Sprint 7.

Algoritmos de aprendizaje supervisado: Clasificación

```
import numpy as np
import matplotlib.pyplot as plt

import pandas as pd
import seaborn as sns

%matplotlib inline


from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split, cross_val_score
```

```
# Configuración warnings
# ==============================================================================
import warnings
warnings.filterwarnings('ignore')
```

```
# Activo Google Drive

from google.colab import drive
drive.mount('/content/drive')
```

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

```
# Abro el fichero

path = ('/content/drive/MyDrive/01_COLAB/wineData.txt')


df=  pd.read_csv(path, sep=',', encoding="latin-1")
df.shape

nRow, nCol = df.shape
print(f'Hay  {nRow} filas con {nCol} columnas')
df1=df.copy()
print('\nImprimo el primer registro, solo para ver como es:\n')
df1.iloc[0]
```

```
Hay  177 filas con 14 columnas

Imprimo el primer registro, solo para ver como es:

1          1.00
14.23     13.20
1.71       1.78
2.43       2.14
15.6      11.20
127      100.00
2.8        2.65
3.06       2.76
.28        0.26
2.29       1.28
5.64       4.38
1.04       1.05
3.92       3.40
1065    1050.00
Name: 0, dtype: float64
```

```

from sklearn.datasets import load_wine
wine=load_wine()

data=pd.DataFrame(data=np.c_[wine['data'],wine['target']],columns=wine['feature_names']+['target'])

# Revisamos los datos
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   alcohol                       178 non-null    float64
 1   malic_acid                    178 non-null    float64
 2   ash                           178 non-null    float64
 3   alcalinity_of_ash             178 non-null    float64
 4   magnesium                     178 non-null    float64
 5   total_phenols                 178 non-null    float64
 6   flavanoids                    178 non-null    float64
 7   nonflavanoid_phenols          178 non-null    float64
 8   proanthocyanins               178 non-null    float64
 9   color_intensity               178 non-null    float64
 10  hue                           178 non-null    float64
 11  od280/od315_of_diluted_wines  178 non-null    float64
 12  proline                       178 non-null    float64
 13  target                        178 non-null    float64
dtypes: float64(14)
memory usage: 19.6 KB
```

Haz doble clic (o pulsa Intro) para editar

```
# Asigno un nomber a las columnas:

```

```
Nombre_Columnas  = ['target','alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium',
       'total_phenols', 'flavanoids', 'nonflavanoid_phenols',
       'proanthocyanins', 'color_intensity', 'hue',
       'od280/od315_of_diluted_wines', 'proline']
```

```
# Asigno una lista con los nombres de las columnas
df1.columns = Nombre_Columnas
df1.iloc[0]
```

```
target                          1.00
alcohol                        13.20
malic_acid                      1.78
ash                             2.14
alcalinity_of_ash              11.20
magnesium                     100.00
total_phenols                   2.65
flavanoids                      2.76
nonflavanoid_phenols            0.26
proanthocyanins                 1.28
color_intensity                 4.38
hue                             1.05
od280/od315_of_diluted_wines    3.40
proline                      1050.00
Name: 0, dtype: float64
```

```
# Busco si hay datos faltantes

(df1.isnull() | df1.empty | df1.isna()).sum()
```

```
target                        0
alcohol                       0
malic_acid                    0
ash                           0
alcalinity_of_ash             0
magnesium                     0
total_phenols                 0
flavanoids                    0
nonflavanoid_phenols          0
proanthocyanins               0
color_intensity               0
hue                           0
od280/od315_of_diluted_wines  0
proline                       0
dtype: int64
```

```
df1.head()
```

	target	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nc
0	1	13.20	1.78	2.14	11.2	100	2.65	2.76	
1	1	13.16	2.36	2.67	18.6	101	2.80	3.24	
2	1	14.37	1.95	2.50	16.8	113	3.85	3.49	
3	1	13.24	2.59	2.87	21.0	118	2.80	2.69	

Analisis de los datos:

```
# Vamos a mostrar un resumen del conjunto de datos donde podemos ver
# los datos estadísticos básicos.
df1.describe()
```

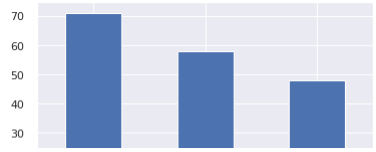
	target	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenol
count	177.000000	177.000000	177.000000	177.000000	177.000000	177.000000	177.0000(
mean	1.943503	12.993672	2.339887	2.366158	19.516949	99.587571	2.2922(
std	0.773991	0.808808	1.119314	0.275080	3.336071	14.174018	0.6264(
min	1.000000	11.030000	0.740000	1.360000	10.600000	70.000000	0.9800(
25%	1.000000	12.360000	1.600000	2.210000	17.200000	88.000000	1.7400(
50%	2.000000	13.050000	1.870000	2.360000	19.500000	98.000000	2.3500(

```
#Veamos la frecuencia del objetivo variable.
##Convertir variable a categórica.
data.target=data.target.astype('int64').astype('category')

#Frequencia.
frecuencia=df1['target'].value_counts()

frecuencia
```

```
2    71
1    58
3    48
Name: target, dtype: int64
```

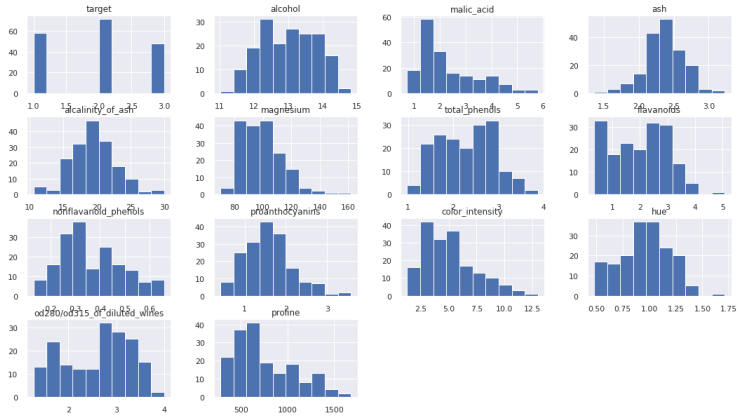
```
#Graficamos la frecuencia
frecuencia.plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd83a9cb850>
```



```
#Vamos a mostrar los histogramas de las variables alcohol, magnesio y color_intensity.....
#Histogramas

df1[df1.columns].hist(figsize=(18,10))
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fd83a93ea10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd83a96fed0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd83a938410>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd83a8ec810>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fd83a8a0cd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd83a863210>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd83a818790>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd83a7cec10>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fd83a7cec50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd83a792290>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd83a700b90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd83a6c30d0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fd83a6f95d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd83a6b0ad0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd83a65cb10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd83a62a510>]],
      dtype=object)
```



https://www.alldatascience.com/classification/wine-dataset-analysis-with-python/

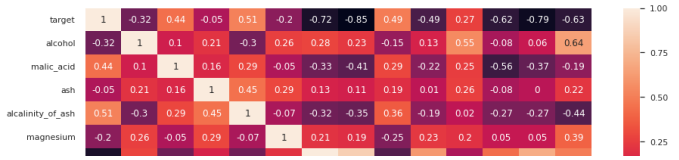
En los puntos anteriores vemos como todas las variables del conjunto de datos, excepto la variable **target**, son numéricas continuas.

No faltan valores en ninguna de las variables. De los valores estadísticos básicos podemos ver que ninguna de las variables sigue una distribución normal, ya que ninguna tiene media 0 y desviación estándar 1.

En los histogramas podemos observar como la variable alcohol tiene una distribución más o menos centrada, con la mayoría de las registros que tienen valores entre 12 y 14 grados, en cuanto a color_intensity y malic_acid, observamos que sus distribuciones están sesgadas a la izquierda.

```
# Creo la matriz de correlacion entre todos los factores.

correlation_matrix = df1.corr().round(2)
sns.set(rc = {'figure.figsize':(15,8)})
sns.heatmap(data=correlation_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd83a297d50>
```



PCA

[ ] ↳ *3 celdas ocultas*

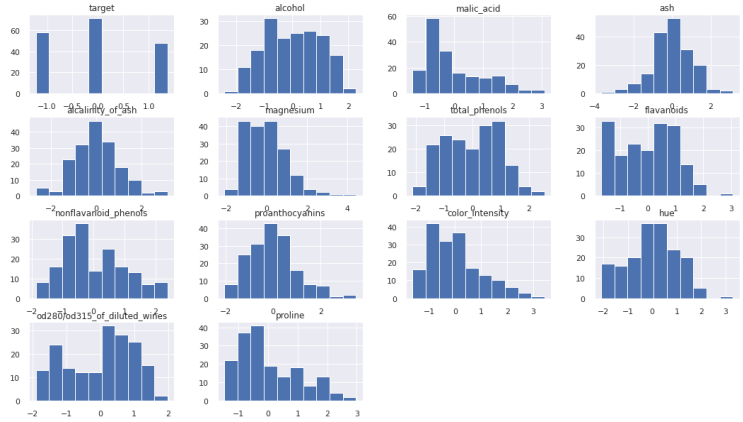
Normalizo los datos

```
# Normalizo los datos para evitar que las columnas de valores grandes
# se impongan a las columnas de valores pequeños:


from sklearn_pandas import DataFrameMapper

mapper = DataFrameMapper([(df1.columns, StandardScaler())])
scaled_features = mapper.fit_transform(df1.copy(), 4)
df3= pd.DataFrame(scaled_features, index=df1.index, columns=df1.columns)

df3
```

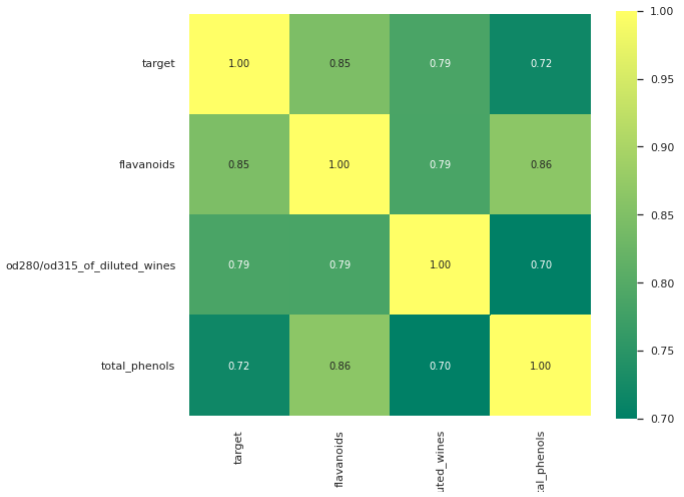
	target	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flav
0	-1.222468	0.255824	-0.501624	-0.824485	-2.500110	0.029180	0.572666	0.
1	-1.222468	0.206229	0.018020	1.107690	-0.275639	0.099932	0.812784	1.
2	-1.222468	1.706501	-0.349315	0.487935	-0.816726	0.948953	2.493609	1.
3	-1.222468	0.305420	0.224086	1.836812	0.445811	1.302712	0.812784	0.
4	-1.222468	1.495719	-0.519543	0.305655	-1.297693	0.878201	1.565153	1.
...	...	...	...	...	...	...	...	
172	1.368871	0.888171	2.965658	0.305655	0.295509	-0.324579	-0.980097	-1.
173	1.368871	0.503803	1.406725	0.415023	1.047020	0.170684	-0.788003	-1.
174	1.368871	0.342617	1.738222	-0.387012	0.145207	1.444215	-1.124168	-1.

```
# Con los datos transformados veo si ha cambiado mucho la forma de cada variable
df3[df3.columns].hist(figsize=(18,10))
```

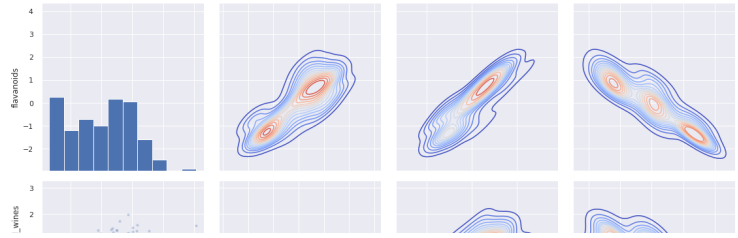
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fd83a528910>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd83aa64e50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd83d9d2cd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd83aad1910>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fd83ab9b9d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd83ac23190>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd83c6bde90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd8407b76d0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fd8407846d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd8406a5b10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd84f30ed10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd83ed76990>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fd83ef62050>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd83c4eedd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd84f3cec50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd83c4b3b90>]],
      dtype=object)
```



```
# Quiero ver que variables tienen mas correlacion con respecto al target

columns_sorted = df3.corr().abs().nlargest(4, 'target').index
correlation_sorted = np.corrcoef(df3[columns_sorted].values.T)

f, ax = plt.subplots(figsize = (9,7.5))
hm = sns.heatmap(abs(correlation_sorted), annot=True, square=True, fmt='.2f', annot_kws={'size': 10}, yticklabels=columns_sorted.values, xticklabels
plt.show()
```

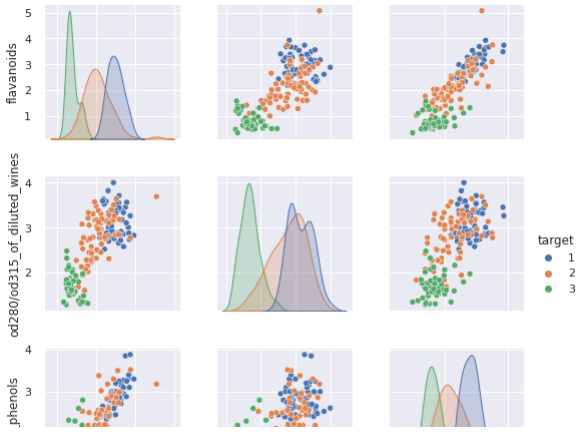



```
# Dibujar relaciones entre los datos de una manera visual de las 3 variables
# con mas correlacion con la salida:


sns.set(rc={'figure.figsize':(15, 10)})

ColumnasCorrelacionAlta  = [ 'flavanoids','od280/od315_of_diluted_wines', 'total_phenols', 'target',]

g = sns.PairGrid(df3, vars=ColumnasCorrelacionAlta, height=4, aspect=1)
g = g.map_diag(plt.hist)
g = g.map_lower(sns.regplot, lowess=True, scatter_kws={'s': 15, 'alpha':0.3},
                line_kws={'color':'red', 'linewidth': 2})
g = g.map_upper(sns.kdeplot, n_levels=15, cmap='coolwarm')

plt.show()
```



```
#scatter plots

df2 = df1[['flavanoids','od280/od315_of_diluted_wines', 'total_phenols', 'target']]
sns.pairplot(df2,hue='target', palette='deep')
```

```
<seaborn.axisgrid.PairGrid at 0x7fd83f0bc290>
```



Hasta aquí el estudio de los datos

- Exercici 1

Crea almenys dos models de classificació diferents per intentar predir el millor les classes de l'arxiu adjunt.

[ ] ↳ *46 celdas ocultas*

- Exercici 2

Compara els models de classificació utilitzant la precisió (accuracy), una matriu de confiança i d'altres mètriques més avançades.

[ ] ↳ *4 celdas ocultas*

- Exercici 3

Entrena'ls usant els diferents paràmetres que admeten per tal de millorar-ne la predicció.

[ ] ↳ *9 celdas ocultas*

- Exercici 4

Compara el seu rendiment fent servir l'aproximació traint/test o cross-validation.

```
datos_x= df2.loc[:, ['flavanoids','od280/od315_of_diluted_wines', 'total_phenols']].values
datos_y = df2.loc[:,['target']].values

#Dividimos el dataset en training set y test set
X_train, X_test, y_train, y_test = train_test_split(datos_x, datos_y, test_size=0.3,random_state=109) # 70% training and 30% test
```

```

# Utilizaré la regresion logística:


logreg = LogisticRegression()

# fit el modelo
logreg.fit(X_train,y_train)

#
y_pred_RLogis=logreg.predict(X_test)
scores_RL = cross_val_score(logreg, X_train, y_train, cv=5)
print("Accuracy Train: %0.2f (+/- %0.2f)" % (scores_RL.mean(), scores_RL.std() * 2))


logreg.fit(X_test,y_test)

```

```
scores_RL = cross_val_score(logreg, X_test, y_test, cv=5)
print("Accuracy Test: %0.2f (+/- %0.2f)" % (scores_RL.mean(), scores_RL.std() * 2))

scores_RL = cross_val_score(logreg, datos_x, datos_y, cv=5)
print("Accuracy Todos: %0.2f (+/- %0.2f)" % (scores_RL.mean(), scores_RL.std() * 2))

```

```
Accuracy Train: 0.81 (+/- 0.15)
Accuracy Test: 0.85 (+/- 0.18)
Accuracy Todos: 0.80 (+/- 0.14)
```

4.2 Cross Validation

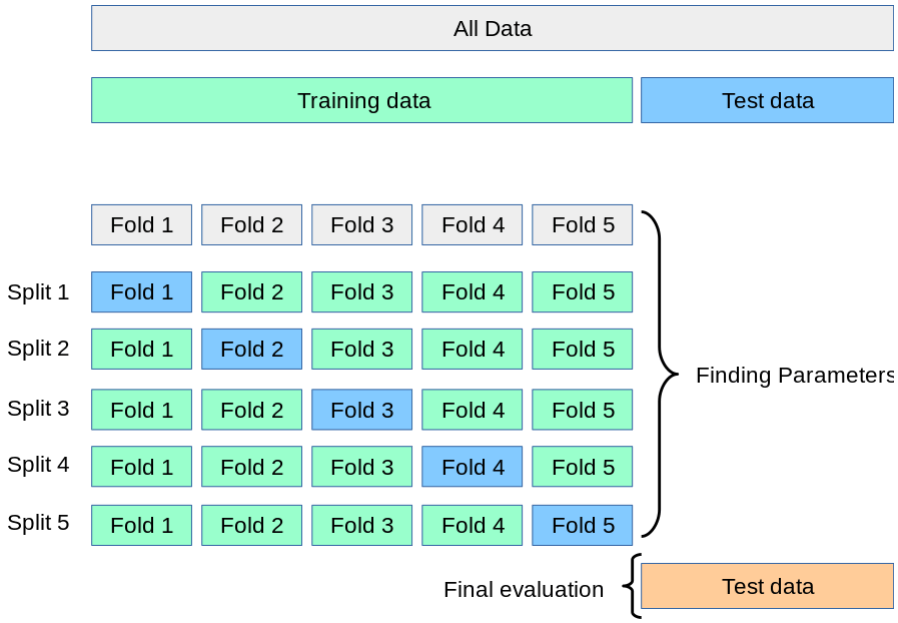
Exercici 5

Aplica algun procés d'enginyeria per millorar els resultats (normalització, estandarditzaci, mostreig...)

El DataFrame que he utilizado los datos han sido normalizados ya

Ver pto del indice ("**Normalizacion de los datos**")

.



Conclusion Spring 7

El proceso para crear algoritmos supervisados, (supervisado = que sabemos que resultado se ha obtenido) .... es el siguiente:

- 1.- Primero separamos la base de datos en datos de Train y Test.
- 2.- Creamos el modelo con los datos de Train
- 3.- Lo validamos con los datos de Test. Introducimos los datos de Test en el modelo y vemos el resutado que obtenemos. Después teenmos que comparar los resultados obtenidos contra los datos de y_test, y aobtendremos la Accuracy y el Error.