Linear Discriminant Analysis (LDA)

Importing the libraries

In [0]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

In [0]:
```
dataset = pd.read_csv('Wine.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

Feature Scaling

In [0]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
```

Splitting the dataset into the Training set and Test set

In [0]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
```

Applying LDA

In [0]:
```
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)
```

Training the Logistic Regression model on the Training set

In [6]:
```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

Out[6]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Predicting the Test set results

In [0]:
```
y_pred = classifier.predict(X_test)
```

Making the Confusion Matrix

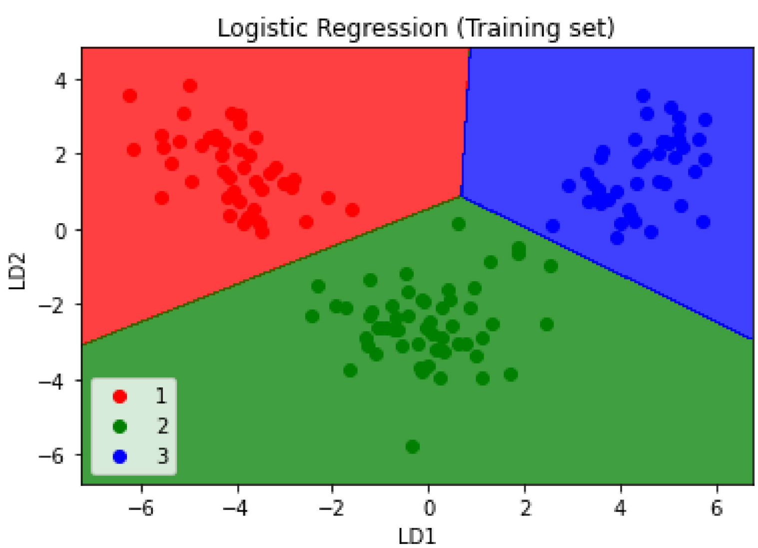
In [8]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[14  0  0]
 [ 0 16  0]
 [ 0  0  6]]
```

Visualising the Training set results

In [9]:
```
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[
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).'
             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 shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
```



Visualising the Test set results

In [10]:
```
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[
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).
             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 shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with 'x' & 'y'.  Please use a 2-D array with a single row if you re
ally want to specify the same RGB or RGBA value for all points.
```