**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика с системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №6

# «Методы обучения без учителя[.](https://github.com/ugapanyuk/courses_current/wiki/LAB_TMO__MISSING)»

Выполнил: Проверил:

студент группы РТ5-61Б преподаватель каф. ИУ5

Мицкевич В.Б. Гапанюк Ю.Е.

Подпись и дата: Подпись и дата:

Москва, 2023 г.

# Методы обучения без учителя.

# Цель лабораторной работы: изучение методов кластеризации и снижения размерности.

# Задание:

1. Выберите набор данных (датасет) для решения задачи обучения без учителя.
2. Выберите подмножество признаков датасета для анализа. Например, подмножество признаков может не включать целевой признак. Количество признаков в подмножестве должно быть более двух. Полученный датасет назовем датасетом D1.
3. С использованием метода главных компонент проведите для датасета D1 снижение размерности данных до двух. Полученный датасет назовем датасетом D2.
4. С использованием алгоритма t-SNE также проведите для датасета D1 снижение размерности данных до двух. Полученный датасет назовем датасетом D3.
5. Визуализируйте точки датасетов D2 и D3. В каком случае кластеры выделены наиболее явно?
6. Для датасетов D1, D2 и D3 проведите кластеризацию с использованием как минимум трех методов кластеризации, рассмотренных в лекции. Оцените качество кластеризации с использованием метрик, рассмотренных в лекции. Сделайте выводы о том, какой метод кластеризации и почему оказался лучше для каждого из датасетов.
7. import numpy as np  
   import pandas as pd  
   from sklearn.decomposition import PCA  
   from sklearn.datasets import \*  
   from sklearn.manifold import TSNE  
   from sklearn.cluster import KMeans, DBSCAN, MeanShift, SpectralClustering, AgglomerativeClustering  
   from sklearn.mixture import GaussianMixture  
   from sklearn.metrics import adjusted\_rand\_score  
   from sklearn.metrics import adjusted\_mutual\_info\_score  
   from sklearn.metrics import homogeneity\_completeness\_v\_measure  
   from sklearn.metrics import silhouette\_score  
   import seaborn as sns  
   import matplotlib.pyplot as plt
8. def make\_dataframe(ds\_function):  
    ds = ds\_function()  
    df = pd.DataFrame(data= np.c\_[ds['data'], ds['target']],  
    columns= list(ds['feature\_names']) + ['target'])  
    return df  
     
   iris = load\_iris()  
     
   df = make\_dataframe(load\_iris)
9. df.head()
10. sepal length (cm) sepal width (cm) petal length (cm) petal width (cm) \  
    0 5.1 3.5 1.4 0.2   
    1 4.9 3.0 1.4 0.2   
    2 4.7 3.2 1.3 0.2   
    3 4.6 3.1 1.5 0.2   
    4 5.0 3.6 1.4 0.2   
      
     target   
    0 0.0   
    1 0.0   
    2 0.0   
    3 0.0   
    4 0.0
11. df.info()
12. <class 'pandas.core.frame.DataFrame'>  
    RangeIndex: 150 entries, 0 to 149  
    Data columns (total 5 columns):  
     # Column Non-Null Count Dtype   
    --- ------ -------------- -----   
     0 sepal length (cm) 150 non-null float64  
     1 sepal width (cm) 150 non-null float64  
     2 petal length (cm) 150 non-null float64  
     3 petal width (cm) 150 non-null float64  
     4 target 150 non-null float64  
    dtypes: float64(5)  
    memory usage: 6.0 KB
13. def count\_nan(data):  
     for col in data.columns:  
     count\_nan = data[data[col].isnull()].shape[0]  
     print('{} имеет NAN: {}'.format(col, count\_nan))  
    count\_nan(df)
14. sepal length (cm) имеет NAN: 0  
    sepal width (cm) имеет NAN: 0  
    petal length (cm) имеет NAN: 0  
    petal width (cm) имеет NAN: 0  
    target имеет NAN: 0
15. X = iris.data  
    X
16. array([[5.1, 3.5, 1.4, 0.2],  
     [4.9, 3. , 1.4, 0.2],  
     [4.7, 3.2, 1.3, 0.2],  
     [4.6, 3.1, 1.5, 0.2],  
     [5. , 3.6, 1.4, 0.2],  
     [5.4, 3.9, 1.7, 0.4],  
     [4.6, 3.4, 1.4, 0.3],  
     [5. , 3.4, 1.5, 0.2],  
     [4.4, 2.9, 1.4, 0.2],  
     [4.9, 3.1, 1.5, 0.1],  
     [5.4, 3.7, 1.5, 0.2],  
     [4.8, 3.4, 1.6, 0.2],  
     [4.8, 3. , 1.4, 0.1],  
     [4.3, 3. , 1.1, 0.1],  
     [5.8, 4. , 1.2, 0.2],  
     [5.7, 4.4, 1.5, 0.4],  
     [5.4, 3.9, 1.3, 0.4],  
     [5.1, 3.5, 1.4, 0.3],  
     [5.7, 3.8, 1.7, 0.3],  
     [5.1, 3.8, 1.5, 0.3],  
     [5.4, 3.4, 1.7, 0.2],  
     [5.1, 3.7, 1.5, 0.4],  
     [4.6, 3.6, 1. , 0.2],  
     [5.1, 3.3, 1.7, 0.5],  
     [4.8, 3.4, 1.9, 0.2],  
     [5. , 3. , 1.6, 0.2],  
     [5. , 3.4, 1.6, 0.4],  
     [5.2, 3.5, 1.5, 0.2],  
     [5.2, 3.4, 1.4, 0.2],  
     [4.7, 3.2, 1.6, 0.2],  
     [4.8, 3.1, 1.6, 0.2],  
     [5.4, 3.4, 1.5, 0.4],  
     [5.2, 4.1, 1.5, 0.1],  
     [5.5, 4.2, 1.4, 0.2],  
     [4.9, 3.1, 1.5, 0.2],  
     [5. , 3.2, 1.2, 0.2],  
     [5.5, 3.5, 1.3, 0.2],  
     [4.9, 3.6, 1.4, 0.1],  
     [4.4, 3. , 1.3, 0.2],  
     [5.1, 3.4, 1.5, 0.2],  
     [5. , 3.5, 1.3, 0.3],  
     [4.5, 2.3, 1.3, 0.3],  
     [4.4, 3.2, 1.3, 0.2],  
     [5. , 3.5, 1.6, 0.6],  
     [5.1, 3.8, 1.9, 0.4],  
     [4.8, 3. , 1.4, 0.3],  
     [5.1, 3.8, 1.6, 0.2],  
     [4.6, 3.2, 1.4, 0.2],  
     [5.3, 3.7, 1.5, 0.2],  
     [5. , 3.3, 1.4, 0.2],  
     [7. , 3.2, 4.7, 1.4],  
     [6.4, 3.2, 4.5, 1.5],  
     [6.9, 3.1, 4.9, 1.5],  
     [5.5, 2.3, 4. , 1.3],  
     [6.5, 2.8, 4.6, 1.5],  
     [5.7, 2.8, 4.5, 1.3],  
     [6.3, 3.3, 4.7, 1.6],  
     [4.9, 2.4, 3.3, 1. ],  
     [6.6, 2.9, 4.6, 1.3],  
     [5.2, 2.7, 3.9, 1.4],  
     [5. , 2. , 3.5, 1. ],  
     [5.9, 3. , 4.2, 1.5],  
     [6. , 2.2, 4. , 1. ],  
     [6.1, 2.9, 4.7, 1.4],  
     [5.6, 2.9, 3.6, 1.3],  
     [6.7, 3.1, 4.4, 1.4],  
     [5.6, 3. , 4.5, 1.5],  
     [5.8, 2.7, 4.1, 1. ],  
     [6.2, 2.2, 4.5, 1.5],  
     [5.6, 2.5, 3.9, 1.1],  
     [5.9, 3.2, 4.8, 1.8],  
     [6.1, 2.8, 4. , 1.3],  
     [6.3, 2.5, 4.9, 1.5],  
     [6.1, 2.8, 4.7, 1.2],  
     [6.4, 2.9, 4.3, 1.3],  
     [6.6, 3. , 4.4, 1.4],  
     [6.8, 2.8, 4.8, 1.4],  
     [6.7, 3. , 5. , 1.7],  
     [6. , 2.9, 4.5, 1.5],  
     [5.7, 2.6, 3.5, 1. ],  
     [5.5, 2.4, 3.8, 1.1],  
     [5.5, 2.4, 3.7, 1. ],  
     [5.8, 2.7, 3.9, 1.2],  
     [6. , 2.7, 5.1, 1.6],  
     [5.4, 3. , 4.5, 1.5],  
     [6. , 3.4, 4.5, 1.6],  
     [6.7, 3.1, 4.7, 1.5],  
     [6.3, 2.3, 4.4, 1.3],  
     [5.6, 3. , 4.1, 1.3],  
     [5.5, 2.5, 4. , 1.3],  
     [5.5, 2.6, 4.4, 1.2],  
     [6.1, 3. , 4.6, 1.4],  
     [5.8, 2.6, 4. , 1.2],  
     [5. , 2.3, 3.3, 1. ],  
     [5.6, 2.7, 4.2, 1.3],  
     [5.7, 3. , 4.2, 1.2],  
     [5.7, 2.9, 4.2, 1.3],  
     [6.2, 2.9, 4.3, 1.3],  
     [5.1, 2.5, 3. , 1.1],  
     [5.7, 2.8, 4.1, 1.3],  
     [6.3, 3.3, 6. , 2.5],  
     [5.8, 2.7, 5.1, 1.9],  
     [7.1, 3. , 5.9, 2.1],  
     [6.3, 2.9, 5.6, 1.8],  
     [6.5, 3. , 5.8, 2.2],  
     [7.6, 3. , 6.6, 2.1],  
     [4.9, 2.5, 4.5, 1.7],  
     [7.3, 2.9, 6.3, 1.8],  
     [6.7, 2.5, 5.8, 1.8],  
     [7.2, 3.6, 6.1, 2.5],  
     [6.5, 3.2, 5.1, 2. ],  
     [6.4, 2.7, 5.3, 1.9],  
     [6.8, 3. , 5.5, 2.1],  
     [5.7, 2.5, 5. , 2. ],  
     [5.8, 2.8, 5.1, 2.4],  
     [6.4, 3.2, 5.3, 2.3],  
     [6.5, 3. , 5.5, 1.8],  
     [7.7, 3.8, 6.7, 2.2],  
     [7.7, 2.6, 6.9, 2.3],  
     [6. , 2.2, 5. , 1.5],  
     [6.9, 3.2, 5.7, 2.3],  
     [5.6, 2.8, 4.9, 2. ],  
     [7.7, 2.8, 6.7, 2. ],  
     [6.3, 2.7, 4.9, 1.8],  
     [6.7, 3.3, 5.7, 2.1],  
     [7.2, 3.2, 6. , 1.8],  
     [6.2, 2.8, 4.8, 1.8],  
     [6.1, 3. , 4.9, 1.8],  
     [6.4, 2.8, 5.6, 2.1],  
     [7.2, 3. , 5.8, 1.6],  
     [7.4, 2.8, 6.1, 1.9],  
     [7.9, 3.8, 6.4, 2. ],  
     [6.4, 2.8, 5.6, 2.2],  
     [6.3, 2.8, 5.1, 1.5],  
     [6.1, 2.6, 5.6, 1.4],  
     [7.7, 3. , 6.1, 2.3],  
     [6.3, 3.4, 5.6, 2.4],  
     [6.4, 3.1, 5.5, 1.8],  
     [6. , 3. , 4.8, 1.8],  
     [6.9, 3.1, 5.4, 2.1],  
     [6.7, 3.1, 5.6, 2.4],  
     [6.9, 3.1, 5.1, 2.3],  
     [5.8, 2.7, 5.1, 1.9],  
     [6.8, 3.2, 5.9, 2.3],  
     [6.7, 3.3, 5.7, 2.5],  
     [6.7, 3. , 5.2, 2.3],  
     [6.3, 2.5, 5. , 1.9],  
     [6.5, 3. , 5.2, 2. ],  
     [6.2, 3.4, 5.4, 2.3],  
     [5.9, 3. , 5.1, 1.8]])

## С использованием метода главных компонент проведите для датасета D1 снижение размерности данных до двух.

## Полученный датасет назовем датасетом D2.

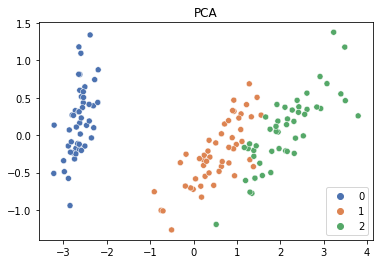
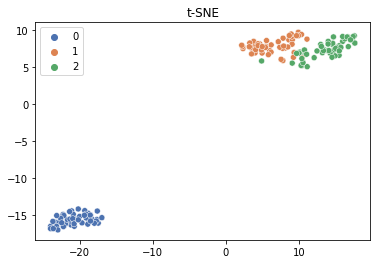
1. # Снижение размерности до 2 методом главных компонент (МГК)  
   pca = PCA(n\_components=2)  
   X\_pca = pca.fit\_transform(X)  
   print('Доля объясненной дисперсии: {}, сумма={}'.format(pca.explained\_variance\_ratio\_, np.sum(pca.explained\_variance\_ratio\_)))  
   X\_pca[:10]
2. Доля объясненной дисперсии: [0.92461872 0.05306648], сумма=0.977685206318795
3. array([[-2.68412563, 0.31939725],  
    [-2.71414169, -0.17700123],  
    [-2.88899057, -0.14494943],  
    [-2.74534286, -0.31829898],  
    [-2.72871654, 0.32675451],  
    [-2.28085963, 0.74133045],  
    [-2.82053775, -0.08946138],  
    [-2.62614497, 0.16338496],  
    [-2.88638273, -0.57831175],  
    [-2.6727558 , -0.11377425]])

## С использованием алгоритма t-SNE также проведите для датасета D1 снижение размерности данных до двух.

## Полученный датасет назовем датасетом D3.

1. # Снижение размерности до 2 методом t-SNE  
   tsne = TSNE(n\_components=2)  
   X\_tsne = tsne.fit\_transform(X)  
   X\_tsne[:10]
2. array([[-21.179821, -15.987457],  
    [-18.820745, -14.786243],  
    [-18.72989 , -15.860402],  
    [-18.384148, -15.514446],  
    [-21.16073 , -16.3791 ],  
    [-23.326126, -16.00032 ],  
    [-18.882215, -16.261795],  
    [-20.57479 , -15.567408],  
    [-17.6368 , -15.634228],  
    [-19.28883 , -15.009454]], dtype=float32)

## Визуализируйте точки датасетов D2 и D3. В каком случае кластеры выделены наиболее явно?

1. # Визуализация полученных выше двух датасетов  
   sns.scatterplot(x=X\_pca[:, 0], y=X\_pca[:, 1], hue=iris.target, palette="deep").set(title="PCA")
2. [Text(0.5, 1.0, 'PCA')]
3. 
4. sns.scatterplot(x=X\_tsne[:, 0], y=X\_tsne[:, 1], hue=iris.target, palette="deep").set(title="t-SNE")
5. [Text(0.5, 1.0, 't-SNE')]
6. 

### Посмотрев на графики, можно сказать, что t-SNE выделил кластеры намного лучше

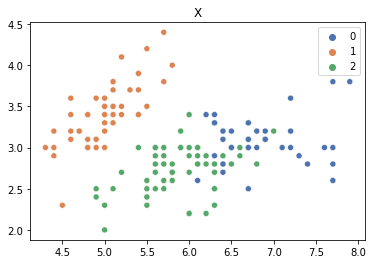
## Для датасетов D1, D2 и D3 проведите кластеризацию с использованием как минимум трех методов кластеризации, рассмотренных в лекции.

## Оцените качество кластеризации с использованием метрик, рассмотренных в лекции.

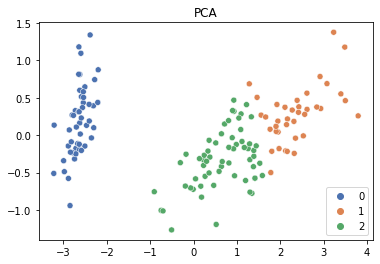
## Сделайте выводы о том, какой метод кластеризации и почему оказался лучше для каждого из датасетов.

### Кластеризация с помощью KMeans

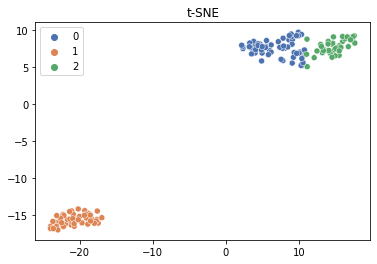
##### обычный датасет

1. k\_means = KMeans(n\_clusters=3)  
   k\_means\_prediction = k\_means.fit\_predict(X)
2. sns.scatterplot(x=X[:, 0], y=X[:, 1], hue=k\_means\_prediction, palette="deep").set(title="X")  
   plt.show()
3. 

#### датасет преобразованный с помощью PCA

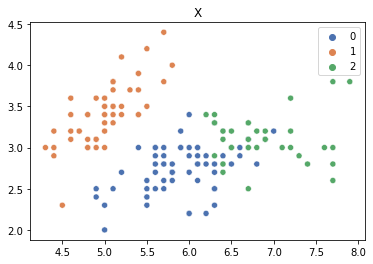
1. k\_means = KMeans(n\_clusters=3)  
   k\_means\_PCA\_prediction = k\_means.fit\_predict(X\_pca)
2. sns.scatterplot(x=X\_pca[:, 0], y=X\_pca[:, 1], hue=k\_means\_PCA\_prediction, palette="deep").set(title="PCA")  
   plt.show()
3. 

#### датасет преобразованный с помощью t-SNE

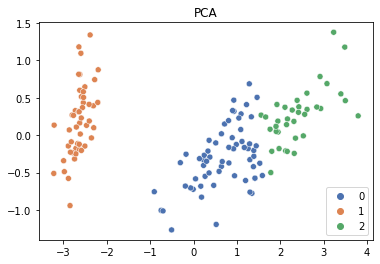
1. k\_means = KMeans(n\_clusters=3)  
   k\_means\_TSNE\_prediction = k\_means.fit\_predict(X\_tsne)
2. sns.scatterplot(x=X\_tsne[:, 0], y=X\_tsne[:, 1], hue=k\_means\_TSNE\_prediction, palette="deep").set(title="t-SNE")  
   plt.show()
3. 

### Кластеризация с помощью Иерархической кластеризации

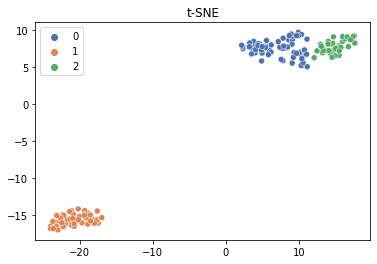
##### обычный датасет

1. agg\_clustering = AgglomerativeClustering(n\_clusters=3)  
   agg\_clustering\_prediction = agg\_clustering.fit\_predict(X)
2. sns.scatterplot(x=X[:, 0], y=X[:, 1], hue=agg\_clustering\_prediction, palette="deep").set(title="X")  
   plt.show()
3. 

#### датасет преобразованный с помощью PCA

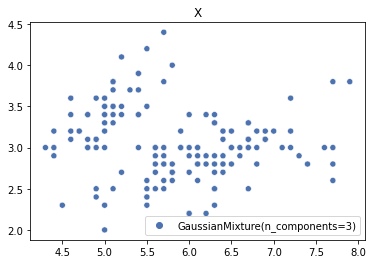
1. agg\_clustering = AgglomerativeClustering(n\_clusters=3)  
   agg\_clustering\_PCA\_prediction = agg\_clustering.fit\_predict(X\_pca)
2. sns.scatterplot(x=X\_pca[:, 0], y=X\_pca[:, 1], hue=agg\_clustering\_PCA\_prediction, palette="deep").set(title="PCA")  
   plt.show()
3. 

#### датасет преобразованный с помощью t-SNE

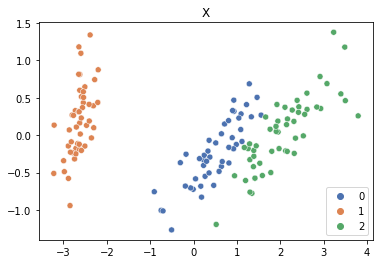
1. agg\_clustering = AgglomerativeClustering(n\_clusters=3)  
   agg\_clustering\_TSNE\_prediction = agg\_clustering.fit\_predict(X\_tsne)
2. sns.scatterplot(x=X\_tsne[:, 0], y=X\_tsne[:, 1], hue=agg\_clustering\_TSNE\_prediction, palette="deep").set(title="t-SNE")  
   plt.show()
3. 

### Кластеризация с помощью Gaussian mixture models

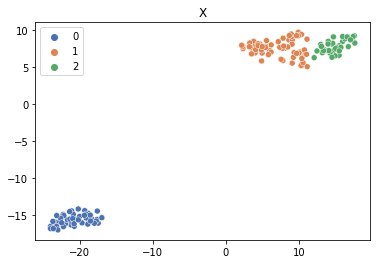
#### обычный датасет

1. gaus\_mix = GaussianMixture(n\_components=3)  
   gaus\_mix\_prediction = gaus\_mix.fit\_predict(X)
2. sns.scatterplot(x=X[:, 0], y=X[:, 1], hue=gaus\_mix, palette="deep").set(title="X")  
   plt.show()
3. 

#### датасет преобразованный с помощью PCA

1. gaus\_mix = GaussianMixture(n\_components=3)  
   gaus\_mix\_PCA\_prediction = gaus\_mix.fit\_predict(X\_pca)
2. sns.scatterplot(x=X\_pca[:, 0], y=X\_pca[:, 1], hue=gaus\_mix\_PCA\_prediction, palette="deep").set(title="X")  
   plt.show()
3. 

#### датасет преобразованный с помощью t-SNE

1. gaus\_mix = GaussianMixture(n\_components=3)  
   gaus\_mix\_TSNE\_prediction = gaus\_mix.fit\_predict(X\_tsne)
2. sns.scatterplot(x=X\_tsne[:, 0], y=X\_tsne[:, 1], hue=gaus\_mix\_TSNE\_prediction, palette="deep").set(title="X")  
   plt.show()
3. 

### Оценка качества кластеризации

1. def claster\_metrics(method, cluster\_datasets, cluster\_true\_y, datasets\_names):  
    """  
    Вычисление метрик кластеризации  
    """  
    ari = []  
    ami = []  
    hl = []  
    cl = []  
    vl = []  
    sl = []  
    for X in cluster\_datasets:  
    temp\_cluster = method.fit\_predict(X)  
    ari.append(adjusted\_rand\_score(cluster\_true\_y, temp\_cluster))  
    ami.append(adjusted\_mutual\_info\_score(cluster\_true\_y, temp\_cluster))  
      
    h, c, v = homogeneity\_completeness\_v\_measure(cluster\_true\_y, temp\_cluster)  
    hl.append(h)  
    cl.append(c)  
    vl.append(v)  
      
    try:  
    sscore = silhouette\_score(X, temp\_cluster)  
    except:  
    sscore = None  
      
    sl.append(sscore)  
      
    result = pd.DataFrame({'Datasets':datasets\_names[0:5],   
    'ARI':ari, 'AMI':ami,   
    'Homogeneity':hl,   
    'Completeness':cl,   
    'V-measure':vl, 'Silhouette':sl})  
    return result
2. cluster\_datasets\_ = [X, X\_pca, X\_tsne]  
   datasets\_names\_ = ["X", "X\_pca", "X\_tsne"]  
   cluster\_true\_y\_ = list(df.target)
3. # Кластеризация с помощью KMeans  
   claster\_metrics(KMeans(n\_clusters=3), cluster\_datasets\_, cluster\_true\_y\_, datasets\_names\_)
4. Datasets ARI AMI Homogeneity Completeness V-measure \  
   0 X 0.730238 0.755119 0.751485 0.764986 0.758176   
   1 X\_pca 0.716342 0.738655 0.736419 0.747487 0.741912   
   2 X\_tsne 0.744526 0.774848 0.769596 0.785902 0.777663   
     
    Silhouette   
   0 0.552819   
   1 0.597676   
   2 0.687133
5. claster\_metrics(AgglomerativeClustering(n\_clusters=3), cluster\_datasets\_, cluster\_true\_y\_, datasets\_names\_)
6. Datasets ARI AMI Homogeneity Completeness V-measure \  
   0 X 0.731199 0.767167 0.760801 0.779596 0.770084   
   1 X\_pca 0.744526 0.774848 0.769596 0.785902 0.777663   
   2 X\_tsne 0.759199 0.803229 0.795982 0.815646 0.805694   
     
    Silhouette   
   0 0.554324   
   1 0.598475   
   2 0.689878
7. claster\_metrics(GaussianMixture(n\_components=3), cluster\_datasets\_, cluster\_true\_y\_, datasets\_names\_)
8. Datasets ARI AMI Homogeneity Completeness V-measure \  
   0 X 0.903874 0.898436 0.898326 0.901065 0.899694   
   1 X\_pca 0.941045 0.929681 0.930042 0.931060 0.930551   
   2 X\_tsne 0.885792 0.868899 0.869675 0.871369 0.870521   
     
    Silhouette   
   0 0.501176   
   1 0.531308   
   2 0.639023