**Міністерство освіти і науки України**

**Національний технічний університет України «КПІ» імені Ігоря Сікорського**

**Кафедра обчислювальної техніки ФІОТ**

**ЗВІТ**

**з модульної контрольної**

**з навчальної дисципліни «Вступ до технології Data Science»**

**Виконав ДужеКрутийІФайнийСтудентТакий-То**

**Перевірив**

**Київ 2023**

**І. Білет №3**

**ІІ. Завдання:**

**ІІІ. Результати виконання модульної контрольної роботи.**

**3.1. Відповідь на теоретичне питання №1.**

**1. Аномальні виміри. Суть та порядок виявлення.**

Аномалії в даних - це несподівані або надто відмінні від норми значення в наборі даних. Вони можуть бути спричинені помилками в даних, непередбачуваними подіями або вказувати на певні незрозумілі закономірності.

Першим кроком є ретельне вивчення даних. Розуміння типів даних, їх розподілів та потенційних відхилень допомагає виявити аномалії.

Використання графіків і візуалізацій допомагає виділити незвичайні патерни чи викиди в даних, які можуть свідчити про аномалії.

Застосування статистичних методів, таких як середнє значення, стандартне відхилення, квантилі і т.д., дозволяє виявляти значення, які виходять за межі типових статистичних параметрів.

Використання алгоритмів машинного навчання, таких як ізоляційний ліс, однокласова машина, метод k-найближчих сусідів тощо, дозволяє автоматично виявляти аномалії в даних.

Для виявлення аномалій в реальному часі використовуються системи моніторингу, які сповіщають або автоматично реагують на надзвичайні ситуації.

Спочатку визначте, що вважається аномалією в конкретному контексті вашого дослідження.

Обробка та Підготовка Даних: Очистіть дані від помилок та пропущених значень, а також стандартизуйте їх для полегшення аналізу.

Використання Методів Виявлення Аномалій: Застосуйте відповідні методи для виявлення аномалій, враховуючи конкретні особливості вашого набору даних.

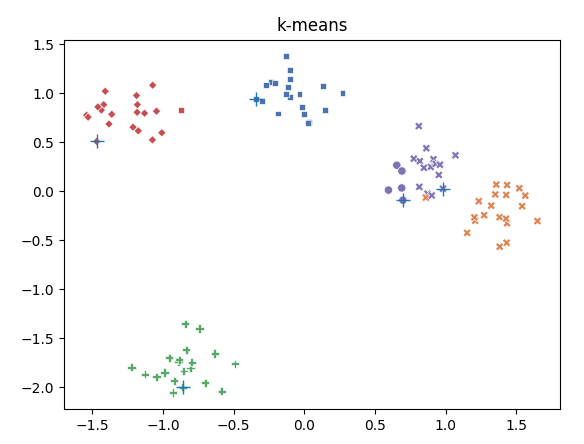
Валідація та Підтвердження: Перевірте, чи справджуються виявлені аномалії за допомогою експертного аналізу або порівняння з іншими джерелами інформації.

Взаємодія з Доменними Експертами: Залучіть фахівців у відповідній області для розуміння природи виявлених аномалій та їх можливих причин.

Виявлення аномалій важливо в різних сферах, таких як фінанси, медицина, виробництво та інші, де розуміння та вчасне реагування на непередбачувані події може мати велике значення.

**3.2. Відповідь на теоретичне питання №2.**

**2. Алгоритм k-середніх.**

** Рисунок 1. Демонстрація роботи алгоритму k середніх**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from sklearn.preprocessing import StandardScaler**

**from numpy.random import uniform**

**from sklearn.datasets import make\_blobs**

**import seaborn as sns**

**import random**

***def* most\_common(*lst*):**

**"""**

**Return the most frequently occuring element in a list.**

**"""**

**return max(set(*lst*), *key*=*lst*.count)**

***def* euclidean(*point*, *data*):**

**"""**

**Return euclidean distances between a point & a dataset**

**"""**

**return np.sqrt(np.sum((*point* - *data*)\*\*2, *axis*=1))**

***class* KMeans:**

***def* \_\_init\_\_(*self*, *n\_clusters*=8, *max\_iter*=300):**

***self*.n\_clusters = *n\_clusters***

***self*.max\_iter = *max\_iter***

***def* fit(*self*, *X\_train*):**

**# Initialize the centroids, using the "k-means++" method, where a random datapoint is selected as the first,**

**# then the rest are initialized w/ probabilities proportional to their distances to the first**

**# Pick a random point from train data for first centroid**

***self*.centroids = [random.choice(*X\_train*)]**

**for \_ in range(*self*.n\_clusters-1):**

**# Calculate distances from points to the centroids**

**dists = np.sum([euclidean(centroid, *X\_train*) for centroid in *self*.centroids], *axis*=0)**

**# Normalize the distances**

**dists /= np.sum(dists)**

**# Choose remaining points based on their distances**

**new\_centroid\_idx = np.random.choice(range(len(*X\_train*)), *size*=1, *p*=dists)[0] # Indexed @ zero to get val, not array of val**

***self*.centroids += [*X\_train*[new\_centroid\_idx]]**

**# This method of randomly selecting centroid starts is less effective**

**# min\_, max\_ = np.min(X\_train, axis=0), np.max(X\_train, axis=0)**

**# self.centroids = [uniform(min\_, max\_) for \_ in range(self.n\_clusters)]**

**# Iterate, adjusting centroids until converged or until passed max\_iter**

**iteration = 0**

**prev\_centroids = *self*.centroids**

**while np.not\_equal(*self*.centroids, prev\_centroids).any() and iteration < *self*.max\_iter:**

**# Sort each datapoint, assigning to nearest centroid**

**sorted\_points = [[] for \_ in range(*self*.n\_clusters)]**

**for x in *X\_train*:**

**dists = euclidean(x, *self*.centroids)**

**centroid\_idx = np.argmin(dists)**

**sorted\_points[centroid\_idx].append(x)**

**# Push current centroids to previous, reassign centroids as mean of the points belonging to them**

**prev\_centroids = *self*.centroids**

***self*.centroids = [np.mean(cluster, *axis*=0) for cluster in sorted\_points]**

**for i, centroid in enumerate(*self*.centroids):**

**if np.isnan(centroid).any(): # Catch any np.nans, resulting from a centroid having no points**

***self*.centroids[i] = prev\_centroids[i]**

**iteration += 1**

***def* evaluate(*self*, *X*):**

**centroids = []**

**centroid\_idxs = []**

**for x in *X*:**

**dists = euclidean(x, *self*.centroids)**

**centroid\_idx = np.argmin(dists)**

**centroids.append(*self*.centroids[centroid\_idx])**

**centroid\_idxs.append(centroid\_idx)**

**return centroids, centroid\_idxs**

**centers = 5**

**X\_train, true\_labels = make\_blobs(*n\_samples*=100, *centers*=centers, *random\_state*=42)**

**X\_train = StandardScaler().fit\_transform(X\_train)**

**kmeans = KMeans(*n\_clusters*=centers)**

**kmeans.fit(X\_train)**

**class\_centers, classification = kmeans.evaluate(X\_train)**

**sns.scatterplot(*x*=[X[0] for X in X\_train],**

***y*=[X[1] for X in X\_train],**

***hue*=true\_labels,**

***style*=classification,**

***palette*="deep",**

***legend*=None**

**)**

**plt.plot([x for x, \_ in kmeans.centroids],**

**[y for \_, y in kmeans.centroids],**

**'+',**

***markersize*=10,**

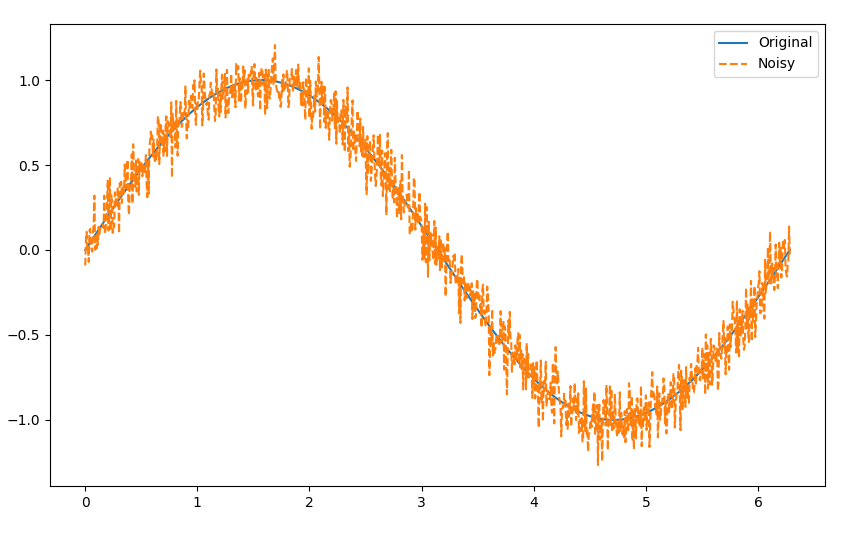
**)**

**plt.title("k-means")**

**plt.show()**

3.3. Відповідь на практичне питання №3.

3. Сформувати модель виміру за періодичним законом з нормальним законом похибки.



**Рисунок 2. Модель виміру**  за періодичним законом з нормальним законом похибки.

import numpy as np

import matplotlib.pyplot as plt

# Define parameters

amplitude = 1

frequency = 1

phase = 0

mean = 0

std\_dev = 0.1

# Generate time points

time = np.linspace(0, 2\*np.pi, 1000)

# Calculate periodic function values

values = amplitude \* np.sin(frequency \* time + phase)

# Generate normally distributed noise

noise = np.random.normal(mean, std\_dev, time.shape)

# Add noise to values

noisy\_values = values + noise

# Plot original and noisy values

plt.figure(*figsize*=(10, 6))

plt.plot(time, values, *label*='Original')

plt.plot(time, noisy\_values, *label*='Noisy', *linestyle*='dashed')

plt.legend()

plt.show()

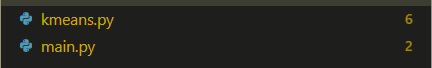


Рисунок 3. Структура проєкту

**Виконав студент: ДужеКрутийІФайнийСтудентТакий-То**