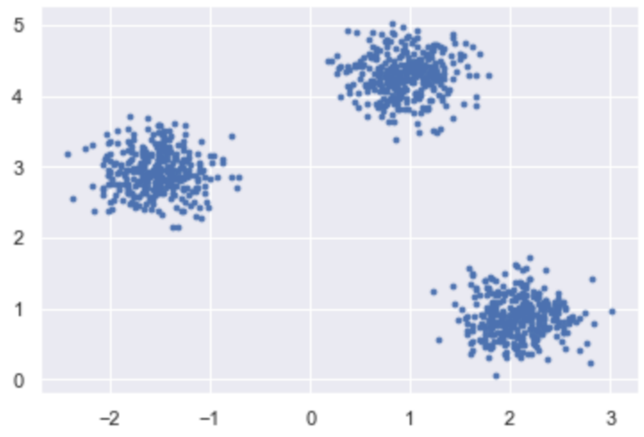
K-means

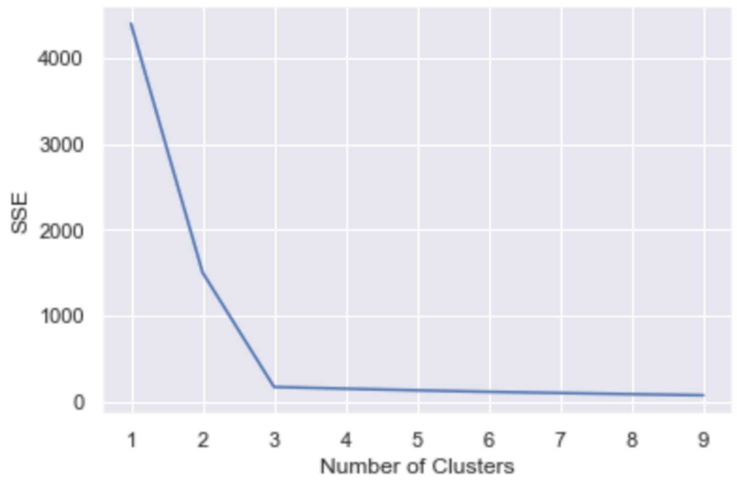
Create a simple dataset use the following code:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.datasets.samples_generator import make_blobs
X, y = make_blobs(n_samples=1000, centers=3,
                       cluster_std=0.30, random_state=0)
plt.scatter(X[:, 0], X[:, 1], marker='.',s=30);
```



Define different K for K-means clusters and compute the SSE:

```
# A list holds the SSE values for each k
from sklearn.cluster import KMeans
SSE = []
for k in range(1,10):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(X,y)
    SSE.append(kmeans.inertia_)

plt.plot(range(1,10), SSE)
plt.xticks(range(1, 10))
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.show()
```



To determine the elbow point in the SSE curve, use:

```
from kneed import KneeLocator
kl = KneeLocator(range(1, 10),SSE, curve="convex", direction="decreasing")
kl.elbow
```

```
3
```

Hierarchical Clustering

Create a dataset use the following code:

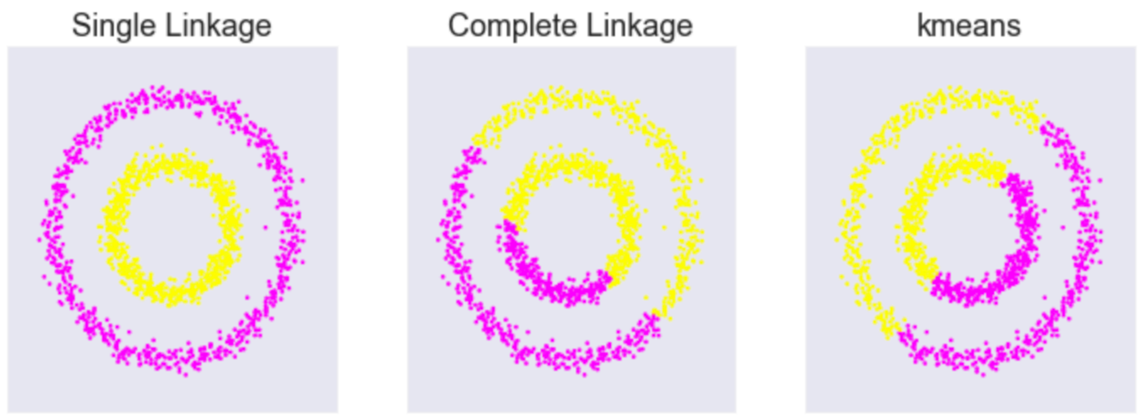
```
import numpy as np
import matplotlib.pyplot as plt

from sklearn import cluster, datasets
from sklearn.preprocessing import StandardScaler


n_samples = 1500
# create 2 datasets
noisy_circles = datasets.make_circles(n_samples=n_samples, factor=.5,noise=.05)
```

Create different clusters for dataset:

```
X, y = noisy_circles

# normalize dataset for easier parameter selection
X = StandardScaler().fit_transform(X)

# Create cluster objects
complete = cluster.AgglomerativeClustering(n_clusters=2, linkage='complete')
single = cluster.AgglomerativeClustering(n_clusters=2, linkage='single')
kmeans = KMeans(n_clusters=2)
```

Fit the model and visualize the result:

```
clustering_algorithms = (
    ('Single Linkage', single),
    ('Complete Linkage', complete),
    ('kmeans', kmeans),
)
plot_num=1
plt.figure(figsize=(12, 4))
for name, algorithm in clustering_algorithms:
    print(name)
    algorithm.fit(X)
    y_pred = algorithm.labels_.astype(int)

    plt.subplot(1, 3, plot_num)
    plt.title(name, size=18)
    plt.scatter(X[:, 0], X[:, 1], s=10,marker='.', c=y_pred, cmap='spring')
    plt.xlim(-2.5, 2.5)
    plt.ylim(-2.5, 2.5)
    plt.xticks(())
    plt.yticks(())
    plot_num += 1

plt.show()
```


Single Linkage
Complete Linkage
kmeans