Методы классификации

В качестве примера, приведена реализация из библиотеки Scikit-learn для Python. Её использование не является обязательным. Язык/библиотека/фреймворк не регламентированы, можно выбрать любое средство, которое посчитаете удобным.

Общие сведения по процессу обучения

Обучение модели в простейшем случае состоит из 3 этапов:

1. Формирование/загрузка набора данных
2. Обучение модели
3. Тестирование обученной модели

Пример использования наивного Байесовского классификатора (и любой другой модели):

Загрузка данных

```
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
```

Создание модели

```
>>> from sklearn.naive_bayes import GaussianNB
>>> gnb = GaussianNB()
```

Обучение модели

```
>>> gnb.fit(iris.data, iris.target)
```

Проверка качества обученной модели

```
>>> y_pred = gnb.predict(iris.data)
>>> print("Number of mislabeled points out of a total %d points : %d"
...       % (iris.data.shape[0],(iris.target != y_pred).sum()))
Number of mislabeled points out of a total 150 points : 6
```

Однако, данный пример имеет мало общего с типичным применением моделей машинного обучения. Оценка качества модели не должна происходить на тех же самых данных, что и её обучение, это приводит к неправильным результатам.

Train/test/val

Обычно набор данных (dataset) разделяют на непересекающиеся обучающую (train) и тестовую (test) выборки (Рис. 1, верхняя часть) и используют их для обучения и оценки качества модели, соответственно. Таким образом, тестируется способность модели работать на данных, не участвовавших в обучении, проверяется генерализируемость классификатора.

Следующим шагом является разбиение набора данных не только на обучающую и тестовую выборки, но ещё и на валидационную (validation) (Рис.1, нижняя часть). В основном применяется, когда необходимо «настроить» (tuning) модель. Модель так же обучается на обучающей выборке, но оценка качества при изменении гиперпараметров происходит на валидационной выборке. Тестовая выборка используется для оценки качества только когда модель уже считается обученной. Так делается для того, чтобы финальная оценка (на тестовой выборке) не была искажена из-за использования данных, на которых модель обучалась, и на данных, которые использовались для «настройки» модели.

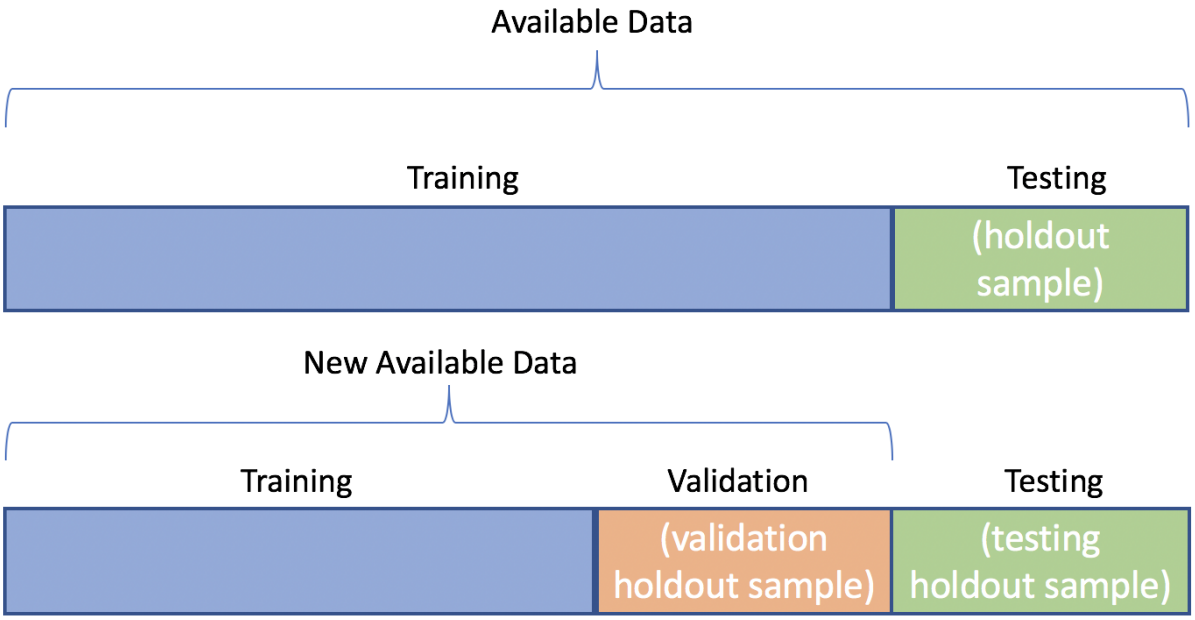

Рисунок 1. Разбиение набора данных

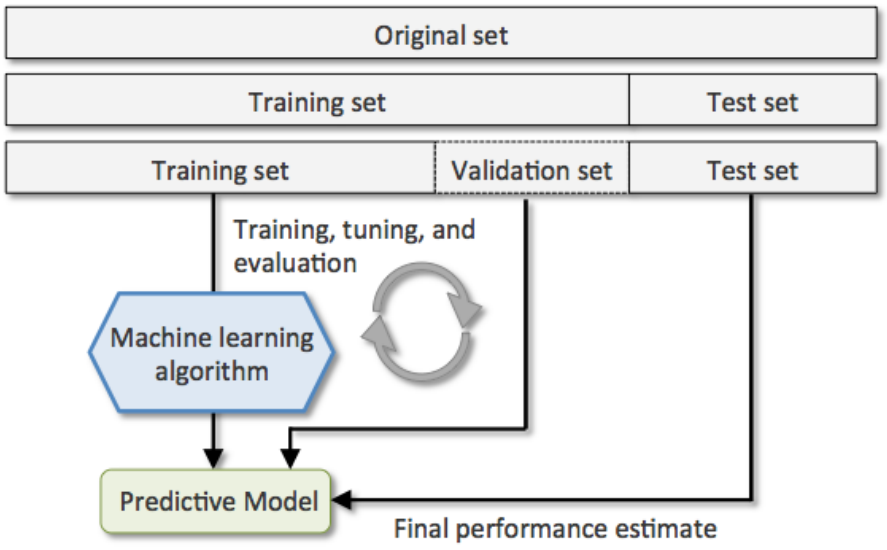

Рисунок 2. Использование выборок

Итого, как указано на Рис. 2, исходный набор данных разделяется на 3 части:

1. Обучающая выборка – на ней происходит обучение модели.
2. Валидационная выборка – на ней происходит оценка модели для её «настройки».
3. Тестовая выборка – итоговая оценка качества модели.

Перекрёстный контроль (cross-validation)

При использовании перекрёстного контроля (cross-validation), правильным будет для этого использовать только обучающую выборку, отделяя тестовую и оставляя её для конечной оценки качества модели (Рис. 3).

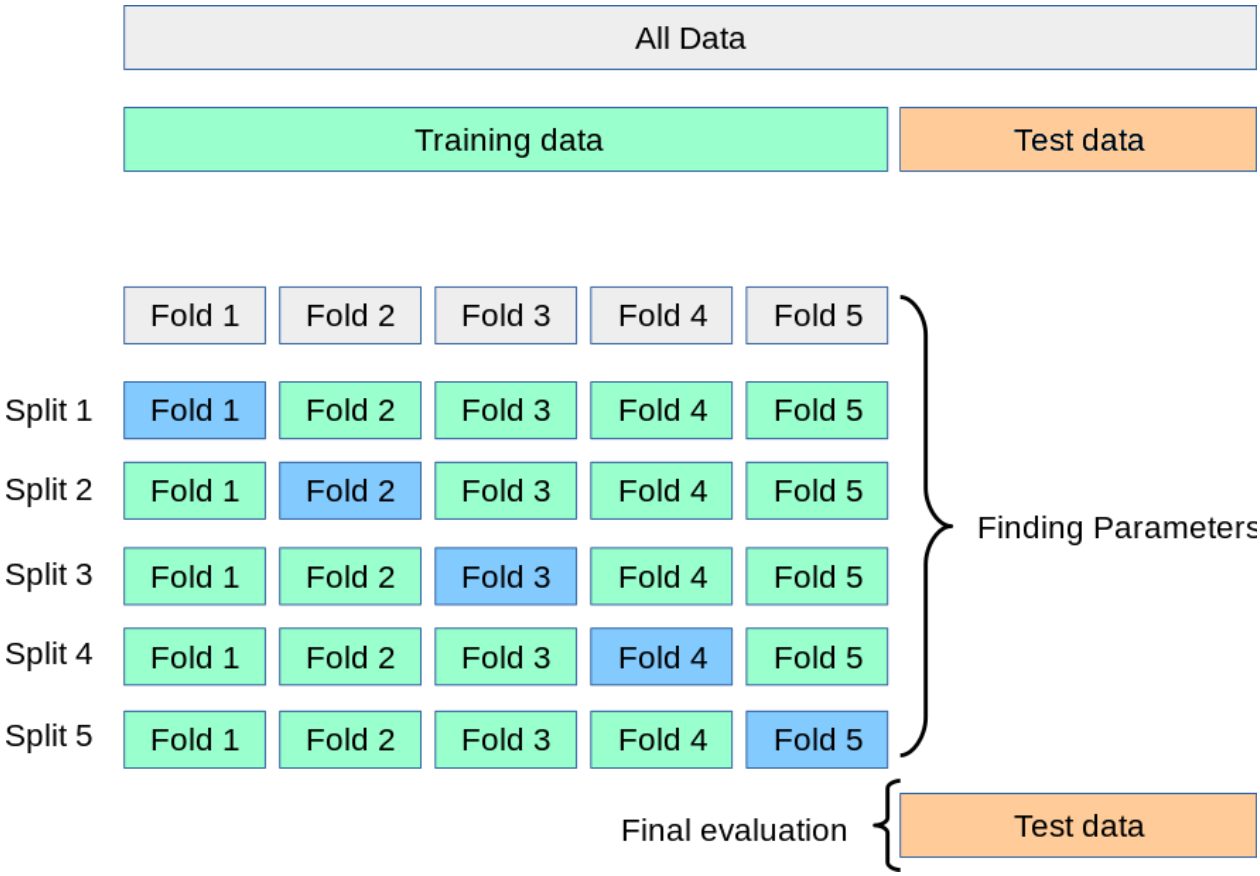

Рисунок 3. Использование кросс-валидации

Кросс-валидация представляет собой формирование нескольких наборов (splits) непересекающихся обучающих и валидационных выборок, используя разбиение исходной обучающей выборки на части одинакового размера (folds). Самым часто применяющимся способом является k-fold cross validation, когда обучающая выборка делится на k частей. В каждом наборе k-1 часть используется для обучения (отмечены зелёным на Рис. 3), и одна часть используется для валидации (отмечена синим на Рис. 3).

Оценка качества моделей классификации

После (а иногда и во время) обучения, настаёт момент оценки производительности модели.

Оценка происходит на валидационной или тестовой части набора данных, сопоставляя предсказанные метки класса с истинными.

Ниже будут приведены методы и подходы к оценке качества, начиная от простых, переходя к более сложным.

Точность (accuracy)

Простейшей мерой качества работы модели является точность.

Точность модели можно посчитать следующим образом:

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

Количество верно предсказанных результатов из общего числа.

Но такая мера часто плохо отражает действительное положение вещей, особенно когда классы в наборе данных не сбалансированы по количеству.

Поэтому переходят к другим методам.

Матрица ошибок (confusion/error matrix)

Рассмотрим типичную задачу классификации. Есть объекты, характеризуемые набором признаков (features). Есть классы, в которые необходимо определить объекты. Есть решение, принимаемое классификатором по объекту на основе его признаков. Есть истинные классы для каждого объекта.

Обладая вышеперечисленной информацией, можно построить удобное представление, отображающее результаты классификации (Рис. 4). Такое представление называется матрица ошибок (confusion matrix/error matrix).

Матрица ошибок является таблицей (подвидом таблицы сопряжённости), где по вертикали и горизонтали откладываются предсказанные и истинные классы (единого стиля нет, где-то предсказанные классы откладываются по горизонтали, а где-то – по вертикали), а в ячейки записывается количество объектов, которые классификатор отнёс к предсказанному классу и которые на самом деле являются представителями истинного класса. Таким образом, ячейка на пересечении столбца и ряда показывает одновременно и что должно было быть предсказано, и что было предсказано по факту. Итого, можно увидеть сводную статистику о том, насколько правильно/неправильно работает классификатор на основе одной таблицы.

Для примера, в Табл. 1 представлена матрица ошибок по классификации хлебобулочных изделий.

		Фактические/истинные		
		Булка	Хлеб	Пончик
Предсказанные	Булка	4	6	3
	Хлеб	1	2	0
	Пончик	1	2	6

Таблица 1. Матрица ошибок

Так как в верхней левой ячейке содержится число 4, то 4 объекта класса булка были действительно определены как булка. В соседней справа ячейке содержится число 6, то есть 6 хлебов были определены как булка. В самой правой ячейке первого ряда содержится число 3, значит 3 пончика были определены как булка. Во второй и третьей ячейке первой колонки содержится число 1, значит по одной булке было определено как хлеб и пончик. И т.д.

Свойства матрицы ошибок:

Сумма всех ячеек – размер выборки, на которой осуществлялось тестирование.

Сумма ячеек колонки – количество истинных объектов заданного класса.

Сумма ячеек ряда – количество предсказаний данного класса.

Очевидно, что правильные предсказания находятся на главной диагонали. Тогда точность можно выразить следующим образом:

$$\text{Accuracy} = \frac{\sum_{i=1}^{n} A_{i,i}}{\sum_{i,j=1}^{n} A_{i,j}}$$

Где A – матрица ошибок, n – число классов

В данном случае, точность составит (4+2+6) / 25 = 0.48

Использование матрицы ошибок позволяет лучше работать с несбалансированными классами в выборке и другими сложными ситуациями. Например, изменим Табл. 1 на случай сильно несбалансированных классов (Табл. 2).

		Фактические/истинные		
		Булка	Хлеб	Пончик
Предсказанные	Булка	40	16	9
	Хлеб	0	0	0
	Пончик	0	0	0

Таблица 2. Матрица ошибок для несбалансированных классов

Точность по Табл. 2 составит 40 / (40+16+9) = 0.61, хотя, по факту, классификатор предсказывает класс булка для всех объектов, что является неприемлемым результатом.

Из матрицы ошибок можно получить гораздо более показательные оценки, чем просто точность. Но для этого необходимо перейти к поклассовым оценкам.

Таблица ошибок (table of confusion)

Для тонкой настройки необходимо высчитывать показатели качества для каждого класса.

Задачу классификации можно рассматривать как несколько гипотез о принадлежности объектов, попадающих под заданные распределения, к определённым классам. Соответственно, сколько классов, столько будет и статистических гипотез.

Последовательно рассматривая гипотезы о принадлежности объекта классу, для каждой из них можно составить сводную статистику о принятии и непринятии гипотезы (Рис. 4). Это представление называется таблица ошибок (table of confusion). Однако, очень часто оно называется всё так же матрицей ошибок (confusion matrix), из-за чего возникает некоторая путаница.

Таблица состоит из колонок, показывающих действительно ли объект должен был быть отнесён к классу или нет, и рядов, показывающих попадает ли объект под критерий гипотезы:

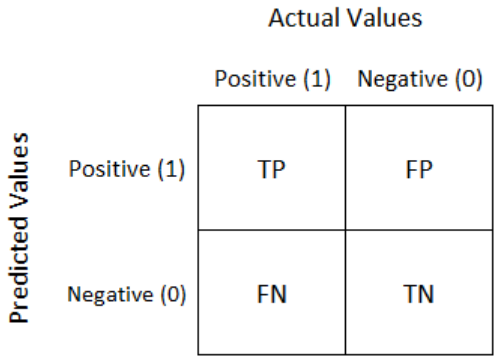

Рисунок 4. Таблица ошибок для гипотезы

Данную таблицу можно получить из матрицы ошибок, «склеив» все классы, кроме интересующего, в один. Например, таблица из Рис. 4, для класса булка и не булка (хлеб и пончик) из Табл. 1.

	Фактические		
Предсказанные		Булка	Не булка
	Булка	4	9
	Не булка	2	10

Таблица 3. Таблица ошибок для класса Булка

Если объект был правильно отнесён к классу, то он попадает в верхнюю левую ячейку, т.к. предсказанный и фактический класс совпадают, значит в отношении этого объекта гипотеза о его принадлежности классу была справедливо подтверждена (True Positive, TP).

Если объект был ошибочно не приписан к классу, но это является ошибкой второго рода (False Negative, FN). Такие предсказания попадают в нижнюю левую ячейку.

Если объект был определён как принадлежащий к исследуемому классу, но на самом деле ему не принадлежит, то это является ошибкой первого рода (False Positive, FP), верхняя правая ячейка.

Если объект был справедливо отвергнут гипотезой, то он попадает в нижнюю правую ячейку (True Negative, TN).

Важно понимать, что в применении к реальным задачам, стоимость ошибок разного рода является разной. Например, при принятии решения о дообследовании пациента с подозрением на раковую опухоль, ложное решение о проведении дополнительных тестов у здорового пациента (FP) не так страшно, как ложное решение о необследовании больного пациента (FN). Противоположным примером может являться система пропускного контроля, где ложноположительные (FP) срабатывания в виде пропуска неавторизованного лица являются недопустимыми.

В связи с этим вводят дополнительные метрики.

Производные метрики качества

Производные метрики качества высчитываются из таблицы ошибок для каждого из классов.

Наиболее часто применяющимися и показательными являются:

- True Positive Rate (TPR), Recall, Sensitivity = TP / (TP + FN) = 1 – FNR, показывает долю правильно предсказанных объектов среди всех предсказаний истинного класса (например, доля булок, которые были идентифицированы как булки)

- True Negative Rate (TNR), Specificity, Selectivity = TN / (TN + FP) = 1– FPR, показывает долю справедливо отвергнутых объектов среди всех предсказаний истинно неподходящих объектов (доля не булок, которые были распознаны как не булки)

- Positive Prediction Value (PPV), Precision = TP / (TP + FP), показывает долю правильно предсказанных объектов из объектов из предсказанных как принадлежащих классу (какая доля булок была предсказана правильно из всех предсказаний булок)

- Negative Prediction Value (NPV) = TN / (TN + FN), показывает долю справедливо отвергнутых объектов среди всех объектов, предсказанных как неподходящие (какая доля не булок была предсказана правильно из всех предсказаний не булок)

- False Positive Rate (FPR) = FP / (TP + FN) = 1 – TNR

- False Negative Rate (FNR) = FN / (FN + TP) = 1 – TPR

- Matthews Correlation Coefficient (MCC)

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

При этом можно выделить 2 группы «противопоставленных» метрик, в которых улучшение одного показателя зачастую приводит к ухудшению другого:

1. Sensitivity и Specificity – правильность определения объектов, принадлежащих и не принадлежащих классу.
2. Precision и Recall – используют данные об ошибках разного рода для определяемого класса.

На их основе строятся диаграммы, наглядно отображающие соотношение внутренних метрик.

Чувствительность и специфичность (sensitivity and specificity), ROC-curve

Представлением, визуально отображающим данную группу, является Receiver Operating Characteristic (ROC-кривая, ROC-curve, кривая ошибок), Рис. 5.

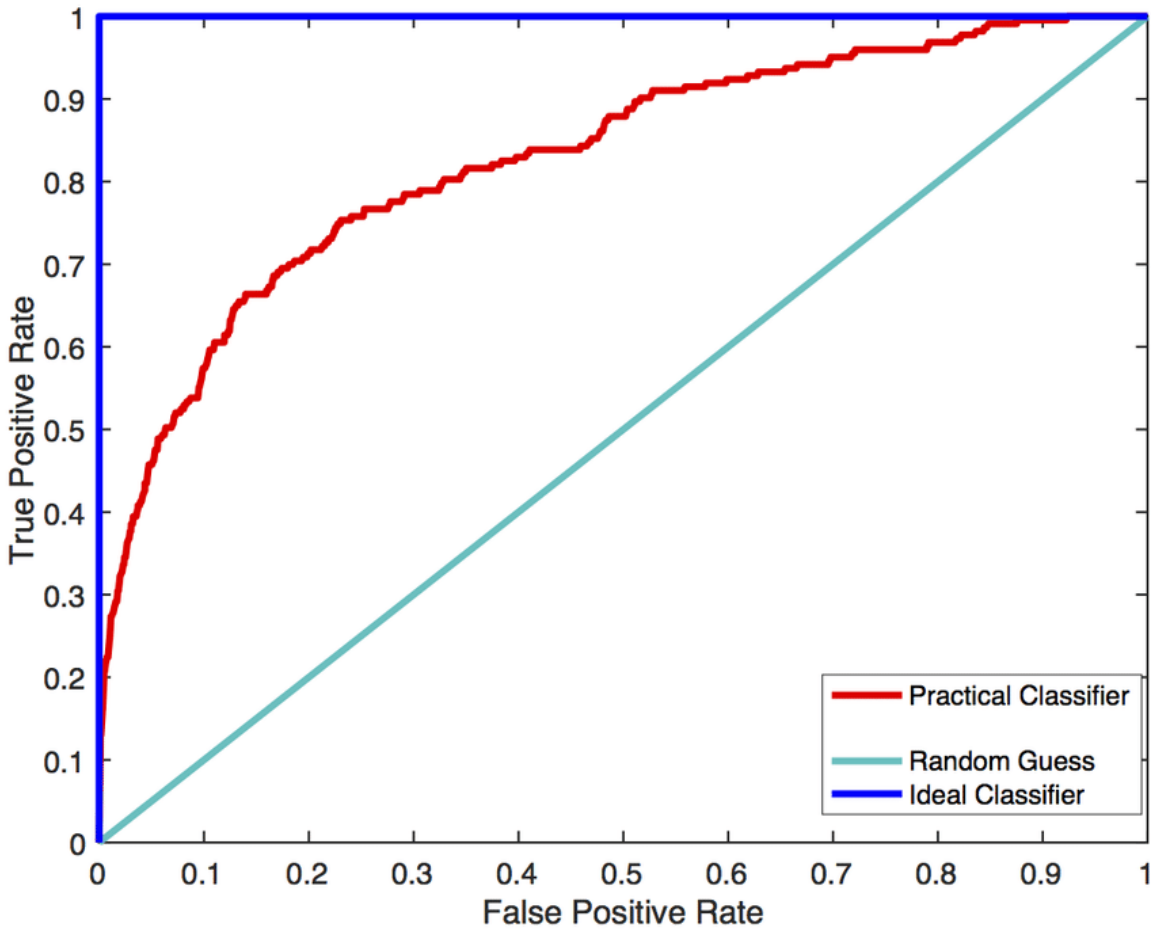

Рисунок 5. ROC-кривая

Применяется для бинарной классификации или отдельно по каждому классу в иных случаях.

По осям откладываются False Positive Rate и True Positive Rate, которые связаны с Sensitivity и Specificity как FPR = 1 – Specificity, TPR = Sensitivity. Это пространство называется ROC-space.

На графике обычно изображается 2 кривых:

1. Кривая случайного принятия решений (изображена лазурной прямой линией)
2. Кривая для анализируемого классификатора (изображена красным)

Кривая отображает соотношение FPR и TPR. Иными словами, как соотносится возможность правильно и неправильно определять объекты класса. Отсюда следует, что идеальной кривой будет кривая, проходящая через (0,1) (на графике отмечена синим цветом).

Кривая случайного принятия решений отличается одинаковым FPR и TPR на всей её протяжённости и соответствует случаям, когда точность (accuracy) = 0.5. Точки, находящиеся выше кривой случайного принятия решения, олицетворяют «хороший» классификатор,

который точнее случайного. Точки ниже – результат хуже случайного. Стоит отметить, что «плохой» классификатор достаточно инвертировать, чтобы сделать его «хорошим».

Чем дальше точки кривой находятся от кривой случайного принятия решения, тем лучше классификатор. Для характеристики удалённости вводят ROC AUC (ROC Area Under Curve, площадь под кривой). Чем больше это значение, тем лучше классификатор. ROC AUC изменяется от 0 до 1.

Для построения ROC-кривой, необходимо получить вероятности принадлежности объектов классу. Затем следует итеративно «сдвигать» порог принятия решения классификатором, начиная с 0 (принимать все объекты), и анализировать FPR с TPR такого классификатора, откладывая их значения на графике.

По построенной ROC-кривой можно выбрать порог определения класса, выбрав наиболее подходящий в зависимости от задачи (важнее FPR или TPR).

Точность и полнота (precision and recall), F_1 score, PR-curve

Если на основе TPR и FPR нет агрегированной метрики, то для Precision и Recall она есть.

Данная мера называется F-мерой (F_1 score, F-score, F-measure) и рассчитывается как среднее гармоническое для Precision и Recall.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Данная метрика позволяет оценить в целом качество классификации. Например, для Табл. 3 Precision будет составлять 4 / (4+9) = 0.31, а Recall 4 / (4+2) = 0.66. При этом F-мера составит 0.42, что говорит о плохом качестве классификации.

Для возможности регулировки важности Precision и Recall в F-мере, существует генерализированная форма с коэффициентом β, отвечающим во сколько раз Recall важнее чем Precision:

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{\beta^2 \cdot precision + recall} = \frac{(1 + \beta^2) \cdot TP}{(1 + \beta^2) \cdot TP + \beta^2 \cdot FN + FP}$$

Наиболее практически применяемыми значениями β являются 0.5, 1 и 2.

Т.к. метрики (F1, Precision, Recall) считаются для каждого из классов по отдельности, то хотелось бы получить единые численные показатели для всех классов сразу. Стратегии могут быть разные, но чаще всего применяются:

- macro – среднее арифметическое между всеми поклассовыми показателями
- micro – вычисление общего показателя путём подсчёта TP, FP и FN на всей выборке

- weighted – усреднение поклассовых показателей с весами, пропорциональными количеству экземпляров каждого класса

PR-кривая (PR-curve) (Рис.6) в целом похожа на ROC-кривую, т.к. тоже зависит от порога классификации, но строится в пространстве Precision и Recall, а не TPR и FPR. Однако, т.к. и Precision, и Recall необходимо максимизировать, в отличие от TPR с FPR, где TPR максимизируется, а FPR минимизируется, то идеальный классификатор будет находиться в точке (1, 1), а не (0, 1). Идеальная же PR-кривая будет проходить через точки (0, 1), (1, 1) и (1, 0).

Кривая случайного выбора будет проходить не по диагонали, а по горизонтали, и значение Precision будет равно соотношению количества экземпляров классов между собой (иными словами, соотношение будет равно P/(P+N), где condition positive P=TP+FN, condition negative N=FP+TN). Например, если соотношение 1:1, то Precision будет 0.5, если 1:2 – 0.33.

Важным достоинством PR-кривой является её независимость от количества экземпляров в классах. Даже если одного класса больше, чем другого, кривая сохранит свой вид.

Аналогично с ROC-кривой, порог классификатора выбирается исходя из задачи классификации. Если Recall важнее Precision, то берутся более правые значения, если важнее Precision – верхние.

Так же, как и для ROC, вводят понятие PR AUC, соответствующее площади под кривой. Она изменяется от 0 до 1 и чем больше, тем лучше.

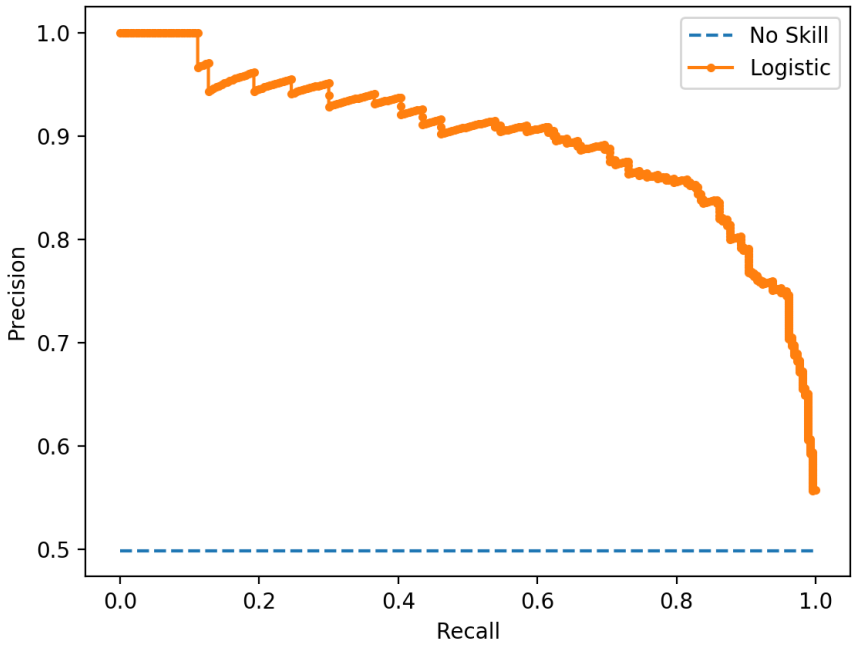

Рисунок 6. PR-кривая

Контекст и интерпретация оценок

Контекст задачи может играть важную роль в оценивании модели и предполагать использование той или иной метрики представленной выше. Дисбаланс классов также влияет на выбор метрики. Например, существует много наборов данных, в которых число меток, принадлежащих разным классам, сильно различается:

- обнаружение мошеннических операций;
- предсказание смертности в палате интенсивной терапии;
- предсказание переходов по ссылкам в вебе.

В этих контекстах общий «процент правильных предсказаний» может ничего не говорить о практической ценности модели.

Цель – обнаружение мошеннических операций с максимальной вероятностью при помощи бинарного классификатора. Трудность моделирования этого набора данных состоит в том, что мошеннические транзакции довольно редки и затрагивают лишь чрезвычайно малую часть деятельности организации. Поэтому в большинстве случаев модель не выявит мошеннических действий.

Поэтому верное обнаружение мошеннических операций является проблемой, например, для оценки качества модели можно использовать следующую формулу:

ОЦЕНКА = MIN(Точность, полнота).

Такая оценка позволяет сосредоточиться на правильном обнаружении мошеннических операций, а не предсказывать их отсутствие и получать хорошую F-меру.

Это пример того, как контекст может изменять представления о качестве модели.

Задание

1. Исследуйте, как объем обучающей выборки и количество тестовых данных, влияет на точность классификации в датасетах про крестики-нолики (tic_tac_toe.txt) и о спаме e-mail сообщений (spam.csv) с помощью наивного Байесовского классификатора. Постройте графики зависимостей точности на обучающей и тестовой выборках в зависимости от их соотношения.
2. Сгенерируйте 100 точек с двумя признаками X1 и X2 в соответствии с нормальным распределением так, что одна и вторая часть точек (класс -1 и класс 1) имеют параметры: мат. ожидание X1, мат. ожидание X2, среднеквадратические отклонения для обеих переменных, соответствующие вашему варианту (указан в таблице). Построить диаграммы, иллюстрирующие данные. Построить Байесовский классификатор и оценить качество классификации с помощью различных методов (точность, матрица ошибок, ROC и PR-кривые). Является ли построенный классификатор «хорошим»?
3. Постройте классификатор на основе метода k ближайших соседей для обучающего множества Glass (glass.csv). Посмотрите заголовки признаков и классов. Перед построением классификатора необходимо также удалить первый признак Id number, который не несет никакой информационной нагрузки.
   a. Постройте графики зависимости ошибки классификации от количества ближайших соседей.
   b. Определите подходящие метрики расстояния и исследуйте, как тип метрики расстояния влияет на точность классификации.
   c. Определите, к какому типу стекла относится экземпляр с характеристиками: RI =1.516 Na =11.7 Mg =1.01 Al =1.19 Si =72.59 K=0.43 Ca =11.44 Ba =0.02 Fe =0.1
4. Постройте классификаторы на основе метода опорных векторов для наборов данных из файлов svmdataN.txt и svmdataNtest.txt, где N – индекс задания:
   a. Постройте алгоритм метода опорных векторов с линейным ядром. Визуализируйте разбиение пространства признаков на области с помощью полученной модели (пример визуализации). Выведите количество полученных опорных векторов, а также матрицу ошибок классификации на обучающей и тестовой выборках.
   b. Постройте алгоритм метода опорных векторов с линейным ядром. Добейтесь нулевой ошибки сначала на обучающей выборке, а затем на тестовой, путем изменения штрафного параметра. Выберите оптимальное значение данного параметра и объясните свой выбор. Всегда ли нужно добиваться минимизации ошибки на обучающей выборке?
   c. Постройте алгоритм метода опорных векторов, используя различные ядра (линейное, полиномиальное степеней 1-5, сигмоидальная функция, гауссово). Визуализируйте разбиение пространства признаков на области с помощью полученных моделей. Сделайте выводы.
   d. Постройте алгоритм метода опорных векторов, используя различные ядра (полиномиальное степеней 1-5, сигмоидальная функция, гауссово). Визуализируйте

разбиение пространства признаков на области с помощью полученных моделей. Сделайте выводы.

e. Постройте алгоритм метода опорных векторов, используя различные ядра (полиномиальное степеней 1-5, сигмоидальная функция, гауссово). Изменяя значение параметра ядра (гамма), продемонстрируйте эффект переобучения, выполните при этом визуализацию разбиения пространства признаков на области.

5. Постройте классификаторы для различных данных на основе деревьев решений:
   a. Загрузите набор данных Glass из файла glass.csv. Постройте дерево классификации для модели, предсказывающей тип (Type) по остальным признакам. Визуализируйте результирующее дерево решения. Дайте интерпретацию полученным результатам. Является ли построенное дерево избыточным? Исследуйте зависимость точности классификации от критерия расщепления, максимальной глубины дерева и других параметров по вашему усмотрению.
   b. Загрузите набор данных spam7 из файла spam7.csv. Постройте оптимальное, по вашему мнению, дерево классификации для параметра yesno. Объясните, как был осуществлён подбор параметров. Визуализируйте результирующее дерево решения. Определите наиболее влияющие признаки. Оцените качество классификации.
6. Загрузите набор данных из файла bank_scoring_train.csv. Это набор финансовых данных, характеризующий физических лиц. Целевым столбцом является «SeriousDlqin2yrs», означающий, ухудшится ли финансовая ситуация у клиента. Постройте систему по принятию решения о выдаче или невыдаче кредита физическому лицу. Сделайте как минимум 2 варианта системы на основе различных классификаторов. Подберите подходящую метрику качества работы системы исходя из специфики задачи и определите, принятие решения какой системой сработало лучше на bank_scoring_test.csv.