Московский авиационный институт
(Национальный исследовательский университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»
Дисциплина: «Машинное обучение»

Лабораторная работа № 2

Тема: алгоритмы классификации

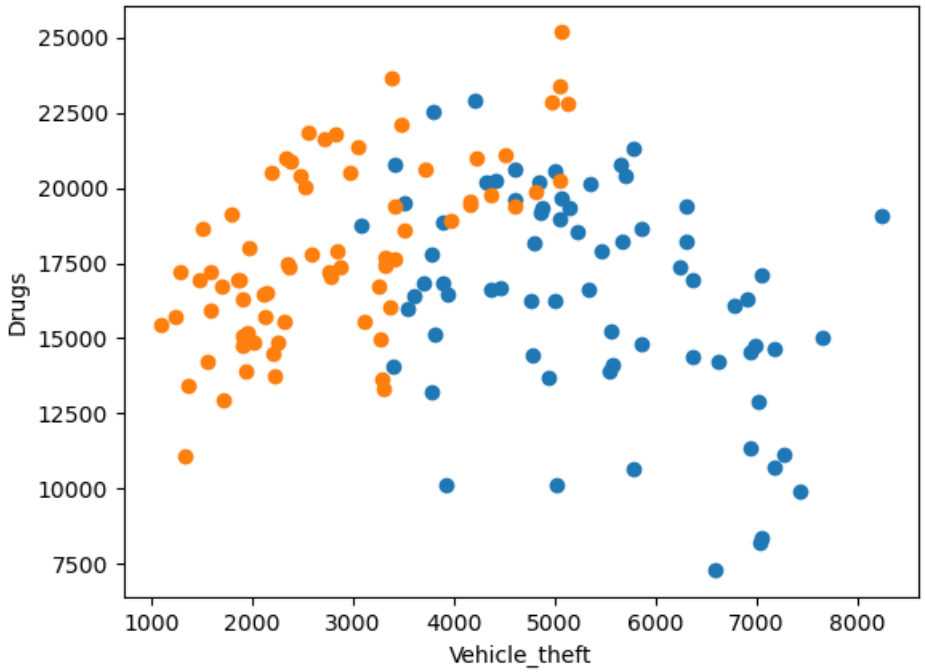
Студент: Хренов Геннадий
Группа: 80-307Б
Преподаватель: Ахмед Самир Халид
Дата:
Оценка:

Москва, 2021

1. Постановка задачи

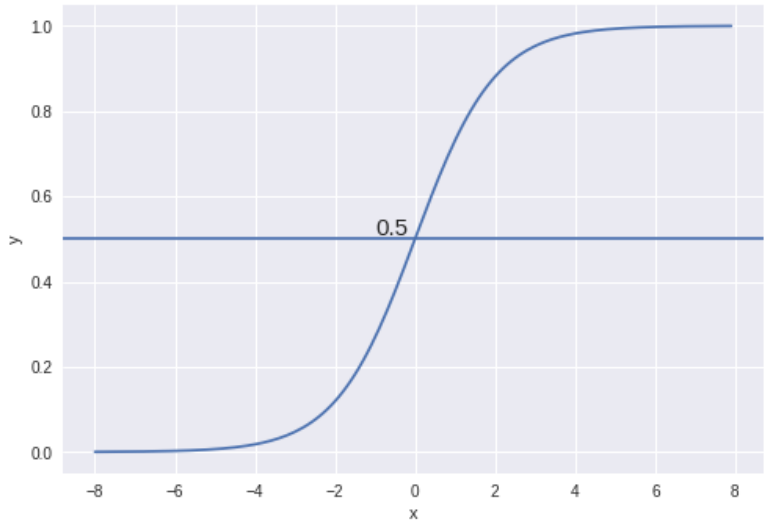
   Необходимо реализовать алгоритмы машинного обучения. Применить данные алгоритмы на наборы данных, подготовленных в первой лабораторной работе. Провести анализ полученных моделей, вычислить метрики классификатора. Произвести тюнинг параметров в случае необходимости. Сравнить полученные результаты с моделями, реализованными в scikit-learn. Аналогично построить метрики классификации. Показать, что полученные модели не переобучились. Также необходимо сделать выводы о применимости данных моделей к вашей задаче.
2. Датасет

   Берем датасет, подготовленный в первой лабораторной.



3. Логистическая регрессия

   Логистическая регрессия — это статистическая модель, используемая для прогнозирования вероятности возникновения некоторого события путём его сравнения с логистической кривой, которая выглядит так:



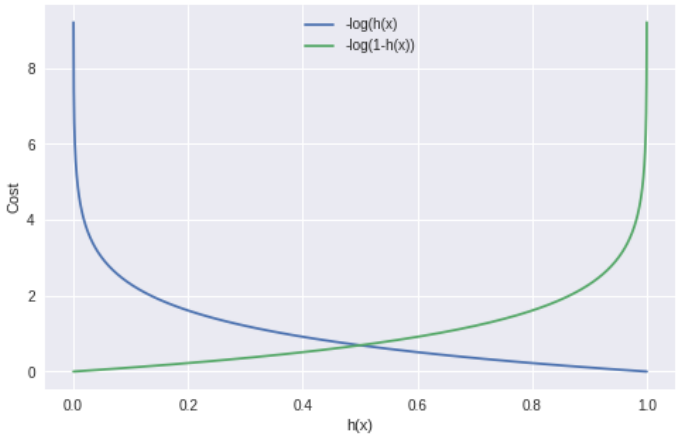
Значение сигмоидной функции всегда лежит между 0 и 1. Значение точно равно 0,5 при X = 0. Мы можем использовать 0,5 в качестве порога вероятности для определения классов. Гипотеза для логистической регрессии:

$$h(x) = \frac{1}{1 + e^{-\theta^T x}}$$

$$h(x) = \begin{cases} > 0.5, & \text{if } \theta^T x > 0 \\ < 0.5, & \text{if } \theta^T x < 0 \end{cases}$$

Функция стоимости задается так:

$$cost = \begin{cases} -log(h(x), & \text{if } y = 1 \\ -log(1 - h(x)), & \text{if } y = 0 \end{cases}$$



Стоимость всех обучающих примеров:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^i log(h(x^i)) + (1 - y^i) log(1 - h(x^i))]$$

Задача стоит в минимизации этой функции.

Результаты:

```
(first) D:\MAI\ML\lab2>python lab2.py
sklearn log:
 [0 1 0 0 0 0 0 1 0 0 1 0 1 1 1 1 1 1 1 1 1 1]
accur:  0.8695652173913043
              precision    recall  f1-score   support

           0       0.80      0.89      0.84         9
           1       0.92      0.86      0.89        14

    accuracy                           0.87        23
   macro avg       0.86      0.87      0.87        23
weighted avg       0.87      0.87      0.87        23

my log:
 [0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
my accur:  0.8695652173913043
              precision    recall  f1-score   support

           0       0.80      0.89      0.84         9
           1       0.92      0.86      0.89        14

    accuracy                           0.87        23
   macro avg       0.86      0.87      0.87        23
weighted avg       0.87      0.87      0.87        23
```

Собственная реализация и sklearn совпадают.

4. Дерево решений

Дерево решений — в основном жадное, нисходящее, рекурсивное разбиение. Энтропия — это мера случайности или неопределенности. Уровень энтропии колеблется от 0 до 1. Для меры энтропии используют примесь Джини. Узел чистый, если все его выборки принадлежат одному и тому же классу, в то время как узел с множеством выборок из разных классов будет иметь Джини ближе к 1.

$$G = 1 - \sum_{k=1}^{n} {p_k}^2$$

Каждый узел делит выборку таким образом, что примесь Джини у детей (точнее, среднее значение Джини у детей, взвешенных по их размеру) сводится к минимуму. Рекурсия останавливается, когда, достигается максимальная глубина, или когда нет разделения, которое может привести к двум детям, чище, чем их родитель.

Результаты:

```
(first) D:\MAI\ML\lab2>python lab2.py
sklearn Dtree:

[0 0 0 0 0 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1]
accur:
0.9047619047619048
              precision    recall  f1-score   support

           0       0.82      1.00      0.90         9
           1       1.00      0.83      0.91        12

    accuracy                           0.90        21
   macro avg       0.91      0.92      0.90        21
weighted avg       0.92      0.90      0.91        21

my Dtree:

[0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
my accur:  0.9047619047619048
              precision    recall  f1-score   support

           0       0.82      1.00      0.90         9
           1       1.00      0.83      0.91        12

    accuracy                           0.90        21
   macro avg       0.91      0.92      0.90        21
weighted avg       0.92      0.90      0.91        21
```

Собственная реализация и sklearn совпадают.

5. Случайный лес

   Random forest — это множество решающих деревьев. В задаче регрессии их ответы усредняются, в задаче классификации принимается решение голосованием по большинству. Все деревья строятся независимо по следующей схеме:

- Выбирается подвыборка обучающей выборки – по ней строится дерево (для каждого дерева — своя подвыборка).
- Для построения каждого расщепления в дереве просматриваем max_features случайных признаков (для каждого нового расщепления — свои случайные признаки).
- Выбираем наилучшие признаки и расщепляем по нему. Дерево строится, как правило, до исчерпания выборки (пока в листьях не останутся представители только одного класса.

Результаты:

```
(first) D:\MAI\ML\lab2>python lab2.py
sklearn RF:

[0 0 0 0 1 0 0 0 0 1 1 1 1 1]
accur:
0.8571428571428571
              precision    recall  f1-score   support

           0       0.88      0.88      0.88         8
           1       0.83      0.83      0.83         6

    accuracy                           0.86        14
   macro avg       0.85      0.85      0.85        14
weighted avg       0.86      0.86      0.86        14

my RF:

[0 0 0 0 1 0 0 0 0 1 1 1 1 1]
my accur:  0.8571428571428571
              precision    recall  f1-score   support

         0.0       0.88      0.88      0.88         8
         1.0       0.83      0.83      0.83         6

    accuracy                           0.86        14
   macro avg       0.85      0.85      0.85        14
weighted avg       0.86      0.86      0.86        14
```

Собственная реализация и sklearn совпадают.

Запуск для тренировочных данных:

```
(first) D:\MAI\ML\lab2>python lab2.py
LR train: 0.8846153846153846
DT train: 0.9538461538461539
RF train: 0.9307692307692308
```

Точность для тестовых упала не сильно, значит модели не переобучены.

6. Выводы о применимости

Оценка применимости сводится к размеру датасета и количеству параметров. Логистическая регрессия хорошо подходит, так как изначально наши данные разделены на два класса. Дерево решений также подходит – из-за небольшого размера и малого количества параметров в дереве производится меньше вычислений и оно быстрее строится. А вот случайный лес плохо подходит из-за малого количества параметров, так как основная идея леса именно в выборе случайных параметров, чтобы уменьшить значимость доминирующих параметров, и в дальнейшем усреднении результатов. По умолчанию для задач классификации количество случайных параметров равняется

корню из числа параметров, а в случае, когда их два это не имеет смысла. Более того, если убрать один из двух параметров это приведет к большим потерям в точности. Хотя, так как в основе лежат деревья решений, которые показывают хороший результат, сам лес также показывает хороший результат.

СПИСОК ЛИТЕРАТУРЫ

1. Построение логистической регрессии
https://www.machinelearningmastery.ru/building-a-logistic-regression-in-python-301d27367c24/

2. Дерево решений в Python

https://www.machinelearningmastery.ru/decision-tree-in-python-b433ae57fb93/

3. Случайный лес

https://dyakonov.org/2016/11/14/%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D1%8B%D0%B9-%D0%BB%D0%B5%D1%81-random-forest/