Министерство науки и высшего образования РФ Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Отчёт по модулю 2

по дисциплине

“Системы искусственного интеллекта”

Выполнил:

Студент группы P3330,

Тарасов Иван Сергеевич

Проверила:

Авдюшина Анна Евгеньевна

 Санкт-Петербург

2024

**Лабораторная 3. Линейная регрессия**

* Получить и визуализировать (графически) статистику по датасету (включая количество, среднее значение, стандартное отклонение, минимум, максимум и различные квантили).
* Провести предварительную обработку данных, включая обработку отсутствующих значений, кодирование категориальных признаков и нормировка.
* Разделить данные на обучающий и тестовый наборы данных.
* Реализовать линейную регрессию с использованием метода наименьших квадратов без использования сторонних библиотек, кроме NumPy и Pandas (для использования коэффициентов использовать библиотеки тоже нельзя). Использовать минимизацию суммы квадратов разностей между фактическими и предсказанными значениями для нахождения оптимальных коэффициентов.
* Построить **три модели** с различными наборами признаков.
* Для каждой модели провести оценку производительности, используя метрику коэффициент детерминации, чтобы измерить, насколько хорошо модель соответствует данным.
* Сравнить результаты трех моделей и сделайте выводы о том, какие признаки работают лучше всего для каждой модели.

**Описание**: линейная регрессия моделирует линейную зависимость между независимыми переменными (признаками) и зависимой переменной (искомым признаком). Основная идея заключается в подборе коэффициентов для функции линейной регрессии вида так, чтобы значение функции потерь была минимальная. При использовании метода наименьших квадратов задача сводится к тому, чтобы подобрать такие коэффициенты , чтобы сумма квадратов, которые образовываются между точкой пересечения осей и координатой признака , была минимальной.

Псевдокод

Матрица А  
Повторить столько раз, сколько обучающихся данных:  
 Повторить столько раз, сколько раз, сколько признаков плюс 1:  
 Если первый столбец матрицы данных или любой другой столбец равен 1:

Иначе:

y = вектор данных признаков  
Коэффициенты =

Линейная регрессия для независимого признака “Hours Studied” и зависимого признака “Performance Index”

Изображение выглядит как снимок экрана, линия, диаграмма, График

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Применение:** Прогнозирование непрерывных данных, таких как цены, температуры, или доходы.

**Лабораторная 4. Метод k-ближайших соседей (k-NN)**

* Провести предварительную обработку данных, включая обработку отсутствующих значений, кодирование категориальных признаков и масштабирование.
* Получить и визуализировать графически статистику по датасету (включая количество, среднее значение, стандартное отклонение, минимум, максимум и различные квантили), постройте 3d-визуализацию признаков.
* Реализовать метод k-ближайших соседей без использования сторонних библиотек, кроме NumPy и Pandas.
* Построить две модели k-NN с различными наборами признаков:
  + Модель 1: Признаки случайно отбираются.
  + Модель 2: Фиксированный набор признаков, который выбирается заранее.
* Для каждой модели провести оценку на тестовом наборе данных при разных значениях k. Выбрать несколько различных значений k, например, k=3, k=5, k=10, и т. д. Постройте матрицу ошибок.

**Описание**: предсказывает класс или значение объекта на основе класса/значений k-ближайших соседей в пространстве признаков. При малом значении k модель более чувствительна к локальным особенностям данных, что может быть полезно для выявления сложных, локальных закономерностей. Однако увеличивается чувствительность к шуму и выбросам. При большом значении k модель становится более устойчивой и менее склонной к переобучению. Однако возможна потеря локальных особенностей данных, так как решение усредняется по большему числу соседей.

Псевдокод

features = *список из названий признаков*y\_list = *список из значение*near\_dict = *словарь из расстояний до точек искомого признака*  
Повторить столько раз, сколько обучающихся данных:  
 cur\_dist = *расстояние (Минковского) до другой точки*  
 near\_dict[cur\_dist] = *значение зависимого признака*  
  
near\_dict = *сортировка словаря по ключам (расстоянию)*  
sum\_classes = {0: 0, 1: 0}  
Повторить k раз:  
 подсчёт количество ближайших точек по классам

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Применение:**

1. Распознавание образов: k-NN используется для классификации изображений, например, для распознавания цифр (MNIST dataset) или лиц.
2. Рекомендательные системы: Метод может применяться для поиска пользователей, похожих на заданного, и рекомендаций продуктов или контента.
3. Анализ текстов: k-NN может использоваться для классификации текста, например, в задачах спам-фильтрации или определения тематики текста.
4. Медицинская диагностика: для классификации заболеваний по медицинским признакам, таким как результаты анализов, симптомы, возраст и т.д.
5. Финансовый анализ: для предсказания стоимости акций или других финансовых данных на основе исторических значений.

**Лабораторная 5. Деревья решений**

1. Отобрать **случайным** образом sqrt(n) признаков
2. Реализовать без использования сторонних библиотек построение дерева решений (дерево не бинарное, numpy и pandas использовать можно, использовать список списков для реализации дерева - нельзя) для решения задачи бинарной классификации
3. Провести оценку реализованного алгоритма с использованием Accuracy, precision и recall
4. Построить кривые AUC-ROC и AUC-PR (в пунктах 4 и 5 использовать библиотеки нельзя)

Псевдокод

info = Оценка среднего количества информации  
если info равен 0:  
 узел = лист

значение листа – “N”, если значение меньше порогового, иначе “Y”  
info\_x = Оценка среднего количества информации после разбиения  
split = Оценка потенциальной информации  
  
max\_x = бесконечность  
max\_value = ''

Повторить столько, сколько элментов в наборе info\_x  
 gain = (info - info\_x[i]) / split[i] если split[i] не 0 иначе бесконечность  
 Если gain больше текущего максимального (max\_x), перезаписать gain и max\_value

признак узла = max\_value

Повторить столько, сколько уникальных значений, которое может принять признак:

Создать рекурсивно новый узел и подсчитать количество N и Y листов

Изображение выглядит как текст, диаграмма, График, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, График, диаграмма, снимок экрана

Автоматически созданное описание

21 -> 1(4), 3(16), 2(<N>),

21/1(4) -> 3(18), 4(8), 5(8), 2(2), 1(<Y>),

21/1(4)/3(18) -> 2(5), 1(<N>), 3(2),

21/1(4)/3(18)/2(5) -> 1(3), 2(24),

21/1(4)/3(18)/2(5)/1(3) -> 3(9), 2(<N>), 1(<N>),

21/1(4)/3(18)/2(5)/1(3)/3(9) -> 1(15),

21/1(4)/3(18)/2(5)/1(3)/3(9)/1(15) -> 2(19),

21/1(4)/3(18)/2(5)/1(3)/3(9)/1(15)/2(19) -> 2(20),

21/1(4)/3(18)/2(5)/1(3)/3(9)/1(15)/2(19)/2(20) -> 1(26),

21/1(4)/3(18)/2(5)/1(3)/3(9)/1(15)/2(19)/2(20)/1(26) -> 2(<N>), 3(<Y>),

21/1(4)/3(18)/2(5)/2(24) -> 1(25), 2(<Y>),

21/1(4)/3(18)/2(5)/2(24)/1(25) -> 3(8), 2(23), 1(<N>),

21/1(4)/3(18)/2(5)/2(24)/1(25)/3(8) -> 1(23), 2(<N>), 3(6),

21/1(4)/3(18)/2(5)/2(24)/1(25)/3(8)/1(23) -> 1(9), 2(<N>),

21/1(4)/3(18)/2(5)/2(24)/1(25)/3(8)/1(23)/1(9) -> 1(<Y>), 4(2),

21/1(4)/3(18)/2(5)/2(24)/1(25)/3(8)/1(23)/1(9)/4(2) -> 2(3),

21/1(4)/3(18)/2(5)/2(24)/1(25)/3(8)/1(23)/1(9)/4(2)/2(3) -> 2(6),

21/1(4)/3(18)/2(5)/2(24)/1(25)/3(8)/1(23)/1(9)/4(2)/2(3)/2(6) -> 2(10),

21/1(4)/3(18)/2(5)/2(24)/1(25)/3(8)/1(23)/1(9)/4(2)/2(3)/2(6)/2(10) -> 2(14),

21/1(4)/3(18)/2(5)/2(24)/1(25)/3(8)/1(23)/1(9)/4(2)/2(3)/2(6)/2(10)/2(14) -> 1(15),

21/1(4)/3(18)/2(5)/2(24)/1(25)/3(8)/1(23)/1(9)/4(2)/2(3)/2(6)/2(10)/2(14)/1(15) -> 2(16),

21/1(4)/3(18)/2(5)/2(24)/1(25)/3(8)/1(23)/1(9)/4(2)/2(3)/2(6)/2(10)/2(14)/1(15)/2(16) -> 1(19),

21/1(4)/3(18)/2(5)/2(24)/1(25)/3(8)/1(23)/1(9)/4(2)/2(3)/2(6)/2(10)/2(14)/1(15)/2(16)/1(19) -> 2(20),

21/1(4)/3(18)/2(5)/2(24)/1(25)/3(8)/1(23)/1(9)/4(2)/2(3)/2(6)/2(10)/2(14)/1(15)/2(16)/1(19)/2(20) -> 1(22),

21/1(4)/3(18)/2(5)/2(24)/1(25)/3(8)/1(23)/1(9)/4(2)/2(3)/2(6)/2(10)/2(14)/1(15)/2(16)/1(19)/2(20)/1(22) -> 1(26),

21/1(4)/3(18)/2(5)/2(24)/1(25)/3(8)/1(23)/1(9)/4(2)/2(3)/2(6)/2(10)/2(14)/1(15)/2(16)/1(19)/2(20)/1(22)/1(26) -> 2(27),

21/1(4)/3(18)/2(5)/2(24)/1(25)/3(8)/1(23)/1(9)/4(2)/2(3)/2(6)/2(10)/2(14)/1(15)/2(16)/1(19)/2(20)/1(22)/1(26)/2(27) -> 3(1),

21/1(4)/3(18)/2(5)/2(24)/1(25)/3(8)/1(23)/1(9)/4(2)/2(3)/2(6)/2(10)/2(14)/1(15)/2(16)/1(19)/2(20)/1(22)/1(26)/2(27)/3(1) -> 3(<N>), 2(<Y>),

21/1(4)/3(18)/2(5)/2(24)/1(25)/3(8)/3(6) -> 2(9),

21/1(4)/3(18)/2(5)/2(24)/1(25)/3(8)/3(6)/2(9) -> 1(10),

21/1(4)/3(18)/2(5)/2(24)/1(25)/3(8)/3(6)/2(9)/1(10) -> 1(22),

21/1(4)/3(18)/2(5)/2(24)/1(25)/3(8)/3(6)/2(9)/1(10)/1(22) -> 1(16),

21/1(4)/3(18)/2(5)/2(24)/1(25)/3(8)/3(6)/2(9)/1(10)/1(22)/1(16) -> 1(<N>), 2(<Y>),

21/1(4)/3(18)/2(5)/2(24)/1(25)/2(23) -> 1(10), 2(<Y>),

21/1(4)/3(18)/2(5)/2(24)/1(25)/2(23)/1(10) -> 2(<N>), 1(3), 3(<Y>),

21/1(4)/3(18)/2(5)/2(24)/1(25)/2(23)/1(10)/1(3) -> 2(14),

21/1(4)/3(18)/2(5)/2(24)/1(25)/2(23)/1(10)/1(3)/2(14) -> 1(15),

21/1(4)/3(18)/2(5)/2(24)/1(25)/2(23)/1(10)/1(3)/2(14)/1(15) -> 2(19),

21/1(4)/3(18)/2(5)/2(24)/1(25)/2(23)/1(10)/1(3)/2(14)/1(15)/2(19) -> 2(6),

21/1(4)/3(18)/2(5)/2(24)/1(25)/2(23)/1(10)/1(3)/2(14)/1(15)/2(19)/2(6) -> 2(27), 1(<Y>),

21/1(4)/3(18)/2(5)/2(24)/1(25)/2(23)/1(10)/1(3)/2(14)/1(15)/2(19)/2(6)/2(27) -> 2(<N>), 3(1),

21/1(4)/3(18)/2(5)/2(24)/1(25)/2(23)/1(10)/1(3)/2(14)/1(15)/2(19)/2(6)/2(27)/3(1) -> 2(2),

21/1(4)/3(18)/2(5)/2(24)/1(25)/2(23)/1(10)/1(3)/2(14)/1(15)/2(19)/2(6)/2(27)/3(1)/2(2) -> 2(9),

21/1(4)/3(18)/2(5)/2(24)/1(25)/2(23)/1(10)/1(3)/2(14)/1(15)/2(19)/2(6)/2(27)/3(1)/2(2)/2(9) -> 1(13),

21/1(4)/3(18)/2(5)/2(24)/1(25)/2(23)/1(10)/1(3)/2(14)/1(15)/2(19)/2(6)/2(27)/3(1)/2(2)/2(9)/1(13) -> 3(20),

21/1(4)/3(18)/2(5)/2(24)/1(25)/2(23)/1(10)/1(3)/2(14)/1(15)/2(19)/2(6)/2(27)/3(1)/2(2)/2(9)/1(13)/3(20) -> 1(22),

21/1(4)/3(18)/2(5)/2(24)/1(25)/2(23)/1(10)/1(3)/2(14)/1(15)/2(19)/2(6)/2(27)/3(1)/2(2)/2(9)/1(13)/3(20)/1(22) -> 1(29),

21/1(4)/3(18)/2(5)/2(24)/1(25)/2(23)/1(10)/1(3)/2(14)/1(15)/2(19)/2(6)/2(27)/3(1)/2(2)/2(9)/1(13)/3(20)/1(22)/1(29) -> 4(7),

21/1(4)/3(18)/2(5)/2(24)/1(25)/2(23)/1(10)/1(3)/2(14)/1(15)/2(19)/2(6)/2(27)/3(1)/2(2)/2(9)/1(13)/3(20)/1(22)/1(29)/4(7) -> 2(<N>), 1(<Y>),

21/1(4)/3(18)/3(2) -> 2(3),

21/1(4)/3(18)/3(2)/2(3) -> 2(8),

21/1(4)/3(18)/3(2)/2(3)/2(8) -> 1(9),

21/1(4)/3(18)/3(2)/2(3)/2(8)/1(9) -> 1(20),

21/1(4)/3(18)/3(2)/2(3)/2(8)/1(9)/1(20) -> 1(22),

21/1(4)/3(18)/3(2)/2(3)/2(8)/1(9)/1(20)/1(22) -> 1(23),

21/1(4)/3(18)/3(2)/2(3)/2(8)/1(9)/1(20)/1(22)/1(23) -> 1(16),

21/1(4)/3(18)/3(2)/2(3)/2(8)/1(9)/1(20)/1(22)/1(23)/1(16) -> 4(<N>), 3(<Y>),

21/1(4)/4(8) -> 1(2), 3(<N>), 2(2), 5(<Y>),

21/1(4)/4(8)/1(2) -> 2(<Y>), 1(20),

21/1(4)/4(8)/1(2)/1(20) -> 1(9), 2(<N>),

21/1(4)/4(8)/1(2)/1(20)/1(9) -> 1(15),

21/1(4)/4(8)/1(2)/1(20)/1(9)/1(15) -> 2(11),

21/1(4)/4(8)/1(2)/1(20)/1(9)/1(15)/2(11) -> 3(<N>), 2(<Y>), 1(<Y>),

21/1(4)/4(8)/2(2) -> 2(6),

21/1(4)/4(8)/2(2)/2(6) -> 2(14),

21/1(4)/4(8)/2(2)/2(6)/2(14) -> 1(15),

21/1(4)/4(8)/2(2)/2(6)/2(14)/1(15) -> 2(20),

21/1(4)/4(8)/2(2)/2(6)/2(14)/1(15)/2(20) -> 1(22),

21/1(4)/4(8)/2(2)/2(6)/2(14)/1(15)/2(20)/1(22) -> 1(24),

21/1(4)/4(8)/2(2)/2(6)/2(14)/1(15)/2(20)/1(22)/1(24) -> 1(27),

21/1(4)/4(8)/2(2)/2(6)/2(14)/1(15)/2(20)/1(22)/1(24)/1(27) -> 2(3),

21/1(4)/4(8)/2(2)/2(6)/2(14)/1(15)/2(20)/1(22)/1(24)/1(27)/2(3) -> 1(<N>), 2(<Y>),

21/1(4)/5(8) -> 1(6), 2(<Y>),

21/1(4)/5(8)/1(6) -> 2(<N>), 1(14),

21/1(4)/5(8)/1(6)/1(14) -> 1(1),

21/1(4)/5(8)/1(6)/1(14)/1(1) -> 1(25), 2(<Y>),

21/1(4)/5(8)/1(6)/1(14)/1(1)/1(25) -> 3(9),

21/1(4)/5(8)/1(6)/1(14)/1(1)/1(25)/3(9) -> 1(<N>), 2(<Y>), 4(<N>),

21/1(4)/2(2) -> 2(14),

21/1(4)/2(2)/2(14) -> 1(3),

21/1(4)/2(2)/2(14)/1(3) -> 2(<N>), 1(<Y>),

21/3(16) -> 2(<N>), 1(<N>), 3(<Y>), 4(<N>),

Accuracy = 0.625

Precision = 0.5

Recall = 0.6666666666666666

Area AUC-ROC: 0.5462915726664325

Area AUC-PR: 0.6110997743994375

**Применение**:

**1) Обнаружение спама**: Деревья решений могут использоваться для классификации электронной почты как спам или не спам на основе содержания, метаданных и других характеристик.

**2)** **Прогнозирование продаж**: Деревья решений могут быть использованы для прогнозирования будущих продаж в зависимости от различных факторов, таких как время года, реклама и экономические условия.

**3)** **Выявление закономерностей**: Анализ данных для поиска ключевых признаков, которые наиболее влияют на целевую переменную.

**Лабораторная 6. Логистическая регрессия**

1. Загрузить выбранный датасет и выполните предварительную обработку данных.
2. Получить и визуализировать (графически) статистику по датасету (включая количество, среднее значение, стандартное отклонение, минимум, максимум и различные квантили).
3. Разделить данные на обучающий и тестовый наборы в соотношении, которое вы считаете подходящим.
4. Реализовать логистическую регрессию "с нуля" без использования сторонних библиотек, кроме NumPy и Pandas. Рализация логистической регрессии включает в себя:
   * Функцию для вычисления гипотезы (sigmoid function).
   * Функцию для вычисления функции потерь (log loss).
   * Метод обучения, который включает в себя градиентный спуск.
   * Возможность варьировать гиперпараметры, такие как коэффициент обучения (learning rate) и количество итераций.
5. Исследование гиперпараметров:
   * Провести исследование влияния гиперпараметров на производительность модели. Варьируйте следующие гиперпараметры:
     + Коэффициент обучения (learning rate).
     + Количество итераций обучения.
     + Метод оптимизации (например, градиентный спуск или оптимизация Ньютона).
6. Оценка модели:
   * Для каждой комбинации гиперпараметров оценить производительность модели на тестовом наборе данных, используя метрики, такие как accuracy, precision, recall и F1-Score.

функция hypothesis\_function(df, row: int, ratios: list):  
 z = np.dot(df.iloc[row], ratios)  
 return 1 / (1 + np.exp(-z))

функция gradient:  
 gradient = инициализация вектора градиентов

Повторить столько раз, сколько обучающихся данных:  
 gradient +=   
 вернуть gradient / кол-во обучающихся данных

функция gradient\_descent:

Повторить заданное количество итераций:  
 gradient = gradient()  
 вектор параметров = – коэффициент обучения \* gradient

Результат

[-0.08237608, -0.07422636, -0.0537892, 0.03548706, 0.04975093 -0.07989111, -0.06422124 0.09660308] – вектор параметров

Функция потерь: 0,682

Accuracy = 0.6168831168831169

Precision = 0.2727272727272727

Recall = 0.05555555555555555

F1 Score = 0.0923076923076923

Применяется в тех же сферах, что и предыдущие методы.

**Сравнение методов**

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Достоинства** | **Недостатки** | **Применение** |
| Линейная  регрессия | 1. Простота  2. Быстро обучается даже на больших объемах данных  3. Эффективна, если зависимость между признаками и целевой переменной действительно линейная | 1. Чувствительность к выбросам  2. Невозможность моделировать нелинейные зависимости  3. Высокая корреляция между признаками может ухудшать результаты | Регрессия |
| kNN | 1. Простота  2. Не требует обучения (ленивый метод)  3. Способен моделировать сложные зависимости, включая нелинейные  4. Подходит для числовых и категориальных признаков | 1.Чувствительность к выбору k  2. Для больших данных метод становится медленным из-за необходимости вычислять расстояния до всех точек  3. Признаки с разными масштабами могут искажать результаты (требуется нормализация).  4. Классы с большим количеством объектов могут доминировать | Классификация,  регрессия |
| Дерево решений | 1. Простота  2. Может работать с признаками в разных масштабах  3. Работает с числовыми и категориальными данными  4. Выбросы меньше влияют на построение дерева  5. Моделирование нелинейных зависимостей | 1. Переобучение  2. Малые изменения в данных могут привести к значительному изменению структуры дерева.  3. Может создавать некорректные границы для классов с разной частотой. | Классификация,  регрессия |
| Логистическая  регрессия | 1.**Интерпретируемость**: Коэффициенты показывают, как признаки влияют на вероятность принадлежности к классу  2. Основана на вероятностной модели и статистически обоснована.  3. Гибкость для бинарной классификации  4. Легко масштабируется на большие данные  5. С весами классов можно корректировать модель для редких событий. | 1. Плохо справляется с задачами, где зависимость между признаками и целевой переменной нелинейная  2. Задачах с высокой сложностью данных, таких как нелинейные зависимости, уступает другим моделям, например, деревьям решений или ансамблевым методам. | Классификация, регрессия |

**Вывод**

В данном модуле я изучил несколько методов машинного обучения, а именно: линейная регрессия, метод k-ближайших соседей, деревья решений и логистическая регрессия. Каждый метод был реализован мною без использования сторонних библиотек, кроме numpy, pandas и matplotlib, которые использовались просто для удобной работы с данными и их визуализации. Также были оценены параметры каждого метода, такие как accuracy, precision, recall и так далее. Они показали, насколько точно была обучена модель.

Что касается моих реализаций, самым успешным методом оказался метод k-ближайших соседей, который показал лучшие характеристики обучения. Возможно, из-за того, что довольно прост и понятен в реализации.