Creado por:

Isabel Maniega

Linear Discriminant Analysis (LDA)

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [3]:
```
dataset = pd.read_csv("Wine.csv")
dataset
```

Out[3]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Proanthocyanins	Color_Intensi
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.6
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.3
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.6
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.8
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.3
...	...	...	...	...	...	...	...	...	...	
173	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	1.06	7.7
174	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	1.41	7.3
175	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	1.35	10.2
176	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	1.46	9.3
177	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	1.35	9.2

178 rows × 14 columns

In [4]:
```
dataset.shape
```

Out[4]:
```
(178, 14)
```

In [5]:
```
X = dataset.iloc[:, 0:13].values
y = dataset.iloc[:, 13].values
```

In [6]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

In [7]:
```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [8]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA

lda = LDA(n_components=2)
X_train = lda.fit_transform(X_train, y_train)
X_test = lda.transform(X_test)
```

In [9]:
```
principal_Df = pd.DataFrame(data = X_train
             , columns = ['principal component 1', 'principal component 2'])
principal_Df
```

Out[9]:

	principal component 1	principal component 2
0	3.573156	1.940189
1	0.854759	-2.081830
2	0.621737	-3.062345
3	4.807864	2.006387
4	-3.857976	0.149873
...	...	...
137	1.686471	-3.834276
138	-0.902058	-2.629893
139	-0.191056	-3.660171
140	-4.206327	0.831072
141	4.529108	3.078393

142 rows × 2 columns

In [10]:
```
from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)
```

Out[10]:
```
▾       LogisticRegression
LogisticRegression(random_state=0)
```

In [11]:
```
y_pred = classifier.predict(X_test)
y_pred
```

Out[11]:
```
array([1, 3, 2, 1, 2, 2, 1, 3, 2, 2, 3, 3, 1, 2, 3, 2, 1, 1, 2, 1, 2, 1,
       1, 2, 2, 2, 2, 2, 2, 3, 1, 1, 2, 1, 1, 1])
```

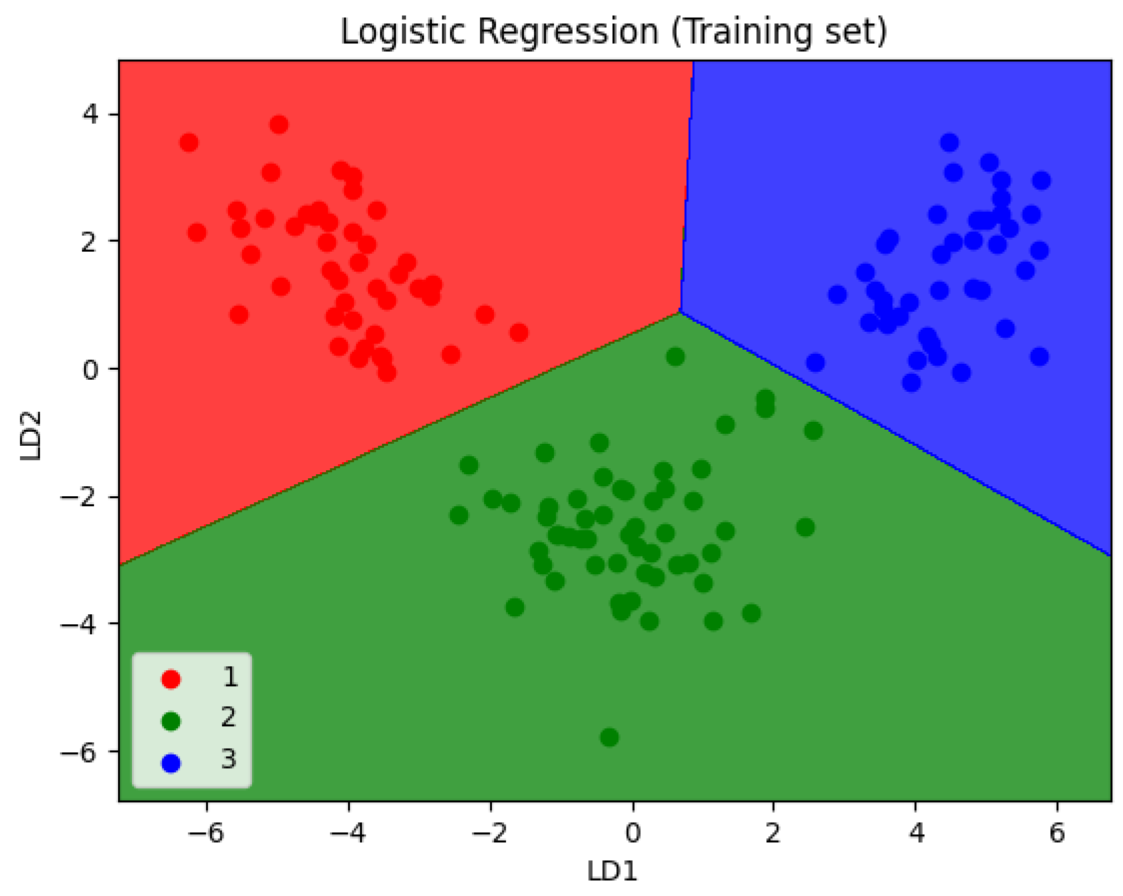
In [12]:
```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)
cm
```

Out[12]:
```
array([[14,  0,  0],
       [ 0, 16,  0],
       [ 0,  0,  6]])
```

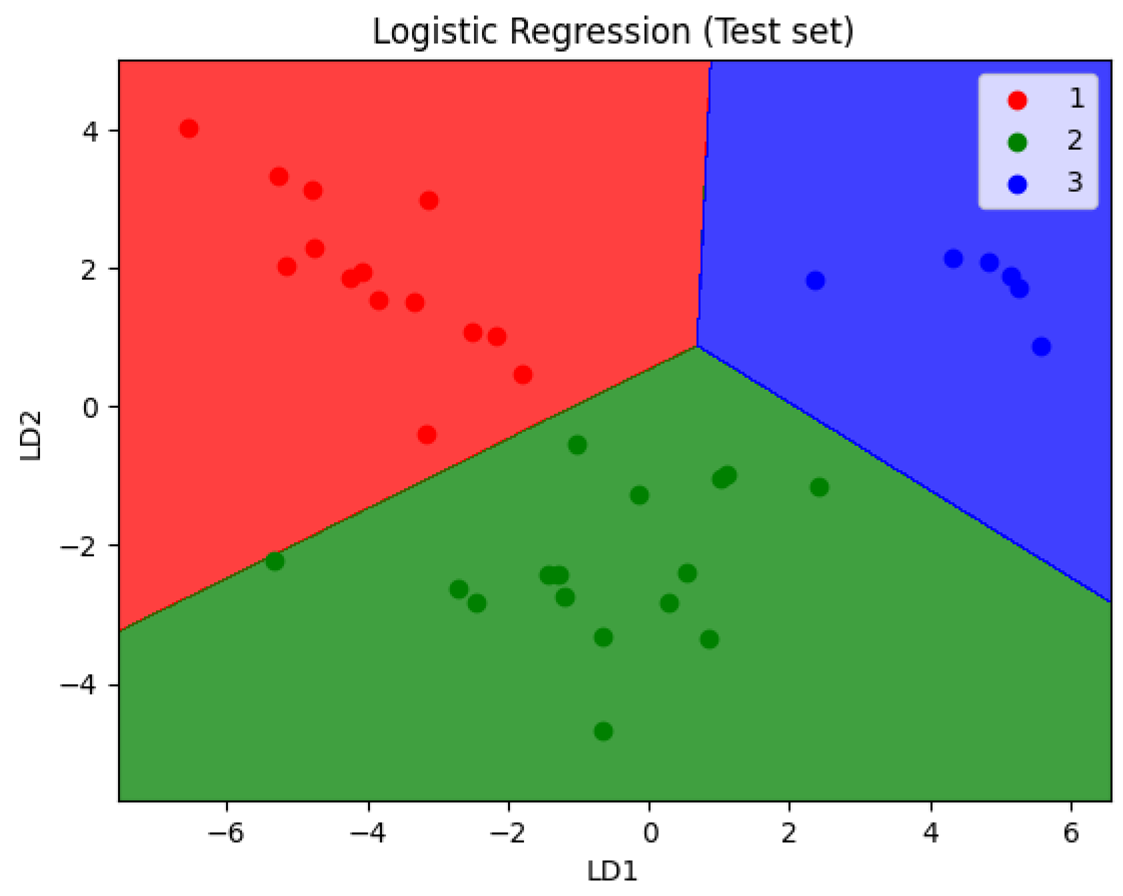
In [13]:
```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green', 'blue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green', 'blue'))(i), label = j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('LD1')
plt.ylabel('LD2')
plt.legend()
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will ha
ve precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a
2D array with a single row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will ha
ve precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a
2D array with a single row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will ha
ve precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a
2D array with a single row if you intend to specify the same RGB or RGBA value for all points.
```



In [14]:
```
# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green', 'blue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green', 'blue'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('LD1')
plt.ylabel('LD2')
plt.legend()
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will ha
ve precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a
2D array with a single row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will ha
ve precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a
2D array with a single row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will ha
ve precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a
2D array with a single row if you intend to specify the same RGB or RGBA value for all points.
```



Creado por:

Isabel Maniega