Machine Learning Python

Antonio Cola, Maristella Simonetti, Rosario Urso

1 Descrizione dataset

Il dataset contiene 178 osservazioni per ognuna delle quali sono state rilevate 13 variabili. I dati sono il risultato di un'analisi chimica di vini prodotti nella stessa regione italiana da tre diversi coltivatori (59 per il primo, 71 per il secondo, 48 per il terzo). Le variabili sono rilevazioni delle componenti chimiche presenti nei vini ad esempio alcol, colore, intensità colore, flavonoidi, magnesio, etc...

2 Descrizione algoritmi

Abbiamo deciso di utilizzare due algoritmi di Machine Learning. Il primo problema da risolvere è di classificazione e il metodo utilizzato è il *Decision Tree*. In questo caso si parla di addestramento supervisionato in quanto l'insieme di ingresso sarà costituito da una serie di dati con le rispettive label. L'algoritmo dovrà, tramite una serie di split successivi dell'insieme di partenza, assegnare una delle tre etichette alle varie osservazioni.
Abbiamo ricavato, grazie all'ausilio del software MATLAB, la matrice di confusione, ovvero lo strumento per analizzare gli errori compiuti dal modello di Machine Learning. Essa va ad incrociare i valori predetti dal modello (sulle colonne della matrice) ed i valori effettivi (sulle righe della matrice). I valori sulla diagonale principale rappresentano i valori correttamente classificati, al di fuori sono presenti gli errori di classificazioni. Al fine di valutare in maniera ottimale ed efficiente la performance del modello è opportuno non utilizzare tutto l'insieme di partenza come training set (ovvero come set con cui costruire l'albero e dunque addestrare il modello) ma operare una *cross validation*: l'insieme di partenza viene diviso in un certo numero di sottoinsiemi, uno viene adibito al ruolo di test set, un altro al ruolo di validation set e i restanti al ruolo di training set (necessari rispettivamente per la valutazione dell'accuratezza del modello predittivo, la selezione dei migliori iperparametri, la costruzione dell'albero). Il ruolo di questi sottoinsiemi varia ad ogni iterazione.

Il secondo algoritmo utilizzato, *K-Means*, è volto alla risoluzione di un problema di *clustering*. In questo caso si tratta di apprendimento non supervisionato in quanto i dati in ingresso non sono corredati da etichette. L'obiettivo è individuare un certo numero k di insiemi nello spazio dei campioni che siano sufficientemente omogenei internamente. Ad ogni iterazione, l'algoritmo andrà ad aggiornare i centroidi dei k insiemi (ovvero i punti medi degli stessi).

3 Implementazione algoritmi

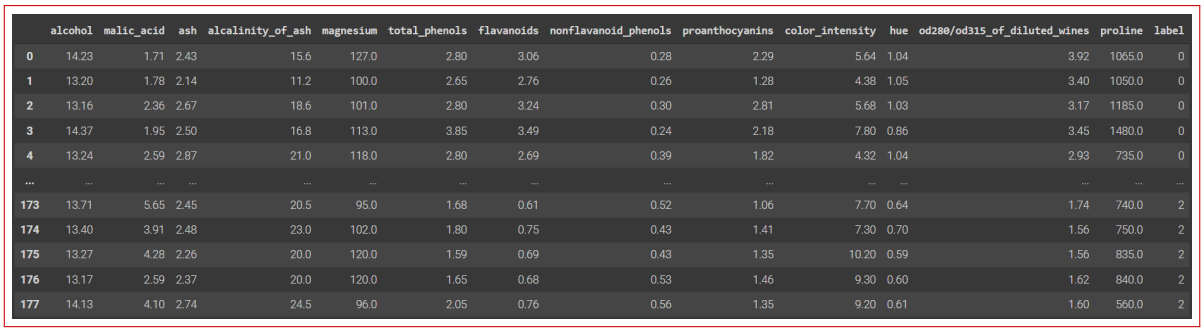
3.1 Decision Tree

Collegamento a Google Drive.

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

Importazioni delle librerie e visualizzazione del dataset

```
from sklearn.datasets import load_wine
import pandas as pd
data = load_wine()
dataset = pd.DataFrame.from_records(data=data.data, columns=data.feature_names)
dataset['label']= data.target
dataset
```

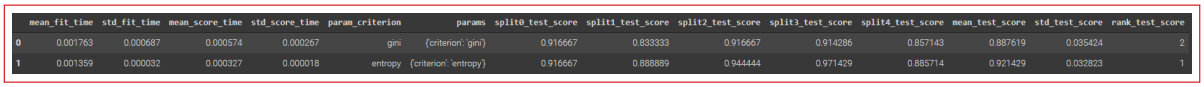


	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline	label
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065.0	0
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050.0	0
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185.0	0
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480.0	0
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
173	13.71	5.65	2.45	20.5	95.0	1.68	0.61	0.52	1.06	7.70	0.64	1.74	740.0	2
174	13.40	3.91	2.48	23.0	102.0	1.80	0.75	0.43	1.41	7.30	0.70	1.56	750.0	2
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	0.43	1.35	10.20	0.59	1.56	835.0	2
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	0.53	1.46	9.30	0.60	1.62	840.0	2
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	0.56	1.35	9.20	0.61	1.60	560.0	2

Figura 1: Tabella dataset

Valutazione del modello: selezione degli iperparametri e Cross Validation

```
from sklearn.model_selection import GridSearchCV
from sklearn import tree
clf = tree.DecisionTreeClassifier()
parameters = {'criterion':('gini', 'entropy')}
clf = GridSearchCV(clf, parameters)
clf.fit(data.data, data.target)
GridSearchCV(estimator=clf, param_grid=parameters, return_train_score=True)
df = pd.DataFrame(clf.cv_results_)
df
```



	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_criterion	params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score	rank_test_score
0	0.001763	0.000687	0.000574	0.000267	gini	{'criterion': 'gini'}	0.916667	0.833333	0.916667	0.914286	0.857143	0.887619	0.035424	2
1	0.001359	0.000032	0.000327	0.000018	entropy	{'criterion': 'entropy'}	0.916667	0.888889	0.944444	0.971429	0.885714	0.921429	0.032823	1

Figura 2: Selezione degli iperparametri e Cross Validation

Processo Decision Tree

```
X, y = load_wine(return_X_y=True)
clf = tree.DecisionTreeClassifier(criterion='entropy')
clf = clf.fit(X, y)
```

Visualizzazione del modello in forma testuale e grafica

```
tree.plot_tree(clf.fit(X, y))
```

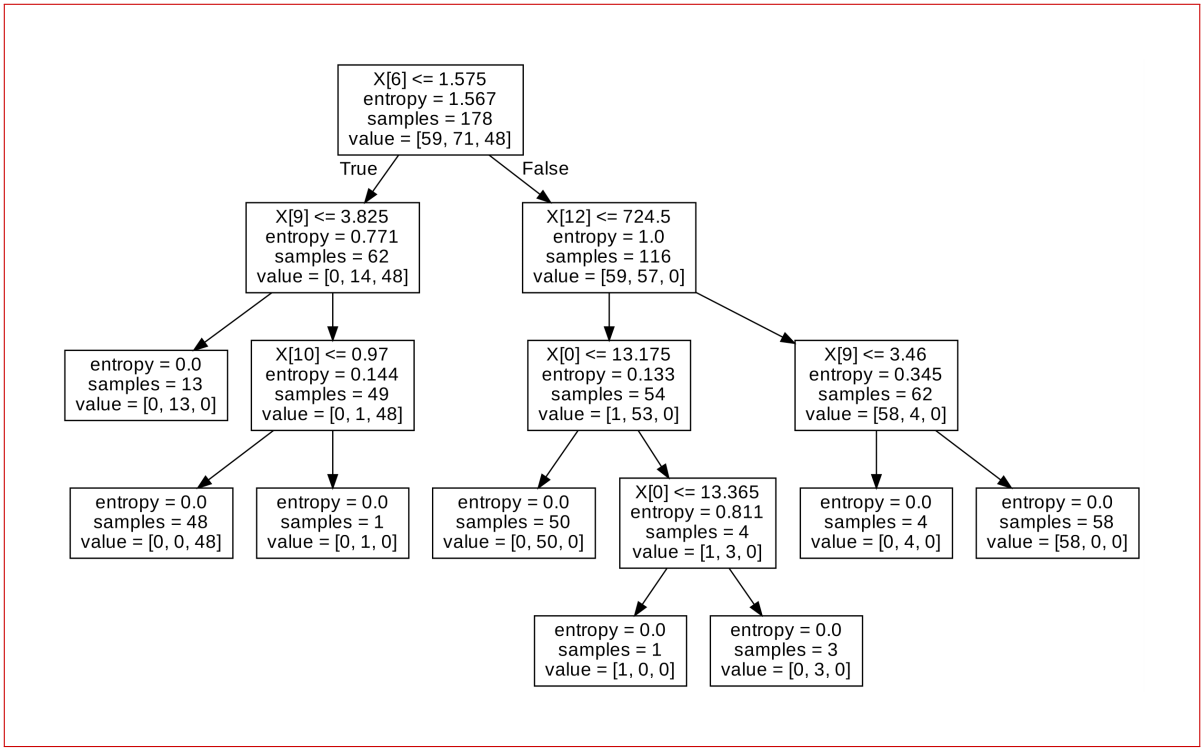


Figura 3: Decision Tree

Salvataggio del Decision Tree in formato pdf

```
import graphviz
dot_data = tree.export_graphviz(clf, out_file=None)
graph = graphviz.Source(dot_data)
graph.render("Vini")
```

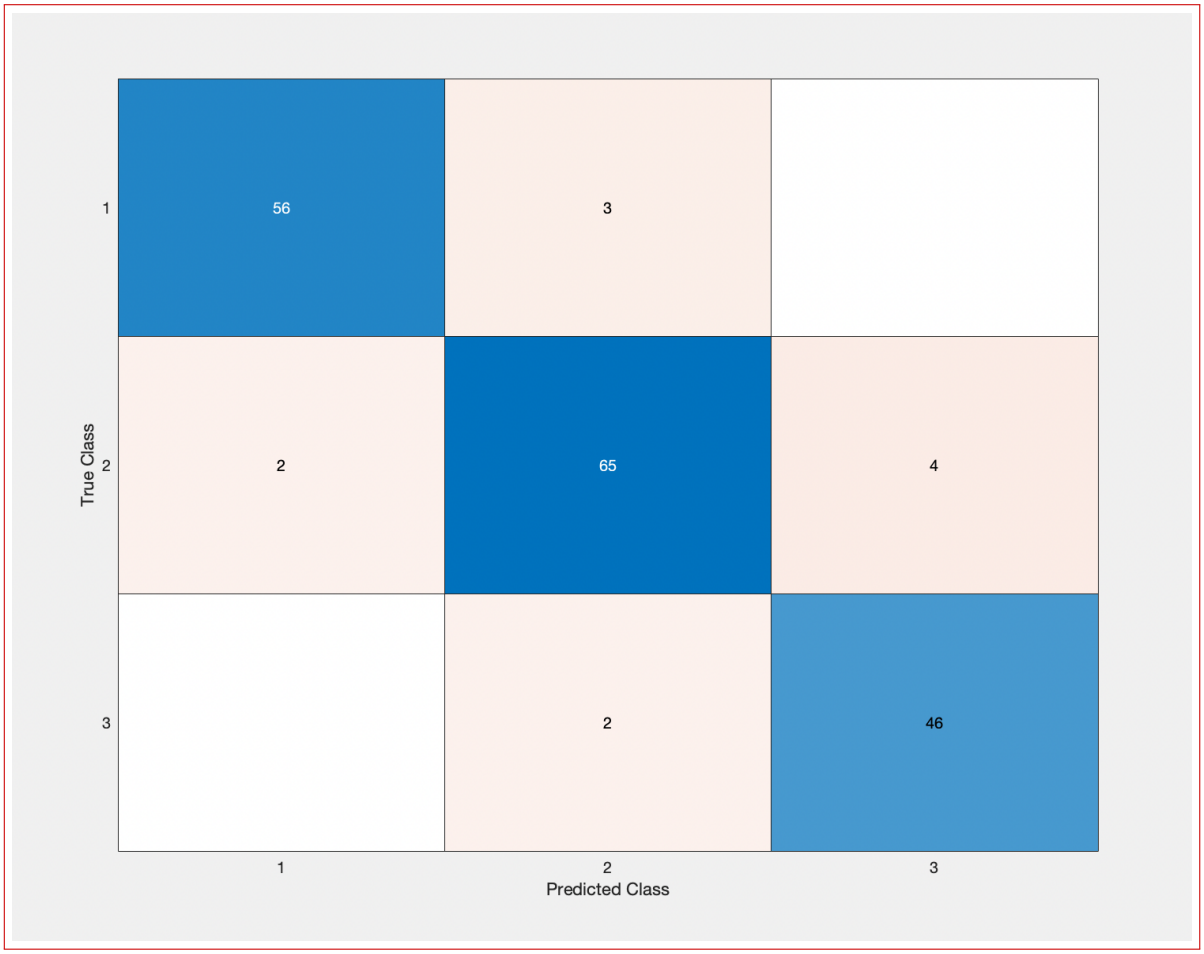


Figura 4: Matrice di confusione in MATLAB

3.2 K-Means

Collegamento a Google Drive

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

Importazioni delle librerie e visualizzazione del dataset

```
from sklearn.datasets import load_wine
import pandas as pd
data = load_wine()
dataset = pd.DataFrame.from_records(data=data.data, columns=data.feature_names)
dataset['label']= data.target
dataset
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline	label
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065.0	0
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050.0	0
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185.0	0
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480.0	0
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
173	13.71	5.65	2.45	20.5	95.0	1.68	0.61	0.52	1.06	7.70	0.64	1.74	740.0	2
174	13.40	3.91	2.48	23.0	102.0	1.80	0.75	0.43	1.41	7.30	0.70	1.56	750.0	2
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	0.43	1.35	10.20	0.59	1.56	835.0	2
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	0.53	1.46	9.30	0.60	1.62	840.0	2
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	0.56	1.35	9.20	0.61	1.60	560.0	2

Figura 5: Tabella dataset

Importazioni delle librerie di supporto

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
import numpy as np
```

Processo K-Means

```
from sklearn.cluster import KMeans
X, y = load_wine(return_X_y=True)
kmeans = KMeans(n_clusters=3,max_iter=400)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
```

Visualizzazione del modello in forma grafica

```
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='plasma')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.7)
```

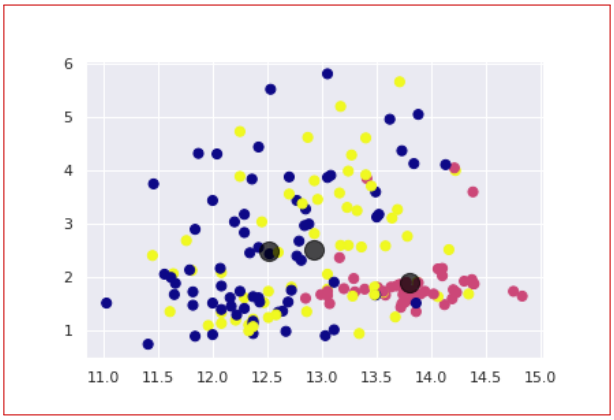


Figura 6: K-Means Plot

centroide n.1: x=12.52, y=2.49
centroide n.2: x=13.8, y=1.88
centroide n.3: x=12.93, y=2.5