CSE4020

LAB-5

Routu Venkat Sai Sampath
19BCE0238

Experiment 1: Adaboost

Algorithm:

1. First assign weights to all the data records. Weights are equal if it’s the first iteration
2. Now start creating stumps with each feature and find its Gini index or Gain based on the type of algorithm that is being used (CART or ID3).
3. Now calculate the amount of say for each stump
4. Now calculate the performance of the stump. Values will be in between 0 and 1, where 0 means horrible.
5. Now select the best stump and wrong predictions will be given more weight this time.
6. Now we need to make a new dataset to see if the errors decreased or not. For this

   $$New\ sample\ weight\ =\ old\ weight\ *\ e^{\pm Amount\ of\ say\ (\alpha)}$$

   we will remove the “sample weights” and “new sample weights” column and then based on the “new sample weights” we will divide our data points into buckets.
7. Selects random numbers from 0-1. Since incorrectly classified records have higher sample weights, the probability to select those records is very high.
8. This will be the new dataset and we will iterate over it till the error reduces.

In [2]:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
X = load_iris().data
y = load_iris().target
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7, random_s
```

In [3]:

```
from sklearn.ensemble import AdaBoostClassifier

abclassifier = AdaBoostClassifier(n_estimators=50, learning_rate=1, random_state=2)
model = abclassifier.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

In [4]:

```
from sklearn.metrics import accuracy_score
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9777777777777777
```

In [ ]:

```

```

Experiment 2: K Means Clustering

Algorithm:

1. Select the number K to decide the number of clusters.
2. Select random K points or centroids. (It can be other from the input dataset).
3. Assign each data point to their closest centroid, which will form the predefined K clusters.
4. Calculate the variance and place a new centroid of each cluster.
5. Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.
6. If any reassignment occurs, then go to step-4 else go to FINISH.

In [1]:

```python
from sklearn.datasets import make_blobs
import pandas as pd
```

In [2]:

```python
dataset, classes = make_blobs(n_samples=300, n_features=2, centers=5, cluster_std=0.
df = pd.DataFrame(dataset, columns=['X', 'Y'])
df.head(2)
```

Out[2]:

	X	Y
0	1.569719	-0.838530
1	-3.750696	-4.419262

In [3]:

```python
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=5, init='k-means++', random_state=0).fit(df)
```

In [4]:

```python
print(kmeans.n_iter_) #total number of iterations to convergence
```

```
3
```

In [5]:

```python
print(kmeans.cluster_centers_) #cluster centers
```

```
[[-5.83451299  2.30779804]
 [-1.35495664 -9.43666895]
 [-4.06321324 -4.81843763]
 [ 1.00220205 -1.26008793]
 [-1.56907217 -3.44508562]]
```

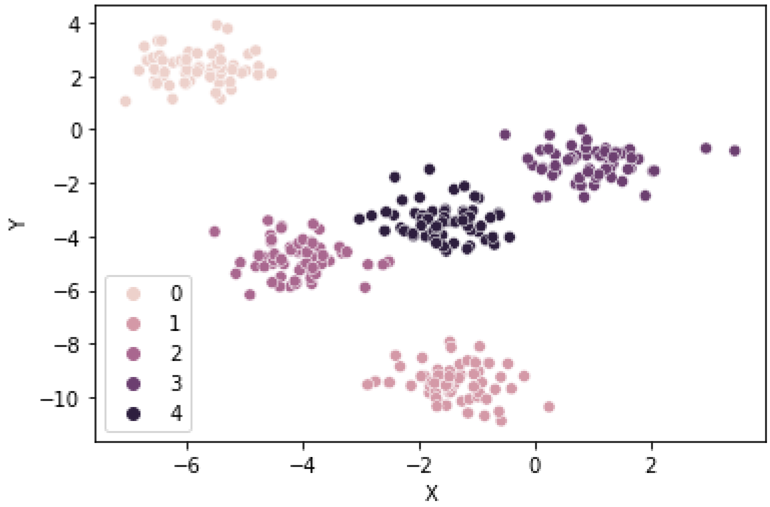
In [6]:

```python
print(kmeans.inertia_) #defines how well the dataset is clustered
```

```
225.49694297568254
```

In [7]:

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.scatterplot(data=df, x="X", y="Y", hue=kmeans.labels_)
plt.show()
```



In [ ]:

Experiment 3: Agglomerative Clustering

Algorithm

1. Make dataset
2. Compute similarity information between every pair of objects in the data set.
3. Using linkage function to group objects into hierarchical cluster tree, based on the distance information generated at step 1. Objects/clusters that are in close proximity are linked together using the linkage function.
4. Determining where to cut the hierarchical tree into clusters. This creates a partition of the data.

In [1]:

```
from sklearn.datasets import make_blobs
import pandas as pd
import numpy as np

dataset, classes = make_blobs(n_samples=300, n_features=2, centers=5, cluster_std=0.
df = pd.DataFrame(dataset, columns=['x1', 'x2'])
X = df.values
x1 = df['x1']
x2 = df['x2']
```

In [2]:

```
from sklearn.cluster import AgglomerativeClustering
ac = AgglomerativeClustering(n_clusters=5)
model = ac.fit(X)
```

In [3]:

```
print(model.n_clusters)
print(model.n_leaves_)
print(model.n_features_in_)
```

```
5
300
2
```

In [4]:

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.scatterplot(x1,x2, hue = model.labels_)
```

```
/Users/sampathroutu/opt/anaconda3/lib/python3.8/site-packages/seaborn/
_decorators.py:36: FutureWarning: Pass the following variables as keyw
ord args: x, y. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

Out[4]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ee8c5cf40>
```