The University of Texas at Arlington

MACHINE LEARNING (CSE 6363)

FINAL PROJECT

Hierarchical Clustering for Seed Categorization

Name: Kuldip Rameshbhai Savaliya
UTA ID: - 1001832000

Project’s Objective

- Implement Hierarchical Clustering on the UCI seed dataset to divide it in groups and implement KNN to identify the species.
- Implement agglomerative clustering and KNN.

Dataset link:

https://archive.ics.uci.edu/ml/machine-learning-databases/00236/

Column Descriptions:

1.area A,
2. perimeter P,
3. compactness C = 4*pi*A/P^2,
4. length of kernel,
5. width of kernel,
6. asymmetry coefficient
7. length of kernel groove.

Steps:

- Grouping the seed dataset into n clusters using hierarchical agglomerative clustering.
- Creating new features for the datapoints in the given dataset based on other clusters (using cluster's centroid) and added them into the existing features of the dataset. Generate new features based on distance.
- Similar procedure is followed for testing dataset.
- New Features are scaled using min max.
- Now, training dataset have extra feature values that is calculated based on similarity between data points.
- Implement KNN classifier on the modified dataset to identify the cluster ids.
- Performance of the model on unknown data is measured by the K-fold cross validation method. (Dataset is split into k subsets)
- Find the suitable number of clusters for different single linkages, complete linkage, and average linkage.
- Make Predictions on unlabeled data.

Code Structure

- Implement Hierarchical clustering (agglomerative clustering) model to divide the training dataset into n clusters.

  class AgglomerativeClustering:
    - __init__ - Initialization
    - fit_predict – fit data and predicting labels
    - clustertolabels – for clusters to labels
    - d_matrix – distance matrix
    - update_d – update distance matrix according to linkage
    - update_cluster – merge cluster
    - distance – distance between two points

- Implement KNN for classification of labels for unlabeled dataset.

  def KNN_classification (sample, k, X, y):

  dist_cartesian – calculate cartesian distance
  lbl_classify – find label
  KNN_classification - classification

- Implement k-fold cross validation method for estimating the performance of the model.

  def splits_CV(dataset_KNN, folds):
    - returns the accuracy for each iteration in the range of k folds.

  def CV_split (dataset, folds):
    - dataset splits with respect to k-fold

- Code Block for the value of clusters, neighbours, and linkage.

  - n_clusters = [3,4,5,6,7]
  - knn = [3,5,7,9,11]
  - linkages = ['single', 'complete', 'average']

- Code block is used to make predictions on test data.
- Run KNN on given dataset (without adding new features) and find the accuracy.

Results

- Experiments with different values of clusters, linkage, and neighbours
- n_clusters = [3,4,5,6,7]
- knn = [3,5,7,9,11]
- linkages = ['single', 'complete', 'average']

    - result = k_Fold (train_set, 8, k)

- Accuracy of KNN is calculated based on mean of the accuracies of the folds found in list. (Steps repeated for different values of k)

- Average accuracy is calculate based on mean of the accuracies of different neighbours. (Steps repeated for different number of clusters)

- Best number of clusters find based on the highest accuracy for the linkage.

    - n_clusters[np.argmax(Accuracy)]

Dataset:

```
df.head()
```

	A	P	C	LK	WK	A_Coef	LKG	target
0	15.26	14.84	0.8710	5.763	3.312	2.221	5.220	0
1	14.88	14.57	0.8811	5.554	3.333	1.018	4.956	0
2	14.29	14.09	0.9050	5.291	3.337	2.699	4.825	0
3	13.84	13.94	0.8955	5.324	3.379	2.259	4.805	0
4	16.14	14.99	0.9034	5.658	3.562	1.355	5.175	0

Training data with features:

```
train_set.head()
```

	A	P	C	LK	WK	A_Coef	LKG	cluster_feature_0	cluster_feature_1	cluster_feature_2	labels
0	12.19	13.36	0.8579	5.240	2.909	4.857	5.158	0.007438	0.654347	0.536302	0
1	18.27	16.09	0.8870	6.173	3.651	2.443	6.197	0.471743	0.018546	0.161483	1
2	11.21	13.13	0.8167	5.279	2.687	6.169	5.275	0.063706	0.877293	0.866551	0
3	11.49	13.22	0.8263	5.304	2.695	5.388	5.310	0.029427	0.787032	0.715293	0
4	10.59	12.41	0.8648	4.899	2.787	4.975	4.794	0.058140	1.000000	0.958808	0

Testing data with features:

```
test_set.head()
```

	A	P	C	LK	WK	A_Coef	LKG	cluster_feature_0	cluster_feature_1	cluster_feature_2
0	13.45	14.02	0.8604	5.516	3.065	3.531	5.097	0.018603	0.426412	0.232044
1	14.79	14.52	0.8819	5.545	3.291	2.704	5.111	0.096871	0.265140	0.108825
2	12.10	13.15	0.8793	5.105	2.941	2.201	5.056	0.041805	0.677900	0.461027
3	13.20	13.66	0.8883	5.236	3.232	8.315	5.056	0.211162	0.725741	0.786708
4	15.01	14.76	0.8657	5.789	3.245	1.791	5.001	0.154330	0.256799	0.117094

Output:

Scenario for single Linkage :

No of clusters :3
Accuracy : 97.46666666666668 %
No of clusters :4
Accuracy : 97.73333333333333 %
No of clusters :5
Accuracy : 97.33333333333334 %
No of clusters :6
Accuracy : 94.80000000000001 %
No of clusters :7
Accuracy : 95.06666666666668 %
Best scenario for no of clusters : 4

Scenario for complete Linkage :

No of clusters :3
Accuracy : 99.06666666666668 %

No of clusters :4
Accuracy : 97.2 %
No of clusters :5
Accuracy : 90.93333333333334 %
No of clusters :6
Accuracy : 94.66666666666669 %
No of clusters :7
Accuracy : 91.73333333333335 %
Best scenario for no of clusters : 3

Scenario for average Linkage :

No of clusters :3
Accuracy : 95.86666666666667 %
No of clusters :4
Accuracy : 97.33333333333334 %
No of clusters :5
Accuracy : 93.06666666666666 %
No of clusters :6
Accuracy : 92.53333333333333 %
No of clusters :7
Accuracy : 89.86666666666666 %
Best scenario for no of clusters: 4

From above results, we can see that for single linkage best number of clusters 4 and for complete linkage best number of clusters is 3 and for average linkage best number of clusters is 4.

Comparision between original dataset vs modified dataset:

K value	Modified dataset (no. of cluster 3)	Original dataset
3	0.9821	0.8666
5	0.9761	0.8933
7	0.9821	0.9066
9	0.9761	0.9133
11	0.9702	0.8933

From above results, we can see that after adding new features in the dataset we get better accuracy compare to original dataset.