In [1]:
```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from pandas import DataFrame

from sklearn import datasets

from sklearn.mixture import GaussianMixture
```

In [2]:
```
iris = datasets.load_iris()

X = iris.data[:, :2]

d = pd.DataFrame(X)
plt.grid()
plt.scatter(d[0], d[1])
plt.title("Before Clustering")
plt.xlabel('sepallength')
plt.ylabel('sepalwidth')
```

Out[2]: Text(0, 0.5, 'sepalwidth')

In [3]:
```
gmm = GaussianMixture(n_components = 3)

gmm.fit(d)

labels = gmm.predict(d)

d['labels']= labels

d0 = d[d['labels']== 0]

d1 = d[d['labels']== 1]

d2 = d[d['labels']== 2]
plt.grid()

plt.scatter(d0[0], d0[1], c ='r')

plt.scatter(d1[0], d1[1], c ='yellow')

plt.scatter(d2[0], d2[1], c ='g')
plt.title("After Clustering using E-M clustering")
plt.xlabel('sepallength')
plt.ylabel('sepalwidth')
```

Out[3]: Text(0, 0.5, 'sepalwidth')

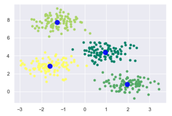
In [4]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
import numpy as np
from sklearn.cluster import KMeans
```

In [5]:
```
from sklearn.datasets.samples_generator import make_blobs
X, y_true = make_blobs(n_samples=400, centers=4, cluster_std=0.60, random_state=0)
```

In [6]:
```
plt.scatter(X[:, 0], X[:, 1], s=20);
plt.show()
```

In [7]:
```
kmeans = KMeans(n_clusters=4)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
```

In [8]:
```
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=20, cmap='summer')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='blue', s=100, alpha=0.9);
plt.show()
```

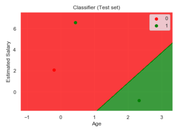


In [10]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Salary_Data.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)

# Training the Simple Linear Regression model on the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predicting the Test set results
y_pred = regressor.predict(X_test)

# Visualising the Training set results
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()

# Visualising the Test set results
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Test set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

In [ ]:
```
```

In [11]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
dataset = pd.read_csv('Book1.csv')
# input
x = dataset.iloc[:, [2, 3]].values

# output
y = dataset.iloc[:, 4].values
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.25, random_state = 0)
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(xtrain)
xtest = sc_x.transform(xtest)

print (xtrain[0:10, :])
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state=0)
classifier.fit(xtrain, ytrain)
y_pred = classifier.predict(xtest)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(ytest, y_pred)

print ("Confusion Matrix : \n", cm)
from sklearn.metrics import accuracy_score
print ("Accuracy : ", accuracy_score(ytest, y_pred))
from matplotlib.colors import ListedColormap
X_set, y_set = xtest, ytest
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,stop = X_set[:, 0].max() + 1, step = 0.01),np.arange(st
art = 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')))

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'))(i), label = j)


plt.title('Classifier (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
C:\Users\pc\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be c
hanged to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have pre
cedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you really want to sp
ecify 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 have pre
cedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you really want to sp
ecify the same RGB or RGBA value for all points.

[[ 2.07689895 -0.9345079 ]
 [-0.32276132  0.0941231 ]
 [ 0.81391986  1.42529262]
 [ 0.81391986 -1.29755413]
 [-0.44905923 -0.51095396]
 [-0.32276132  0.63869245]
 [-1.20684669  0.94123097]
 [-1.20684669 -1.35806183]
 [-0.19646341  1.00173868]]
Confusion Matrix :
 [[1 0]
 [1 1]]
Accuracy :  0.6666666666666666
```



In [ ]:
```
```