Assignment

1. Knn algorithm

Knn 알고리즘은 주어진 데이터를 분류하는 알고리즘 중에서도 가장 간단한 알고리즘이다.

새로운 데이터가 들어왔을 때 그 데이터가 어떤 분류에 속할지 결정하는 것이 핵심으로

이번의 예제는 sklearn의 iris(붓꽃속)라는 데이터를 이용해서

4개의 feature를 가진 데이터를 Iris Versicolor , Iris Setosa , Iris Virginica 세종류로 구분한다.

전체 코드

```
from sklearn.datasets import load_iris
iris = load_iris()

from sklearn.model_selection import train_test_split

X = iris.data
y = iris.target

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size =
0.2,random_state=4)
print(X_train.shape)
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

knn = KNeighborsClassifier(n_neighbors=6)
knn.fit(X_train,y_train)

y_pred = knn.predict(X_test)
scores = metrics.accuracy_score(y_test,y_pred)
print(scores)
```

y_pred 와 y_test 의 값을 비교하면 knn의 학습을 통해 예측의 정확도가 어느 정도인지 알 수 있다.

```
[2 0 2 2 2 1 2 0 0 2 0 0 0 1 2 0 1 0 0 2 0 2 1 0 0 0 0 0 0 2]
[2 0 2 2 2 1 1 0 0 2 0 0 0 1 2 0 1 0 0 2 0 2 1 0 0 0 0 0 0 2]
```

(위쪽이 y_pred , 아래가 y_test)

결론적으로 약 96.7%의 예측 정답률을 가지게 된다.

2. Advanced knn

1번의 예시는 knn 알고리즘을 사용하며 데이터 사이의 distance값을 가중치로 고려하지 않은 예측방식으로 이번에는 데이터를 분석할 때 distance 값을 데이터 구별에 사용할 것이다.

제공된 코드는 KneighborsClassifier 에 weights를 제공하는 파트가 빠져 있어 목표의 동작을 하기 어려울 것 이기에 임의로 코드를 수정하였다.

```
knn = KNeighborsClassifier(n_neighbors=5,weights="distance")
```

전체코드

```
from sklearn.datasets import load_iris
iris = load_iris()

from sklearn.model_selection import train_test_split

X = iris.data
y = iris.target

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size =
0.2,random_state=4)
print(X_train.shape)
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

knn = KNeighborsClassifier(n_neighbors=5,weights="distance")
knn.fit(X,y)

classes = {0:'setosa', 1:'versicolor', 2:'virginica'}

x_new = [[3,4,5,2],[5,4,2,2]]
y_predict = knn.predict(x_new)
print(classes[y_predict[0]])
print(classes[y_predict[1]])
```

이번 코드에서는 train과 test를 구분하지 않고 모든 데이터를 이용해 학습한다.

그후 knn을 이용해 데이터를 fit 한 후,

임의의 데이터를 이용해 예측의 결과를 확인하는 방식으로 예제를 진행하였다.

임의의 데이터는 기존의 데이터들과 같이 4개의 feature를 가지고 잇다. `[3,4,5,2]`

해당 데이터를 이용해 이러한 feature를 가진 붓꽃속이 어느 부류에 속하는지 알 수 잇다.

`classes = {0:'setosa', 1:'versicolor', 2:'virginica'}`

결과

```
(120, 4)
versicolor
setosa
```

3. K-means

이번 예제에서는 K-means algorithm를 이용하여 어떠한 데이터들을 clustering 하여 군집화 할 것이다. Knn과 유사해 보일 수 있으나 이 알고리즘은 비지도 방식에 적용이 가능하다. (학습시 라벨이 없는 데이터로도 학습이 가능하다)

Sklearn을 이용하여 쉽게 적용할 수 있었다.

이번 예제에는 matplotlib.pyplot 을 이용해서 실행의 결과를 그림으로 확인하는 것이 목표에 있었으나 코드가 누락되어 예제만으로는 결과를 확인할 수 없기에 다음의 코드를 추가하였다.

```
plt.show()
```

전체코드

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans

X = np.array([[6,3],[11,15],[17,12],[24,10],[20,25],[22,30],[85,70],[71,81],[60,79],[56,52],[81,91],[80,81]])

plt.scatter(X[:,0],X[:,1])

kmeans = KMeans(n_clusters = 2)
kmeans.fit(X)

print(kmeans.cluster_centers_)
print(kmeans.labels_)
plt.scatter(X[:,0],X[:,1],c=kmeans.labels_,cmap='rainbow')

plt.show()
```

단순히 KMeans() 함수를 통해 실행이 가능하며 몇 개의 군집으로 분류할 것인지는 직접 입력이 가능하다.

결과

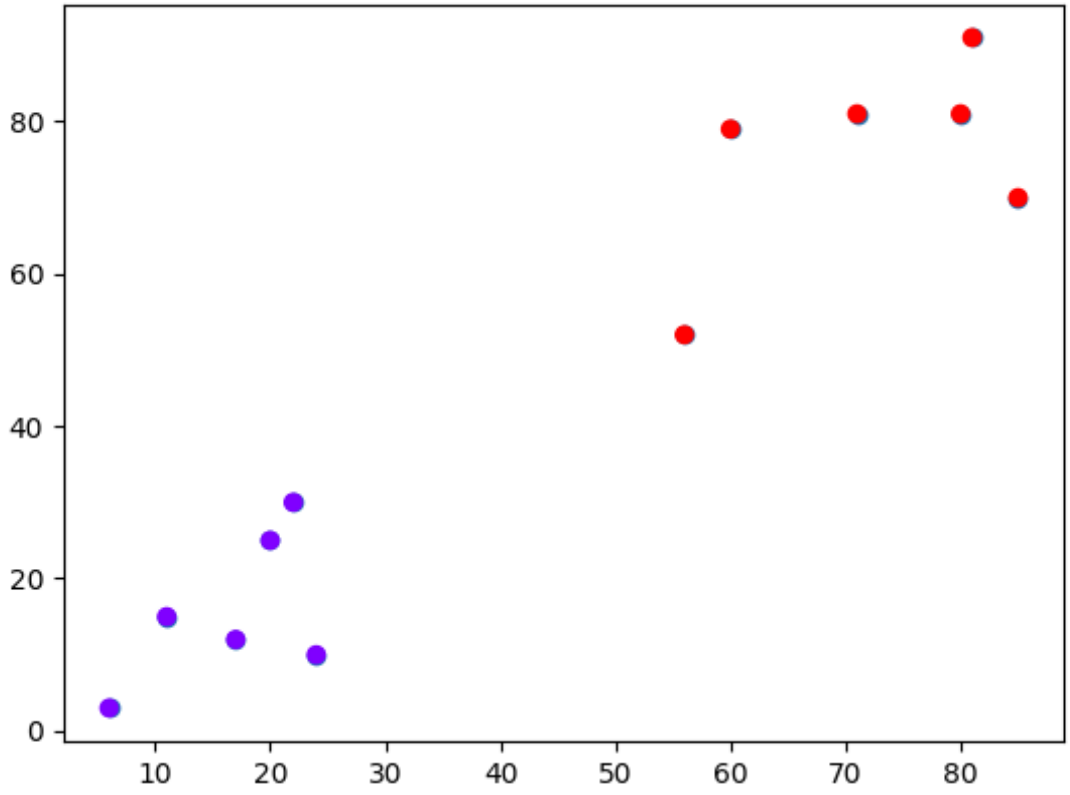
```
[[16.66666667 15.83333333]
 [72.16666667 75.66666667]]
```

각 군집의 중심은 [16.6666,15.83333] 과 [72.16666,75.66666] 이다

```
[0 0 0 0 0 0 1 1 1 1 1 1]
```

입력에 따른 군집화는 다음과 같다.

이를 그림으로 표현하면 이렇게 표현된다.



4. Elbow method

Knn을 이용하여 데이터를 군집화 할 때 군집의 개수를 두고 비교를 한 그래프를 통해

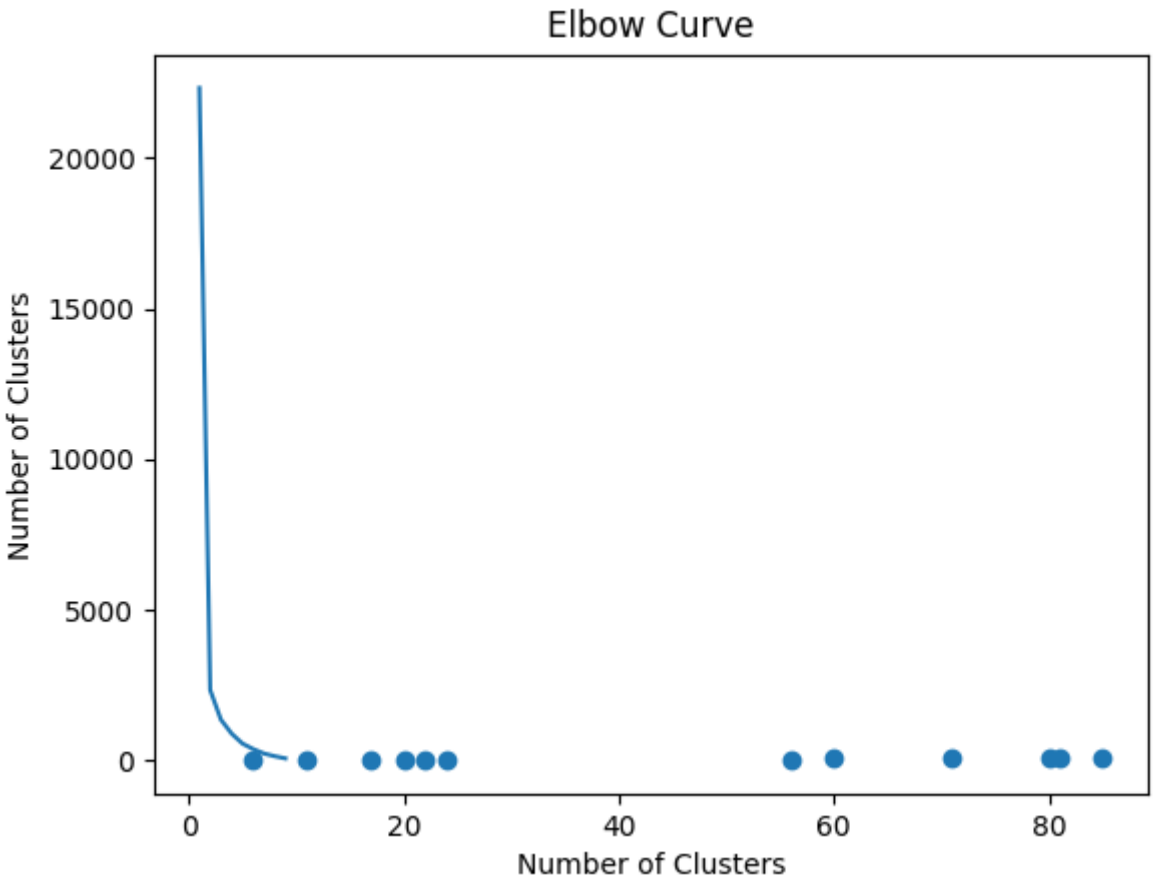
군집이 많아질수록 SSE(sum of squred error)즉 군집 정확도가 높아짐을 알 수 있다.

우리는 이를 분석하여 완만한 경사를 보이는 군집 수를 목표로 두어 데이터를 더 정밀하게 분석할 수 잇다.

전체 코드

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans

X =
np.array([[6,3],[11,15],[17,12],[24,10],[20,25],[22,30],[85,70],[71,81],[60
,79],[56,52],[81,91],[80,81]])

plt.scatter(X[:,0],X[:,1])


n_clusters = range(1,10)
kmeans = [KMeans(n_clusters=i) for i in n_clusters]
score = [kmeans[i].fit(X).inertia_ for i in range(len(kmeans))]

plt.plot(n_clusters, score)
plt.xlabel('Number of Clusters')
plt.ylabel('Number of Clusters')
plt.title('Elbow Curve')
plt.show()
```

결과



데이터의 총 수가 12 개이기 때문에 군집의 개수는 12개 이상으로 늘어나지 않으며 12개의 군집을 가진 경우의 SSE 는 0이다.

코드상 군집의 최대개수는 9 이며 이때의 SSE 값은 매우 작다

내 생각에는 군집의 개수를 5 이상으로만 설정하면 전체적인 오차율에 큰 문제가 없으므로

프로그램의 실용성을 생각하면 군집의 개수를 5로 설정하는 것이 좋아 보인다.