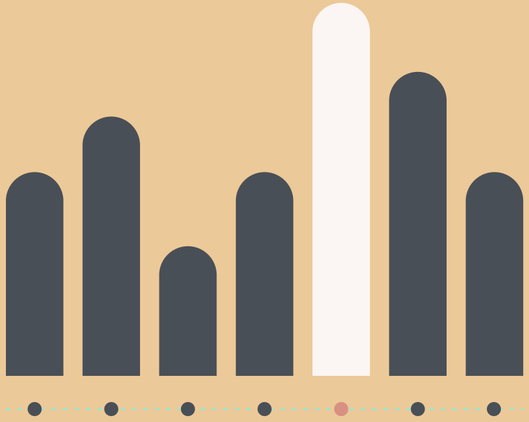
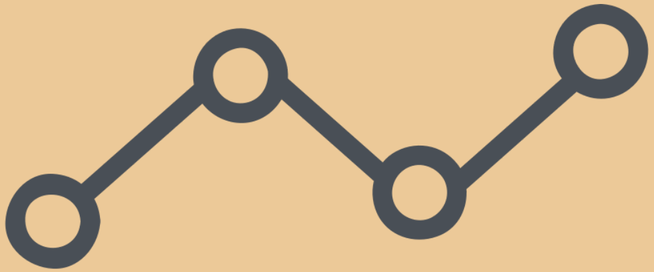
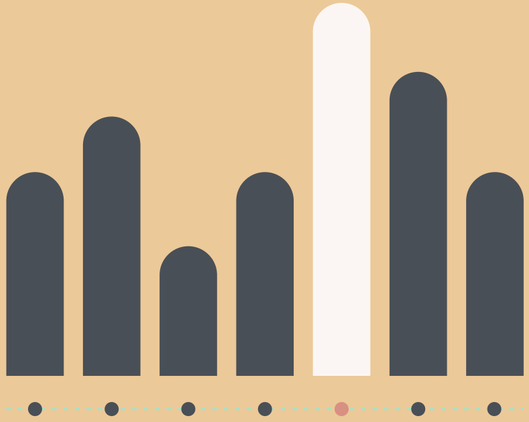
第一組程式報告(SVM)

組員: 魏品華、黃裕權、洪碩廷、陳南天、梅可函、吳晟菖、王璁婕、賴宥晴、鄭樂詩







目標

根據葡萄酒的化學特性將葡萄酒分類成三種不一樣的葡萄酒類型

1

```python
#1.讀檔"wine.data",並放在一個data frame裡面.
df = pd.read_csv("wine.data.orig",
                 names=["class",
                        "Alcohol",
                        "Malicacid",
                        "Ash",
                        "Alcalinity_of_ash",
                        "Magnesium",
                        "Total_phenols",
                        "Flavanoids",
                        "Nonflavanoid_phenols",
                        "Proanthocyanins",
                        "Color_intensity",
                        "Hue",
                        "0D280_0D315_of_diluted_wines",
                        "Proline"])
```

將資料檔讀入並設立每個欄位的名稱
(第一欄為預測的類別.後十三欄為特徵)

特徵介紹:

Alcohol(酒精): 葡萄酒中的酒精含量。
Malic acid(蘋果酸): 葡萄酒中的蘋果酸含量。
Ash(灰分): 葡萄酒中的灰分含量。
Alcalinity of ash(灰的鹼性): 葡萄酒中灰的鹼性。
Magnesium(鎂): 葡萄酒中的鎂含量。
Total phenols(總酚): 葡萄酒中的總酚含量。
Flavanoids(黃酮): 葡萄酒中的黃酮類化合物含量
Nonflavanoid phenols(非黃酮酚): 葡萄酒中的非黃酮類酚含量
Proanthocyanins(原花青素): 葡萄酒中的原花青素含量。
Color intensity(顏色強度): 葡萄酒的顏色強度。
Hue(色調): 葡萄酒的色調。
OD280/OD315 of diluted wines(稀釋葡萄酒的 OD280/OD315): 透過光學密度測量的葡萄酒的稀釋度。
Proline(脯氨酸):葡萄酒中的脯氨酸含量。

2

```
#2.列印前20筆資料
df.head(20)
```

	class	Alcohol	Malicacid	Ash	Alcalinity_of_ash	Magnesium	Total_phenols	Flavanoids	Nonflavanoid_phenols	Proanthocyanins	Color_intensity	Hue	0D280_0D315_of_diluted_wines	Proli
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	10
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	10
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	11
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	14
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	7
5	1	14.20	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.75	1.05	2.85	14
6	1	14.39	1.87	2.45	14.6	96	2.50	2.52	0.30	1.98	5.25	1.02	3.58	12
7	1	14.06	2.15	2.61	17.6	121	2.60	2.51	0.31	1.25	5.05	1.06	3.58	12
8	1	14.83	1.64	2.17	14.0	97	2.80	2.98	0.29	1.98	5.20	1.08	2.85	10
9	1	13.86	1.35	2.27	16.0	98	2.98	3.15	0.22	1.85	7.22	1.01	3.55	10
10	1	14.10	2.16	2.30	18.0	105	2.95	3.32	0.22	2.38	5.75	1.25	3.17	15
11	1	14.12	1.48	2.32	16.8	95	2.20	2.43	0.26	1.57	5.00	1.17	2.82	12
12	1	13.75	1.73	2.41	16.0	89	2.60	2.76	0.29	1.81	5.60	1.15	2.90	13
13	1	14.75	1.73	2.39	11.4	91	3.10	3.69	0.43	2.81	5.40	1.25	2.73	11
14	1	14.38	1.87	2.38	12.0	102	3.30	3.64	0.29	2.96	7.50	1.20	3.00	15
15	1	13.63	1.81	2.70	17.2	112	2.85	2.91	0.30	1.46	7.30	1.28	2.88	13
16	1	14.30	1.92	2.72	20.0	120	2.80	3.14	0.33	1.97	6.20	1.07	2.65	12
17	1	13.83	1.57	2.62	20.0	115	2.95	3.40	0.40	1.72	6.60	1.13	2.57	11
18	1	14.19	1.59	2.48	16.5	108	3.30	3.93	0.32	1.86	8.70	1.23	2.82	16
19	1	13.64	3.10	2.56	15.2	116	2.70	3.03	0.17	1.66	5.10	0.96	3.36	8

(使用.head(20)印出資料集中的前20筆資料)

3

印出資料集的大小，
得知一共有178筆資料，
以及14個欄位

```
#3.列印此data frame 的size
df.shape

(178, 14)
```

將資料進行分割，
train_data佔80%
test_data佔20%

```
#4.將資料隨機分割成訓練集train_set和測試集test_set, 各佔80%和20%
from sklearn.model_selection import train_test_split
train_set, test_set = train_test_split(df, test_size=0.2, random_state=1)
```

4

提取出train_data和test_data各自的特徵和標籤

```
#5.對兩集合分別取第0行當class label, 第1~13行當資料特徵
X_train = train_set.iloc[:, 1:14]
Y_train = train_set["class"]
X_test = test_set.iloc[:, 1:14]
Y_test = test_set["class"]
```

匯入不同機器學習分類器的函式庫

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn import svm
```

5

```python
classifiers = [
    KNeighborsClassifier(1),
    DecisionTreeClassifier(max_depth=2),
    LogisticRegression(C=10**10),
    LinearDiscriminantAnalysis(),
    RandomForestClassifier(),
    MLPClassifier(),
    GaussianNB(),
    svm.SVC(),
    svm.SVC(kernel='poly', C=0.1, degree=2)
]

results_df = pd.DataFrame(columns=["Classifier", "Training Accuracy", "Testing Accuracy"])

#6.分別用以下方法訓練分類器並列印出訓練集的辨識率和測試集的辨識率
for clf in classifiers:
    # 訓練分類器
    clf.fit(X_train, Y_train)

    #訓練集和測試集的辨識率
    train_accuracy = clf.score(X_train, Y_train)
    test_accuracy = clf.score(X_test, Y_test)

    print(f"\n分類器：{clf}")
    print(f"訓練集辨識率：{train_accuracy:.4f}")
    print(f"測試集辨識率：{test_accuracy:.4f}")

    results_df = results_df.append({
        "Classifier": str(clf),
        "Training Accuracy": train_accuracy,
        "Testing Accuracy": test_accuracy
    }, ignore_index=True)
```

首先先建立一個包含不同機器學習分類器的列表，
然後以迴圈的方式訓練每個分類器(.fit)並印出
它們各自訓練集的辨識率和測試集的辨識率

最後建立了一個results_df來存取每個分
類器的辨識率方便於後面視覺化

ConvergenceWarning: Stochastic Optimizer:
Maximum iterations (200) reached and the
optimization hasn't converged yet.

我們發現在我們使用MLPClassifier()和
LogisticRegression時因為迭代次數太少
所以導致優化過程尚未達到收斂，
因此我們可能需要增加他它們的max_iter

5 印出results_df

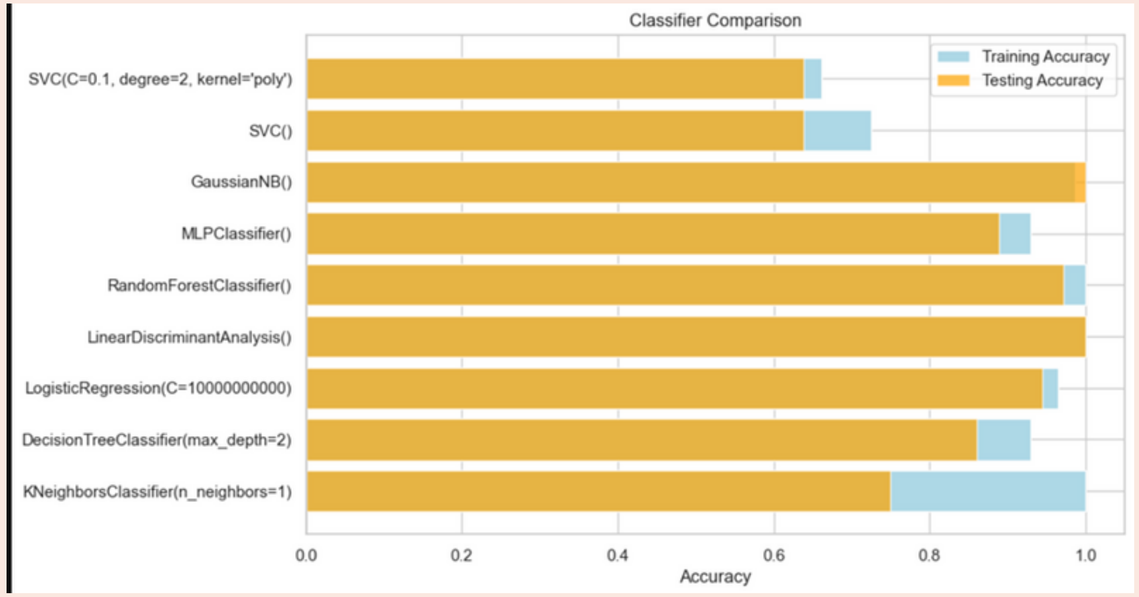
```
results_df
```

	Classifier	Training Accuracy	Testing Accuracy
0	KNeighborsClassifier(n_neighbors=1)	1.000000	0.750000
1	DecisionTreeClassifier(max_depth=2)	0.929577	0.861111
2	LogisticRegression(C=10000000000)	0.964789	0.944444
3	LinearDiscriminantAnalysis()	1.000000	1.000000
4	RandomForestClassifier()	1.000000	0.972222
5	MLPClassifier()	0.929577	0.888889
6	GaussianNB()	0.985915	1.000000
7	SVC()	0.725352	0.638889
8	SVC(C=0.1, degree=2, kernel='poly')	0.661972	0.638889

6

將每個分類器訓練集的辨識率和測試集的辨識率進行比較

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6)) #創建圖表 寬度10單位、高度6單位。
plt.title("Classifier Comparison")
plt.barh(results_df["Classifier"], results_df["Training Accuracy"], color="lightblue", label="Training Accuracy")
plt.barh(results_df["Classifier"], results_df["Testing Accuracy"], color="orange", label="Testing Accuracy", alpha=0.7) #alpha=0.7 設置條形的透明度
plt.xlabel("Accuracy")
plt.legend(loc="upper right")
plt.show()
```



結論與心得

從結果可以看出，不同的分類器在處理同一個資料集時的表現會有所差異
我們必須要依據不同的情況選擇表現相對較好的分類器
又或說可能會需要對一些辨識率較低的模型進行進一步參數的調整來達到更好的訓練效果

謝謝聆聽~