МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра програмних систем і технологій

Дисципліна

«Спеціалізоване програмування автоматизованих систем »

Екзаменаційна робота

Білет №

Виконав:	Гоша Давід Олександрович	**Перевірила**:	Ковтун Оксана Іванівна
Група	ІПЗ-33	Дата перевірки	
Форма навчання	денна	Бали	
Спеціальність	121		
2023			

Завдання:

Проведіть кластерізацію даних за алгоритмом k-means, використавши відстань Мінковського, із застосуванням бібліотеки Scikit-Learn.

- Підготуйте набір даних з файлу «seeds.csv».
- Виведіть у консоль координати центроїдів кожного кластера.
- Виведіть у консоль загальну внутрішньо-кластерну суму квадратів відстаней від екземплярів до найближчого центроїда.
- Виведіть у консоль кількість ітерацій, які виконує алгоритм k-means.
- Побудуйте графіки результатів кластеризації для декількох параметрів попарно (не менше 3-х), додайте на графіки центроїди.
- Порахуйте кількість екземплярів у кожному кластері, порівняйте з відомим розподілом на класи.

Код:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from itertools import combinations

def load_and_scale_data(filename):
    # Load data
    df = pd.read_csv(filename)
    X = df.drop(columns=['class']).values

    # Scale the data
    sc = StandardScaler()
    X = sc.fit_transform(X)

    return X, df['class'].values

def run_kmeans(X, n_clusters=3):
    # KMeans
    kmeans = KMeans(n_clusters=n_clusters, random_state=0)
    kmeans.fit(X)
    return kmeans

def print_kmeans_info(kmeans):
```

```
    print("Centroids: ", kmeans.cluster_centers_)
    print("Sum of squared distances: ", kmeans.inertia_)
    print("Number of iterations: ", kmeans.n_iter_)

def plot_clusters(X, kmeans):
    pairs = list(combinations(range(X.shape[1]), 2))[:3]
    for i, (feat1, feat2) in enumerate(pairs):
        plt.figure(i)
        plt.scatter(X[:, feat1], X[:, feat2],
c=kmeans.labels_)
        plt.scatter(kmeans.cluster_centers_[:, feat1],
kmeans.cluster_centers_[:, feat2], s=300, c='red')
        plt.show()

def count_cluster_instances(kmeans):
    clusters, counts = np.unique(kmeans.labels_,
return_counts=True)
    for cluster, count in zip(clusters, counts):
        print(f"Cluster {cluster}: {count} instances")

def compare_to_classes(kmeans, classes):
    class_counts = np.unique(classes, return_counts=True)
    for class_, count in zip(*class_counts):
        print(f"Class {class_}: {count} instances")

def main():
    X, classes = load_and_scale_data('seeds.csv')
    kmeans = run_kmeans(X)
    print_kmeans_info(kmeans)
    plot_clusters(X, kmeans)
    count_cluster_instances(kmeans)
    compare_to_classes(kmeans, classes)

if __name__ == "__main__":
    main()
```