## Министерство образования Республики Беларусь

Учреждение образования

## БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Инженерно-экономический факультет

### Кафедра ЭИ

Отчет

по лабораторной работе №4

«Кластерный анализ»

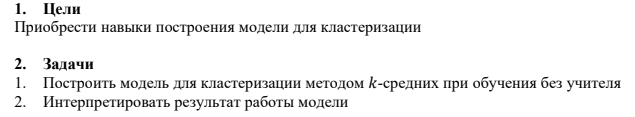
по курсу

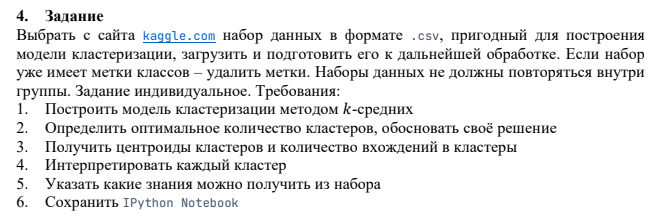
«Машинное обучение»

Вариант 7

|  |  |
| --- | --- |
| Выполнил:  Ермаков К.Ю. | Проверил:  Милентьев В.А. |

Минск 2024

****



В качестве исходных данных используется набор данных "Electricity Consumption", содержащий следующие признаки:

temperature: температура

var1: переменная 1

pressure: давление

windspeed: скорость ветра

var2: категориальный признак

*import pandas as pd*

*import numpy as np*

*import matplotlib.pyplot as plt*

*from sklearn.cluster import KMeans*

*from sklearn.preprocessing import StandardScaler, OneHotEncoder*

*from sklearn.compose import ColumnTransformer*

*from sklearn.pipeline import Pipeline*

*from sklearn.metrics import silhouette\_score*

*from scipy.spatial.distance import cdist*

*data = pd.read\_csv('test.csv')*

*print(data.head())*

*print(data.info())*

*features = ['temperature', 'var1', 'pressure', 'windspeed', 'var2']*

*X = data[features]*

*X.dropna(inplace=True)*

*X = X.astype({'temperature': float, 'var1': float, 'pressure': float, 'windspeed': float})*

*numeric\_features = ['temperature', 'var1', 'pressure', 'windspeed']*

*categorical\_features = ['var2']*

*numeric\_transformer = StandardScaler()*

*categorical\_transformer = OneHotEncoder()*

*preprocessor = ColumnTransformer(*

*transformers=[*

*('num', numeric\_transformer, numeric\_features),*

*('cat', categorical\_transformer, categorical\_features)])*

*pipeline = Pipeline(steps=[('preprocessor', preprocessor)])*

*X\_scaled = pipeline.fit\_transform(X)*

*distortions = []*

*sil\_scores = []*

*K = range(2, 11)*

*for k in K:*

*kmeans = KMeans(n\_clusters=k, random\_state=42)*

*kmeans.fit(X\_scaled)*

*distortions.append(sum(np.min(cdist(X\_scaled, kmeans.cluster\_centers\_, 'euclidean'), axis=1)) / X\_scaled.shape[0])*

*sil\_scores.append(silhouette\_score(X\_scaled, kmeans.labels\_))*

*plt.figure(figsize=(12, 6))*

*plt.subplot(1, 2, 1)*

*plt.plot(K, distortions, 'bx-')*

*plt.xlabel('Number of Clusters')*

*plt.ylabel('Distortion')*

*plt.title('Elbow Method For Optimal k')*

*plt.subplot(1, 2, 2)*

*plt.plot(K, sil\_scores, 'bx-')*

*plt.xlabel('Number of Clusters')*

*plt.ylabel('Silhouette Score')*

*plt.title('Silhouette Method For Optimal k')*

*plt.show()*

*optimal\_k = sil\_scores.index(max(sil\_scores)) + 2*

*print(f"Optimal number of clusters: {optimal\_k}")*

*kmeans = KMeans(n\_clusters=optimal\_k, random\_state=42)*

*kmeans.fit(X\_scaled)*

*centroids = kmeans.cluster\_centers\_*

*cluster\_labels = kmeans.labels\_*

*unique, counts = np.unique(cluster\_labels, return\_counts=True)*

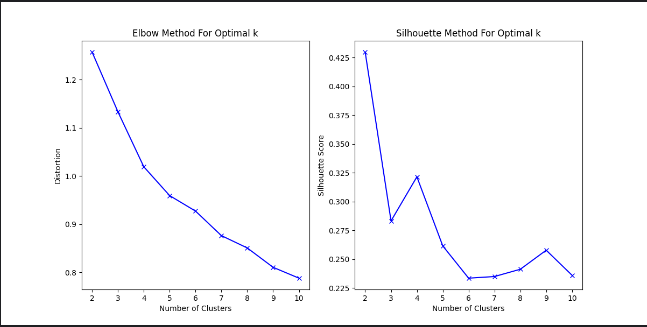
*print(f"Centroids:\n{centroids}")*

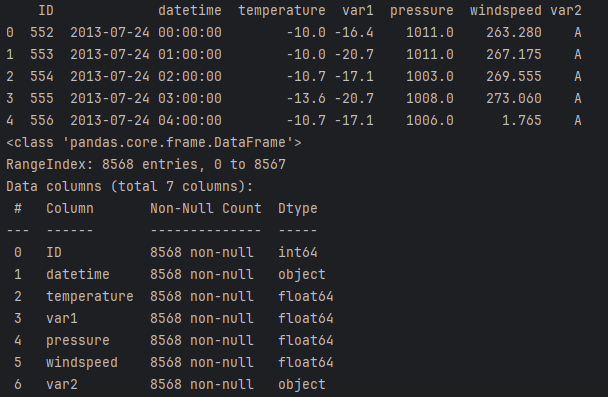
*print(f"Number of entries in each cluster:\n{dict(zip(unique, counts))}")*

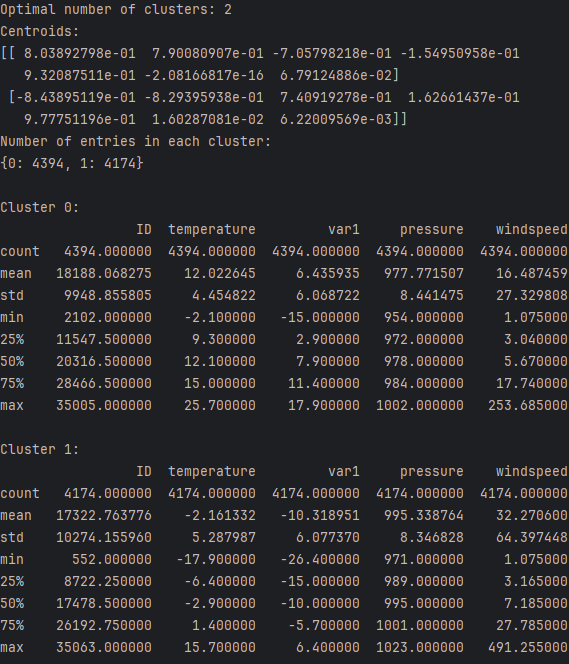
*for i in range(optimal\_k):*

*print(f"\nCluster {i}:")*

*print(data.iloc[X.index][cluster\_labels == i].describe())*







В ходе выполнения лабораторной работы была проведена кластеризация данных набора "Electricity Consumption" с использованием метода k-средних. Было определено оптимальное количество кластеров, а также проведена интерпретация каждого кластера на основе статистического анализа.