Задание (ТРАРД)

Горская Елена

В ходе данной работы рассматривался датасет для классификации машин: https://archive.ics.uci.edu/ml/datasets/car+evaluation.

Описание датасета:

У каждого объекта имеется 6 признаков:

- ✓ стоимость автомобиля
- ✓ стоимость обслуживания
- ✓ количество дверей
- ✓ вместимость: количество пассажиров
- ✓ размер багажника
- ✓ безопасность автомобиля

Далее, целевой признак – это доступность автомобиля (4 значения: unacc, acc, good, vgood).

В данной работе будут использованы следующие термины (взятые из описания задания):

$|g' \cap g_i^+|$ – мощность пересечения

$|(g' \cap g_i^+)^+|$ – поддержка (релизована в прилагаемом скрипте)

$|(g' \cap g_i^+)^-|$ – достоверность

Способ 1

Рассматривалась задача бинарной классификации. В качестве отрицательных примеров рассматривались те, у которых целевой признак равен «unacc», а в качестве положительных – у которых он другой (acc, good, vgood). В целом, итоговые данные оказались поделены в процентном соотношении 30% : 70%.

Остальные признаки также имеют бинарную природу. И для каждого объекта список признаков задается в виде сета (set), включающего в себя все те и только те признаки, которыми данный объект обладает.

Для данного способа было использовано три алгоритма:

Алгоритм 1

1. В функцию классификации подаем два параметра: threshold и num_misses.

2. Рассматриваем пересечение описания классифицируемого объекта с описанием каждого объекта из плюс-контекста. Если оно не вкладывается в описание ни одного из минус-примеров (а точнее, оно может вкладываться максимум в num_misses примеров), то начисляем голос, равный нормированной мощности пересечения.

3. Аналогично делаем и для минус-примеров.

4. Далее, рассматриваем также threshold (среди двух голосов берется максимальный, и его доля сравнивается с этим значением threshold: если она больше, то присваиваем метку соответствующего класса, а если меньше, то присваиваем неопределенную метку).

В работе также применялась кросс валидация, и метрики считались как средние по всем проходам. Вот какие значения получились при параметрах threshold = 0, num_misses = 0:

```
total accuracy =  0.9936355899128685
total precision =  0.9862614393773954
total recall =  0.9921404374140692
total F1 =  0.9890364338731111
```

Алгоритм 2

Основным отличием данного алгоритма является то, что тут используется еще и минимальная поддержка, представляющая собой число от 0 до 1. Это число, умноженное на длину положительных примеров, представляет собой минимальное значение поддержки при голосовании за «+». Аналогично при умножении данного числа на длину отрицательных примером, получим минимальное значение поддержки при голосовании за «-».

Так, алгоритм будет выглядеть следующим образом:

1. В функцию классификации подаем три параметра: threshold, num_misses и min_supp_part.

2. Считаем минимальную поддержку для положительных и отрицательных примеров (так, как описано чуть выше).

3. Рассматриваем пересечение описания классифицируемого объекта с описанием каждого объекта из плюс-контекста. Если оно не вкладывается в описание ни

одного из минус-примеров (а точнее, оно может вкладываться максимум в num_misses примеров), и в то же время поддержка больше минимальной, то начисляем голос, равный нормированной мощности пересечения.

4. Аналогично делаем и для минус-примеров.

5. Далее, рассматриваем threshold (среди двух голосов берется максимальный, и его доля сравнивается с этим значением threshold: если она больше, то присваиваем метку соответствующего класса, а если меньше, то присваиваем неопределенную метку).

В работе при параметрах threshold = 0, num_misses = 0, min_supp_part = 0.001 получились следующие значения:

```
total accuracy =  0.9947946790052054
total precision =  0.9862614393773954
total recall =  0.9960363020064512
total F1 =  0.9910141336278622
```

Алгоритм 3

Данный алгоритм очень похож на предыдущий, однако голоса теперь начисляются иначе. А именно, на третьем шаге второго алгоритма в качестве голоса мы начисляем не мощность пересечения, а именно значение поддержки (в том случае, конечно, если оно больше заданного минимального значения).

При значениях параметров threshold=0, num_misses=0, min_supp_part = 0.001 получились следующие результаты:

```
total accuracy =  0.995373048004627
total precision =  0.988216024028157
total recall =  0.9960363020064512
total F1 =  0.9920269879406852
```

Сравнение метрик в зависимости от параметров

В каждом из алгоритмов использовалось не менее 2 параметров (изначально почти все они были заданы нулевыми). Теперь же рассмотрим, как меняются все 4 метрики при изменении этих параметров. В первую очередь, рассматривать буду accuracy.

Однако также стоит рассмотреть еще одну метрику, назовем ее CO (classified objects) – доля классифицированных объектов. Ведь в каждом из алгоритмов мы относили

некоторые объекты к «неизвестному» классу, а число точно классифицированных объектов также хотелось бы максимизировать.

Алгоритм 1

Использовалось два параметра:

- threshold

- num_misses

Значения метрики accuracy в зависимости от параметров получились следующие:

	threshold				
num_misses		*0.5*	*0.55*	*0.6*	*0.7*
	0	0.99	0.99	1.0	1.0
	1	0.80	0.94	1.0	1.0
	3	0.80	0.94	1.0	1.0
	5	0.80	0.94	1.0	1.0

Значения метрики precision в зависимости от параметров получились следующие:

	threshold				
num_misses		*0.5*	*0.55*	*0.6*	*0.7*
	0	0.99	0.99	0.99	0.99
	1	0.61	0.89	1.00	0.00
	3	0.61	0.89	1.00	0.00
	5	0.61	0.89	1.00	0.00

Значения метрики CO (classified objects) в зависимости от параметров получились следующие:

	threshold				
num_misses		*0.5*	*0.55*	*0.6*	*0.7*
	0	1.00	0.99	0.98	0.96
	1	1.00	0.54	0.16	0.01
	3	1.00	0.54	0.16	0.01
	5	1.00	0.54	0.16	0.01

В таблицах зеленым цветам выделены клетки с лучшим значением метрики (а желтым – все остальные). Лучшим считала как 1.00, так и 0.99.

Так, исходя из табличек для каждой из трех метрик, получаем, что лучшие значения параметров – это threshold=0.5 или 0.55, и num_misses=0, хотя стоит отметить, что и при threshold=0.6 значения также получились неплохие (хотя немного упала доля классифицированных объектов).

Алгоритм 2

Использовалось теперь уже три параметра:

- threshold

- num_misses

- min_supp_part

Значения метрик при разных значениях параметров представлены в ноутбуке. Наилучше результаты для первых двух метрик получились при:

threshold = 0.6, num_misses = 0, min_supp_part = 0.

Значения вышли следующие:

```
total accuracy =  0.995315865155845
total precision =  0.9901186115471831
classified objects =  0.9796747967479674
```

Важно отметить, что при увеличении значения параметра threshold увеличивались первые две метрики, однако сильно падала последняя метрика (то есть много объектов оставались с неопределенным классом).

В этом алгоритме важно понимать, что именно мы хотим максимизировать. Если мы не хотим, чтобы число классифицированных объектов составляло 97% (как выше), то можем добиться увеличения этого значения за счет взятия других параметров:

threshold = 0.5, num_misses = 0, min_supp_part = 0.001

Тогда метрики будут следующие:

```
total accuracy =  0.9953659947241462
total precision =  0.9881640268964212
classified objects =  0.9959349593495935
```

Так, в зависимости от степени важности каждого из параметров, можно рассматривать различные значения параметров.

Алгоритм 3

Использовалось снова три параметра:

- threshold

- num_misses

- min_supp_part

Значения метрик при разных значениях параметров также представлены в ноутбуке.

1) Если хотим, чтобы все объекты были классифицированы (classified = 1), то лучшие результаты были при

threshold = 0.5, num_misses = 0, min_supp_part = 0

и они составляли:

```
total accuracy =  0.9947876257247247
total precision =  0.9901186115471831
classified objects =  1.0
```

2) Если нам не так важно классифицировать *все* объекты, а мы хотим повысить точность, то лучшие результаты были при

threshold = 0.6, num_misses = 0, min_supp_part = 0.001

и они составляли:

```
total accuracy =  0.9965180113434025
total precision =  0.992129850825503
classified objects =  0.9878048780487805
```

Можно заметить, что при росте значения третьего параметра (поддержка) увеличивалась точность, но уменьшалось число классифицированных объектов. Итак, в зависимости от цели можем выбрать соответствующие параметры.

Способ 2

Теперь рассмотрим классификацию с 4 классами. При этом будем преобразовывать и остальные признаки объектов. Логика преобразования простая: «-high» означает, что значение признака меньше или равно «high», а «+high» означает, что значение больше или равно «high». Так, для каждого из изначальных признаков сопоставляю 2 новых, например для первых двух признаков:

$$low = \{-low, -med\}$$

$$med = \{-med, +med\}$$

$$high = \{+med, +high\}$$

$$vhigh = \{+high, +vhigh\}$$

Подробно преобразование описано в ноутбуке с кодом. Основной смысл состоит в том, чтобы два соседних значения (например, *low* и *med* пересекались, потому что они по значению ближе друг к другу чем, например *low* и *high*).

Алгоритм 4

1. В функцию классификации подаем два параметра: threshold и num_misses.

2. Рассматриваем пересечение описания классифицируемого объекта с описанием каждого объекта из первого контекста. Если оно не вкладывается в описание ни одного из примеров других контекстов (а точнее, оно может вкладываться максимум в num_misses примеров), то начисляем голос, равный нормированной мощности пересечения.

3. Аналогично делаем и для примеров из других трех контекстов.

4. Далее, рассматриваем также threshold (среди четырех голосов берется максимальный, и его доля сравнивается с этим значением threshold: если она больше, то присваиваем метку соответствующего класса, а если меньше, то присваиваем неопределенную метку).

При нулевых значениях параметров была получена следующая точность:

```
total accuracy =  0.9762680622757643
```

Алгоритм 5

Отличие заключается только в том, что теперь также имеется параметр минимальной поддержки (min_supp), и голос мы зачисляем только если поддержка больше этого минимального значения.

И снова, результат при нулевых значениях параметров:

```
total accuracy =  0.9768487823686798
```

Сравнение метрик в зависимости от параметров

В каждом из алгоритмов сравнение проводилось так же, как и в первом способе (для алгоритмов 1 – 3).

Алгоритм 4

Использовалось два параметра:

- threshold

- num_misses

Полученные значения метрик представлены в ноутбуке с кодом. Лучшее значение точности получилось при параметрах:

threshold = 0.5, num_misses = 0

И равнялось:

```
threshold = 0.5 num_misses = 0  :
total accuracy =  0.9760736504927718
classified objects =  0.9878048780487805
```

И снова, здесь не все объекты были распределены по классам. Если же мы хотим достигнуть распределения 100%, то подходят параметры:

threshold = 0.25 (или 0.3), num_misses = 0

Тогда значения метрик будут составлять:

```
-------------
threshold = 0.3 num_misses = 0  :
total accuracy =  0.972219479279813
classified objects =  1.0
```

Алгоритм 5

Использовалось уже три параметра:

- threshold

- num_misses

- min_supp

Полученные значения метрик представлены в ноутбуке с кодом. Лучшее значение точности получилось при параметрах:

threshold = 0.5, num_misses = 0, min_supp = 3

И равнялось:

```
threshold = 0.5 num_misses = 0 min_supp = 3  :
total accuracy =  0.9802740760850321
classified objects =  0.9959349593495935
```

Однако, как видим, не все объекты были классифицированы. Если же мы хотим, чтобы абсолютно все объекты были распределены по классам, то можем использовать параметры:

threshold = 0.25, num_misses = 0, min_supp = 3

Значения метрик при этом будут равны:

```
threshold = 0.25 num_misses = 0 min_supp = 3  :
total accuracy =  0.9785862404604382
classified objects =  1.0
```

Выбор лучших алгоритмов

Как уже говорилось, значения параметров могут варьироваться в зависимости от того, что именно нужно в задаче. Я рассмотрю алгоритмы с наибольшим значением accuracy, в которых число классифицированных объектов составляло не менее 99%.

Алгоритм бинарной классификации

Самым качественным оказался алгоритм №2, итоговые параметры получились следующие:

threshold = 0.53, num_misses = 0, min_supp_part = 0.0012

Значения метрик:

```
total accuracy =  0.9959372720578872
total precision =  0.9881640268964212
total recall =  0.9978354978354977
total F1 =  0.9928925083507588
total TPR =  0.9978354978354977
total FPR =  0.004872215389742555
classified =  0.996525083816483
```

Алгоритм многоклассовой классификации:

Самым качественным оказался алгоритм №5, итоговые параметры получились следующие:

threshold = 0.55, num_misses = 0, min_supp = 3

Значения метрик:

```
total accuracy =  0.9836441825469822
classified =  0.9907343405417859
```

Проверка при помощи стандартных алгоритмов.

Рассматривалась сразу задача многоклассовой классификации. Полученные метрики представлены ниже:

1. KNN

```
KNN metrics:
accuracy:  0.9335260115606936
```

2. SVM

```
SVM metrics:
accuracy:  0.953757225433526
```

3. Decision Tree

```
Decision Tree metrics:
accuracy:  0.9682080924855492
```

4. Random Forest

```
Random Forest metrics:
accuracy:  0.9710982658959537
```

Самый хороший результат вышел для Random Forest, но он меньше, чем тот, который был получен в данной работе при помощи алгоритма №5.

Вывод:

В ходе работы были написаны несколько алгоритмов, из которых были выбраны два лучших (первый – для задачи бинарной классификации, второй – для задачи многоклассовой классификации). Алгоритм для многоклассовой классификации работает с хорошей точностью (более 0.98), при этом дает лучшую точность, чем

использованные стандартные алгоритмы. Из минусов же можно отметить, что время работы у него больше, чем у стандартных алгоритмов.

Дополнение: вероятностное приближение

Алгоритм 6:

Был написан также алгоритм, представляющий собой модификацию алгоритма 5. Здесь изменен подсчет поддержки (используется вероятностное приближение): поддержки подсчитываются на случайной подвыборке.

Используется параметр prob – это число (от 0 до 1), показывающее на какой части всей выборки мы будем искать поддержки.

В таблице ниже представлены значения метрик в зависимости от prob (threshold=0.55, num_misses=0, min_supp=3 остались прежние):

prob	accuracy	classified
1.0	0.9836	0.9907
0.9	0.9842	0.9896
0.8	0.9837	0.99015
0.7	0.9826	0.9878
0.6	0.9824	0.9872
0.5	0.9842	0.9872
0.4	0.9840	0.9821
0.3	0.9858	0.9658
0.2	0.9798	0.9473

Что интересно, при значении (prob=0.9) метрика accuracy даже немного возросла (хоть и незначительно).

Самый хороший результат вышел при prob=0.9 – потому что это самое маленькое значение, при котором обе метрики довольно неплохие (при prob=0.3 точность очень хорошая, но число классифицированных объектов уже сильно упало: на 2%).

Далее, можно также проверить другие параметры (threshold, num_misses и min_supp): возможно, в таком вероятностном алгоритме результат окажется лучше при других значениях этих параметров.

В данном случае параметры подбирались вручную и лучший результат получился при:

threshold=0.55, num_misses=0, min_supp=3, prob=0.4

Получились метрики:

```
total accuracy =  0.9840531329939871
classified =  0.9820588055504614
```

Стоит отметить, что для min_supp значение «3» в каком-то смысле оптимальное: как при его увеличении, так и при его уменьшении падает точность (а вот число классифицированных объектов падает при увеличении и возрастает при уменьшении этого параметра). То же самое выполнено и для параметра threshold.

Так, благодаря новому алгоритму смогли сократить время работы, а точность все еще осталась довольно хорошая.