ML Assignment 2

Active Learning

Group No -27

Manasvi Agarwal- 2017A7PS0542P

Ashish Prabhune- 2017A7PS0231P

Lakshya Kwatra – 2017A7PS0365P

Date – 26th November,2019

Active Learning:

1. Uncertainty Sampling:
2. Query by Committee
3. Clustering

Data

We used a UCI ML repository data- Balance Data

It has three classes:

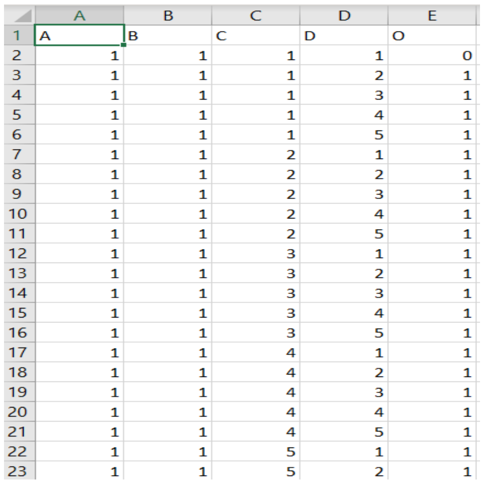
1.Balanced

2.Left

3.Right

It has 600 instances.

We have assumed 150 data points to be labelled and 450 points to be unlabeled.



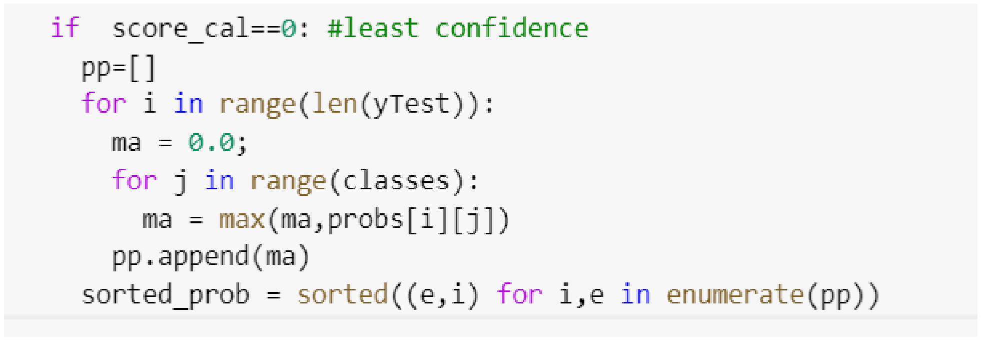
	A	B	C	D	E
1	A	B	C	D	O
2	1	1	1	1	0
3	1	1	1	2	1
4	1	1	1	3	1
5	1	1	1	4	1
6	1	1	1	5	1
7	1	1	2	1	1
8	1	1	2	2	1
9	1	1	2	3	1
10	1	1	2	4	1
11	1	1	2	5	1
12	1	1	3	1	1
13	1	1	3	2	1
14	1	1	3	3	1
15	1	1	3	4	1
16	1	1	3	5	1
17	1	1	4	1	1
18	1	1	4	2	1
19	1	1	4	3	1
20	1	1	4	4	1
21	1	1	4	5	1
22	1	1	5	1	1
23	1	1	5	2	1

1.Uncertainty Sampling:

The data was divided into labelled and unlabeled data.
We used a Logistic Regression model for building a classifier on labelled data.
Then we used three methods for selecting the points to label.

1.Least Confidence

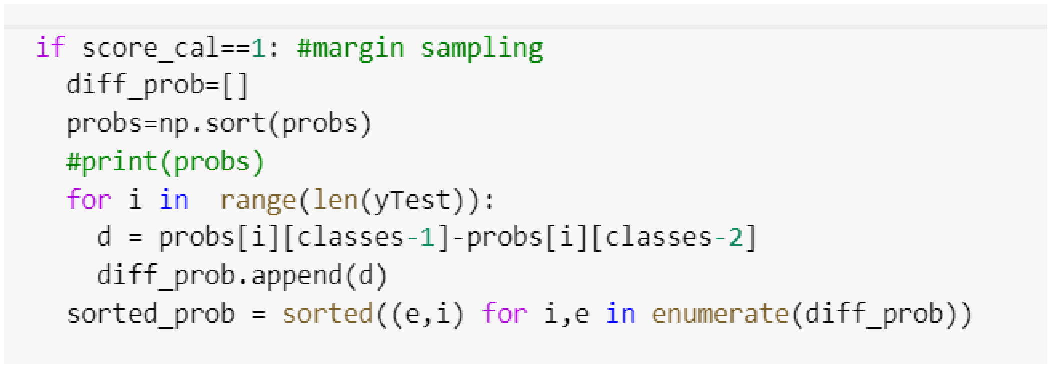
We stored the probability of the most probable class for each data point. Then we sorted the probability in increasing order.



```
if  score_cal==0: #least confidence
  pp=[]
  for i in range(len(yTest)):
    ma = 0.0;
    for j in range(classes):
      ma = max(ma,probs[i][j])
    pp.append(ma)
  sorted_prob = sorted((e,i) for i,e in enumerate(pp))
```

2.Margin Sampling

We sorted the probabilities for each data point and then stored the difference of probabilities of two most probable classes. Then we sorted the probabilities scores remembering the indexes.



```
if score_cal==1: #margin sampling
  diff_prob=[]
  probs=np.sort(probs)
  #print(probs)
  for i in  range(len(yTest)):
    d = probs[i][classes-1]-probs[i][classes-2]
    diff_prob.append(d)
  sorted_prob = sorted((e,i) for i,e in enumerate(diff_prob))
```

3.Entropy method

We calculated the entropy score using probabilities for each data point.

Entropy = S = -sum(pk * log(pk), axis=0).

```
  if score_cal==2: #entropy method
    entropy=[]
    for i in range(len(yTest)):
      x =0.0
      for j in range(classes):
        x += probs[i][j]*math.log(probs[i][j])
      entropy.append(-x)
    #entropy
    sorted_prob =sorted((e,i) for i,e in enumerate(entropy))
```

Then we implemented two approaches for selection of the points which are to be labelled.

1.Stream-based approach

For stream based approach, we set a threshold . Then we selected all the points to be labelled which have scores lesser than the threshold.

```
  if sel_method==0: #stream_based approach
    to_label = []
    threshold = 0.6
    for i in range(len(xTest)):
      if sorted_prob[i][0] <= threshold:
        to_label.append(sorted_prob[i][1])
```

2.Pool based approach

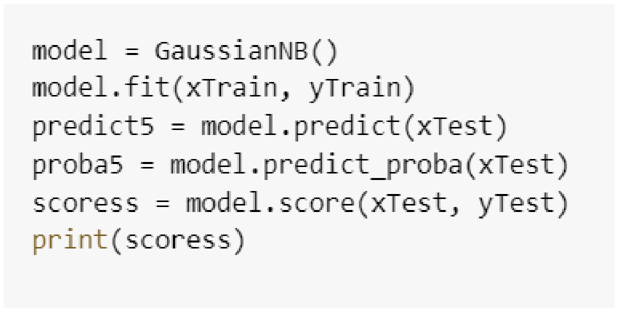
For pool -based approach, we set a parameter that is the number of points to be labelled. Since the list is already sorted, we selected the top k points to be labelled.

```
if sel_method==1: #pool based approach
  to_label=[]
  number_to_label=20
  if number_to_label > len(xTest):
    print('Error')
    #return(0)
  for i in range(number_to_label):
    to_label.append(sorted_prob[i][1])
```

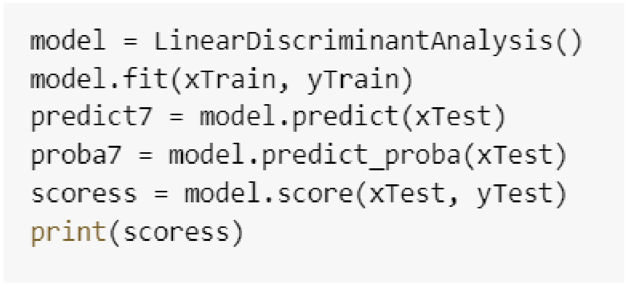
2.Query-by-Committee

We built a committee of models (7) including

- Logistic Regression
- Polynomial Regression(2)
- Naïve-Bayes Classifier
- Gaussian Classifier
- Linear Discriminant Analysis classifier
- Decision Tree classifier



```
model = GaussianNB()
model.fit(xTrain, yTrain)
predict5 = model.predict(xTest)
proba5 = model.predict_proba(xTest)
scoress = model.score(xTest, yTest)
print(scoress)
```



```
model = LinearDiscriminantAnalysis()
model.fit(xTrain, yTrain)
predict7 = model.predict(xTest)
proba7 = model.predict_proba(xTest)
scoress = model.score(xTest, yTest)
print(scoress)
```

Then we calculated the most probable class for each model.

1. Vote-Entropy method:

We calculated the probability of belonging to each class for each data point. Then we calculated the entropy scores for each data point.

Entropy = S = -sum(pk * log(pk), axis=0).

2. KL-Divergence method

We calculated the KL divergence score of each data point using the formula

S = sum(pk * log(pk / qk), axis=0).

Then we used two approaches for selecting the points to be labelled.

1. Stream-based approach

For stream based approach, we set a threshold. Then we selected all the points to be labelled which have scores lesser than the threshold.

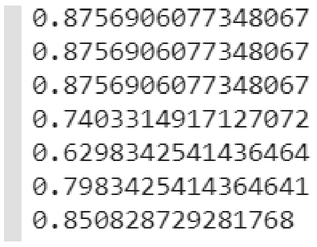
2. <u>Pool based approach</u>

For pool -based approach, we set a parameter that is the number of points to be labelled. Since the list is already sorted, we selected the top k points to be labelled.

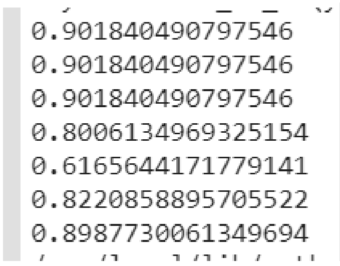
Results:

We run 3 iterations for active learning QBC

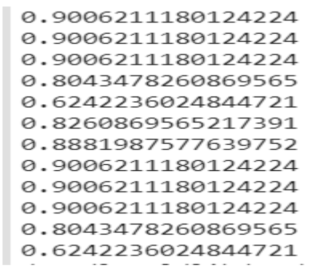
These are the results for 3 iterations for the models.



```
0.8756906077348067
0.8756906077348067
0.8756906077348067
0.7403314917127072
0.6298342541436464
0.7983425414364641
0.850828729281768
```



```
0.901840490797546
0.901840490797546
0.901840490797546
0.8006134969325154
0.6165644171779141
0.8220858895705522
0.8987730061349694
```

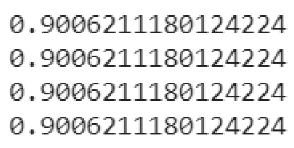


```
0.9006211180124224
0.9006211180124224
0.9006211180124224
0.8043478260869565
0.6242236024844721
0.8260869565217391
0.8881987577639752
0.9006211180124224
0.9006211180124224
0.9006211180124224
0.8043478260869565
0.6242236024844721
```

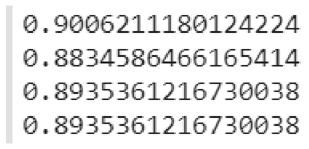
Iteration1 Iteration 2 Iteration 3

.We can clearly see that the model wise average accuracy is increasing with each iteration. This is because with each iteration ,model has more correctly labelled point(by Oracle).

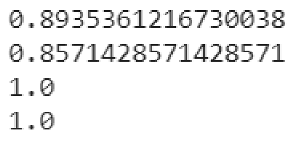
We also checked the accuracy values for Uncertainty sampling on Stream-based approach.



```
0.9006211180124224
0.9006211180124224
0.9006211180124224
0.9006211180124224
```



```
0.9006211180124224
0.8834586466165414
0.8935361216730038
0.8935361216730038
```



```
0.8935361216730038
0.8571428571428571
1.0
1.0
```

Least confidence Margin Sampling Entropy method

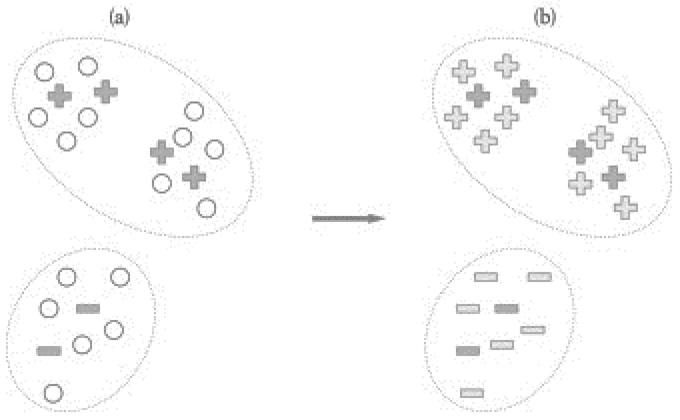
We can observe that entropy method has the best accuracy ,it reaches accuracy of 1 in 3rd iteration as it takes into account all the probabilities.

3. Cluster-based strategy for data labelling, given a limited budget.

Clustering is similar to classification, but the basis is different. In Clustering you don't know what you are looking for, and you are trying to identify some segments or clusters in your data. When you use clustering algorithms on your dataset, unexpected things can suddenly pop up like structures, clusters and groupings you would have never thought of otherwise.

The second active learning regime exploits cluster structure in data. A key advantage of cluster-based heuristics is that the framework can naturally utilise the unlabelled data X_u, as well as optimising the selection of the training data X_l .

Roughly speaking, various cluster-based methods follow a similar framework, introduced by Dasgupta and Hsu. In an ideal scenario, defined, separable clusters will exist that are pure in terms of labels. Following definition by unsupervised learning, a few informative points X_l can be selected from each cluster; any remaining unlabelled points X_u can then be assumed to have their most confident (majority) label — as in the figure below. A supervised classifier can then be trained on the labelled dataset X_L, including queried and propagated labels Y_L, such that $X_L = (X_l \cup X_u, Y_L)$.

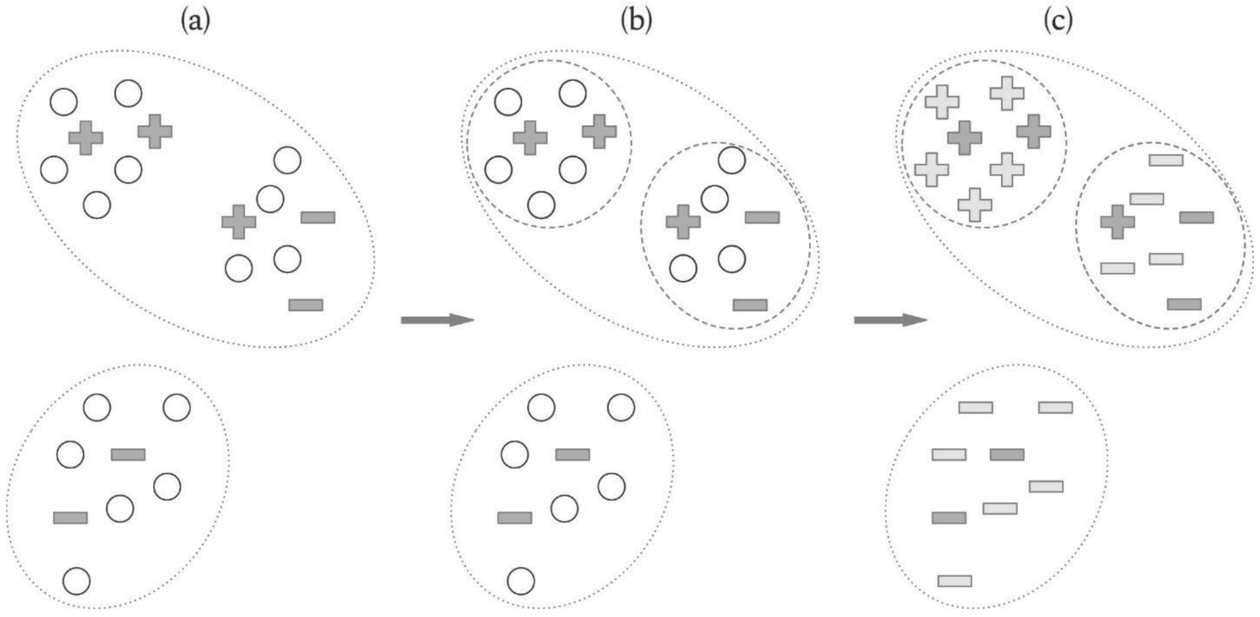


Ideal clusters (separable and pure): (a) clustering of query points [+ ⁄ −] and unlabelled instances [○]; (b) query points and propagated labels (X_L).

The active/guided sampling element of cluster-based techniques is defined by the sampling procedure. Various methods have been proposed. Dasgupta and Hsu suggest a heuristic that favours instances from clusters that appear mixed as querying progresses. Alternatively, the density clustering algorithm, by Wang et al. , favours queries in regions populated by (relatively) dense groups of data.

The relationship between labels and clusters could be insignificant, or there might be viable (near pure) clusters but at many different resolutions. For this reason, the performance of cluster-based methods heavily depends on the quality of the clustering results.

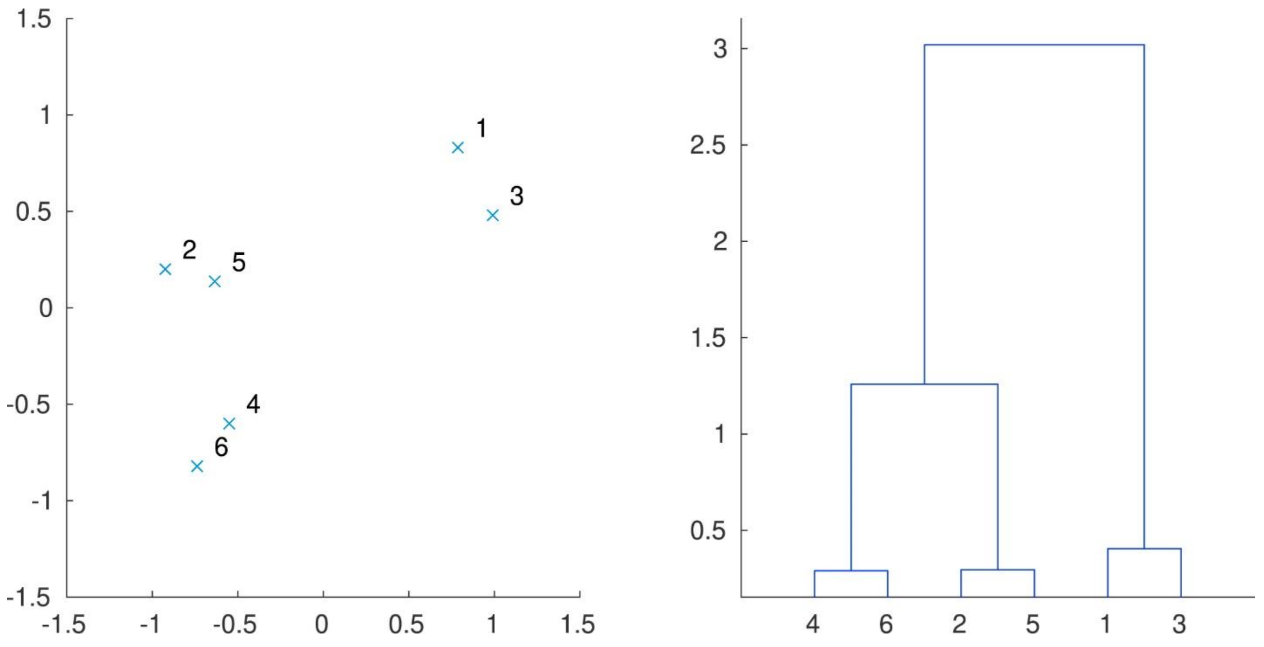
Thus, data clustering must be adaptive — *actively* changing as more information becomes available. Provided that there is some relationship between clustered groups of data and diagnostic labels, at whatever resolution, cluster-based active learning can exploit these patterns.



Algorithm 1. Agglomerative clustering

```
1: *compute* dissimilarity matrix $d$ between all observations in $X$
2: *initialise* clusters as singletons: **for** $i \leftarrow 1$ **to** $N$ **do** $Ci \leftarrow \{i\}$
3: *initialise* set of clusters available for merging: $S \leftarrow \{1, \ldots, N\}$
4: **repeat**
5: Pick the two most similar clusters to merge: $(j, k) \leftarrow \text{argmin}_{j,k \in S}(d_{j,k})$
6: Create new cluster $C_l \leftarrow C_j \cup C_k$
7: Mark $j$ and $k$ as unavailable: $S \leftarrow S \{j, k\}$
```

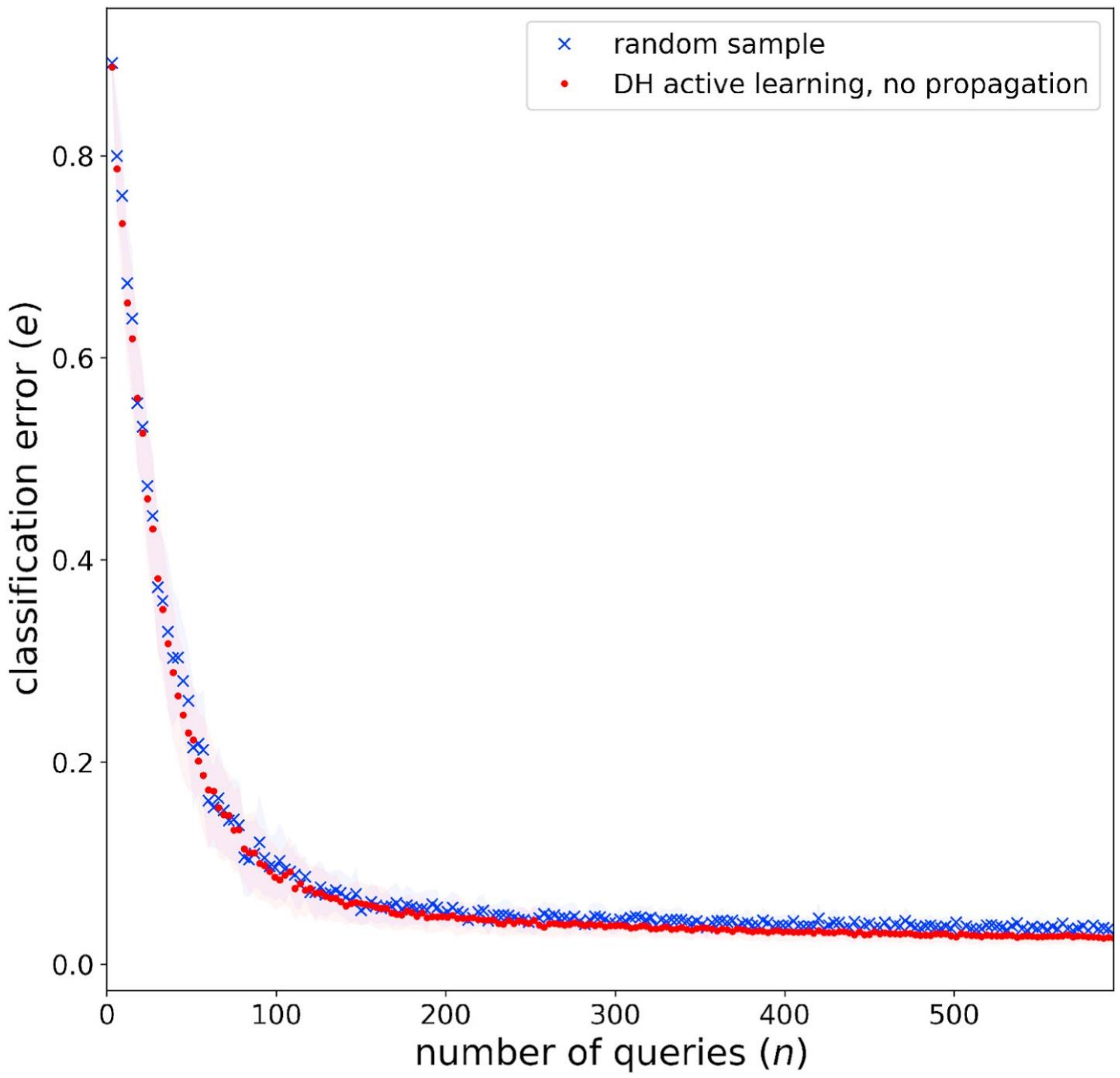
```
8: **if** $C_l \neq \{1, \ldots, N\}$ **then**
9: Mark $l$ as available, $S \leftarrow S \cup \{l\}$
10: **end if**
11: **for** $i \in S$ **do**
12: Update dissimilarity matrix $d(i, l)$
13: **end for**
14: **until** *no more clusters are available for merging*
```



Algorithm 2. Cluster-adaptive active learning.

```
    Input: Agglomerative clustering $T = \{u_i\}_{i=1}^{N-1}$ of the input observations $X$
 1: $P \leftarrow \{\text{root}\}$                         $\triangleright$ Initialise current pruning as the root node
 2: $L \leftarrow \{0\}$                                   $\triangleright$ Initialise arbitrary root label
 3: #——— GUIDED SAMPLING ———#
 4: for $t = 1 : T$ do                                     $\triangleright$ Algorithm run budget $T$
 5:     for $i = 1 : B$ do                                 $\triangleright$ Guided sampling, batch size $B$
 6:         $v \leftarrow \text{select}(P)$                $\triangleright$ select $v$ from $P$ according to Equation 16
 7:         randomly sample $\hat{x}$ form $T_v$           $\triangleright$ Adding to $X_l$
 8:         query $l$ the label of $\hat{x}$, update $X_L$ $\triangleright$ Provided by engineer/oracle
 9:         update $(n_u(t), p_u(t))$                      $\triangleright$ For all nodes containing new sample $\hat{x}$
10:     end for
11:     for all nodes $u \in T$ do $\triangleright$ Compute the admissibility and error for all nodes
12:         update $(\mathcal{A},\ \tilde{\epsilon}_{u,L(u)})$
13:     end for
14:     #—— PRUNING REFINEMENTS ——#
15:     for each $v \in P$ do                              $\triangleright$ Refine the current pruning, node by node
16:         let $(P_v, L_v)$ be the best pruning
            and labelling of $T_v$:                        $\triangleright$ According to the rules in Section 3.3
17:         $P \leftarrow P_v \cup (P \setminus v)$        $\triangleright$ Update node $v$ to refine $P$
18:         $L(v) \leftarrow L_v(\hat{v})$ for all $\hat{v} \in T_v$ $\triangleright$ Update $L(v)$ to reflect the refined pruning
19:     end for
20: end for
21: #——— LABEL PROPAGATION ———#
22: for each cluster $v \in P$ do                          $\triangleright$ For each cluster in the *final* pruning
23:     if $|P| \geq |l(t)|$ then                          $\triangleright$ Additional rule, Equation 18
24:         assign each unlabelled point in                $\triangleright$ Propagate labels to $X_u$
            $T_v$ the label $L(v)$, update $X_L$
25:     end if
26: end for
27: return: final pruning and labelling $(P, L)$, labelled data $X_L$
```

Misclassification error *e* for an increasing query budget *n*. Plots are provided for classifiers trained using guided sampling (the DH learner *without* label propagation) vs. random sample training.


random sample
DH active learning, no propagation
classification error (e)
number of queries (n)