MATH 4/5388: Machine Learning Methods

Homework 4

Due date: Thursday, March 20

Submission Instruction:

- Submit both the Jupyter notebook file (.ipynb) and a PDF copy of the notebook.
- Ensure that your notebook runs properly before submitting it:
  - Kernel -> Restart & Run All to ensure that there are no errors.
- To generate a PDF of your notebook:
  - File -> Print Preview followed by printing to a PDF from your browser; or:
  - File -> Download as -> PDF via LaTeX.
- If this doesn't work, try first exporting as an HTML file and then converting that to PDF (load it in a web browser and print it to PDF).

Problem 1 (40 points)

In this problem, we explore an important mathematical property of cluster variance. Consider a data set $\{x_i\}_{i=1}^N$, where each observation x_i belongs to a cluster $C_k, k = 1, \ldots, K$. Each observation x_i is a p-dimensional vector:

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{ip}).$$

For each cluster C_k, define the centroid (mean vector):

$$\bar{x}_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i,$$

where $|C_k|$ is the number of points in cluster C_k.

Prove the following identity, which relates the average squared distance between all pairs of points in a cluster to the sum of squared deviations from the centroid:

$$\frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = 2 \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})^2.$$

Question1

Problem 2 (20 points)

In this problem, we explore the California housing dataset, leveraging location data (latitude and longitude) as features, i.e., $x \in \mathbb{R}^2$. The data set consists of geospatial information that allows us to analyze regional groupings. The following steps outline how to import and preprocess these two features for each data point.

Additionally, we visualize the data set by plotting the location data to identify possible natural clusters. Since the feature scales may differ, we apply data normalization to standardize the values before clustering.

Task: Cluster this data set into $K = 5$ components using three clustering techniques:

1. K-means Clustering
2. Gaussian Mixture Model (GMM)
3. DBSCAN

Visualize the clustering results using a scatter plot and analyze the differences between the clustering approaches.

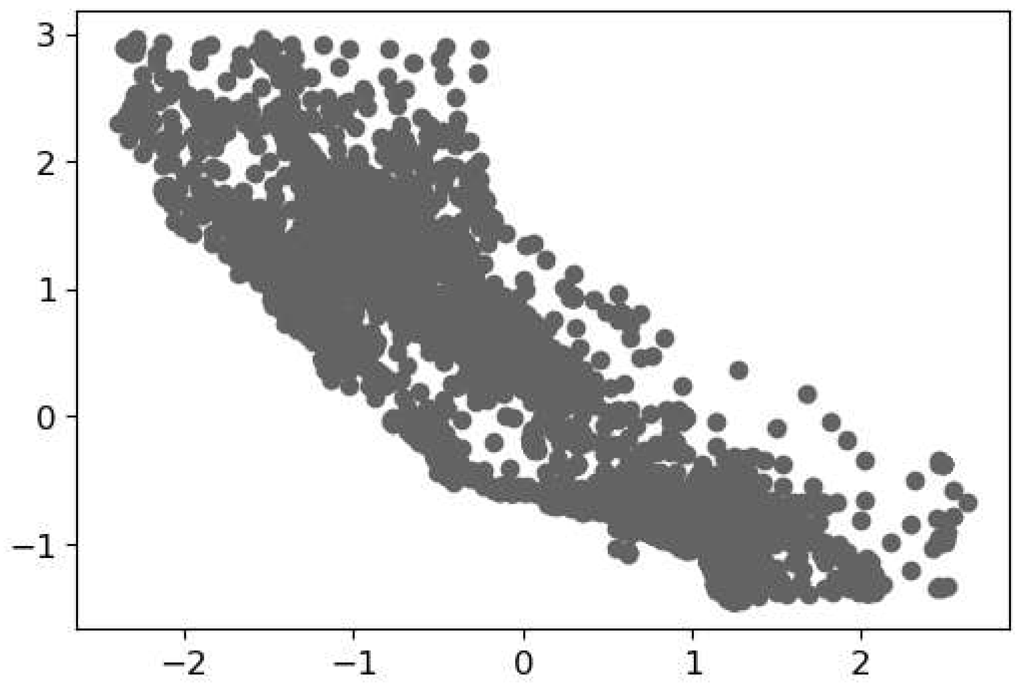
In [1]:
```
import pandas as pd

home_data = pd.read_csv('housing.csv', usecols = ['longitude', 'latitude'])

home_data.head()
```

Out[1]:

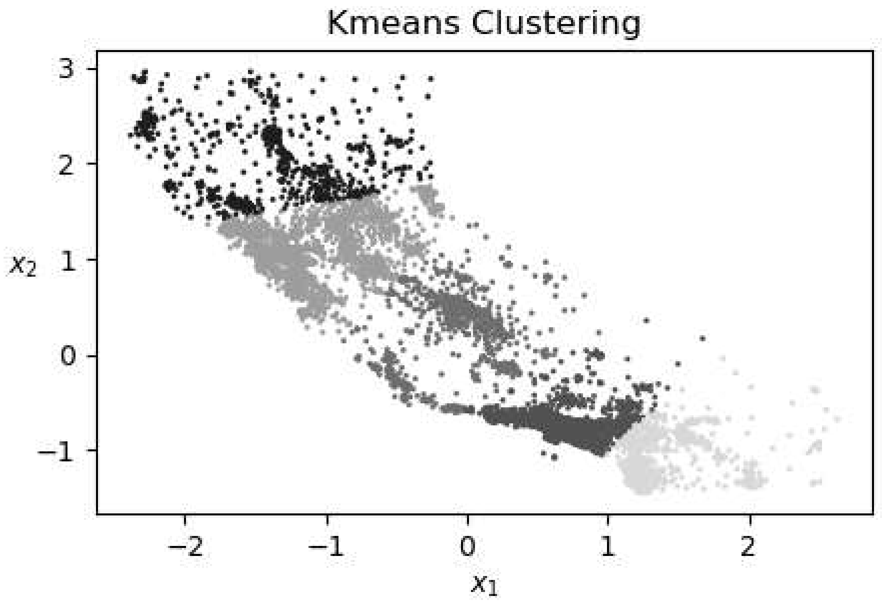
	longitude	latitude
0	-122.23	37.88
1	-122.22	37.86
2	-122.24	37.85
3	-122.25	37.85
4	-122.25	37.85

In [2]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X = home_data.to_numpy()

X_normalized = scaler.fit_transform(X)
```

In [3]:
```
import matplotlib.pyplot as plt

plt.rcParams.update({'font.size': 12, "figure.figsize": (6,4)})

plt.scatter(X_normalized[:,0], X_normalized[:,1])

plt.show()
```

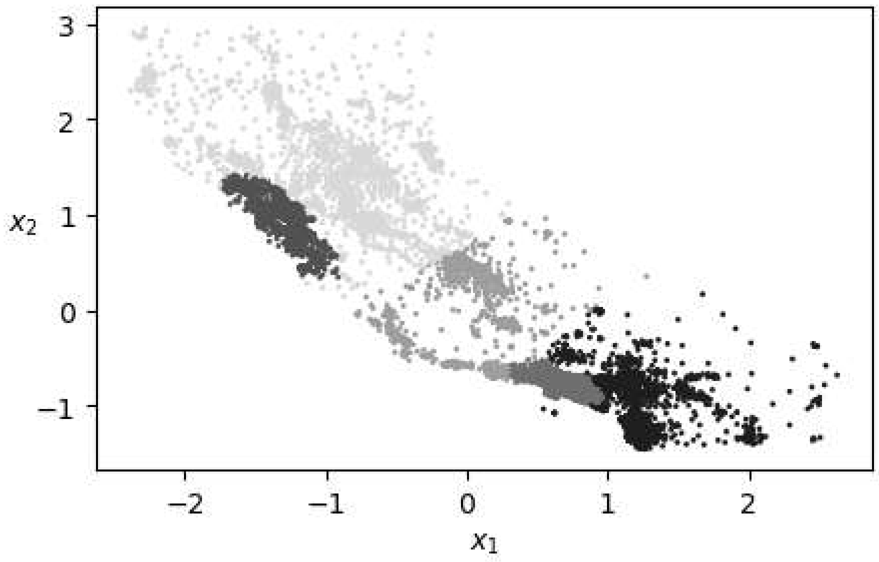


In [4]:
```
# K-means
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=5, random_state=1, n_init=20)

y_pred = kmeans.fit_predict(X_normalized)

plt.rcParams.update({'font.size': 10, "figure.figsize": (5,3)})
plt.scatter(X_normalized[:, 0], X_normalized[:, 1], c=y_pred, s=1)
plt.xlabel("$x_1$")
plt.ylabel("$x_2$", rotation=0)
plt.title('Kmeans Clustering')
plt.show()
```

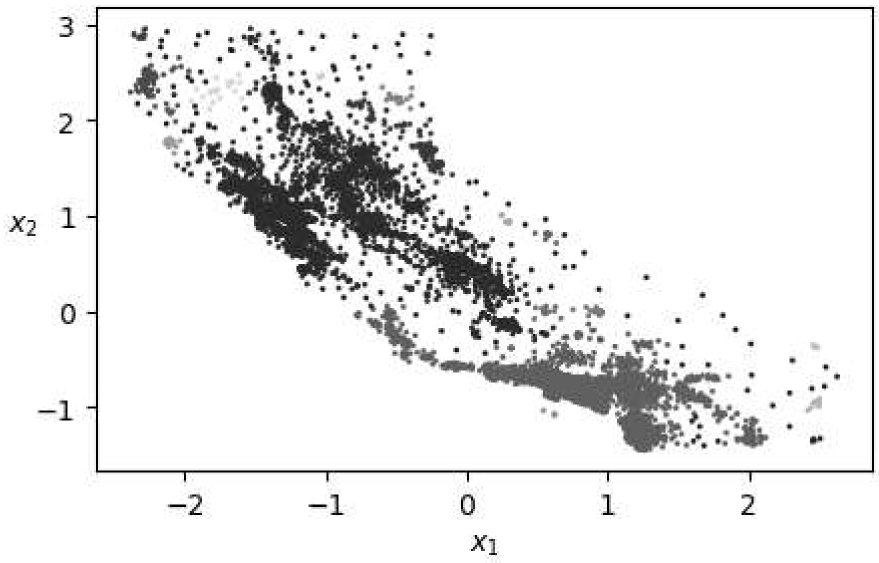


In [5]:
```
# GMM
from sklearn.mixture import GaussianMixture
```

```python
gm = GaussianMixture(n_components=5, n_init=10, random_state=42)

gm.fit(X_normalized)

y_pred = gm.fit_predict(X_normalized)

plt.scatter(X_normalized[:, 0], X_normalized[:, 1], c=y_pred, s=1)
plt.xlabel("$x_1$")
plt.ylabel("$x_2$", rotation=0)
plt.show()
```



In [6]:
```python
# DBSCAN
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=0.1, min_samples=5)

dbscan.fit(X_normalized)

plt.scatter(X_normalized[:, 0], X_normalized[:, 1], c=dbscan.labels_, s=1)
plt.xlabel("$x_1$")
plt.ylabel("$x_2$", rotation=0)
plt.show()
```



Problem 3 (40 points)

The 20 Newsgroups data set consists of thousands of news articles categorized into 20 topics (e.g., politics, sports, technology). Your task is to apply K-Means clustering to group similar articles together and evaluate the clustering performance using Normalized Mutual Information (NMI).

Understanding TF-IDF Vectorization

Before applying clustering, we transform the raw text into numerical features using **TF-IDF (Term Frequency-Inverse Document Frequency)**. TF-IDF represents the importance of words in a document relative to the entire dataset. Words that appear frequently in a document but rarely across other documents receive higher importance, making it useful for text analysis. This ensures that common words (e.g., "the", "is", "and") do not dominate the clustering process.

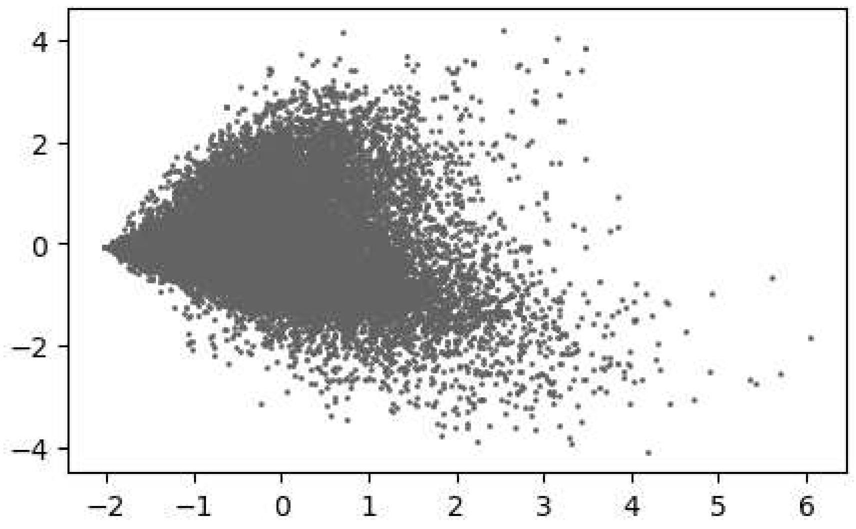
Tasks

1. Apply K-Means Clustering to the preprocessed data set.
2. Determine the best number of clusters using an appropriate method (e.g., Elbow Method or Silhouette Score).
3. Compute NMI to compare your clusters with the actual labels and analyze the clustering performance.

- Run the provided preprocessing code to generate the feature matrix.

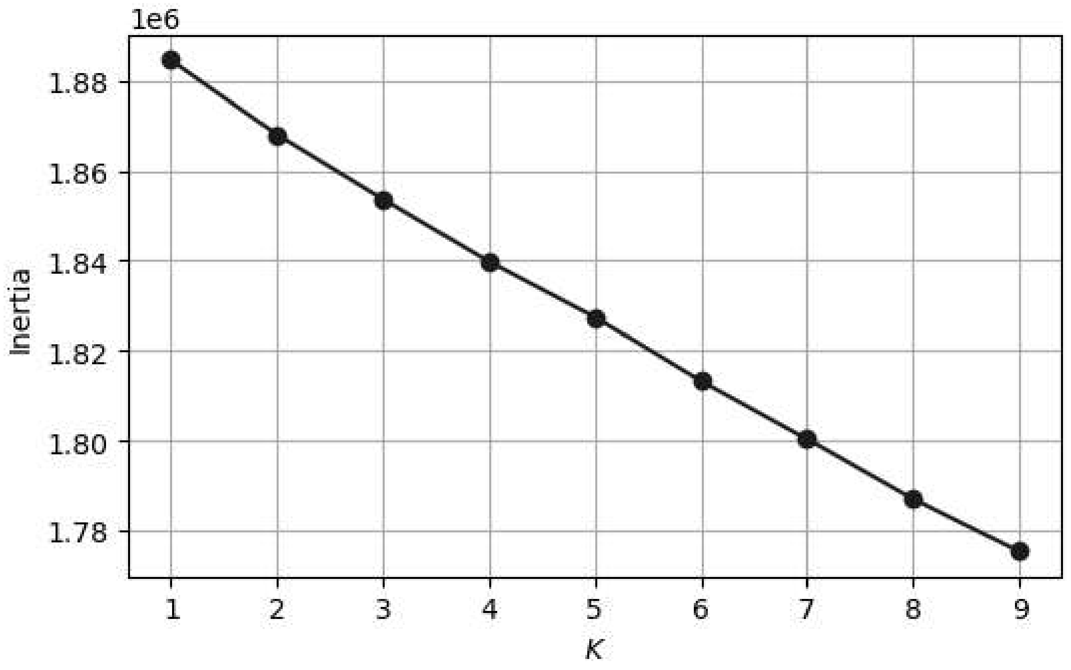
In [7]:
```
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import StandardScaler

# Load dataset
newsgroups = fetch_20newsgroups(subset='all', remove=('headers', 'footers', 'quotes'))
texts, y = newsgroups.data, newsgroups.target  # Text data and labels

# Convert text to TF-IDF features
vectorizer = TfidfVectorizer(stop_words='english', max_features=2000)
X_tfidf = vectorizer.fit_transform(texts)

# Reduce dimensionality for better clustering
svd = TruncatedSVD(n_components=100, random_state=42)
X_reduced = svd.fit_transform(X_tfidf)

# Normalize features
scaler = StandardScaler()
X = scaler.fit_transform(X_reduced)

# X is now preprocessed and ready for clustering
```

In [8]:
```
plt.scatter(X[:,0], X[:,1], s=1)

plt.show()
```

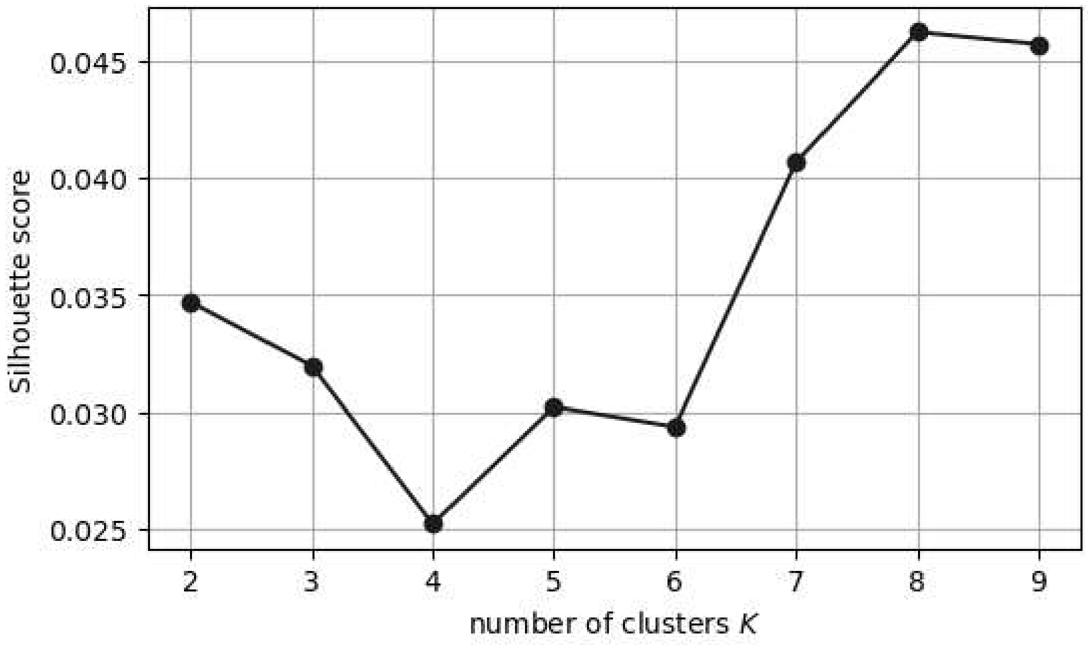


In [9]:
```
kmeans_per_k = [KMeans(n_clusters=k, random_state=42).fit(X)
                for k in range(1, 10)]
inertias = [model.inertia_ for model in kmeans_per_k]

plt.figure(figsize=(6, 3.5))
plt.plot(range(1, 10), inertias, "bo-")
plt.xlabel("$K$")
plt.ylabel("Inertia")
plt.grid()
plt.show()
```

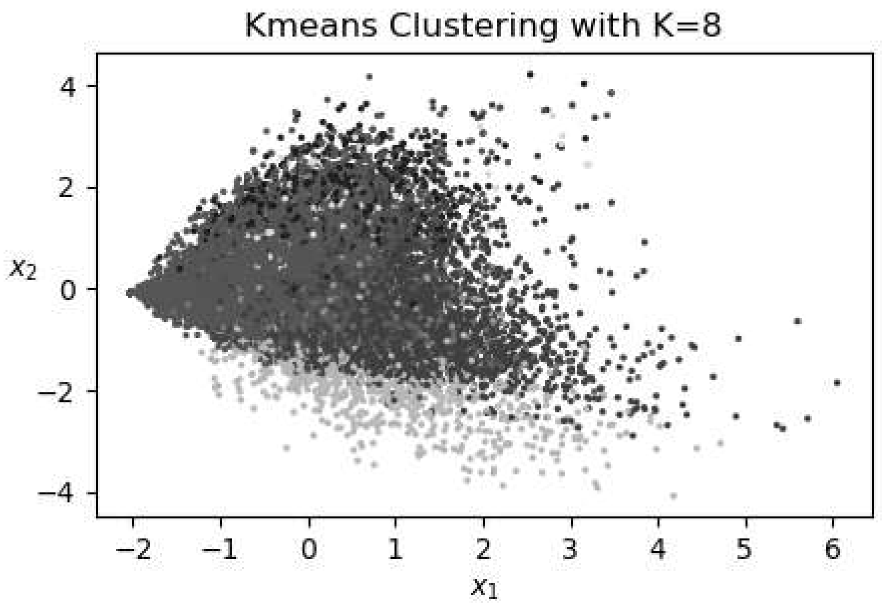
```
C:\Users\Andre\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Andre\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Andre\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Andre\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Andre\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Andre\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Andre\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Andre\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Andre\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



In [10]:
```python
from sklearn.metrics import silhouette_score

silhouette_scores = [silhouette_score(X, model.labels_)
                     for model in kmeans_per_k[1:]]

plt.figure(figsize=(6, 3.5))
plt.plot(range(2, 10), silhouette_scores, "bo-")
plt.xlabel("number of clusters $K$")
plt.ylabel("Silhouette score")
plt.grid()
plt.show()
```



In [11]:
```python
kmeans = KMeans(n_clusters=8, random_state=1, n_init=20)

y_pred = kmeans.fit_predict(X)

plt.scatter(X[:, 0], X[:, 1], c=y_pred, s=2)
plt.xlabel("$x_1$")
plt.ylabel("$x_2$", rotation=0)
plt.title('Kmeans Clustering with K=8')
plt.show()
```



```
In [12]: from sklearn.metrics import normalized_mutual_info_score

         NMI_sklearn = normalized_mutual_info_score(y, y_pred)
         print(f"Normalized Mutual Information (NMI): {NMI_sklearn:.4f}")
```

```
Normalized Mutual Information (NMI): 0.1691
```

```
In [ ]:
```