Comp4433 individual Project report

21100602d

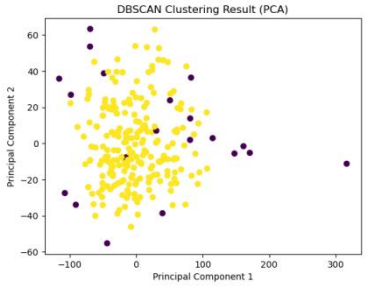
Liu Yuzhou

*** notice : 0 in the data set represents female, 1 represents male.**

Data Preprocessing:

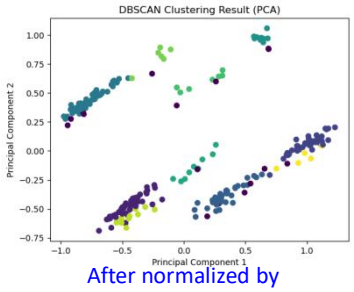
Before start other operations, I have applied DBSCAN algorithm to clear the outlier first. And following is the result which plot after applied the PCA method to project the data on a 2 dimensional space, and the purple point is the outlier. Here I find the result is really sensitive when epsilon value is among 25-30, seems majority of the points' neighbor are fallen in such a region.

dbscan = DBSCAN(eps=27, min_samples=4)



Based on original data set

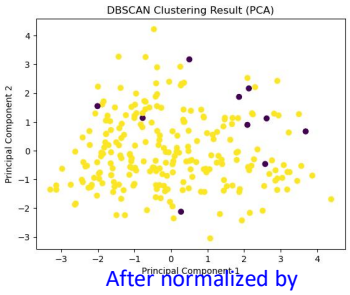
dbscan = DBSCAN(eps=1, min_samples=5)



After normalized by

min-max normalization

dbscan = DBSCAN(eps=3.5, min_samples=5)



After normalized by

Z-score normalization

We can see that different normalization method actually can achieve different result, the min-max normalization method tends to make data much more scatter plotted, which means more clusters tends to be formed, but for z score it seems uniformly distributed.

Thinking: For data clustering it might be better to use min-max normalization, which tends to make data fall in different clusters respectively.

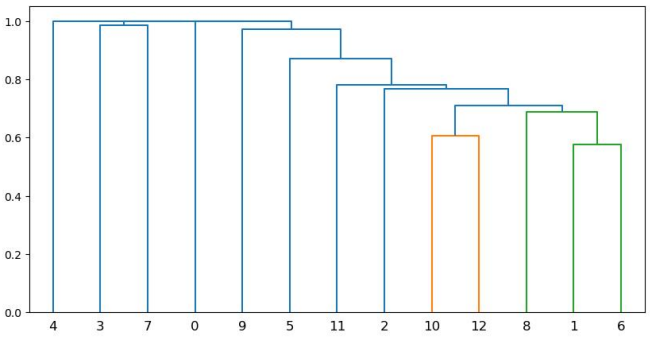
So after sweep out the noisy point , I start training next.

Over all view: I majorly focus on the bagging approach that is based on features or rows.

Approach 1: Single linkage(cluster features) + multiple decision tree voting.

So the first way I have tried is to use single linkage clustering algorithm to cluster the attributes, and then for different attribute sets, I will build a decision tree based for it, and finally based on majority voting result, I will finally get the predict output.

Single linkage(Original data set):
So as we can see below it is the single linkage result.



Thinking: Based on view of the graph, I choose the threshold to form the clusters at 0.7, because here we can view it tends to achieve the best grouping result for us, the orange and green things are more likely highly related and tends to be clustered together in the earliest stage, but at higher threshold, it tends to group with other attributes, which means mixed.

```
Cluster 9: ['age']
Cluster 4: ['sex', 'restecg', 'exng']
Cluster 5: ['cp']
Cluster 1: ['trtbps']
Cluster 10: ['chol']
Cluster 7: ['fbs']
Cluster 2: ['thalachh']
Cluster 8: ['oldpeak']
Cluster 3: ['slp', 'thall']
Cluster 6: ['caa']
```

Clustering result

And then I applied feature bagging to form multiple decision trees and then finally do voting to predict the result(Following is the result I have obtained based on different conditions).

Result obtained based on original dataset:

```
Evaluation Metrics:
--------------------
Accuracy: 0.85
Precision: 0.93
Recall: 0.78
F1-Score: 0.85
```

Original data

```
Evaluation Metrics:
--------------------
Accuracy: 0.82
Precision: 0.84
Recall: 0.81
F1-Score: 0.83
```

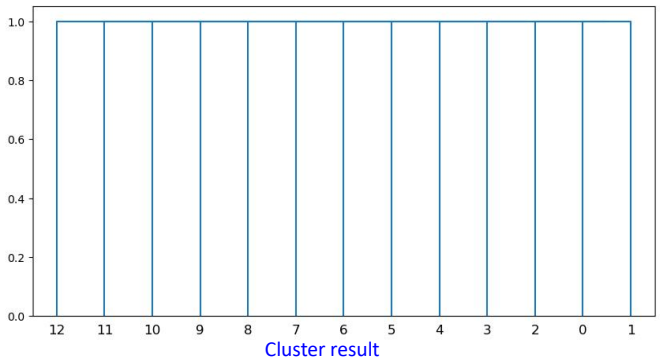
Z-score Normalization

```
Evaluation Metrics:
--------------------
Accuracy: 0.75
Precision: 0.74
Recall: 0.81
F1-Score: 0.78
```

Min-max Normalization

Description: Here by normalizing the result by Z-score normalization seems not to be a good strategy for clustering, seems that to make the data uniformly distributed will eliminate some important information for clustering different clusters. We can also see from the clustering result that all the attributes seems to have similar distance, so 13 groups have been formed. But the interesting thing is that seems build thirteen different decision trees for each attributes seems also achieves a good result, which is close but a little bit worsen than the original trained one, that means some of the data seems have correlation with each other and somehow worsen the training

result.



Thinking: The min-max and z-score normalization tends to produce the worse training result, it might caused the changing of the distribution of original data, and might hide some important information inside the dataset.

Attempt for making improvement and Potential Innovation:

(Similar matrix + K-means + multiple decision trees)

Innovation point: *Conventional way is always to minimize the intra-distance of a group, and maximize the inter distance between different clusters. What about cluster the most dissimilar items into same group? Maybe it can somehow handle the correlation between different columns?(Two stages of finding approach to solve the problem)*

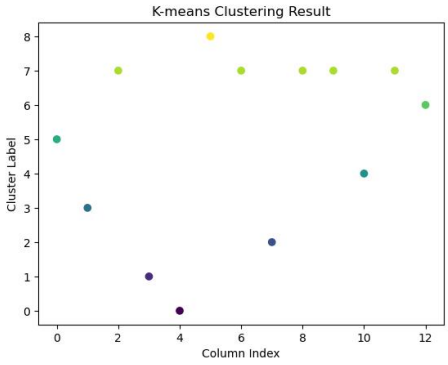
What I attempted to do here is try to put dissimilar columns together to build several different decision tree.

First stage(failed at implementation by using single linkage): I am think about to cluster based on Jaccard matrix and each time choose the largest value to combine together, but unfortunately, I failed at last, seems that python built in single linkage cannot have an option to cluster the most dissimilar things together, but theoretically it is achievable, instead of choose the smallest value to cluster together each time, we can choose the largest value.

Second stage(achieve the goal): I considered to use the cosine similarity matrix and treat the similarity value to be distance, so originally the large value inside the cosine similarity matrix means the higher similarity, but if we treat them as distance, then the most similar things will tend to be pushed far away from each other, but the most dissimilar columns which has the small cosine value will tend to pulled together, and then we modify the diagonal elements of the matrix to be 0. So by now we can use k-means algorithm to cluster those points.

So the work flow will become: Cosine similarity matrix(treat as distance matrix) + K-means to cluster the columns based on the distance matrix -> Build multiple decision tree based on different set of clusters -> voting to predicting

Here by taking the k-means algorithm(Which is somehow a draw back of this algorithm), we need to determine how many decision trees to form.
Best result attain at k=9:




```
[("cluster_['chol']", DecisionTreeClassifier()),
 ("cluster_['trtbps']", DecisionTreeClassifier()),
 ("cluster_['thalachh']", DecisionTreeClassifier()),
 ("cluster_['sex']", DecisionTreeClassifier()),
 ("cluster_['slp']", DecisionTreeClassifier()),
 ("cluster_['age']", DecisionTreeClassifier()),
 ("cluster_['thall']", DecisionTreeClassifier()),
 ("cluster_['cp', 'restecg', 'exng', 'oldpeak', 'caa']",
  DecisionTreeClassifier()),
 ("cluster_['fbs']", DecisionTreeClassifier())]
```

The clustering result of the attributes

```
Evaluation Metrics:
--------------------
Accuracy: 0.84
Precision: 0.87
Recall: 0.81
F1-Score: 0.84
```

Result of the innovated model

```
Evaluation Metrics:
--------------------
Accuracy: 0.85
Precision: 0.93
Recall: 0.78
F1-Score: 0.85
```

Result of the original model

Though the innovated model only improved the recall value than the original one, from other perspective it even act worsen, but I still think it is a good attempt, and a potential way to improve the final result. Maybe we could change another way to measure the similarity, or choose another algorithm to cluster the data, there many potential areas to improve the final result.

Some Insight of the model:(Analysis of the data based on innovative model)

Second thing I also did during this stage is that I try to see whether the innovative model is more predictable for female or for male.

Infect on sex:

```
dataset[dataset['sex'] != 0].shape
```

```
(207, 14)
```

Number of male in dataset

```
dataset[dataset['sex'] != 1].shape
```

```
(96, 14)
```

Number of female in the dataset

- **Train and test male and female individually**

So first thing I did is to drop all the male in the dataset, and train the rest, or drop all the female in the dataset, to train all the male individually.

```
Evaluation Metrics:
--------------------
Accuracy: 0.90
Precision: 0.93
Recall: 0.93
F1-Score: 0.93
```

Female(0) only

```
Evaluation Metrics:
--------------------
Accuracy: 0.71
Precision: 0.56
Recall: 0.71
F1-Score: 0.63
```

Male(1) only

I surprisingly find that for female only the result is somehow accurate(There might because the population of female is less in the dataset), but if I train male individually, I will get the result which is really poor.

Conclusion: **for this dataset our model is more predictable for female.**

Thinking of why it is more predictable for female:

potential reason (1):*it might because some important information which is useful is hide inside of the female part of the dataset. For male it might be less.*

Potential reason (2): *Though for male though their population is large, but maybe there are a lot of noise in the data set which could not be handle by my current method, and the noise might hide the important information in the dataset. In this case, the male dataset can still contain lot of useful information.*

To justify the correct the reason, I continue to do the experiment:

- **Training based on female only to predict male only:**

```
random.seed(123)
dataset = pd.read_csv("heart.csv")
# dataset = dataset[dataset['sex'] != 0]
train_set, test_set = train_test_split(dataset, test_size=0.2, random_state=42)
heart_test=test_set[test_set['sex'] != 1]# select all male
heart_data=train_set[train_set['sex']!=0]# select all female
heart_data =heart_data.drop('sex', axis=1)
heart_test =heart_test.drop('sex', axis=1)
```

Deal with dataset

```
Evaluation Metrics:
--------------------
Accuracy: 0.65
Precision: 0.61
Recall: 0.74
F1-Score: 0.67
```

Prediction result

Thinking: *First thing which is really interesting is that, training based on female only to predict male in the dataset achieves a better performance than we training based on male and to predict male only.*

And then I am considering what about add male into the training dataset, will it become more predictable for female?

- **Also we use the both female and male for training and to predict male:**

```
random.seed(123)
dataset = pd.read_csv("heart.csv")
# dataset = dataset[dataset['sex'] != 0]
train_set, test_set = train_test_split(dataset, test_size=0.2, random_state=42)
heart_test=test_set[test_set['sex'] != 1]# select all male
heart_data=train_set# select all female
heart_data =heart_data.drop('sex', axis=1)
heart_test =heart_test.drop('sex', axis=1)
```

With full training data to predict male only

```
Evaluation Metrics:
--------------------
Accuracy: 0.81
Precision: 0.85
Recall: 0.74
F1-Score: 0.79
```

Prediction result

Thinking:

Point1:*We can see that the accuracy of training based on female only to predict male is not enough, so that means there is still some important information hide in the male dataset which is important for prediction, but with put the female data to predict together, we can somehow correct the noise, which improves the training*

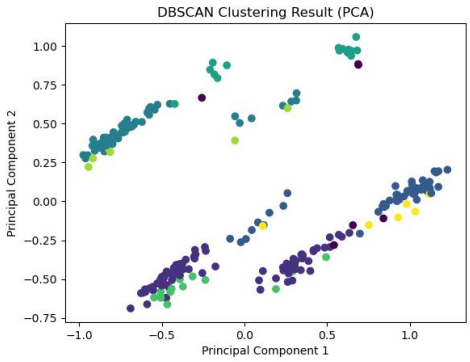
result.

Point 2: *from here we can introduce boosting idea to improve the model, which is to put more weight on the 'sex' attribute or to put more weight on male data points, but due to knowledge limitation on boosting method, I am not going to implement that, just put it here as a potential way to improve in future.*

Approach 2: try to use the DBSCAN result of min-max normalization result to build the decision tree.

$$v' = \frac{v - min_A}{max_A - min_A}(newmax_A - newmin_A) + newmin_A$$

Because we can see from the DBSCN clustering result for normalizing the outlier that the min-max normalization tends to make result fall into different clusters, so I am thinking whether we can build decision tree for different cluster of data points and then use basic bagging based on different groups of people.



Result be observed

```
array([ 0,  1,  0,  2,  6,  2, -1,  2,  0,  2,  0,  3,  0,  0,  0,  0,  4,
        4,  0,  0,  1,  2,  1,  0,  2,  1,  2,  2,  1,  2,  2,  0,  2,  0,
        0,  2,  4,  0,  1,  0,  2,  0, -1,  0,  2,  2,  2,  0,  1,  2,  0,
        1,  0,  0,  2,  0,  0,  0,  1,  3,  1,  1,  6,  0,  1,  0,  1,  3,
        0,  3,  2,  1,  0,  2,  2,  0,  0,  1,  1,  1,  1,  2,  2,  2,  2,
        2,  2, -1,  1,  2,  2,  4,  4,  0,  3,  0,  0,  0,  0,  1,  2, -1,
        2,  0,  0,  0,  3,  5,  2,  1,  2,  0,  0,  1,  3,  1,  0,  2,  4,
        5,  4,  3,  0,  1,  4,  0,  0,  0,  0,  4,  0,  0,  5,  0,  0,  1,
        1,  1,  2,  0,  0,  0,  0,  1,  3,  3,  0,  1,  0,  0,  1,  1,  0,
        2,  2,  0,  4,  0,  0,  4,  0,  1,  0,  2,  2,  5,  2,  2,  6,  3,
        3,  0,  2,  0,  2,  0,  1, -1,  0,  5,  0,  2,  3,  1,  0,  1,  2,
        2,  6,  0,  3,  2,  1,  0,  2,  2,  0,  2,  0,  1,  1,  1,  0,  1,
        3,  1,  2,  1, -1,  2,  0,  1,  2,  2,  1,  0,  1,  1,  6,  0,  2,
        1,  5,  1,  0,  0,  1,  1,  0,  2,  2,  4,  4,  6,  0,  2,  0,  0,
        1,  4,  0,  2], dtype=int64)
```

Label of different clusters

We can observe from the label sets that 7 clusters are formed in the dataset, the -1 indicate outliers, which have been removed from our dataset. So we will have 7 experts (different decision tree) to voting together.

```
Evaluation Metrics:
--------------------
Accuracy: 0.75
Precision: 0.74
Recall: 0.81
F1-Score: 0.78
```

Min-max normalized Prediction result of previous model(naive)

```
Evaluation Metrics:
--------------------
Accuracy: 0.80
Precision: 0.88
Recall: 0.72
F1-Score: 0.79
```

Prediction result of current model

```
[('cluster_0', DecisionTreeClassifier()),
 ('cluster_1', DecisionTreeClassifier()),
 ('cluster_2', DecisionTreeClassifier()),
 ('cluster_3', DecisionTreeClassifier()),
 ('cluster_4', DecisionTreeClassifier()),
 ('cluster_5', DecisionTreeClassifier()),
 ('cluster_6', DecisionTreeClassifier())]
```

The decision trees have been built

The result shows is not as good as the previous naive training based on the naive model without normalizing the dataset, but it somehow improved a lot on the training which is based on the normalized dataset, which means that basic bagging is a more suitable for normalized data.

Thinking: *and the normalization tends to increase the difference between different people in the dataset, but it could also make the training result somehow worsen, now I consider it is because normalization also eliminate some important information which is contained in the dataset, so my next trial is instead of using the normalized data for training, I try to use the original dataset. But different decision tree is still built based on the normalized data.*

Attempt to make improvement:

So the change I made here is to change the training data of the model: min-max normalized data -> original data.

```
Evaluation Metrics:
--------------------
Accuracy: 0.80
Precision: 0.88
Recall: 0.72
F1-Score: 0.79
```

Training based on min-max normalized data

```
Evaluation Metrics:
--------------------
Accuracy: 0.84
Precision: 0.89
Recall: 0.78
F1-Score: 0.83
```

Training based on Original data

```
[('cluster_0', DecisionTreeClassifier()),
 ('cluster_1', DecisionTreeClassifier()),
 ('cluster_2', DecisionTreeClassifier()),
 ('cluster_3', DecisionTreeClassifier()),
 ('cluster_4', DecisionTreeClassifier()),
 ('cluster_5', DecisionTreeClassifier()),
 ('cluster_6', DecisionTreeClassifier())]
```

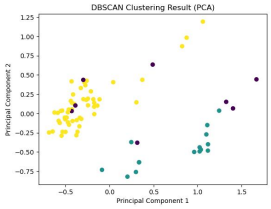
The decision trees have been built

By doing comparison we can see that the result have improved bit, which means that our original data is more suitable for prediction than the data we obtained after normalization. Which means that though min-max normalized data is not good for training, but it can play a role in separate data into different groups, and the grouping information is somehow useful for prediction.

Some Insight of the model:(Analysis of the data based on improved system here)

Here I also choose to analyse the affect of age, and the result is somehow interesting:

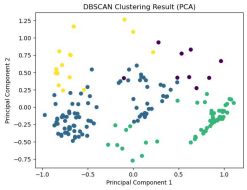
- **Train and test male and female individually**




```
Evaluation Metrics:
--------------------
Accuracy: 0.65
Precision: 0.73
Recall: 0.79
F1-Score: 0.76
```

Female(0) only

```
Evaluation Metrics:
--------------------
Accuracy: 0.69
Precision: 0.53
Recall: 0.57
F1-Score: 0.55
```



Male(1) only

We can observe from here that the model is still more predictable for female than for male, but the result is quite interesting, we can see that it is both bad for female and for male if we train the male and female individually, the result obtained is poor.

Thinking: *That means sex have a crucial impact on the training result, which cannot be omit here. So when male and female data are putting together the model can learn the impact on sex, it may gain some information such as how likely the person will get heart attack given the sex of the person is male/female.*

Conclusion: *Sex is a crucial factor for heart attack prediction.*

Summarize of this project:

In this project I build two types of systems, both of them uses the idea of bagging. And before the start of other procedure we use DBSCAN clustering method to clear the outlier contains in the dataset, So the first model is based on feature bagging, and the second one is based on basic bagging.

For feature bagging: I first use single linkage to cluster the features, and then based on the observed result, I conducted a innovative way which is to cluster the feature attributes based on dissimilarity(similarity matrix works as distance matrix, and k-means works for clustering).

For basic bagging: I used the DBSCAN clustering result based on min-max normalized dataset to build different clusters of data points respectively, and build different decision trees for different clusters of data, and finally apply voting to do the classification.

Insight from the dataset:

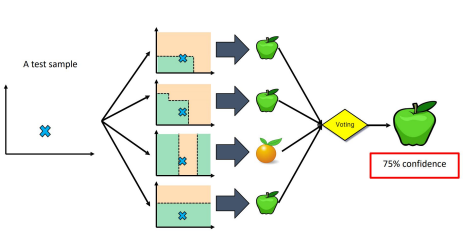
All in all, the female(0) tends to be more predictable than male(1).

First approach: The male dataset is large but seems that it contains some noise which could worsen the training result, and the noise can be somehow reduced by contain male points into the dataset.

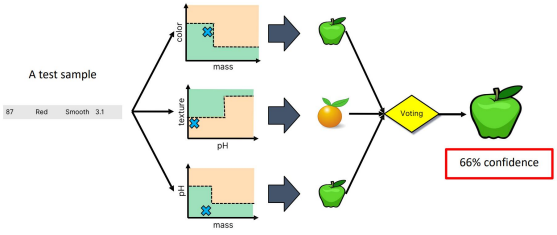
Second approach: Sex is a crucial attribute of the dataset which plays significant rule in heart attack prediction.

References:

Classification more(p29, p31):



Basic bagging



Feature bagging