Оценка качества методов

Здравствуйте, уважаемые слушатели! Тема нашей лекции – Оценка качества методов. План лекции:

- Метрики оценки качества классификации. Матрица путанницы
- Подбор параметров по сетке
- Кривые Precision-Recall и ROC
- Микро- и макросреднее (Micro- and macro-average)
- Перекрестная оценка модели
- Показатели оценки качества регрессии

1. Вступительное слово

Основная тема сегодняшнего лекуии является оценивание качества моделей. Некоторые функции мы уже применяли в предыдущих занятиях

2. Метрики оценки качества классификации. Матрица путанницы

Для оценки качества работы алгоритмов машинного обучения было разработано множество метрик. Вычисление метрик осуществляется с помощью матрица путанницы (confusion matrix). Данная матрица позволяет увидеть, где модель классификации совершает больше ошибок. Матрица путанницы содержит в себе 4 комбинаций:

- True Negative (TN) – объекты, которые являются верно-отрицательными;
- True Positive (TP) – объекты, которые являются верно-положительными;
- False Positive (FP) – объекты, которые классифицированы как положительные, но в действительности являются отрицательными;
- False Negative (FN) – количество решений, которые классификатор предсказал как отрицательные объекты, но в действительности являющиеся положительными.

Столбцы матрица путанницы резервируются за фактическими решениями, а строки – за решениями, предсказанных классификатором.

		Реальный класс (Actual class)	
		Positive (1)	Negative (0)
Предсказанный класс (Predicted class)	Positive (1)	TP	FP
	Negative (0)	FN	TN

Функции recall, precision, accuracy используют два или три комбинаций матрицы неточностей и не дают объективной оценки результатов классификации. А метрика F1 использует все элементы матрицы ошибок и оценивает результаты классификатора при несбалансированных данных. Ниже представлены формулы нахождения этих метрик:

- Метрика recall определяется с помощью верно-положительных результатов, но вместо ложно-положительных решений (FP) учитывает количество объектов, классифицированных как отрицательные, но фактически являющиеся положительными (FN). Данная метрика оценки демонстрирует способность обнаруживать определенный класс, и показывает сколько примеров из положительных было потеряно в результате классификации. Чем выше значение метрики recall, тем меньше и значение потери правильных предсказаний.

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

- Precision отвечает за способность отличать заданный класс от всех остальных классов, но в отличие от recall, зависит только от положительных результатов, то есть precision – соотношение между верно-положительными результатами (TP) и всеми положительно классифицированными объектами (TP и FP).

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

- Accuracy рассчитывает долю правильных классификаций, и определяется как соотношение всех истинных результатов и суммы всех комбинаций матрицы ошибок.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

- F1-measure является метрикой, которая сводит к одному числу две основных метрики оценки: precision и recall. Она нужна для сбалансирования, когда максимальное значение precision и recall не достижимы одновременно.

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

3. Подбор параметров по сетке

Для подбора параметров устанавливается сочетание параметров классификатора и затем вызывается метод GridSearchCV(). Пример ниже показывает, как это сделать:

Отметим, что параметры, среди которых выполняется поиск, записываются в виде словаря param_grid. Поиск наилучшего сочетания ведется по фолдам (cv=5). В результате будет найдено такое сочетание параметров, которое показывает наилучший результат. В нашем случае accuracy вырасло с 0.94 до 0.97.

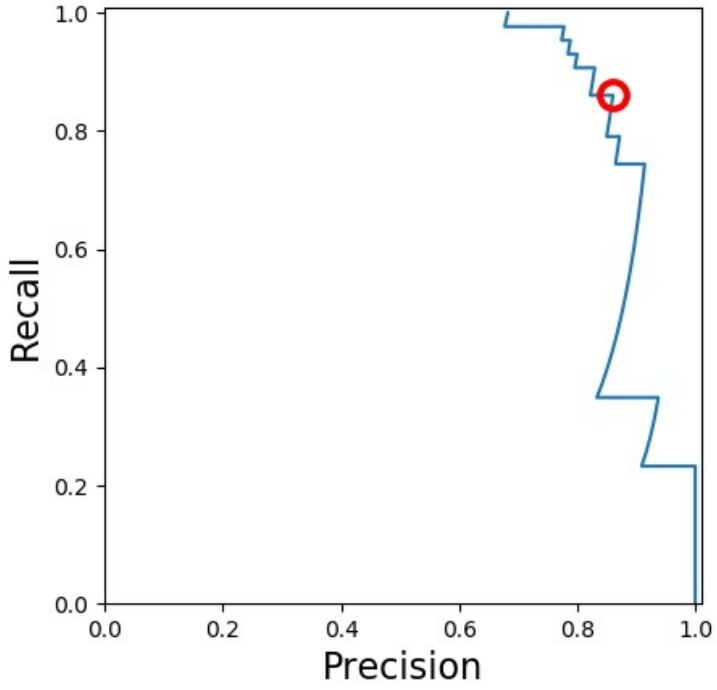
Отметим, что, несмотря на удобство применения GridSearchCV(), в случае большого объема данных для обучения и сложных классификаторов поиск может потребовать много времени.

4. Кривые Precision-Recall и ROC

Как мы поняли показатели качества очень важны при оценке классификтора, но для наглядного представления поведения precision, recall в зависимости от границы между двумя классами используется кривая Precision-Recall. Данная кривая позволяет показать компромисс между precision и recall.

Каждое значение оценки классификатора для каждого тестового примера показывает, насколько уверенно классификатор предсказывает положительный класс или отрицательный класс.

Выбор фиксированного порога принятия решения (treshold) определяет правило классификации. Обычно правило классификации таково, что объект принадлежит к классу или является «позитивным», если $h_\theta(x) \geq 0.5$, в противном случае, если $h_\theta(x) < 0.5$, объект является «негативным» (не принадлежит к классу).



Кривая Precision-Recall. Окружность на кривой идентифицирует оптимальное соотношение между точностью и полнотой

Изменяя порог принятия решения по всему диапазону возможных оценок классификатора, мы получаем серию результатов классификации, которые образуют указанную выше кривую. Если порог высокий, например, $h_\theta(x) \geq 0.7$, мы получаем «консервативный» классификатор с высоким уровнем precision, но низким recall. Уменьшая порог, получаем более «либеральный» классификатор с большим значением recall, зато низким precision. Нужно отметить, что не все классификаторы (например, SVC) имеют метод predict_proba(), который необходим для расчета вероятности принадлежности объекта к классу. Однако в таком случае для построения кривой можно воспользоваться методом decision_function(). Разница между этими методами заключается в том, что predict_proba() возвращает значение сигмоиды

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}},$$

в то время как decision_function() – значение полинома $z = \Theta^T x$ [1]. Соответственно, в первом случае мы получаем оценку вероятности принадлежности объекта к классу, а во втором – некоторое значение полинома, по которому можно судить, насколько объект близок к гиперповерхности, разделяющей классы.

Кривая Receiver Operating Characteristic (ROC), иногда называемая кривой ошибок, связывает показатели True positive (TP) и False positive (FP), рассмотренные выше. Для обобщения ROC-кривой используется оценка площади под кривой – Area Under the Curve (AUC). Использование AUC, с одной стороны, удобно тем, что показатели классификатора отображаются одной цифрой. Детали построения ROC AUC описаны в [2]. Однако, как и в случае других обобщающих оценок, часть информации, описывающей классификатор, теряется. Например, форма ROC-кривой не учитывается.

Рассмотрим пример.

---

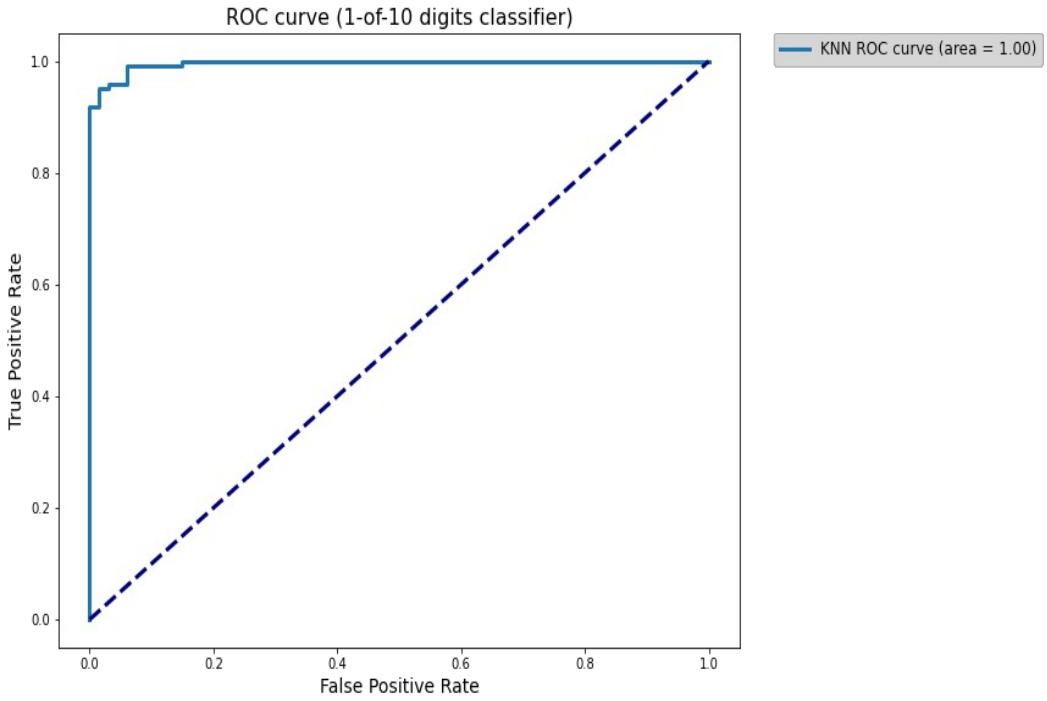
1 What is the difference between decision_function, predict_proba, and predict function for logistic regression problem. – https://stats.stackexchange.com/questions/329857/what-is-the-difference-between-decision-function-predict-proba-and-predict-fun

2 AUC ROC (площадь под кривой ошибок). – https://dyakonov.org/2017/07/28/auc-roc-площадь-под-кривой-ошибок/

Импортируем необходимые методы и воспользуемся найденными выше параметрами KNN-классификатора для получения оценок с помощью predict_proba():

```
from sklearn.metrics import roc_curve, auc

y_score = KNN. predict_proba (x_test)[:, 1 ]

fpr_knn, tpr_knn, _ = roc_curve(y_test, y_score)
roc_auc = auc(fpr_knn, tpr_knn)

plt.figure(figsize=(10,8))
plt.plot(fpr_knn, tpr_knn, lw=3, label='KNN ROC curve (area = {:0.2f})'.format(roc_auc))
plt.xlabel('False Positive Rate', fontsize=14)
plt.ylabel('True Positive Rate', fontsize=14)
plt.title('ROC curve (1-of-10 digits classifier)', fontsize=16)
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=12)
plt.plot([0, 1], [0, 1], color='navy', lw=3, linestyle='--')
plt.show()
```

В результате получим следующий график (рисунок 3.5):



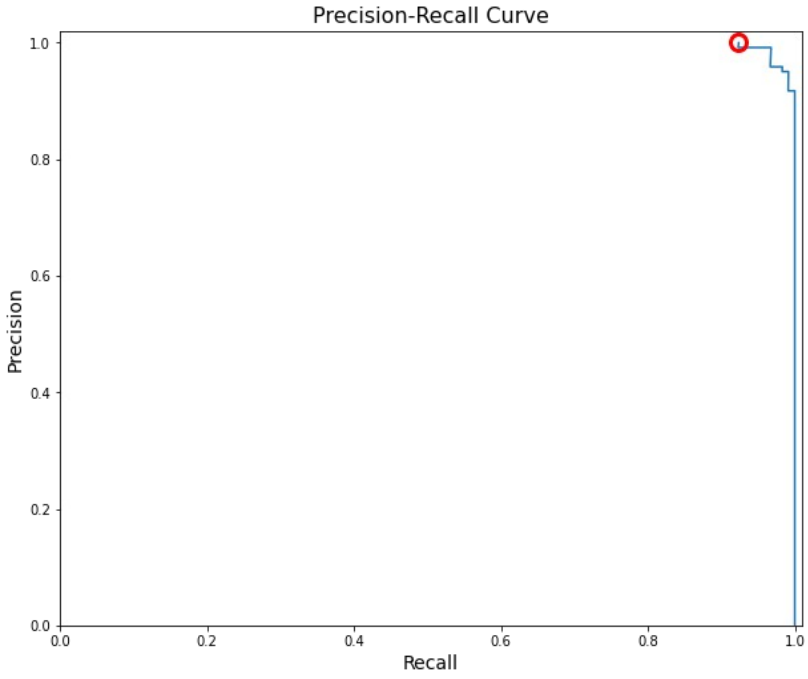
ROC-кривая для оптимально настроенного классификатора на базе KNN

Кривую precision_recall можно получить с помощью следующего фрагмента кода:

```
from sklearn. metrics import precision_recall_curve


precision, recall, thresholds = precision_recall_curve(y_test, y_score, pos_label=None)
closest_zero = np.argmin(np.abs(thresholds))
closest_zero_p = precision[closest_zero]
closest_zero_r = recall[closest_zero]
```

```
plt.figure(figsize=(10,8))
plt.xlim([0.0, 1.01])
plt.ylim([0.0, 1.04])
plt.plot(precision, recall, label='Precision-Recall Curve')
plt.plot(closest_zero_p, closest_zero_r, 'o', markersize = 12, fillstyle = 'none', c='r', mew=3)

plt.title('Precision-Recall Curve', fontsize=16)
plt.ylabel('Precision', fontsize=14)
plt.xlabel('Recall', fontsize=14)

plt.show()
```

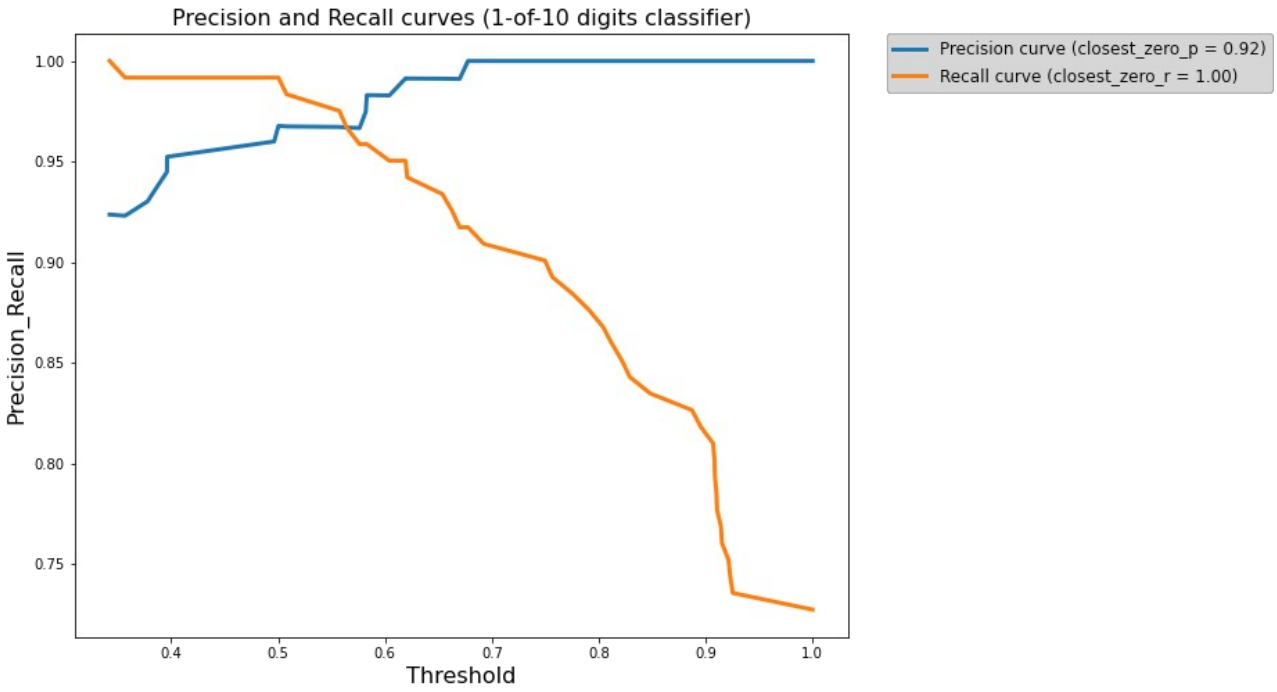
который для упомянутого ранее классификатора дает следующую картину (рисунок 3.6):



Кривая Precision-Recall для оптимально настроенного классификатора KNN

Следующий фрагмент кода позволяет отобразить кривые recall и precision в зависимости от пороговых значений (threshold):

```
plt.figure(figsize=(10,8))
plt.plot(thresholds,precision[0:44], lw=3, label='Precision curve (closest_zero_p =
{:0.2f})'.format(closest_zero_p))
plt.plot(thresholds,recall[0:44], lw=3, label='Recall curve (closest_zero_r =
{:0.2f})'.format(closest_zero_r))
plt.xlabel('Threshold', fontsize=16)
plt.ylabel('Precision_Recall', fontsize=16)
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=12)
plt.title('Precision and Recall curves (1-of-10 digits classifier)', fontsize=16)
plt.show()
```



Кривые recall и precision в зависимости от значений пороговых значений (threshold)

5. Микро- и макросреднее (Micro- and macro-average)

При классификации несбалансированных наборов данных, когда объектов одних классов существенно больше или меньше, чем других, возможны два варианта оценки основных показателей. Первый из них, называемый**macro-average,** предполагает, что каждый класс, несмотря на количество его членов, эквивалентен по весу, вносимому в общую оценку показателя. Это означает, что метрика качества рассчитывается внутри объектов каждого класса и затем усредняется по всем классам. Второй вариант усреднения, **micro-average,** предполагает, что объекты всех классов вносят равный вклад в показатели качества.

Расмотрим пример. У нас имеется 4 класса.

№ объекта	Класс	Предсказание классификатора
1	1	1
2	3	3
3	3	3
4	2	1
5	3	3
6	4	1
7	4	4
8	1	1
9	1	1
10	2	4
11	2	2
12	2	2
13	4	4
14	4	3
15	1	1
16	2	2
17	2	4

18	1	4
19	1	2
20	3	3

Micro-average precision = 13/20=0.65
Macro-average precision = (4/6+3/6+4/4+2/4)/4=0.67

6. Перекрестная оценка модели (Cross-validation)

Предположим, у нас небольшой набор данных для обучения. Мы разделили его на тренировочный и тестовый. Однако может получиться так, что разделение на тренировочный и тестовый наборы выполняется неравномерно. Например, в тренировочном наборе много объектов одного класса и мало объектов других классов. Тогда классификатор настроится на один из классов, в то время как другие объекты будут классифицироваться с затруднениями. Для того чтобы избежать таких ситуаций и получить более объективную оценку классификатора, выполняется так называемая кросс-валидация.

Алгоритм работы при этом следующий. Набор данных по-прежнему делится на тренировочный и тестовый. Однако вместо однократного разделения выполняется несколько разделений. Затем модель обучается и оценивается столько раз, сколько раз выполнялось разделение. Средний результат и будет более объективной оценкой модели.

Более детально набор данных разбивается на две группы. Первая часть набора – тренировочные и тестовые данные, которые разделяются несколько раз. Вторая часть – данные для окончательной проверки модели (validation), которые не участвуют в процессе кросс-валидации. Их задача – дать окончательную оценку модели в том случае, если в процессе перекрестной проверки выполняется настройка макропараметров модели. Если настройка модели не предусматривается, а выполняется лишь ее оценка, то этими данными можно пренебречь, а точнее, не выделять их в наборе данных. Например:

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(KNN, X, y, cv=5)
```

Разделение данных на тренировочный и тестовый можновыполнить, путем случайного выбора данных в тестовую и тренировочную части (ShuffleSplit). В результате выполнения следующего фрагмента кода:

```
from sklearn.model_selection import ShuffleSplit
cv = ShuffleSplit(n_splits=5, test_size=0.25, random_state=0)
scores = cross_val_score(KNN, X, y, cv=5)
```

Более подробно о перекрестном оценивании можно почитать в [3].

7. Показатели оценки качества регрессии

Для оценки качества регрессионной зависимости применяют следующие показатели:

1. r2_score.
2. mean_absolute_error (absolute difference of target & predicted values).
3. mean_squared_error (squared difference of target & predicted values).
4. median_absolute_error (robust to outliers).

[3] Cross-validation: evaluating estimator performance. – https://scikit-learn.org/stable/modules/cross_validation.html

В большинстве случаев достаточно использовать r2_score, иначе называемый коэффициентом детерминации, который в существующих библиотеках рассчитывается следующим образом:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

$$SS_{res} = \sum_{i=1}^{m_k}$$

$$SS_{tot} = \sum_{i=1}^{m_k} \quad ,$$

где$y^{(i)}$– фактическое значение;
$h^{(i)}$ – расчетное значение (значение функции гипотезы) для i-го примера;
$m_k \in m$ – часть обучающей выборки (множества размеченных объектов).

Наилучшее значение r2_score = 1. Минимальное значение 0 означает, что модель делает случайный выбор. Отрицательные значения коэффициента детерминации говорят о том, что модель фактически не работает.