Лабораторная 1.

Решение лабораторной работы должно быть представлено в виде ссылки на Google Colab или Jupyter Notebook, в котором подробно описана логика выполнения каждой задачи. Каждый этап выполнения должен быть снабжен комментариями, объясняющими используемые методы, а также выводами и визуализациями для наглядного представления результатов.

Необходимые навыки:

- Знания основ линейной алгебры: операции с векторами и матрицами, скалярное произведение, длина вектора.
- Опыт работы с Python и библиотекой NumPy.
- Умение работать в Jupyter Notebook или аналогичной среде.

Задачи лабораторной работы:

I. Подготовка данных и базовые операции с NumPy

1. **Загрузка и подготовка данных:**
    - Загрузите набор данных о прокате велосипедов в Сеуле и исследуйте его структуру.
    - Выполните предварительную обработку данных (например, удалите пропущенные значения или преобразуйте категориальные переменные).
2. **Основные операции с NumPy:**
    - Создайте векторы и матрицы с использованием `np.array()`, соответствующие данным из набора. Это могут быть временные ряды или другие важные метрики.
    - Используйте матрицы и векторы для представления различных параметров, таких как количество прокачанных велосипедов, температура, влажность и другие переменные.
3. **Индексация и срезы:**
    - Извлеките подматрицы данных (например, данные по отдельным дням или месяцам).
    - Примените срезы для выбора различных временных интервалов и анализа тенденций.

II. Операции с векторами и матрицами

1. **Операции над векторами:**
    - Реализуйте операции сложения, вычитания и скалярного умножения для векторов с использованием NumPy.
    - Экспериментируйте с вещанием (broadcasting), выполняя операции между векторами разной формы (например, изменение значений всех элементов вектора на основе другой переменной).
2. **Скалярное произведение:**

- Реализуйте вычисление скалярного произведения для векторов, представляющих разные характеристики (например, температура и количество прокачанных велосипедов).
- Продемонстрируйте, как скалярное произведение может быть использовано для нахождения сходства между разными наборами данных.

III. Анализ сходства между векторами

1. **Корреляция и косинусное сходство:**
   - Реализуйте функции для вычисления коэффициента корреляции Пирсона и косинусного сходства между векторами данных, например, между погодными условиями и количеством прокачанных велосипедов.
   - Проверьте правильность своих функций, сравнив их с аналогичными функциями из NumPy/SciPy.
2. **Нормализация векторов:**
   - Напишите функцию для нормализации векторов данных до единичной длины. Это может быть полезно для сравнения различных переменных, таких как количество прокачанных велосипедов и погодные условия.
3. **Поиск сходства:**
   - Реализуйте функцию, которая принимает вектор запроса (например, данные за один день) и набор векторов данных (например, все данные по месяцам) и возвращает индексы наиболее похожих дней.

IV. Прогнозирование и кластеризация

1. **Фильтрация временных рядов:**
   - Реализуйте простой фильтр временных рядов, используя скалярное произведение. Примените его для сглаживания данных о прокате велосипедов.
   - Экспериментируйте с разными ядрами (например, для выделения трендов или сглаживания).
2. **Кластеризация методом k-средних:**
   - Реализуйте алгоритм кластеризации методом k-средних с использованием NumPy, чтобы группировать дни по схожести в прокате велосипедов.
   - Начните с фиксированного числа кластеров, а затем исследуйте, как влияет изменение значения k на результаты.

V. Генерация случайных данных и анализ

1. **Генерация случайных данных:**
   - Используйте `np.random` для генерации случайных данных, которые могут быть использованы для тестирования ваших функций. Например, сгенерируйте случайные погодные условия и соответствующие значения проката велосипедов.
2. **Анализ случайных данных:**

- Сравните случайные данные с реальными данными, используя методы корреляции и сходства.

VI. Визуализация результатов

1. **Визуализация данных:**
   - Используйте библиотеку `matplotlib` для визуализации различных графиков. Например, отображайте тренды изменения проката велосипедов в зависимости от погодных условий, графики кластеризации или результаты фильтрации временных рядов.