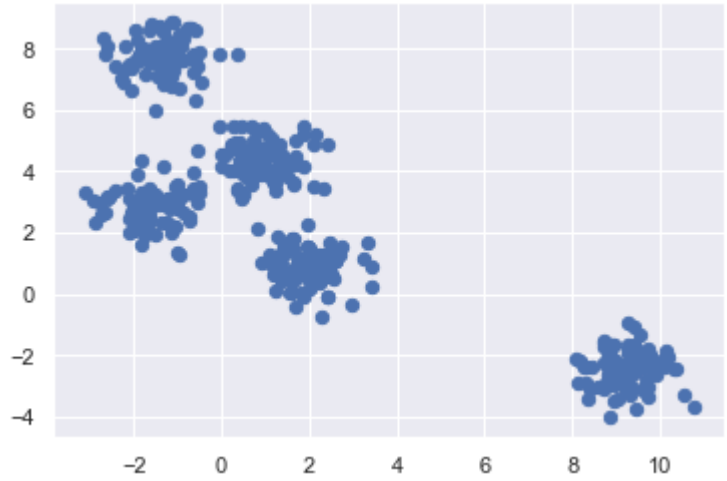
In [1]:
```
#importing required libraries
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set(color_codes=True)
```

In [3]:
```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples = 400, centers = 5, cluster_std = 0.60, random_state = 0)
```

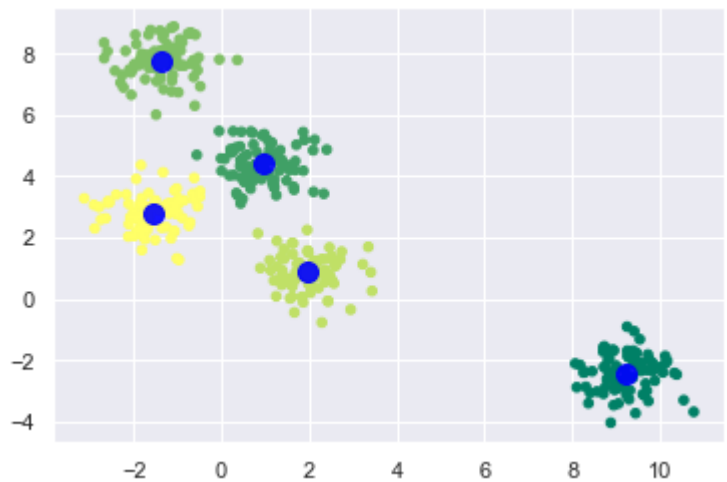
In [4]:
```
plt.scatter(X[:, 0], X[:, 1], s = 40);
plt.show()
```



In [6]:
```
#make an object of kmeans providing number of clusters,train the model and do the pre
kmeans = KMeans(n_clusters = 5)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
```

In [7]:
```
#plot and visualize the cluster's centers picked my kmeans Python estimator
X, y = make_blobs(n_samples = 400, centers = 5, cluster_std = 0.60, random_state = 0)
```

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 [9]:
```
#load digit dataset from sklearn and make an object of it
#dataset is having 1797 samples with 64 features
from sklearn.datasets import load_digits
digit = load_digits()
digit.data.shape
```

Out[9]:
```
(1797, 64)
```

In [11]:
```
#checking its type
type(digit.data)
```

Out[11]:
```
numpy.ndarray
```

In [15]:
```
#using kmeans creating 10 clusters with 64 features
kmeans = KMeans(n_clusters = 10, random_state = 0)
clusters = kmeans.fit_predict(digit.data)
kmeans.cluster_centers_.shape
```

Out[15]:
```
(10, 64)
```

In [33]:
```
fig, ax = plt.subplots(2, 5, figsize=(8, 3))
centers = kmeans.cluster_centers_.reshape(10, 8, 8)
for axi, center in zip(ax.flat, centers):
    axi.set(xticks=[], yticks=[])
    axi.imshow(center, interpolation='nearest', cmap=plt.cm.binary)
```

In [70]:
```
from scipy.stats import mode
label = np.zeros_like(clusters)
for i in range(10):
    mask = (clusters == i)
    label[mask] = mode(digit.target[mask])[0]
```

In [72]:
```
from sklearn.metrics import accuracy_score
accuracy_score(digit.target, label)
```

Out[72]:
```
0.7935447968836951
```

In [ ]:
```

```