float64
Especie          object
dtype: object
```

Summary statistics, which include things like the mean, min, and max of the data, can be useful to get a feel for how large some of the variables are and what variables may be the most important.

In [87]:
```
# Summary statistics for the quantitative variables
df.describe()
```

Out[87]:

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [90]:
```
# Drop observations with NaN values
df.Largo_Sepalo.dropna().describe()
#df.'nameOfColumn'.dropna().describe()
```

Out[90]:
```
count    150.000000
mean       5.843333
std        0.828066
min        4.300000
25%        5.100000
50%        5.800000
75%        6.400000
max        7.900000
Name: Largo_Sepalo, dtype: float64
```

It is also possible to get statistics on the entire data frame or a column as follows

- `df.mean()` Returns the mean of all columns
- `df.corr()` Returns the correlation between columns in a data frame
- `df.count()` Returns the number of non-null values in each data frame column
- `df.max()` Returns the highest value in each column
- `df.min()` Returns the lowest value in each column
- `df.median()` Returns the median of each column
- `df.std()` Returns the standard deviation of each column

In [91]:
```
print("MEAN: \ndf" + str(df.mean()))
print("CORRELATION: \ndf" + str(df.corr()))
print("COUNT: \ndf" + str(df.count()))
print("MAX: \ndf" + str(df.max()))
print("MIN: \ndf" + str(df.min()))
print("MEDIAN: \ndf" + str(df.median()))
print("STD: \ndf" + str(df.std()))
```

```
MEAN:
dfLargo_Sepalo    5.843333
Ancho_Sepalo    3.057333
Largo_Petalo    3.758000
Ancho_Petalo    1.199333
dtype: float64
CORRELATION:
df              Largo_Sepalo  Ancho_Sepalo  Largo_Petalo  Ancho_Petalo
Largo_Sepalo      1.000000     -0.117570      0.871754      0.817941
Ancho_Sepalo     -0.117570      1.000000     -0.428440     -0.366126
Largo_Petalo      0.871754     -0.428440      1.000000      0.962865
Ancho_Petalo      0.817941     -0.366126      0.962865      1.000000
COUNT:
dfLargo_Sepalo    150
Ancho_Sepalo    150
Largo_Petalo    150
Ancho_Petalo    150
Especie         150
dtype: int64
MAX:
dfLargo_Sepalo              7.9
Ancho_Sepalo              4.4
Largo_Petalo              6.9
Ancho_Petalo              2.5
Especie         Iris-virginica
dtype: object
MIN:
dfLargo_Sepalo            4.3
Ancho_Sepalo            2.0
Largo_Petalo            1.0
Ancho_Petalo            0.1
Especie         Iris-setosa
dtype: object
MEDIAN:
dfLargo_Sepalo    5.80
Ancho_Sepalo    3.00
Largo_Petalo    4.35
Ancho_Petalo    1.30
dtype: float64
STD:
dfLargo_Sepalo    0.828066
Ancho_Sepalo    0.435866
Largo_Petalo    1.765298
Ancho_Petalo    0.762238
dtype: float64
```

```
<ipython-input-91-05c1408d5d26>:1: FutureWarning: Dropping of nuisance
columns in DataFrame reductions (with 'numeric_only=None') is deprecate
d; in a future version this will raise TypeError.  Select only valid co
lumns before calling the reduction.
  print("MEAN: \ndf" + str(df.mean()))
<ipython-input-91-05c1408d5d26>:6: FutureWarning: Dropping of nuisance
columns in DataFrame reductions (with 'numeric_only=None') is deprecate
d; in a future version this will raise TypeError.  Select only valid co
lumns before calling the reduction.
  print("MEDIAN: \ndf" + str(df.median()))
<ipython-input-91-05c1408d5d26>:7: FutureWarning: Dropping of nuisance
columns in DataFrame reductions (with 'numeric_only=None') is deprecate
d; in a future version this will raise TypeError.  Select only valid co
lumns before calling the reduction.
  print("STD: \ndf" + str(df.std()))
```

In [30]:
```
df.to_csv('myDataFrame.csv')
```

In [95]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remoun
t, call drive.mount("/content/drive", force_remount=True).
```

In [100]:
```
df = df.rename(columns={"Largo_Sepalo": "Petalo"})
df.head()
```

Out[100]:

	Petalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [101]:
```
# Back to the original name
df = df.rename(columns={"Petalo": "Largo_Sepalo"})
df.head()
```

Out[101]:

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [102]:
```
a = df.Largo_Sepalo
b = df["Largo_Sepalo"]
c = df.loc[:, "Largo_Sepalo"]
d = df.iloc[:, 1]
print(c)
```

```
0      5.1
1      4.9
2      4.7
3      4.6
4      5.0
      ...
145    6.7
146    6.3
147    6.5
148    6.2
149    5.9
Name: Largo_Sepalo, Length: 150, dtype: float64
```

In [105]:
```
# Return all observations of CWDistance
df.loc[:,["Largo_Sepalo", "Ancho_Sepalo"]]

# Select multiple columns, ["Gender", "GenderGroup"]me
#keep = ['Gender', 'GenderGroup']
#df_gender = df[keep]

# Select range of rows for all columns
#df.loc[10:15,:]
```

Out[105]:

	Largo_Sepalo	Ancho_Sepalo
0	5.1	3.5
1	4.9	3.0
2	4.7	3.2
3	4.6	3.1
4	5.0	3.6
...	...	...
145	6.7	3.0
146	6.3	2.5
147	6.5	3.0
148	6.2	3.4
149	5.9	3.0

150 rows × 2 columns

The attribute **iloc()** is an integer based slicing.

In [106]:
```
# .
#df.iloc[:, :4]

# .
df.iloc[:4, :]

# .
#df.iloc[:, 3:7]

# .
#df.iloc[4:8, 2:4]

# This is incorrect:
#df.iloc[1:5, ["Gender", "GenderGroup"]]
```

Out[106]:

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa

In [107]:
```
# List unique values in the df['Gender'] column
df.Largo_Sepalo.unique()
```

Out[107]:
```
array([5.1, 4.9, 4.7, 4.6, 5. , 5.4, 4.4, 4.8, 4.3, 5.8, 5.7, 5.2, 5.5,
       4.5, 5.3, 7. , 6.4, 6.9, 6.5, 6.3, 6.6, 5.9, 6. , 6.1, 5.6, 6.7,
       6.2, 6.8, 7.1, 7.6, 7.3, 7.2, 7.7, 7.4, 7.9])
```

In [108]:
```
#Filter, Sort and Groupby
df[df['Largo_Sepalo']>5]
```

Out[108]:

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie
0	5.1	3.5	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
10	5.4	3.7	1.5	0.2	Iris-setosa
14	5.8	4.0	1.2	0.2	Iris-setosa
15	5.7	4.4	1.5	0.4	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

118 rows × 5 columns

In [110]:
```
#Sort
df.sort_values("Largo_Sepalo")
#df.sort_values(ColumnName, ascending=False)
```

Out[110]:

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie
13	4.3	3.0	1.1	0.1	Iris-setosa
42	4.4	3.2	1.3	0.2	Iris-setosa
38	4.4	3.0	1.3	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
41	4.5	2.3	1.3	0.3	Iris-setosa
...	...	...	...	...	...
122	7.7	2.8	6.7	2.0	Iris-virginica
118	7.7	2.6	6.9	2.3	Iris-virginica
117	7.7	3.8	6.7	2.2	Iris-virginica
135	7.7	3.0	6.1	2.3	Iris-virginica
131	7.9	3.8	6.4	2.0	Iris-virginica

150 rows × 5 columns

In [111]:
```
df.notnull()
#df.isnull()
```

Out[111]:

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie
0	True	True	True	True	True
1	True	True	True	True	True
2	True	True	True	True	True
3	True	True	True	True	True
4	True	True	True	True	True
...	...	...	...	...	...
145	True	True	True	True	True
146	True	True	True	True	True
147	True	True	True	True	True
148	True	True	True	True	True
149	True	True	True	True	True

150 rows × 5 columns

In [112]:
```
df.isnull().sum()
#df.notnull().sum()
```

Out[112]:
```
Largo_Sepalo    0
Ancho_Sepalo    0
Largo_Petalo    0
Ancho_Petalo    0
Especie         0
dtype: int64
```

In [114]:
```
df.Largo_Sepalo.notnull().sum()
```

Out[114]:
```
150
```

In [115]:
```
# Extract all non-missing values of one of the columns into a new variab
le
x = df.Largo_Sepalo.dropna().describe()
x
```

Out[115]:
```
count    150.000000
mean       5.843333
std        0.828066
min        4.300000
25%        5.100000
50%        5.800000
75%        6.400000
max        7.900000
Name: Largo_Sepalo, dtype: float64
```

In [116]:
```
df.head()
```

Out[116]:

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [125]:
```
# Add a new column with new data
# Create a column data
NewColumnData = df.Largo_Sepalo/df.Ancho_Sepalo
# Insert that column in the data frame
df.insert(5, "L_Sepalo/A_Sepalo", NewColumnData, True)
df.head()
```

Out[125]:

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie	L_Sepalo/A_Sepalo	Largo_
0	5.1	3.5	1.4	0.2	Iris-setosa	1.457143	
1	4.9	3.0	1.4	0.2	Iris-setosa	1.633333	
2	4.7	3.2	1.3	0.2	Iris-setosa	1.468750	
3	4.6	3.1	1.5	0.2	Iris-setosa	1.483871	
4	5.0	3.6	1.4	0.2	Iris-setosa	1.388889	

In [127]:
```
# # Eliminate inserted column
df.drop("L_Sepalo/A_Sepalo", axis=1, inplace = True)
# #df.drop(columns=['ColumnInserted'], inplace = True)
# # Remove three columns as index base
# #df.drop(df.columns[[12]], axis = 1, inplace = True)
df.head()
```

Out[127]:

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [128]:
```
# # Add new column derived from existing columns
# # The new column is a function of another column
df["Largo_Sepalo*10"] = df["Largo_Sepalo"] * 10
df.head()
```

Out[128]:

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie	Largo_Sepalo*10
0	5.1	3.5	1.4	0.2	Iris-setosa	51.0
1	4.9	3.0	1.4	0.2	Iris-setosa	49.0
2	4.7	3.2	1.3	0.2	Iris-setosa	47.0
3	4.6	3.1	1.5	0.2	Iris-setosa	46.0
4	5.0	3.6	1.4	0.2	Iris-setosa	50.0

In [129]:
```
# # Eliminate inserted column
df.drop("Largo_Sepalo*10", axis=1, inplace = True)
df.head()
```

Out[129]:

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Add and eliminate rows

In some cases it is requiered to add new observations (rows) to the data set

In [130]:
```
# Print tail
df.tail()
```

Out[130]:

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

In [131]:
```
df.loc[len(df.index)] = [26, 24, 'F', 1, 'Y']
#
df.tail()
```

Out[131]:

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica
150	26.0	24.0	F	1.0	Y

In [133]:
```
## Eliminate inserted row
df.drop([150], inplace = True )
#
df.tail()
```

Out[133]:

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie
144	6.7	3.3	5.7	2.5	Iris-virginica
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica

Activity: work with the iris dataset

Repeat this tutorial with the iris data set and respond to the following inquiries

1. Calculate the statistical summary for each quantitative variables. Explain the results
   - Identify the name of each column
   - Identify the type of each column
   - Minimum, maximum, mean, average, median, standar deviation

1. Are there missing data? If so, create a new dataset containing only the rows with the non-missing data

1. Create a new dataset containing only the petal width and length and the type of Flower

1. Create a new dataset containing only the setal width and length and the type of Flower

1. Create a new dataset containing the setal width and length and the type of Flower encoded as a categorical numerical column

At the end with the statistical summary of the information, it is seen how the dimension of the flower relate each other to the type of flower and the distinct dimension.

In [148]:
```
#1. Calculate the statistical summary for each quantitative variables. E
xplain the results
#
df.describe()
```

Out[148]:

	Largo_Sepalo	Ancho_Sepalo	Ancho_Petalo
count	149.000000	149.000000	149.000000
mean	5.842953	3.057718	1.195302
std	0.830846	0.437311	0.763202
min	4.300000	2.000000	0.100000
25%	5.100000	2.800000	0.300000
50%	5.800000	3.000000	1.300000
75%	6.400000	3.300000	1.800000
max	7.900000	4.400000	2.500000

In [145]:
```
#2. Are there missing data? If so, create a new dataset containing only
the rows with the non-missing data
df2 = df.dropna()
df2
```

Out[145]:

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
144	6.7	3.3	5.7	2.5	Iris-virginica
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica

149 rows × 5 columns

In [137]:
```
#3. Create a new dataset containing only the petal width and length and
the type of Flower
keep = ['Largo_Petalo', 'Ancho_Petalo', 'Especie']
df_flower = df[keep]
df_flower
```

Out[137]:

	Largo_Petalo	Ancho_Petalo	Especie
0	1.4	0.2	Iris-setosa
1	1.4	0.2	Iris-setosa
2	1.3	0.2	Iris-setosa
3	1.5	0.2	Iris-setosa
4	1.4	0.2	Iris-setosa
...	...	...	...
144	5.7	2.5	Iris-virginica
145	5.2	2.3	Iris-virginica
146	5.0	1.9	Iris-virginica
147	5.2	2.0	Iris-virginica
148	5.4	2.3	Iris-virginica

149 rows × 3 columns

In [139]:
```
#4. Create a new dataset containing only the setal width and length and the type of Flower
keep = ['Largo_Sepalo', 'Ancho_Sepalo', 'Especie']
df_flower2 = df[keep]
df_flower2
```

Out[139]:

	Largo_Sepalo	Ancho_Sepalo	Especie
0	5.1	3.5	Iris-setosa
1	4.9	3.0	Iris-setosa
2	4.7	3.2	Iris-setosa
3	4.6	3.1	Iris-setosa
4	5.0	3.6	Iris-setosa
...	...	...	...
144	6.7	3.3	Iris-virginica
145	6.7	3.0	Iris-virginica
146	6.3	2.5	Iris-virginica
147	6.5	3.0	Iris-virginica
148	6.2	3.4	Iris-virginica

149 rows × 3 columns

In [140]:
```
#5. Create a new dataset containing the setal width and length and the type of Flower encoded as a categorical numerical column
dfT = df.copy()
dfT["Especie"] = dfT["Especie"].replace({"Iris-setosa":0,"Iris-virginica":1,"Iris-versicolor":2})
keep = ['Largo_Sepalo', 'Ancho_Sepalo', 'Especie']
df5 = dfT[keep]
df5
```

Out[140]:

	Largo_Sepalo	Ancho_Sepalo	Especie
0	5.1	3.5	0
1	4.9	3.0	0
2	4.7	3.2	0
3	4.6	3.1	0
4	5.0	3.6	0
...	...	...	...
144	6.7	3.3	1
145	6.7	3.0	1
146	6.3	2.5	1
147	6.5	3.0	1
148	6.2	3.4	1

149 rows × 3 columns