Конспект по теме «Метрики классификации»

Баланс и дисбаланс классов

В некоторых задачах наблюдается сильный **дисбаланс классов**, что плохо сказывается на обучении модели. Классы несбалансированны, когда их соотношение далеко от 1:1. **Баланс классов** наблюдается, если их количество примерно равно.

Accuracy в случае дисбаланса классов не подходит. Нужна новая метрика!

Вы уже знаете, что класс с меткой «1» называется **положительным**, с меткой «0» — **отрицательным**. Если сравнить эти ответы с предсказаниями, получается такое деление:

- **истинно положительные ответы** (*True Positive, TP*): модель пометила объект меткой «1», и его настоящая метка тоже «1»;
- **истинно отрицательные ответы** (*True Negative, TN*): модель пометила объект меткой «0», и его настоящая метка тоже «0»;
- **ложноположительные ответы** (*False Positive, FP*): модель предсказала «1», а действительное значение класса — «0»;
- **ложноотрицательные ответы** (*False Negative, FN*): модель предсказала «0», а действительное значение класса — «1».

Матрица ошибок

TP, *FP*, *TN*, *FN* собираются в одну таблицу — **матрицу ошибок**, или **матрицу неточностей** (*confusion matrix*). Матрица формируется так:

- по горизонтали («Предсказания») располагаются метки алгоритма от 0 до 1;
- по вертикали («Ответы») — истинные метки класса от 0 до 1.

Что получаем:

1. По главной диагонали (от верхнего левого угла) выстроены правильные прогнозы:
   - *TN* в левом верхнем углу;
   - *TP* в правом нижнем углу.
2. Вне главной диагонали — ошибочные варианты:
   - *FP* в правом верхнем углу;
   - *FN* в левом нижнем углу.

Матрица неточностей находится в знакомом модуле *sklearn.metrics*. Функция *confusion_matrix()* принимает на вход верные ответы и предсказания, а возвращает матрицу ошибок.

```
from sklearn.metrics import confusion_matrix
```

```
print(confusion_matrix(target, predictions))
```

Полнота

Матрица ошибок поможет построить новые метрики. Начнём с **полноты** (*recall*).

Полнота выявляет, какую долю положительных среди всех ответов выделила модель. *Recall* рассчитывается по такой формуле:

$$Recall = \frac{TP}{TP + FN}$$

Полнота — это доля *TP*-ответов среди всех с истинной меткой «1». Хорошо, когда значение *recall* близко к единице: модель хорошо ищет положительные объекты. Если ближе к нулю — модель надо перепроверить и починить.

В модуле *sklearn.metrics* находится метод *recall_score()*. Он принимает на вход верные ответы и предсказания, а возвращает значение полноты.

```
from sklearn.metrics import recall_score

print(recall_score(target, predictions))
```

Точность

Ещё одна метрика для оценки качества прогноза целевого класса — **точность** (*precision*).

Точность определяет, как много отрицательных ответов нашла модель, пока искала положительные. Чем больше отрицательных, тем ниже точность. *Precision* рассчитывается по такой формуле:

$$Precision = \frac{TP}{TP + FP}$$

Модель считается хорошей, если точность близка к единице.

В модуле *sklearn.metrics* находится метод *precision_score()*. Он принимает на вход верные ответы и предсказания, а возвращает значение точности.

```
from sklearn.metrics import precision_score

print(precision_score(target, predictions))
```

F1-мера

По отдельности полнота и точность не слишком информативны. Нужно одновременно повышать показатели обеих. Или обратиться к новой метрике, которая их объединит.

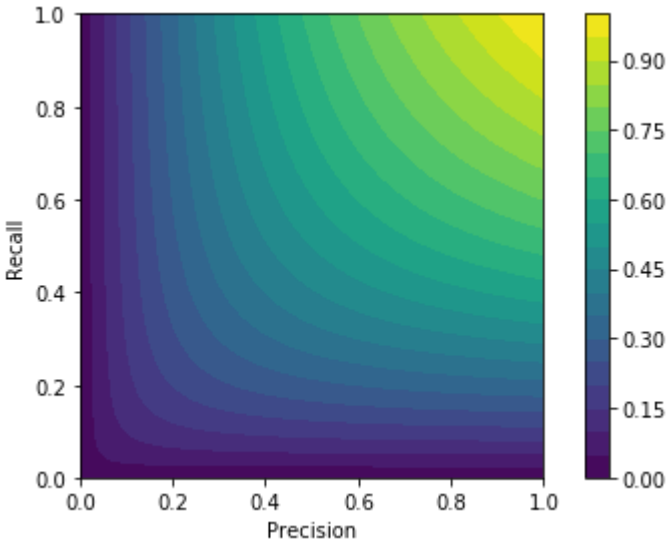
Полнота и точность оценивают качество прогноза положительного класса с разных позиций. *Recall* описывает, как хорошо модель разобралась в особенностях этого класса и распознала его. *Precision* выявляет, не переусердствует ли модель, присваивая положительные метки.

Важны обе метрики. Контролировать их параллельно помогают агрегирующие метрики, одна из которых — **F1-мера** (*F1-score*). Это среднее гармоническое полноты и точности. Единица в *F1* означает, что соотношение полноты и точности равно 1:1.

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

Важно: когда полнота или точность близки к нулю, то к 0 приближается и само среднее гармоническое.

На графике отображены значения *F1*-меры при разных значениях точности и полноты. Синий цвет соответствует нулю, а жёлтый — единице.



Если положительный класс плохо прогнозируется по одной из шкал (*Recall* или *Precision)*, то близкая к нулю *F1*-мера покажет, что прогноз класса «1» не удался.

В модуле *sklearn.metrics* находится метод *f1_score()*. Он принимает на вход верные ответы и предсказания, а возвращает среднее гармоническое точности и полноты.

```
from sklearn.metrics import f1_score

print(f1_score(target, predictions))
```