Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №2 по курсу
«Искусственый интеллект(Машинное обучение)»

Студент: А. О. Дубинин
Преподаватель:
Группа: М8О-306Б
Дата:
Оценка:
Подпись:

Москва, 2020

Лабораторная работа №2

Задача: 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.

- Логистическая регрессия
- KNN
- Дерево Решений
- Случайный лес

1 Логистическая Регрессия

Теория

Гипотиза логистической регрессии

Гипотиза линейной регрессии.

$$h(x) = \theta^T x$$

Применим sigmoid function, чтобы получить гипотизу для логистической регрессии.

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

Функция стоимости

$$cost(h(x), y) = -ylog(h(x)) - (1 - y)log(1 - h(x))$$

$$J(\theta) = \frac{1}{5} \sum_{i=1}^{m} [y^i log(h(x^i)) + (1 - y^i)log(1 - h(x^i))]$$

Градиент

$$\frac{\delta J(\theta)}{\delta \theta_j} = \frac{1}{m} sum_{i=1}^{m} (h(x^i) - y^i) x_j^i$$

Результат работы

Собственная реализация:

```
My log reg:

Train:

accuracy: 0.7221876942200124
report:
              precision    recall  f1-score   support

           0       0.67      0.68      0.67      1359
           1       0.76      0.75      0.76      1859

    accuracy                           0.72      3218
   macro avg       0.72      0.72      0.72      3218
weighted avg       0.72      0.72      0.72      3218

**********************************************************

Test:

accuracy: 0.7419558359621451
report:
              precision    recall  f1-score   support

           0       0.69      0.70      0.69       657
           1       0.78      0.77      0.78       928

    accuracy                           0.74      1585
   macro avg       0.73      0.74      0.73      1585
weighted avg       0.74      0.74      0.74      1585
```

Sklearn:

```
Scikit-learn:

Train:

accuracy: 0.7688004972032318
report:
              precision    recall  f1-score   support

           0       0.72      0.74      0.73      1349
           1       0.81      0.79      0.80      1869

    accuracy                           0.77      3218
   macro avg       0.76      0.76      0.76      3218
weighted avg       0.77      0.77      0.77      3218

**********************************************************

Test:

accuracy: 0.7753943217665615
report:
              precision    recall  f1-score   support

           0       0.73      0.73      0.73       660
           1       0.81      0.80      0.81       925

    accuracy                           0.78      1585
   macro avg       0.77      0.77      0.77      1585
weighted avg       0.78      0.78      0.78      1585
```

2 Knn

Теория

1. Загрузить данные
2. Инициализировать k - число соседей
3. Для каждого примера данных
   (a) Рассчитаем расстояние между примером запроса и текущим примером по данным.
   (b) Добавим расстояние и индекс примера в упорядоченную коллекцию.
4. Отсортируем упорядоченный набор расстояний и индексов от наименьшего к наибольшему (в порядке возрастания) по расстояниям.
5. Выберем первые K записей из отсортированной коллекции.
6. Получим лэйблы K записей.
7. Вернем наиболее встречающейся из K записей лэйбл.

Результат работы

Собственная реализация:

```
My knn:

Train:

accuracy: 0.6488502175264139
report:
              precision    recall  f1-score   support

           0       0.18      1.00      0.31       251
           1       1.00      0.62      0.76      2967

    accuracy                           0.65      3218
   macro avg       0.59      0.81      0.54      3218
weighted avg       0.94      0.65      0.73      3218

******************************************************

Test:

accuracy: 0.6126182965299685
report:
              precision    recall  f1-score   support

           0       0.12      0.75      0.20       102
           1       0.97      0.60      0.74      1483

    accuracy                           0.61      1585
   macro avg       0.54      0.68      0.47      1585
weighted avg       0.92      0.61      0.71      1585
```

Sklearn:

```
Scikit-learn:

Train:

accuracy: 0.7921068986948415
report:
              precision    recall  f1-score   support

           0       0.77      0.75      0.76      1402
           1       0.81      0.82      0.82      1816

    accuracy                           0.79      3218
   macro avg       0.79      0.79      0.79      3218
weighted avg       0.79      0.79      0.79      3218

******************************************************

Test:

accuracy: 0.6971608832807571
report:
              precision    recall  f1-score   support

           0       0.66      0.63      0.65       692
           1       0.72      0.75      0.74       893

    accuracy                           0.70      1585
   macro avg       0.69      0.69      0.69      1585
weighted avg       0.70      0.70      0.70      1585
```

3 Решающее дерево(CART)

Алгоритм

1. Посчитать индекс gini

   Индекс Джини – это имя функции стоимости, используемой для оценки разбиений в наборе данных.

   $G = |1 - \sum_{k=2}^{n}(X_k - X_{k-1})(Y_k + Y_{k-1})|$

2. Создать Сплит

   (a) Разделение набора данных:

   Разделение набора данных означает разделение набора данных на два списка строк с учетом индекса атрибута и значения разделения для этого атрибута.

   Получив две группы, мы можем использовать приведенный выше показатель Джини для оценки стоимости разделения.

   (b) Оценка всех разделений:

   Проверяются значения каждого атрибута как разделение кандидатов, оценивается стоимость разделения и находится наилучшее возможное разделение, которое мы могли бы сделать.

   Как только будет найдено лучшее разделение, оно используется в качестве узла в дереве решений.

3. Выборка делится на левую и правую часть.

4. Процедура повторяется рекурсивно к каждой из частей, пока энтропия не окажется равной нулю или очень малой величине.

Результат работы

Собственная реализация:

```
My ds:

Train:

accuracy: 0.7454940957116222
report:
              precision    recall  f1-score   support

           0       0.77      0.68      0.72      1556
           1       0.73      0.81      0.77      1662

    accuracy                           0.75      3218
   macro avg       0.75      0.74      0.74      3218
weighted avg       0.75      0.75      0.74      3218

**********************************************************

Test:

accuracy: 0.7299684542586751
report:
              precision    recall  f1-score   support

           0       0.75      0.66      0.70       760
           1       0.72      0.80      0.75       825

    accuracy                           0.73      1585
   macro avg       0.73      0.73      0.73      1585
weighted avg       0.73      0.73      0.73      1585
```

Sklearn:

```
Scikit-learn:

Train:

accuracy: 0.7454940957116222
report:
              precision    recall  f1-score   support

           0       0.77      0.68      0.72      1556
           1       0.73      0.81      0.77      1662

    accuracy                           0.75      3218
   macro avg       0.75      0.74      0.74      3218
weighted avg       0.75      0.75      0.74      3218

**********************************************************

Test:

accuracy: 0.7305993690851735
report:
              precision    recall  f1-score   support

           0       0.75      0.66      0.70       761
           1       0.72      0.80      0.76       824

    accuracy                           0.73      1585
   macro avg       0.73      0.73      0.73      1585
weighted avg       0.73      0.73      0.73      1585
```

4 Случайный лес

Алгоритм

1. Сгенерируем случайную подвыборку с повторениями размером N из обучающей выборки.

2. Построим решающее дерево, классифицирующее образцы данной подвыборки, причём в ходе создания очередного узла дерева будем выбирать набор признаков, на основе которых производится разбиение (не из всех M признаков, а лишь из m случайно выбранных).

3. Дерево строится до полного исчерпания подвыборки и не подвергается процедуре прунинга (англ. pruning — отсечение ветвей).

Результат работы

Собственная реализация:

```
My rf:

Train:

accuracy: 0.6541330018645121
report:
              precision    recall  f1-score   support

           0       0.33      0.71      0.45       632
           1       0.90      0.64      0.75      2586

    accuracy                           0.65      3218
   macro avg       0.61      0.68      0.60      3218
weighted avg       0.79      0.65      0.69      3218

**********************************************************

Test:

accuracy: 0.661198738170347
report:
              precision    recall  f1-score   support

           0       0.32      0.72      0.44       299
           1       0.91      0.65      0.76      1286

    accuracy                           0.66      1585
   macro avg       0.61      0.68      0.60      1585
weighted avg       0.80      0.66      0.70      1585
```

Sklearn:

```
Scikit-learn:

Train:

accuracy: 0.6991920447482909
report:
              precision    recall  f1-score   support

           0       0.53      0.70      0.60      1047
           1       0.83      0.70      0.76      2171

    accuracy                           0.70      3218
   macro avg       0.68      0.70      0.68      3218
weighted avg       0.73      0.70      0.71      3218

**********************************************************

Test:

accuracy: 0.7041009463722397
report:
              precision    recall  f1-score   support

           0       0.51      0.71      0.59       479
           1       0.85      0.70      0.77      1106

    accuracy                           0.70      1585
   macro avg       0.68      0.70      0.68      1585
weighted avg       0.74      0.70      0.71      1585
```

5 Выводы

Из-за того, что метрики на train выборке и test выборки сильно не отличались, можно сделать вывод, что модели не переобучились. Лучше всего для моей задачи классификации подходит алгоритм Решающего дерева.