```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

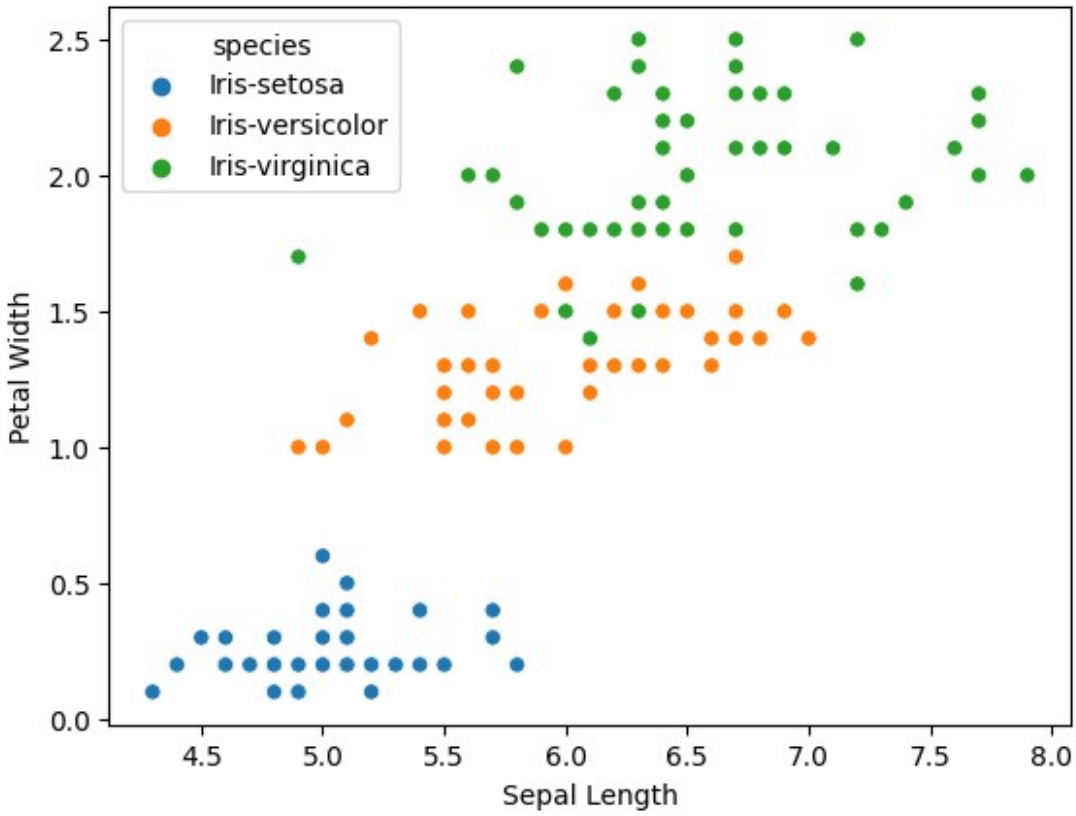
Preparing Data

```
data = pd.read_csv("datas/Iris.csv",index_col=0)
data.columns = ["Sepal Length","Sepal Width","Petal Length","Petal
Width","species"]
df = data

df.head(3)
```

```
    Sepal Length  Sepal Width  Petal Length  Petal Width      species
Id
1            5.1          3.5           1.4          0.2  Iris-setosa
2            4.9          3.0           1.4          0.2  Iris-setosa
3            4.7          3.2           1.3          0.2  Iris-setosa
```

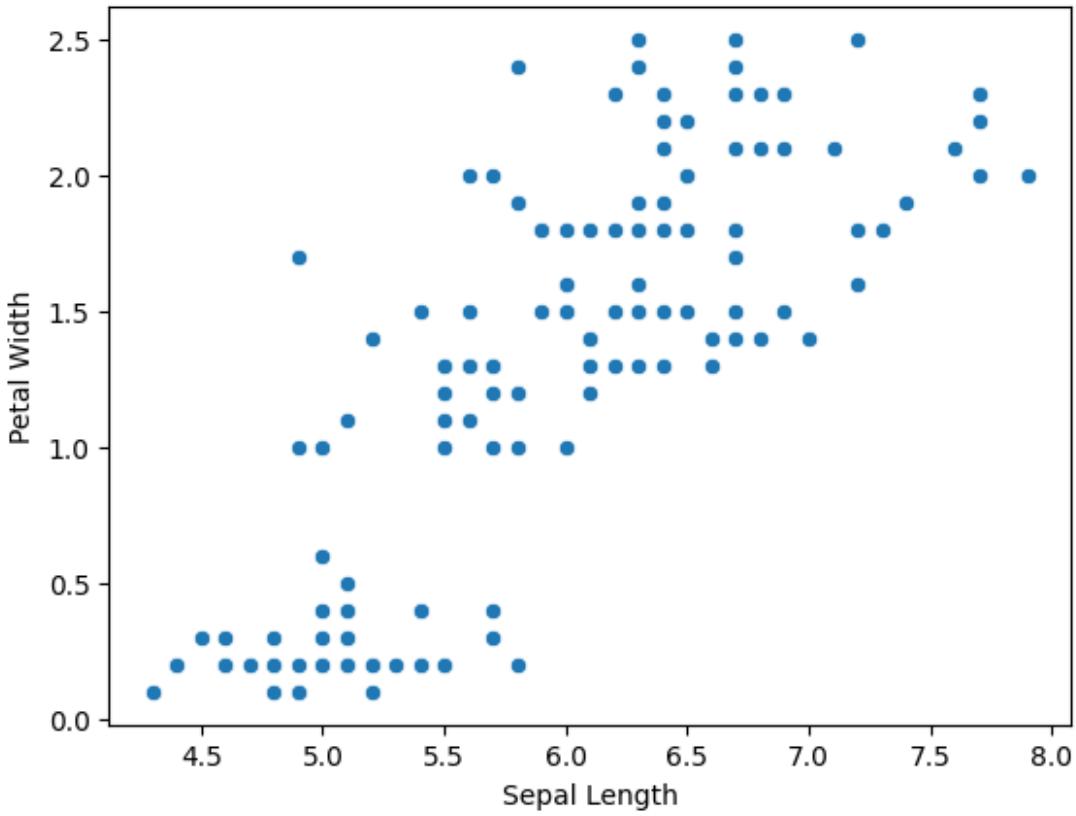
```
sns.scatterplot(data=df,x='Sepal Length',y='Petal
Width',hue='species')
```

```
<Axes: xlabel='Sepal Length', ylabel='Petal Width'>
```




```
X = df[['Sepal Length','Petal Width']]
sns.scatterplot(x=X['Sepal Length'],y=X['Petal Width'])
```

```
<Axes: xlabel='Sepal Length', ylabel='Petal Width'>
```



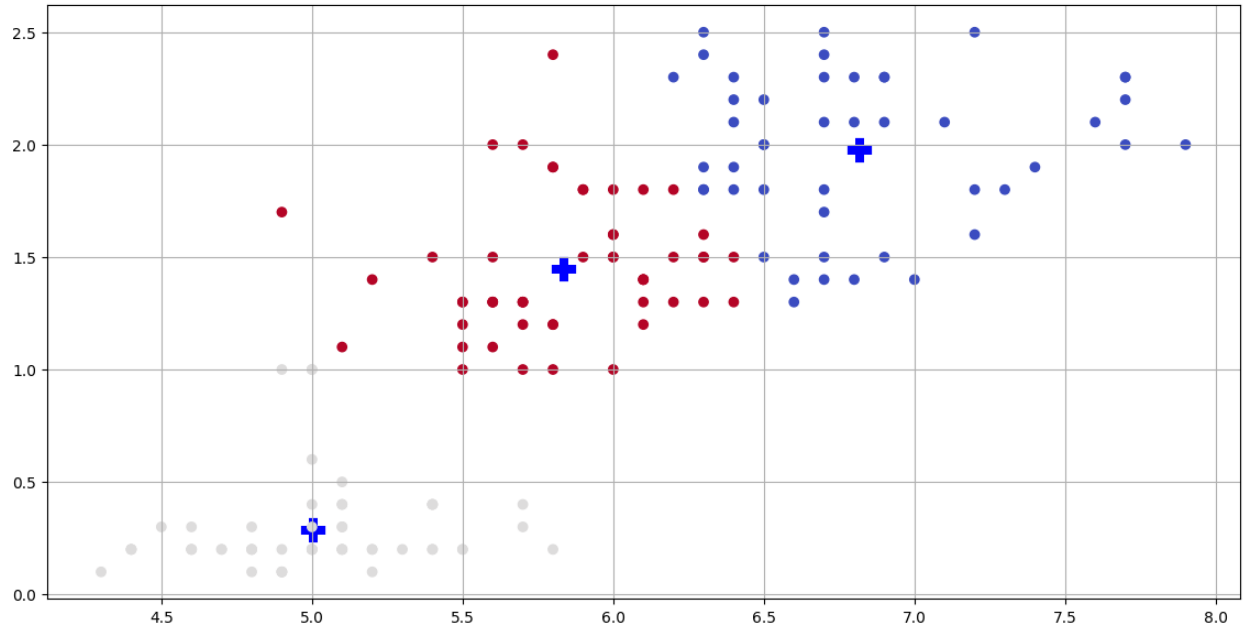
Model

```
class KMeans:
    def __init__(self,n_cluster=1):
        self.k = n_cluster
    def _find_closest_centroids(self,X,centroids):
        n = len(X)
        k = self.k
        index = np.zeros(n)
        for i in range(0,n):
            distances = []
            for j in range(0,k):

                dist = np.linalg.norm(X[i] - centroids[j])
                distances.append(dist)
            index[i] = np.argmin(distances)

        return index
    def _compute_centroid(self,X,index):
        centroids = np.zeros((self.k,X.shape[1]))
        for k in range(0,self.k):
            cluster = X[index == k]
            centroids[k] = np.mean(cluster,axis=0)
```

```
        return centroids

    def _random_initialize_centroids(self,X,K):
        indices = np.random.choice(X.shape[0], size=K, replace=False)
        centroids = X[indices]
        return centroids

    def fit(self,X,max_iter=10):

        initial_centroids =
self._random_initialize_centroids(X,self.k)
        index = np.zeros(X.shape[0])
        centroids = initial_centroids

        for i in range(max_iter):
            index=self._find_closest_centroids(X,centroids)
            centroids=self._compute_centroid(X,index)

        self.index=index
        self.centroids = centroids

def plot_cluster(X,index,centroids):
    plt.figure(figsize=(14, 7))
    plt.scatter(x=centroids[:, 0], y=centroids[:, 1], marker="+",
c='b', linewidth=16)
    plt.scatter(X[:, 0], X[:, 1], c=index, cmap='coolwarm')
    plt.grid()
    plt.savefig("K-Means_iris")
    plt.show()
```

Model Evaluation

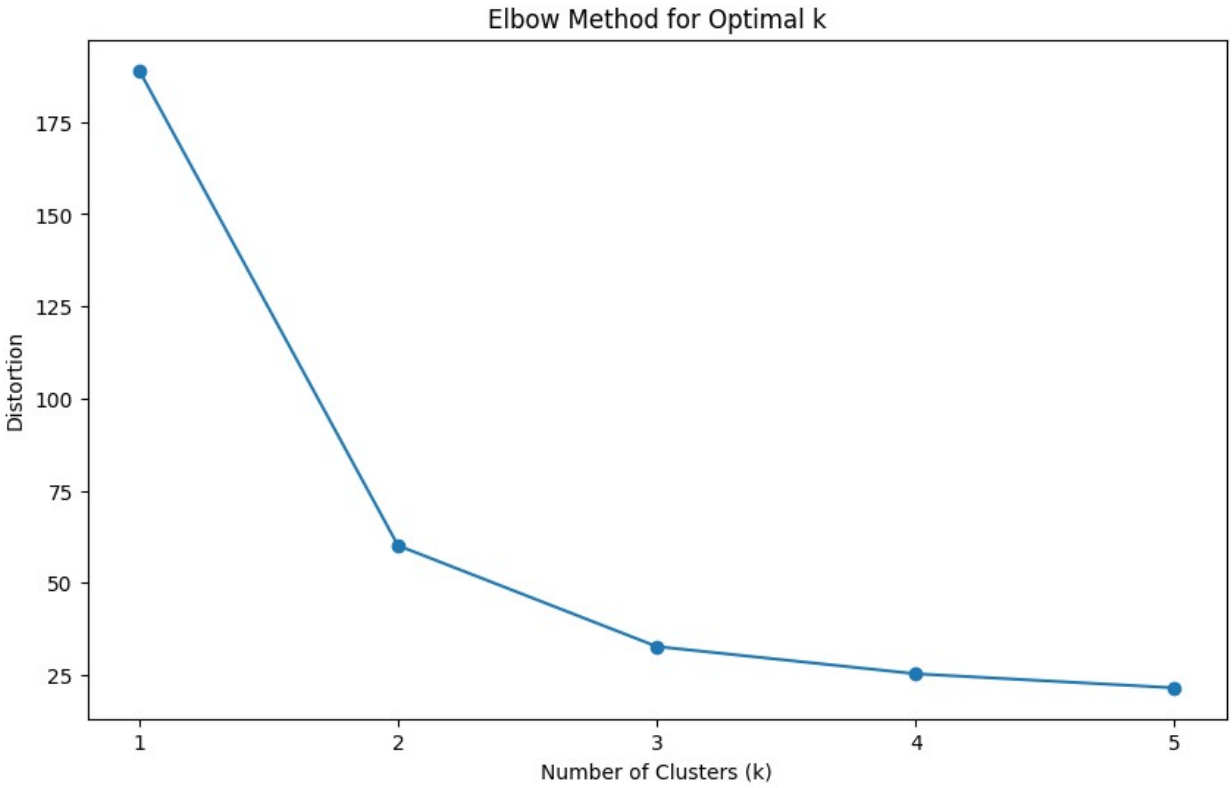
```
# initial_centroids = np.array([[5.0,0.3],[6.5,1.5],[7.8,2.4]])
model = KMeans(3)
model.fit(X.values,10)

plot_cluster(X.values,model.index,model.centroids)
```




```
from sklearn.metrics import silhouette_score

def calculate_distortion(X, centroids, cluster_indices):
    distortion = 0

    for i in range(0,centroids.shape[0]):
        cluster_points = X[cluster_indices == i]
        centroid = centroids[i]
        distortion += np.sum(np.linalg.norm(cluster_points - centroid,
axis=1)**2)
    return distortion

def elbow_method(X, max_k, max_iter):
    distortions = []
    silhouette_scores =[]
    for k in range(1, max_k +1 ):
        model_kmeans = KMeans(k)
        model_kmeans.fit(X, max_iter)

        cluster_indices, centroids =
model_kmeans.index,model_kmeans.centroids
        distortion = calculate_distortion(X, centroids,
cluster_indices)
        distortions.append(distortion)
        if k > 1:  # Silhouette Score requires at least 2 clusters
            silhouette = silhouette_score(X, cluster_indices)
            silhouette_scores.append(silhouette)
            print(f"At k = {k} distortion is {distortion}and silhoutee
score is {silhouette}")
    # Plot the elbow graph
```

```
    plt.figure(figsize=(10, 6))
    plt.plot(range(1, max_k + 1), distortions, marker='o')
    plt.title("Elbow Method for Optimal k")
    plt.xlabel("Number of Clusters (k)")
    plt.ylabel("Distortion")
    plt.xticks(range(1, max_k + 1))
    plt.show()
    return distortions,silhouette_scores

distortions,silhouette = elbow_method(X.values,5,10)
```

```
At k = 2 distortion is 60.13669920795727and silhoutee score is
0.5705593986178453
At k = 3 distortion is 32.76801587301587and silhoutee score is
0.5051309106461155
At k = 4 distortion is 25.380192158385093and silhoutee score is
0.4457401981463308
At k = 5 distortion is 21.57256938490981and silhoutee score is
0.40381998058843466
```



Using SK-Learn

```
from sklearn.cluster import KMeans as sklKMeans

k = 3
model_1 =
sklKMeans(n_clusters=k,init='random',n_init='auto',max_iter=10).fit(X.
values)

print("Distortion Value
",calculate_distortion(X,model_1.cluster_centers_,model_1.labels_))
print("Silhouette ",silhouette_score(X.values,model_1.labels_))
```

```
Distortion Value  32.757081714581716
Silhouette  0.5038084350159749
```