УДК 004.896

ФУНДАМЕНТАЛЬНЫЕ ТИПЫ КРОСС-ВАЛИДАЦИИ ДЛЯ ОЦЕНКИ КАЧЕСТВА МОДЕЛЕЙ МАШИННОГО И ГЛУБОКОГО ОБУЧЕНИЯ

Пылов П. А., студент группы ИТм-201, I курс
Ивина О.А., к.т.н., доцент
Кузбасский государственный технический университет имени Т.Ф. Горбачева г. Кемерово

В данной статье рассматриваются различные типы кросс – валидации алгоритмов машинного и глубокого обучения. Валидация – это один из самых главных этапов в разработке модели искусственного интеллекта, так как именно он позволяет сделать выводы о точности созданного решения.

Кросс – валидация (от английского «Cross – validation», также «Перекрестная проверка») — это метод оценки модели, эффективность которого выше, чем использование остатков датасета из набора обучения. Проблема валидации на основе остаточных наборов заключается в том, что они не дают представление о том, насколько хорошо модель предскажет результат на новых данных, которые не использовались в обучение. Один из способов решения складывающейся проблемы – не использовать весь набор данных при обучении модели. Некоторые данные будут отложены до начала обучения. Затем, когда обучение завершено, данные, которые были извлечены изначально, будут использованы для проверки производительности изученной модели как «новые» данные. Эта основная идея для целого класса методов оценки моделей, которые носят общее название перекрестной проверки.

Особенности использования метода перекрёстной проверки:

- Позволяет оценить ожидаемую величину ошибки прогноза;
- Помогает выбрать наиболее подходящую модель из представленных;
- Способ проверки и доказательства некорректности работы модели.

Кросс – валидация – это целый набор различных типов проверки. Реализации могут отличаться в зависимости от конкретной общепринятой методики. Существуют следующие типы перекрёстной проверки:

1. Holdout method;

2. K-fold;
3. Leave One Out;
4. Bootstrap method.

Рассмотрим каждый тип по отдельности:

1. Метод на основе отложенных данных (от английского «holdout») – является самым простым видом перекрестной проверки. Набор данных должен быть разделен на два набора, называемых обучающим набором и тестовым набором. Аппроксиматор соответствует функции, составляемой только при использование обучающего набора. Затем предполагается, что аппроксиматор функции предсказывает выходные значения для данных в тестовом наборе (никогда ранее аппроксиматор «не видел» эти выходные значения). Ошибки, которые совершаются аппроксиматором, накапливаются для того, чтобы получить среднюю абсолютную ошибку тестового набора, которая используется для оценки модели.

   Преимущество этого метода в том, что он не требует больших вычислений. Однако, датасет может иметь высокую дисперсию, и задача сильно зависит от того, какие семплы данных оказываются в обучающем наборе, а какие – в наборе тестов, и, таким образом, оценка может существенно отличаться в зависимости от того, как организовано разбиение датасета.

   Пример разбиения данных на тренировочные и тестовые образцы данных приведен на рисунке 1. Процентное соотношение может варьироваться в зависимости от выбора исследователей и самой задачи, но, в большинстве случаев, составляет соотношение 70% : 30% тренировочного набора данных к тестовому соответственно. Гораздо реже датасет разделяется на две равные части.

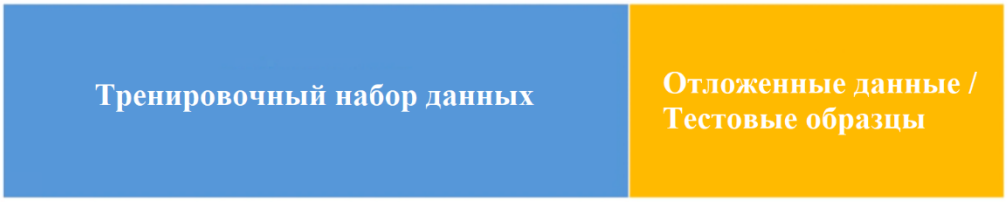

Рисунок 1 – Графическое представление разбиения датасета данных

2. Метод разделения по k фолдам (от английского «k-fold cross – validation») – один из способов улучшить метод отложенной выборки. Набор данных делится на k подмножеств, и методика holdout повторяется k раз. Каждый раз одно из k подмножеств используется в качестве тестового набора, а другие (k-1) поднаборы объединяются для формирования обучающего набора. Затем вычисляется средняя ошибка по всем k испытаниям.

   Преимущество этого метода в том, что менее важно, как данные разделяются. Каждая точка данных попадает в набор тестов ровно один раз, и она входит в набор тренировок (k-1) раз. Дисперсия полученной оценки уменьшается с увеличением числа k.

   Недостатком этого метода является то, что алгоритм обучения должен быть перезапущен с нуля k раз, что означает, что для выполнения оценки требуется в k раз больше вычислений. Вариация этого метода состоит в том, чтобы случайным образом разделить данные на тестовый и обучающий набор k в разное время. Преимущество этого варианта состоит в том, что появляется возможность самостоятельно выбирать размер каждого набора тестов и количество испытаний, которое было получено в среднем.

   Рассмотрим в качестве графического примера выборку. Разбиение совершим на k = 9 частей. Пример выборки, с произведённым над ней разбиением, приведен на рисунке 2

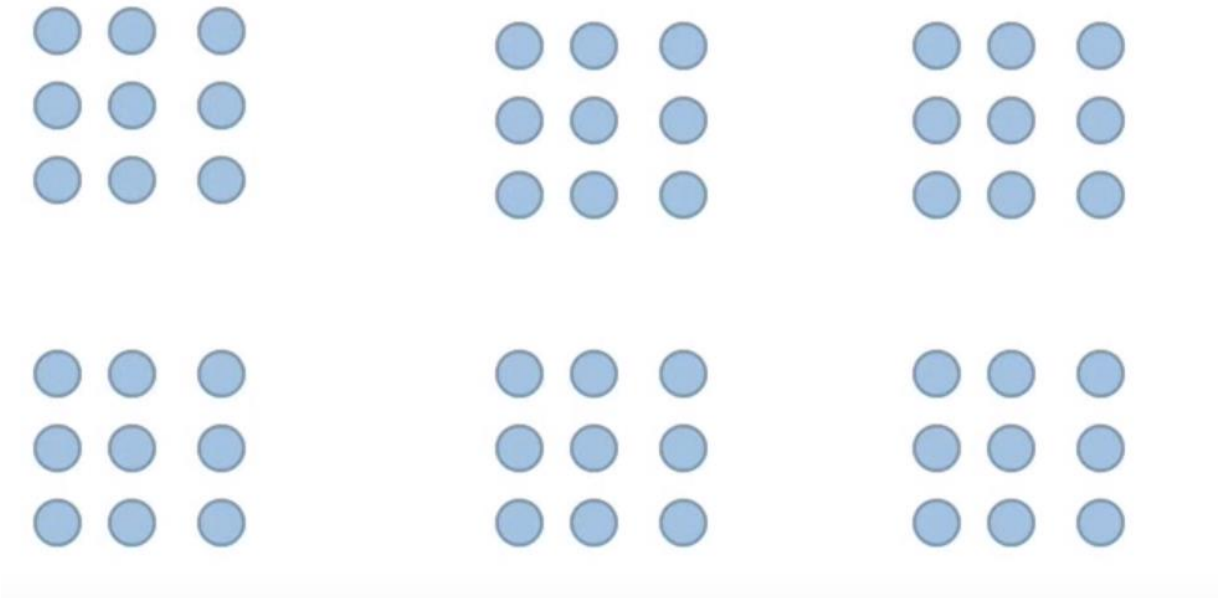

Рисунок 2 – Разделённая выборка

После этого поочерёдно будем использовать 5/6 частей выборки для тренировки алгоритма и лишь 1/6 для валидации. Выполним это ровно 6 раз,

так как данные валидации и тренировки будут изменяться в зависимости от того, какая часть будет относиться к валидационным данным в текущий момент времени. Отметим валидационные данные красным цветом, а тренировочные – синим. Пример последовательных обучений и валидации алгоритма машинного обучения на основе метода разбиения по k-частям приведены на рисунках 3 – 8.

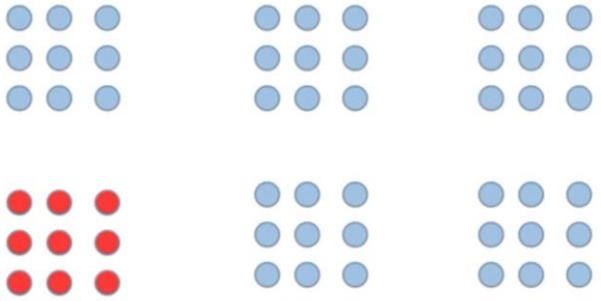

Рисунок 3 – 1 этап кросс – валидации

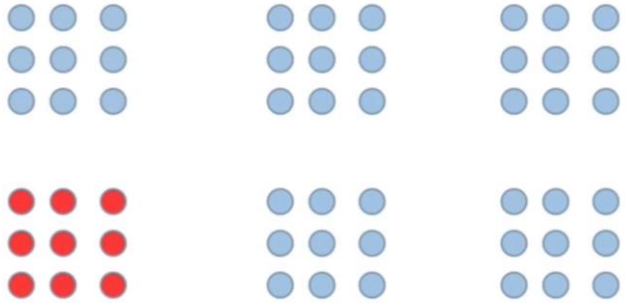

Рисунок 4 – 2 этап кросс – валидации

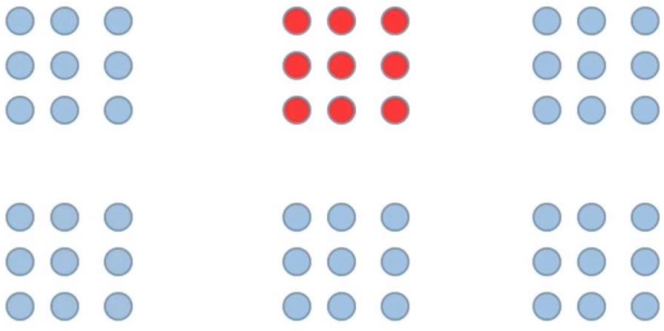

Рисунок 5 – 3 этап кросс – валидации

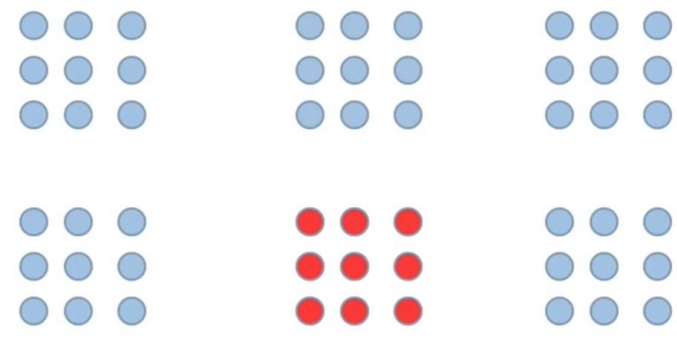

Рисунок 6 – 4 этап кросс – валидации

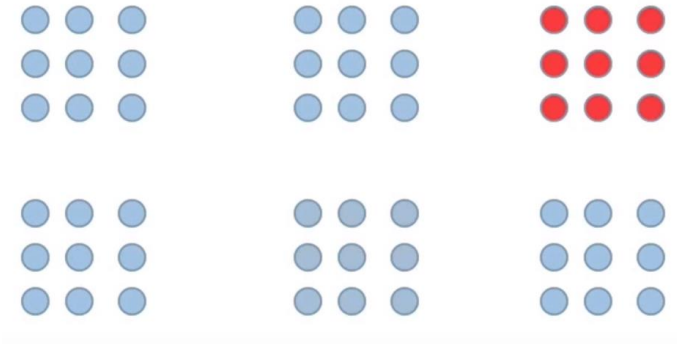

Рисунок 7 – 5 этап кросс – валидации

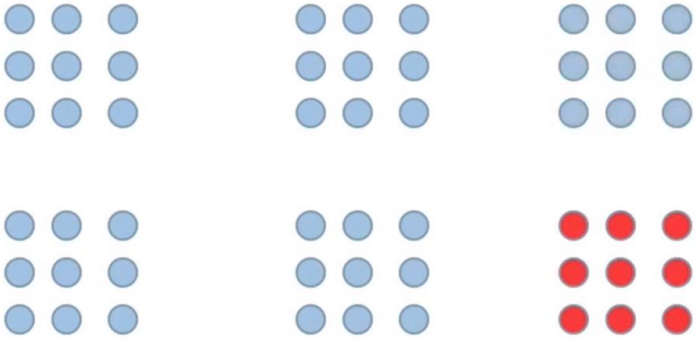

Рисунок 8 – 6 этап кросс – валидации

3. Метод валидации по отдельным компонентам объектов выборки

Перекрестная проверка по отдельным компонентам объектов выборки — это K-кратная перекрестная проверка, доведенная до логического предела, где K равно N, – числу точек данных в исходном датасете. Это означает, что N раз аппроксиматор функции обучается на всех данных, за исключением одной точки, и для этой точки делается прогноз. Частный случай предыдущего метода, с самым наибольшим числом проверок. Не применяется для выборок, где объём данных превышает миллиона значений, поскольку требуются огромные вычислительные мощности для обучения модели.

Схематичный пример выборки, при использование данного метода приведён на рисунке 9 (где T – тренировочные подмножества данных, V – данные для тестирования модели)

- K = N

Рисунок 9 – Схематичное представление исходной выборки для метода перекрестной проверки

4. Метод бутстрэпа (от английского «bootstrap») – это способ проверки, основанный на многократной генерации выборки подмножества при помощи метода Монте-Карло на базе имеющегося датасета.

Особенности метода:

- случайным образом выбирает набор данных из обучающей выборки;
- каждый образец получает тот же размер, что и тренировочный образец;
- существует возможность переоснастить модель при помощи данного метода;
- позволяет подробнее изучить модель.

Пример генерации подмножеств приведён на рисунке 10. Из тренировочного набора данных получаются подмножества такого же размера, как и сам набор тренировочных данных.

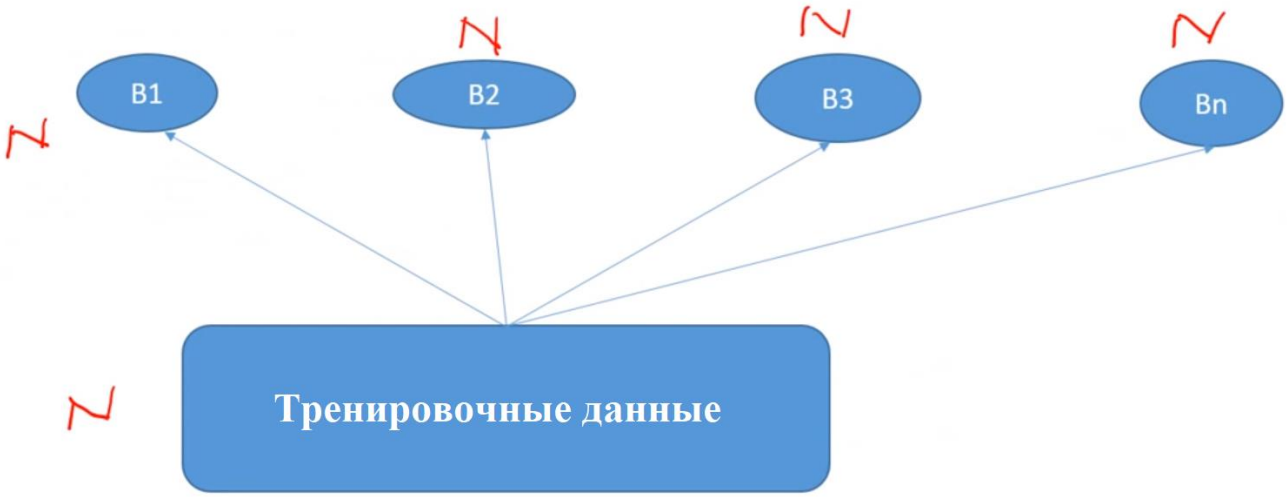

Рисунок 10 – Метод бутстрэпа в графической схеме

Подводя итог, стоит отметить, что выбор определённого типа валидации всегда зависит от конкретной прикладной задачи и многих её особенностей (размер выборки, количество параметров). Поэтому знание особенностей каждого типа во многом может помочь сделать правильный выбор в практической задаче разработчику и, как следствие, получить наиболее точный и интерпретируемый показатель действительной точности модели.

Список литературы:

1. Christopher Bishop. Pattern Recognition and Machine Learning, Plenum press, New York – London, 1971
2. M. Narasimha Murty, V. Susheela Devy. Introduction to pattern recognition and machine learning, IISc press, New Jersey – London, 2015
3. A. Geron. Hands-on Machine learning with Scikit-Learn, Keras, and TensorFlow, OREILLY Sebastopol, California – USA, 2019
4. Ivan Idris. Python Data Analysis. – Packt Publishing. 2014. – 319 с
5. Coding for Python – Black Dog i-Tech Series. 2019. - 160 с
6. Sunil Kapil. Clean Python – Apress. 2019. – 261 с
7. Дэвид Колец. Классические задачи Computer Science на языке Python – СПб.: Питер. 2020 – 256 с