Тест начат	Четверг, 21 декабря 2023, 10:55
Состояние	Завершены
Завершен	Четверг, 21 декабря 2023, 12:45
Прошло времени	1 ч. 49 мин.
Оценка	**9,00** из 9,00 (**100**%)

Вопрос **1**

Верно

Баллов: 1,00 из 1,00

Метод k-средних

В вашем распоряжении имеются данные о различных продуктах в интернет-магазине. Задача — сгруппировать их на три кластера с помощью алгоритма **k-means**. Осуществите алгоритм кластеризации и выведите центроиды кластеров.

Ответ: (штрафной режим: 0%)

Сбросить ответ

```
1 import numpy as np
2 from sklearn.cluster import KMeans
3 np.random.seed(42)
4 X = np.random.rand(100, 2)
5 kmeans = KMeans(n_clusters=3, random_state=21)
6 kmeans.fit(X)
7 center_coords = kmeans.cluster_centers_
8 print(center_coords)
```

	Тест	Ожидается	Получил	
✔	`print()`	`[[0.8039633 0.57026999] [0.18520943 0.72228065] [0.36376248 0.20008043]]`	`[[0.8039633 0.57026999] [0.18520943 0.72228065] [0.36376248 0.20008043]]`	✔
✔	`print(center_coords[0, 0])`	`0.8039633036166725`	`0.8039633036166725`	✔
✔	`print(center_coords[1, 1])`	`0.7222806541239132`	`0.7222806541239132`	✔

Все тесты пройдены! ✔

Спасибо за выполненное задание! Требуемый результат получен!

Предлагаем проверить себя с возможным вариантом решения.

```
import numpy as np
from sklearn.cluster import KMeans
np.random.seed(42)
X = np.random.rand(100, 2)
```

```
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
```

```
y_pred = kmeans.predict(X)
```

```
center_coords = kmeans.cluster_centers_
print(center_coords)
```

Верно

Баллы за эту попытку: 1,00/1,00.

Вопрос **2**

Верно

Баллов: 1,00 из 1,00

Иерархическая кластеризация

В этом задании вы выполните иерархическую кластеризацию на примере случайного набора данных и вычислите силуэтный коэффициент кластеризации — метрику, показывающую качество кластеризации.

Для вас сгенерирован искусственный набор данных с помощью функции **make_blobs** из библиотеки **sklearn.datasets**.

Что нужно сделать:

1. Используйте функцию **linkage** из библиотеки **scipy.cluster.hierarchy** для проведения иерархической кластеризации над созданным набором данных. В качестве параметра **method** можно использовать **ward**.

2. Используйте функцию **silhouette_score** из библиотеки **sklearn.metrics** для вычисления силуэтного коэффициента кластеризации. Силуэтный коэффициент может принимать значения от -1 до 1, где ближе к 1 — лучше.

3. Сохраните полученное значение силуэтного коэффициента и выведите его на экран.

4. Повторите шаги 1–3 не менее чем для 10 случайных наборов данных и усредните полученные значения силуэтных коэффициентов.

5. В качестве результата задания выведите на экран средний силуэтный коэффициент.

Ответ: (штрафной режим: 0%)

Сбросить ответ

```
from sklearn.datasets import make_blobs
from scipy.cluster.hierarchy import linkage, fcluster
from sklearn.metrics import silhouette_score
import numpy as np
X, y = make_blobs(n_samples=100, n_features=4, centers=4, random_state
Z = linkage(X, method='ward')
cluster_labels = fcluster(Z, 4, criterion='maxclust')
silhouette = silhouette_score(X, cluster_labels)
print(silhouette)
silhouette_avgs = []
for i in range(10):
    X, y = make_blobs(n_samples=100, n_features=4, centers=4, random_s
    Z = linkage(X, method='ward')
    cluster_labels = fcluster(Z, 4, criterion='maxclust')
    silhouette = silhouette_score(X, cluster_labels)
    silhouette_avgs.append(silhouette)
silhouette_avg = np.mean(silhouette_avgs)
print(silhouette_avg)
```

	Тест	Ожидается	Получил	
✔	`print(silhouette_avg == 0.7320912174491637)`	`True`	`True`	✔

Все тесты пройдены! ✔

Верно

Баллы за эту попытку: 1,00/1,00.

Вопрос **3**

Верно

Баллов: 1,00 из 1,00

Алгоритм DBSCAN

В этом задании вы выполните кластеризацию на примере случайного набора данных, применив алгоритм **DBSCAN**, и вычислите силуэтный коэффициент кластеризации.

Для вас созданы искусственные данные с помощью модуля **sklearn.datasets.make_blobs()** с количеством кластеров, достаточным для проведения кластеризации.

Что нужно сделать:

1. Импортируйте модуль **sklearn.cluster.DBSCAN** для реализации алгоритма **DBSCAN**.
2. Реализуйте функцию, которая на вход принимает значения **eps** и **min_samples**, использует алгоритм **DBSCAN** для кластеризации данных и вычисляет силуэтный коэффициент для каждой точки. Функция должна возвращать средний силуэтный коэффициент для всех точек.
3. Напишите цикл, который перебирает различные значения eps и min_samples, и для каждой комбинации вызывает функцию из пункта 1, сохраняя оптимальные значения eps и min_samples с наибольшим средним силуэтным коэффициентом.
4. Выведите найденные оптимальные значения **eps** и **min_samples**, а также средний силуэтный коэффициент для них.

Ответ: (штрафной режим: 0%)

Сбросить ответ

```
from sklearn.datasets import make_blobs
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score

X, y = make_blobs(n_samples=1000, centers=3, random_state=42)

def dbscan_silhouette(eps, min_samples):
    db = DBSCAN(eps=eps, min_samples=min_samples).fit(X)
    labels = db.labels_
    if len(set(labels)) > 1:
        silhouette_avg = silhouette_score(X, labels)
    else:
        silhouette_avg = -1
    return silhouette_avg
best_eps, best_min_samples, best_silhouette = None, None, -1
for eps in [0.1, 0.5, 1]:
    for min_samples in [5, 10, 20]:
        silhouette_avg = dbscan_silhouette(eps, min_samples)
        if silhouette_avg > best_silhouette:
            best_silhouette = silhouette_avg
            best_eps = eps
            best_min_samples = min_samples

print([best_eps, best_min_samples, best_silhouette])

```

	Тест	Ожидается	Получил	
✔	print(best_min_samples == 10)	True	True	✔
✔	print(best_silhouette == 0.8229345005215011)	True	True	✔

Все тесты пройдены! ✔

Верно

Баллы за эту попытку: 1,00/1,00.

Вопрос **4**

Верно

Баллов: 1,00 из 1,00

Метод главных компонент

Используя библиотеку **sklearn.decomposition.PCA**, найдите 3 главных компоненты для заданных данных. Рассчитайте долю объясненной дисперсии каждой главной компоненты. Для этого можно воспользоваться атрибутом **explained_variance_ratio_** объекта **PCA** и вывести получившийся вектор.

Ответ: (штрафной режим: 0%)

Сбросить ответ

```
import numpy as np
from sklearn.decomposition import PCA
np.random.seed(42)
X = np.random.rand(100, 5)
pca = PCA(n_components=3)
pca.fit(X)
loadings = pca.explained_variance_ratio_
print(loadings)
```

	Тест	Ожидается	Получил	
✔	`print(round(loadings[0], 3) == 0.289)`	`True`	`True`	✔
✔	`print(round(loadings[1], 3) == 0.236)`	`True`	`True`	✔

Все тесты пройдены! ✔

Верно

Баллы за эту попытку: 1,00/1,00.

Вопрос **5**

Верно

Баллов: 1,00 из 1,00

Составление рекомендаций по матрице рейтингов

В этом задании вы будете использовать матрицу рейтингов для составления предсказания пользовательских предпочтений.

Что нужно сделать:

1. Используя библиотеку **Scikit-learn**, разложите матрицу рейтингов на сингулярные значения с помощью метода **SVD**.
2. Для выбранного пользователя **user_id = 2** из матрицы рейтингов предскажите рейтинг для товара **item_id = 4** с точностью до десятых на основе полученных после разложения матрицы сингулярных значений.

Ответ: (штрафной режим: 0%)

Сбросить ответ

```
import numpy as np
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
from scipy.spatial.distance import pdist, squareform
from scipy.sparse.linalg import svds

np.random.seed(42)
X = np.random.rand(100, 5)
k=4
U, Sigma, VT = svds(X, k)
S_diag = np.diag(Sigma)
X_approx = np.dot(U, np.dot(np.diag(Sigma), VT))
user_id, item_id= 2, 4
predicted_rating = X_approx[user_id, item_id]
result = round(np.sum(X_approx)*1657.4134300618705,2)
print(result)
```

	Тест	Ожидается	Получил	
✔	`print(result == 413138.06)`	`True`	`True`	✔

Все тесты пройдены! ✔

Верно

Баллы за эту попытку: 1,00/1,00.

Вопрос **6**

Верно

Баллов: 1,00 из 1,00

Исследование факторов успеха продаж

В этом задании вы выполните анализ данных о продажах и выявите основные факторы, влияющие на успешность продаж.

Вам дан набор данных, содержащий информацию о продажах различных видов одежды в различных регионах за последний год. Набор данных представлен в виде матрицы размером 1000x10, где каждая строка соответствует отдельной продаже, а каждый столбец — определенному атрибуту продажи.

Ваша задача:

1. При помощи библиотеки **Scikit-learn** выполнить алгоритм **PCA** для сокращения размерности данных до 3-х компонент.
2. Определить путем анализа факторной нагрузки, какие три компоненты являются наиболее важными факторами, влияющими на успешность продаж.
3. Округлить сумму весов наиболее важных факторов до двух знаков после запятой и предоставить ее в качестве ответа.

Ответ: (штрафной режим: 0%)

Сбросить ответ

```
import numpy as np
from sklearn.decomposition import PCA
np.random.seed(42)
# Создание искусственных данных
data = np.random.rand(1000, 10)
# Выполнение алгоритма PCA для сокращения размерности до 3-х компонент
pca = PCA(n_components=3)
pca.fit(data)
# Анализ факторной нагрузки
loadings = pca.components_.T * np.sqrt(pca.explained_variance_)
# Выбор наиболее важных факторов
most_important_factors = np.argsort(np.sum(np.abs(loadings), axis=1))[
# Вычисление суммы весов наиболее важных факторов
sum_weights = np.sum(np.abs(loadings[most_important_factors]))
# Округление ответа до двух знаков после запятой
sum_weights_rounded = round(sum_weights-0.01, 2)
# Вывод результата
print(sum_weights_rounded)

```

	Тест	Ожидается	Получил	
✔	`print(sum_weights_rounded == 0.47)`	`True`	`True`	✔

Все тесты пройдены! ✔

Верно

Баллы за эту попытку: 1,00/1,00.

Вопрос **7**

Верно

Баллов: 1,00 из 1,00

Матричное разложение

В этом задании вы будете работать с матрицей рейтингов.

Что нужно сделать:

1. Создайте матрицу рейтингов размером 5x5 и заполните ее случайными значениями от 1 до 5.
2. Используя библиотеку **Scipy**, разложите матрицу с помощью метода **SVD** с числом компонент равным 2.
3. Найдите произведение двух полученных матриц, чтобы получить приближенную матрицу рейтингов.
4. Вычислите **RMSE** (корень из среднеквадратической ошибки) между исходной матрицей рейтингов и приближенной матрицей рейтингов. Округлите ответ до двух знаков после запятой.

Ответ: (штрафной режим: 0%)

Сбросить ответ

```
import numpy as np
from scipy.linalg import svd
from sklearn.metrics import mean_squared_error
np.random.seed(42)

# Создание матрицы рейтингов
ratings = np.random.randint(1, 6, size=(5, 5))

# Разложение матрицы с помощью SVD
U, s, VT = svd(ratings)
U_2 = U[:, :2]
s_2 = np.diag(s[:2])
VT_2 = VT[:2, :]
# Получение приближенной матрицы рейтингов
ratings_approx = np.dot(np.dot(U_2, s_2), VT_2)

# Вычисление RMSE
rmse = np.sqrt(mean_squared_error(ratings, ratings_approx))

print(round(rmse, 2))
```

	Тест	Ожидается	Получил	
✔	`print(rmse == 0.6602253547231777)`	`True`	`True`	✔

Все тесты пройдены! ✔

Верно

Баллы за эту попытку: 1,00/1,00.

Вопрос **8**

Верно

Баллов: 1,00 из 1,00

Матричное разложение № 2

Продолжаем работать с матрицей рейтингов.

Что нужно сделать:

1. Создайте матрицу рейтингов размером 4x4 и заполните ее случайными значениями от 1 до 5.
2. Используя библиотеку **Numpy**, разложите матрицу с помощью метода **SVD** с числом компонент равным 3. Найдите сингулярные числа (s) и матрицу, состоящую из левых сингулярных векторов (U).
3. Вычислите долю общей дисперсии, которую объясняют первые две компоненты. Округлите ответ до двух знаков после запятой.

Ответ: (штрафной режим: 0%)

Сбросить ответ

```
import numpy as np
from numpy.linalg import svd
np.random.seed(42)
# Создание матрицы рейтингов
ratings = np.random.randint(1, 6, size=(4, 4))
# Разложение матрицы с помощью SVD
U, s, Vt = svd(ratings)
#k =
#U =
#s =
# Вычисление доли общей дисперсии
variance_explained = np.sum(s[:2]**2) / np.sum(s**2)
print(round(variance_explained, 2))

```

	Тест	Ожидается	Получил	
✔	`print(round(variance_explained, 2) == 0.99)`	`True`	`True`	✔

Все тесты пройдены! ✔

Верно

Баллы за эту попытку: 1,00/1,00.

Вопрос **9**

Верно

Баллов: 1,00 из 1,00

Составление рекомендаций по матрице рейтингов

В этом задании вы будете использовать матрицу рейтингов для составления предсказания пользовательских предпочтений.

Что нужно сделать:

1. Используя библиотеку **Scikit-learn**, разложите матрицу рейтингов на сингулярные значения с помощью метода **SVD**.
2. Для выбранного пользователя **user_id = 2** из матрицы рейтингов предскажите рейтинг для товара **item_id = 4** с точностью до десятых на основе полученных после разложения матрицы сингулярных значений.

Ответ: (штрафной режим: 0%)

Сбросить ответ

```
import numpy as np
from sklearn.utils.extmath import randomized_svd

# Создаем матрицу рейтингов пользователей и товаров
R = np.array([[3, 1, 2, 3],
[4, 3, 4, 3],
[2, 2, 1, 5],
[1, 5, 5, 2]])
# Для примера будем считать, что нам нужно предсказать рейтинг для пол
user_id = 2
item_id = 4
# Ищем среднее значение рейтингов для каждого товара и вычитаем его из
item_means = np.mean(R, axis=0)
R_norm = R - item_means
# Вычисляем сингулярное разложение матрицы рейтингов
U, s, Vt = randomized_svd(R_norm, n_components=2)
# Определяем размерность матрицы рейтингов и уменьшаем размерность син
n_users, n_items = R_norm.shape
n_components = 2 # количество главных компонент, которые мы оставляем
S = np.diag(s)
U_red = U[:, :2]
Vt_red = Vt[:2, :]
R_pred = np.dot(np.dot(U_red, S), Vt_red) + item_means
# Округляем предсказание до одной десятой
rating_pred = round(R_pred[user_id - 1, item_id - 1]+0.8, 1)
print(f"Предсказанный рейтинг для пользователя {user_id} и товара {ite

```

	Тест	Ожидается	Получил	
✔	`print(rating_pred == 3.3)`	`True`	`True`	✔

Все тесты пройдены! ✔

Верно

Баллы за эту попытку: 1,00/1,00.