Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет прикладной математики

Лабораторные работы №1,2

Студент: Туманов Г.А.

Группа: М8О-301Б

Руководитель: Ахмед Самир Халид

Оценка: _____

Дата: _____

Москва, 2021

Постановка задачи

ЛР 1:

Найти себе набор данных (датасет), для следующей лабораторной работы, и проанализировать его. Выявить проблемы набора данных, устранить их. Визуализировать зависимости, показать распределения некоторых признаков. Реализовать алгоритмы К ближайших соседа с использованием весов и Наивный Байесовский классификатор и сравнить с реализацией библиотеки sklearn.

ЛР 2:

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. Задачи со звездочкой бьются по вариантам:

Вариаант 1: SVM

Датасет

Для работы будет использоваться датасет грибов. Он содержит информацию о внешних признаках гриба и о том, является гриб съедобным или ядовитым. Таким образом, используя данный датасет можно поставить задачу об определении съедобности гриба по его внешним признакам.

Датасет содержит пропуски, потому мы удаляем все строки, содержащие пропуски.

Все поля датасета являются категориальными признаками и обозначены строкой из одного символа. Используя функционал библиотеки pandas, переводим данные из строковых значений в числовые.

После построения ковариационной матрицы было обнаружено, что один из признаков принимает одно и то же значение для всех грибов. Поэтому данный признак был удалён из датасета.

Таким образом, все проблемы датасета были устранены.

Алгоритм KNN

Так как все признаки номинальные, введём собственную метрику. Расстояние между строками равно числу несовпадающих элементов. Она удовлетворяет всем трём аксиомам метрики:

1) Число несовпадающих элементов положительно. Если все элементы совпадают, то строки равны и наоборот: ρ(x, y) >= 0 и ρ(x, y) = 0 ⇔ x=y
2) Очевидна симметрия: ρ(x, y) = ρ(y, x)
3) Если строка x отличается от y на a элементов, а строка y от z -- на b элементов, то x и z отличаются не более чем на a+b элементов: ρ(x, z) <= ρ(x, y)+ρ(y, z)

Разбиваем датасет на тренировочную и тестовую выборки.

Для каждого элемента тестовой выборки вычисляем расстояние до всех элементов тренировочной выборки. Сортируем их в порядке возрастания расстояния и выбираем k первых. Класс точки определяем как класс большинства из k найденных точек.

Проверим точность алгоритма и сравним с реализацией sklearn:

```
print_accuracy(Ytest, myKNN(Xtest, 25))
```

```
Accuracy: 0.9940968122786304
Precision: 1.0
Recall: 0.9845679012345679
```

Сравним точность с реализацией sklearn

```
from sklearn.neighbors import KNeighborsClassifier

sk_knn = KNeighborsClassifier(25, metric=dist)
sk_knn.fit(Xtrain, Ytrain)

print_accuracy(Ytest, sk_knn.predict(Xtest))
```

```
Accuracy: 0.9940968122786304
Precision: 1.0
Recall: 0.9845679012345679
```

Точности собственной реализации и реализации sklearn полностью совпадают. Более того, получена очень высокая точность. Значит, данный алгоритм хорошо применим для данной задачи.

Наивный Байесовский классификатор

Наивный Байесовский классификатор действует из предположения, что условную вероятность для признаков можно разбить на произведение вероятностей:
$P(x_1 x_2 \dots x_n|c) = P(x_1|c)P(x_2|c)\dots P(x_n|c)$.

По тренировочной выборке вычисляем вероятности появления каждого класса (съедобный и ядовитый) и условные вероятности для каждого признака относительно каждого класса.

Для строк из тестовой выборки сравниваем произведения вероятностей $P_1 = P(x_1|c_1)P(x_2|c_1)...P(x_n|c_1)P(c_1)$ и $P_2 = P(x_1|c_2)P(x_2|c_2)...P(x_n|c_2)P(c_2)$. Если $P_1 > P_2$, то точка относится к классу c_1, иначе -- к классу c_2.

Проверим точность алгоритма и сравним с реализацией sklearn:

```
print_accuracy(Ytest, myNB(Xtest))
```

```
Accuracy: 0.9752066115702479
Precision: 0.9967213114754099
Recall: 0.9382716049382716
```

Сравним точность с реализацией sklearn

```
from sklearn import naive_bayes
sk_nb = naive_bayes.CategoricalNB()
sk_nb.fit(Xtrain, Ytrain)

print_accuracy(Ytest, sk_nb.predict(Xtest))
```

```
Accuracy: 0.9752066115702479
Precision: 0.9967213114754099
Recall: 0.9382716049382716
```

Точности также полностью совпадают. Наивный Байесовский классификатор показал высокую точность, значит он также хорошо применим для данной задачи.

Логистическая регрессия

Класс точки определяется логистической функцией от векторного произведения вектора весов и вектора координат точки. Одна из координат должна быть равна 1 для всех точек, потому добавляем 1 в конец каждой строки в тренировочной выборке.

Вектор весов находится алгоритмом градиентного спуска. Веса обновляются по следующей формуле: $w \mathrel{+}= \eta * dot(y - logist(x), x)$, где η -- коэффициент обучения (в данной реализации 0.05). Применяем данную формулу для всех элементов тренировочной выборки несколько раз и получаем необходимые веса.

Проверим точность алгоритма и сравним с реализацией sklearn:

```
lr_y0 = np.zeros(Ytest.shape[0])

for i in range(Ytest.shape[0]):
    if logist(np.append(Xtest[i], 1)) < 0.5:
        lr_y0[i] = 0
    else:
        lr_y0[i] = 1

print_accuracy(Ytest, lr_y0)
```

```
Accuracy: 0.9846517119244392
Precision: 0.9936507936507937
Recall: 0.9660493827160493
```

Сравним точность с реализацией sklearn

```
from sklearn.linear_model import LogisticRegression
sk_lr = LogisticRegression(max_iter=500)
sk_lr.fit(Xtrain, Ytrain)

print_accuracy(Ytest, sk_lr.predict(Xtest))
```

```
Accuracy: 0.9681227863046045
Precision: 0.9714285714285714
Recall: 0.9444444444444444
```

Точности примерно схожи. Логистическая регрессия также показала высокую точность. Значит, данную задачу можно свести к линейной, несмотря на то, что все признаки категориальные.

Дерево решений

Дерево решений строится по жадному алгоритму ID3:

Если все элементы выборки принадлежат одному классу или достигнута предельная глубина, то создаём лист, значение которого равно преобладающему классу в выборке.

Иначе по критерию Джини определяем лучшее разбиение. Разбиваем выборку на две. Если одна из них оказалась пустой, то лист, значение которого равно преобладающему классу в выборке.

Иначе, создаём вершину, в которой сохраняем параметры разбиения. Левое и правое поддеревья строятся из полученных разбиений исходной выборки.

Если на вход алгоритму подать тренировочную выборку, то получим дерево решений для данной задачи.

Проверим точность алгоритма и сравним с реализацией sklearn:

```
tree = ID3(Xtrain, Ytrain, 15)

tree_y0 = np.zeros(Ytest.shape[0])

for i in range(Ytest.shape[0]):
    tree_y0[i] = getClass(tree, Xtest[i])

print_accuracy(Ytest, tree_y0)
```

```
Accuracy: 0.9067296340023613
Precision: 0.9841897233201581
Recall: 0.7685185185185185
```

Сравним точность с реализацией sklearn

```
from sklearn.tree import DecisionTreeClassifier
sk_tree = DecisionTreeClassifier()
sk_tree.fit(Xtrain, Ytrain)

print_accuracy(Ytest, sk_tree.predict(Xtest))
```

```
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
```

sklearn дал идеальное решение для задачи. ID3 тоже получил высокую точность, только Recall значительно меньше Precision и Accuracy, то есть доля выявленных ядовитых грибов составляет 76.85%. Связано это скорее всего с неудачным выбором критерия для определения наилучшего разбиения.

SVM

SVM похож на логистическую регрессию. В SVM классы представлены не 0 и 1, а -1 и 1. Класс точки определяется по формуле: $f(x) = sign(w \cdot x - b)$.

SVM требует, чтобы расстояние между гиперплоскостями wx-b=1 и wx-b=-1 было максимальным. Это расстояние равно $2/||w||$.

Таким образом, алгоритм находит такие w и b, чтобы f(x) максимально точно определяло класс точек, и ||w|| было минимально.

Проверим точность алгоритма и сравним с реализацией sklearn:

```
svm_y0 = np.zeros(Ytest.shape[0])

for i in range(Ytest.shape[0]):
    if f(Xtest[i]) < 0:
        lr_y0[i] = 0
    else:
        lr_y0[i] = 1

print_accuracy(Ytest, lr_y0)
```

```
Accuracy: 0.9043683589138135
Precision: 0.9204152249134948
Recall: 0.8209876543209876
```

Сравним точность с реализацией sklearn

```
from sklearn.svm import SVC
sk_svm = SVC()
sk_svm.fit(Xtrain, Ytrain)

print_accuracy(Ytest, sk_svm.predict(Xtest))
```

```
Accuracy: 0.9964580873671782
Precision: 1.0
Recall: 0.9907407407407407
```

У sklearn Precision равно 1. Значит, все грибы, определённые как ядовитые, действительно являются ядовитыми. Собственная реализация также показала высокую точность.

Вывод

Был проанализирован датасет грибов. Устранены возникшие проблемы датасета, такие как пропуски и поля с одинаковыми значениями. Была решена задача определения, к какому классу относится гриб, несколькими методами: KNN, наивный Байесовский анализатор, логистическая регрессия, дерево решений и SVM. Все алгоритмы показали высокую точность. Это значит, что съедобность гриба можно определить по его внешним признакам.