

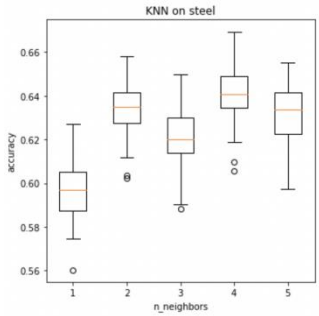
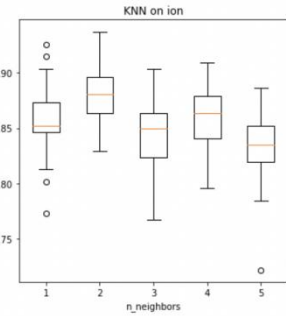
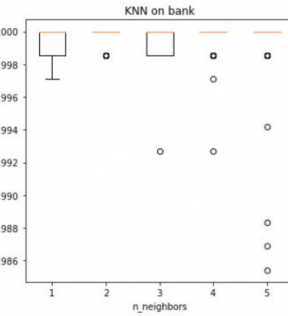
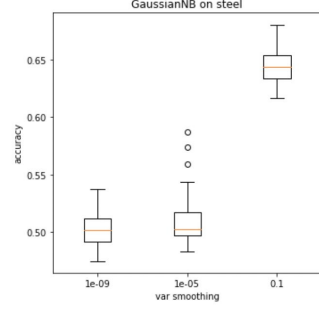
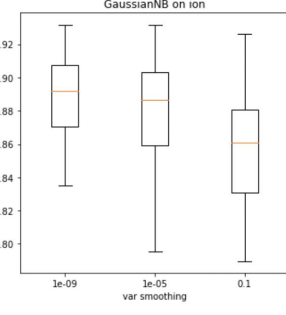
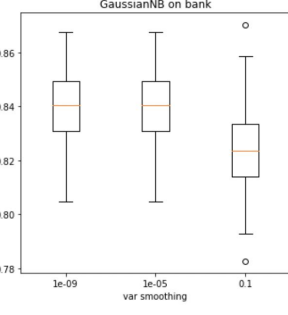
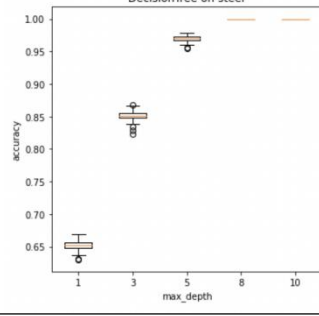
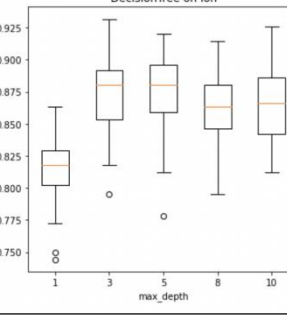
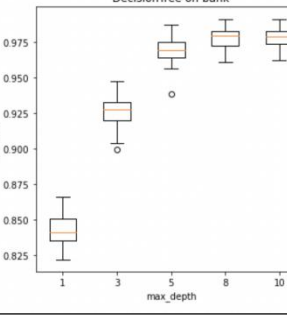
# comp309 ass1 report

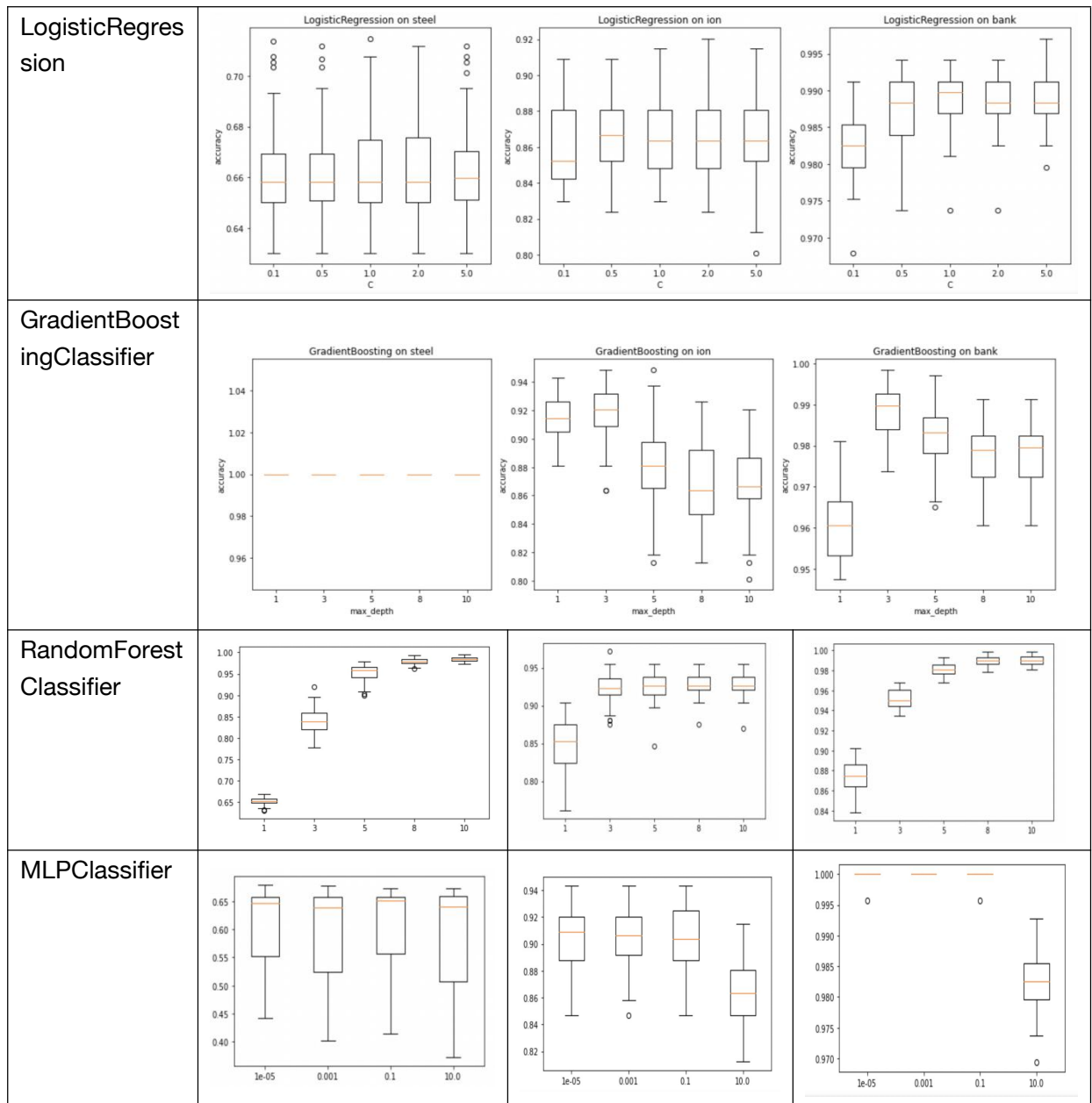
ID:300595912 Name:ZihaoZhang

## 1 Classification [70 marks]

1.3 Tasks:

(i) a 7-by-3 (7 classifiers and 3 datasets) table to present the boxplots on the classifier accuracy versus parameter values.(one per classifier, each consisting of several subplots for different datasets.)

	steel-plates-fault	ionosphere	banknotes
KNN			
GaussianNB			
DecisionTreeC lassifier			



(ii) Table (1) is to contain the best mean value of the test errors.

	steel-plates-fault	ionosphere	banknotes
KNN	0.3599	0.1186	0.0002
GaussianNB	0.3552	0.1116	0.1600
DecisionTreeClassifier	0.0	0.1266	0.0226
LogisticRegression	0.3351	0.1334	0.0110

GradientBoostingClassifier	0.0	0.0811	0.0111
RandomForestClassifier	0.0158	0.0724	0.0099
MLPClassifier	0.3920	0.0958	0.0

Table (2) is to contain the corresponding hyperparameter values for obtaining the best test errors.

	steel-plates-fault	ionosphere	banknotes
KNN	4	2	2
GaussianNB	0.1	1e-09	1e-09
DecisionTreeClassifier	8	5	10
LogisticRegression	2	2	1
GradientBoostingClassifier	1	3	3
RandomForestClassifier	10	8	10
MLPClassifier	1e-05	0.001	0.001

(iii) Write a paragraph to compare and analyse the overall results as captured in these two tables including which model has the best performance and why, and how sensitive these models are to the complexity control hyperparameter.

Analyst:

First, I analyzed the characteristics of the three data sets:

*steel-plates-fault:*

number of variables:33

number of instances:1941

*ionosphere:*

number of variables:34

number of instances:351

*banknotes:*

number of variables:4

number of instances:1372

Obviously, *steel-plates-fault* has huge number of instances. *ionosphere* dataset has large number of instances, but less sample size. While *banknotes* dataset has large sample size with small number of features.

performance:

Depends on Table(i){the best mean value of the test errors}, we can see:

**st:**

The values of accuracy in steel dataset are generally low except some algorithm with tree structure like **DecisionTreeClassifier**, **GradientBoostingClassifier** and **RandomForestClassifier**.

The one reason I think is this dataset(*steel-plates-fault*) is a non-linear dataset, the algorithm with tree structure can handle this kind of dataset well.

**io:**

**RandomForestClassifier** and **GradientBoostingClassifier** show a very good performance. compares to the other algorithm, there are more than 5% extra accuracy. This two model get very good score on ion datasets.

The reason I think is this two algorithm are pretty good to deal with the dataset which has large sample size.

**bank:**

**MLP** have best score at bank which is 100%, **KNN** which accuracy is also more than 99%, so this two model may good at handle with datasets which have large sample size with small number of features.

It is possible that the data is overfitted, so the error of MLP on this dataset show 0, cause *banknotes* has few number of variables, it is more easy to accur overfitted.

In general,

**RandomForestClassifier** and **GradientBoostingClassifier** show good performance in handling with dataset which has big number of variables. **KNN** is very good at linear relational dataset with large sample size, but it seems not good at nonlinear relation.(get not good score at steel). **MLP** model is also good at predict linear relation model, and not preform well on nonlinear relation model.

For *ionosphere* and *banknotes*, we can see both of them have higher accuracy than *steel-plates-fault*. Those two dataset has already standardized, the data is between 0 to 1, and all these algorithms perform well in low dimensional data.

sensitive:

**KNN:**

HyperParameter we test for this model is `n_neighbor` which means the number of adjacent points.

We can see that it is sensitive in steel and ionosphere dataset, the diffrent between the highest value score and the lowest score is nearly 5%, but only have a little change with different hyperparameter in banknotes dataset. I think the reason for this is that the score for banknotes dataset is too high.

**GaussianNB:**

HyperParameter we used for model is `var_smoothing`. The largest variance of all feature variances is added to the estimated variance in a certain proportion which is determined by `var_Smoothing` parameter control.

It is very sensitive in steel, because there is almost 15% difference of accuracy between two hyperparameters(1e-09, 0.1). And it is not sensitive to the other two datasets, we can see that var\_ Smoothing not impact results a lot.

#### **DecisionTreeClassifier:**

HyperParameter we used for model is max\_depth which means the maximum depth of decision tree.

The Chart shows that the difference between the highest value score and the lowest score is more that 30% on steel datasets, 5% on ion dataset, and nearly 10% on bank datasets. This HyperParameter influence this model a lot especially in nonlinear relational dataset.

#### **LogisticRegression:**

The HyperParameter we used for this model is C, high C value indicate the great trust in this training data, low C value indicate the data may not fully represent the data of the real world.

To all of these datasets, it is not sensitive on those three dataset. This might is influence by the database's default value.

#### **GradientBoostingClassifier:**

The HyperParameter we used for this model is max\_depth which is same with DT. But GB using regression tree.

It is not sensitive in steel but in ionosphere and banknotes it is sensitive according to its volatility.

#### **RandomForestClassifier:**

HyperParameter for this model is also max\_depth, based on classification tree.

It is sensitive in all three datasets, the result is similar to Decision tree, the difference between the highest value score and the lowest score is more that 30% on steel datasets, 8% on ion dataset, and nearly 10% on bank datasets.

#### **MLPClassifier:**

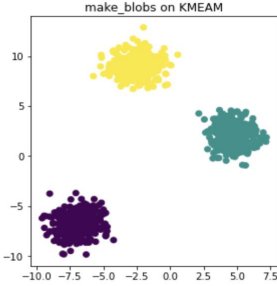
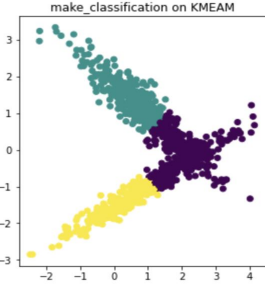
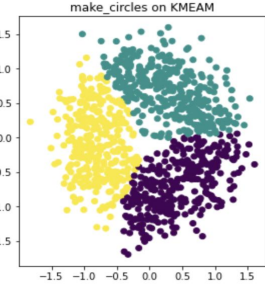
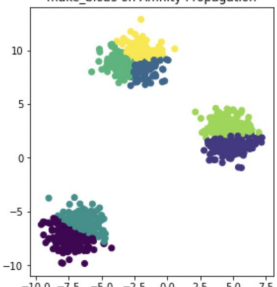
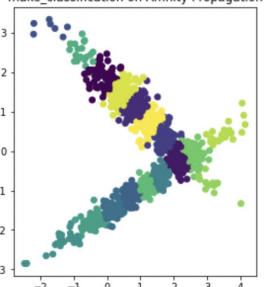
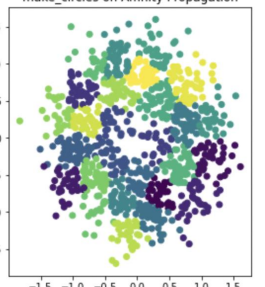
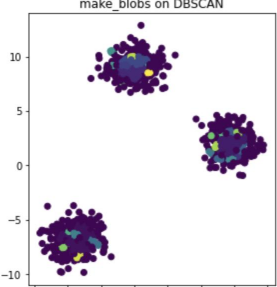
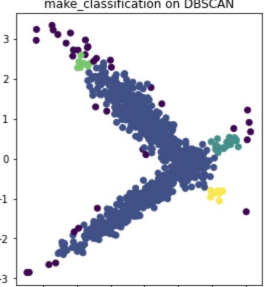
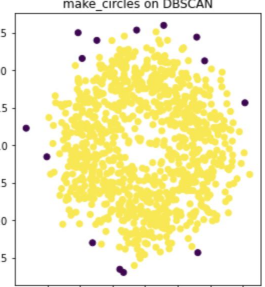
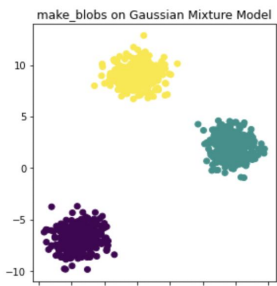
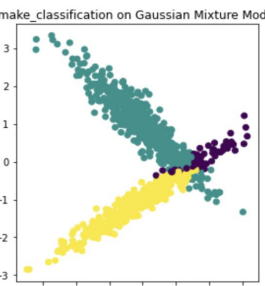
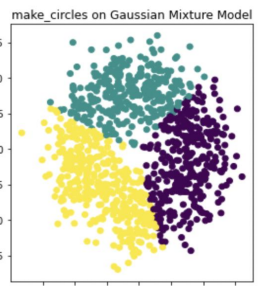
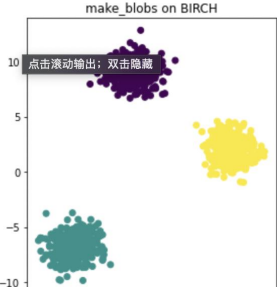
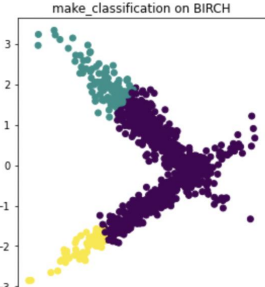
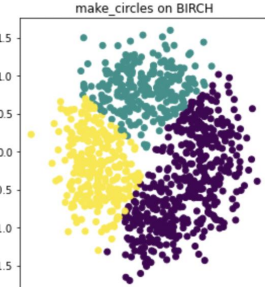
HyperParameter for this model is alpha which in this model means weight.

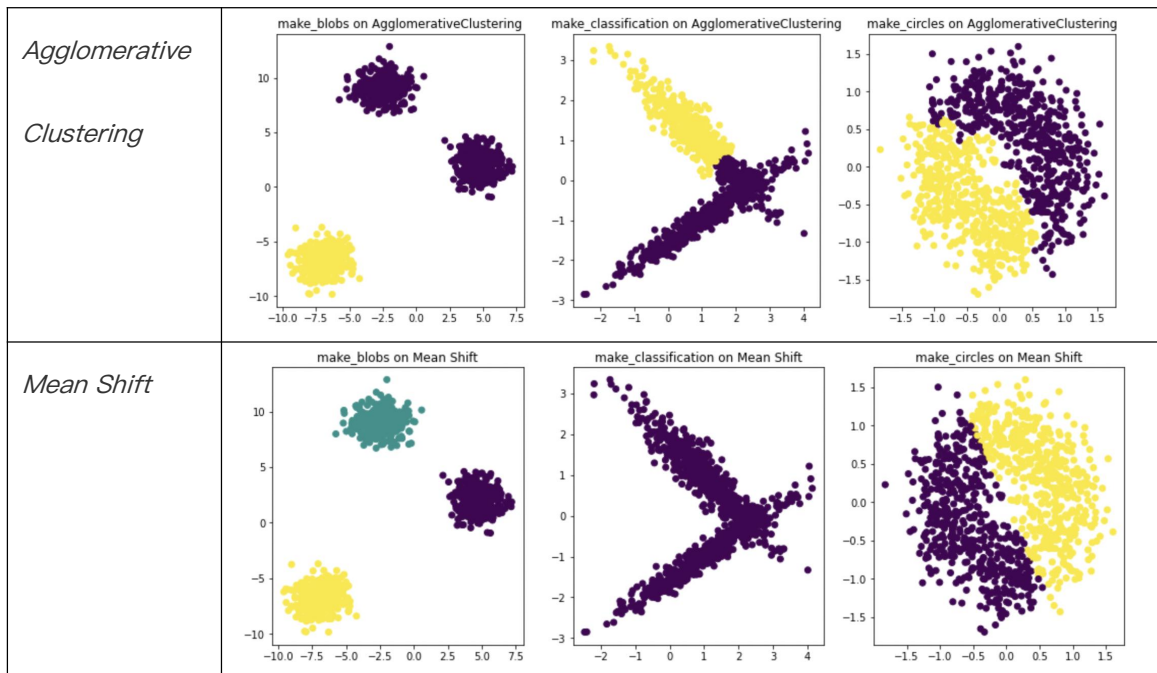
And we can see that this model is not quit sensitive to alpha. And two of them(io, bank) has not converged.

```
/Users/zhangzihao/opt/anaconda3/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
warnings.warn(
```

## **2. Clustering [30 marks]**

(i) Fit the clustering models on the datasets and predicts a cluster for each example in the datasets. Build a 7-by-3 (7 clustering algorithms and 3 datasets) table to present the scatter plots showing the clusters generated by each algorithm.

	<i>make_blobs</i>	<i>make_classification</i>	<i>make_circles</i>
<i>K-Means</i>			
<i>Affinity Propagation</i>			
<i>DBSCAN</i>			
<i>Gaussian Mixture Model</i>			
<i>BIRCH</i>			



(ii) Write a paragraph to compare and analyse the results of the clustering algorithms on the three datasets, highlight the characteristics of these algorithms.

#### (a) K-Means

For KMEAN model, the sample set is divided into K clusters, according to the distance between samples in a given sample set. The default n\_clusters value of k is 7.

So it will choose 7 cluster centers, I give the estimated (n\_clusters=3) to get the result.

base on the table above, we set n\_clusters=3 for blobs and 2 for classification we can get a better result.

#### (b) Affinity Propagation

Same with K-Mean algorithms, this model also need to find cluster center. what different is AP can decide the number of clusters, we don't need to set a n\_clusters value. And the existing data points are used as the final cluster center instead of generating a new cluster center.

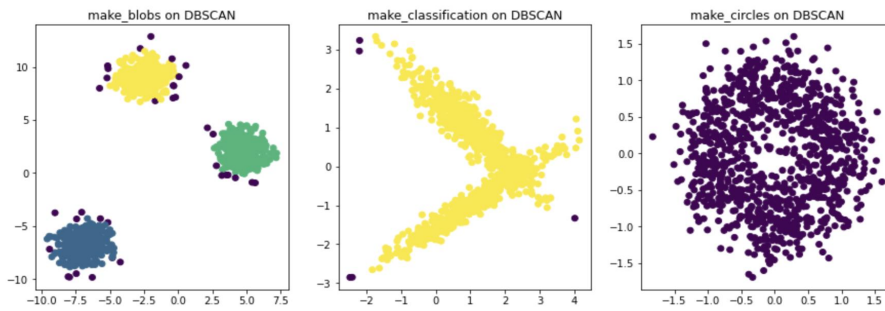
That's why we can see there are more center for one class in AP, and each point are clustering around a center.

#### (c) DBSCAN

Unlike K-means-based algorithms that need to pre-specify the number of clusters, DBSCAN can automatically determine the number of clusters. DBSCAN is a density-based clustering algorithm that can identify data points with a sufficiently high density and group them into the same cluster.

We can see the it performs well in those toysets(eps=0.2). Parameter eps is a variable used to measure the distance, it can not be too large, otherwise the dataset would be one cluster, like: if i set eps=0.5:

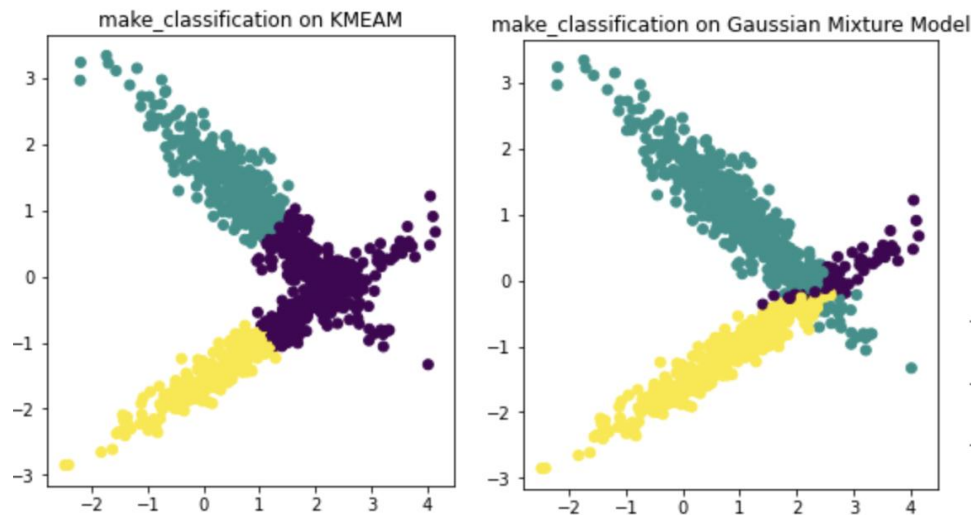




#### (d) **Gaussian Mixture Model**

GMM calculates the probability of belonging to each cluster for each data point, and assigns data points to each cluster center according to the probability.

From the results, its performance is very similar to the kmean algorithm, What the different is K-Mean let each data point is assigned to a unique cluster center, forming a definite cluster, while data point in GMM can belong to multiple clusters.



#### (e) **BIRCH**

BIRCH is a hierarchical clustering algorithm for high-dimensional data clustering. Its main goal is to efficiently build clustering structures when dealing with large-scale data, and to be able to control memory usage.

It attempts to divide the data sets at different levels. From the table in (i) we can see the advantage of this algorithm is that it identify noise points better than Kmean or AP.

#### (f) **Agglomerative Clustering**

Agglomerative Clustering is a common hierarchical clustering algorithm whose main goal is to gradually merge data points into larger and larger clusters until a complete cluster structure is formed.

Compared with BIRCH, both are hierarchical clustering algorithms, which can generate hierarchical clustering results and form a clustering tree structure.

BIRCH has advantages in dealing with high-dimensional data, while agglomerative hierarchical clustering is relatively weak in dealing with high-dimensional data and is susceptible to the curse of dimensionality.



### **(g) Mean Shift**

Mean Shift is a clustering algorithm based on kernel density estimation for finding cluster centers in a set of data points. Its main idea is to iteratively find the direction where the density of data points is maximized, and move the data points to gradually gather in the area with the highest density until it converges to the cluster center.

As we can see, the cluster centers of these plots are all at the highest density, so the `make_blobs` and `make_circles` datasets show good performance.

### Reference:

*classification:*

*Collection of machine learning notes:*

<https://www.cnblogs.com/St-Lovaer/category/1638548.html>

Fetch dataset from openml by name or dataset id:

[https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch\\_openml.html](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_openml.html)

How to build a knn model:

<https://towardsdatascience.com/building-a-k-nearest-neighbors-k-nn-model-with-scikit-learn-51209555453a>

Getting the best value for the hyperparameter:

<https://numpy.org/doc/stable/reference/generated/numpy.argmin.html>

How to Change Plot Size in Python with `plt.figure()`:

<https://www.freecodecamp.org/news/matplotlib-figure-size-change-plot-size-in-python/>

Subplot building:

[https://matplotlib.org/stable/gallery/subplots\\_axes\\_and\\_figures/subplots\\_demo.html](https://matplotlib.org/stable/gallery/subplots_axes_and_figures/subplots_demo.html)

### *Cluster:*

Why we need to add `"n_redundant=0"` in `"make_classification"`:

<https://stackoverflow.com/questions/68372105/valueerror-number-of-informative-redundant-and-repeated-features-must-sum-to-1>

Random State:

<https://stackoverflow.com/questions/28064634/random-state-pseudo-random-number-in-scikit-learn>

Scatter params setting:

<https://www.geeksforgeeks.org/matplotlib-pyplot-scatter-in-python/>