

Тестовое задание

Дмитрий Донецков (ddonetskov@gmail.com)

Содержание

| | | | | | |
|---|-------------------------|---|---|-------------------|---|
| 1 | Теоретическая задача №1 | 1 | 4 | Урок | 3 |
| 2 | Теоретическая задача №2 | 2 | 5 | Упражнение (кейс) | 5 |
| 3 | Практическая задача | 2 | A | Прочее | 5 |

1. Теоретическая задача №1

Вопрос:

Можно ли использовать ковариацию двух признаков для анализа зависимостей между ними?

Ковариация двух признаков (двух случайных переменных) показывает насколько сильно они меняются *совместно*. Значение ковариации может быть от $-\infty$ до $+\infty$. Если посмотреть на логику формулы (см. ниже) вычисления эмпирического значения ковариации двух случайных переменных X и Y , то можно видеть, что это сумма таких произведений, что каждое произведение получает тем большее значение, чем более "синхронно" два признака отклоняются от своих средних значений, а знак ковариации показывает отклоняются ли признаки в одном направлении (положительное значение) или в разных (отрицательное значение):

$$\text{Теоретическая ковариация: } cov(X, Y) = E[(X - E X)(Y - E Y)] \quad (1)$$

$$\text{Выборочная ковариация: } cov(X, Y) = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) \quad (2)$$

Таким образом, более высокие значения ковариации получаются при более "синхронных" колебаниях переменных, что несёт информацию об их связи, об их зависимости между собой. Но! данную информацию тяжело интерпретировать в силу её ненормализованности (ковариация принимает значения от $-\infty$ до $+\infty$), особенно тяжело её использовать для сравнения силы зависимости между парами разных признаков, например, между двумя парами признаков: рост и вес человека, размер обуви и вес человека. На основе значений ковариаций для этих двух пар невозможно ответить на вопрос, что сильнее связано с весом человека. Данное затруднение исходит из того, что в ковариации содержится размерность исходных признаков, она не свободна от природы измерения самих признаков, которая может быть разной. Поэтому, для *оценки* силы связи между двумя признаками, *сравнения* силы связи между различными парами признаков используют нормализованное значение ковариации - корреляцию, которая уже свободна от размерностей, и принимает своё значение на интервале $[-1; +1]$ вне зависимости от того, множество каких значений могут иметь сами признаки.

💡 Что можно добавить в материал:

- таблицу сравнения свойств ковариации и корреляции,
- пояснения, почему ковариация всё равно остаётся очень важным понятием (например, для ковариационных матриц).

2. Теоретическая задача №2

Вопрос:

Ваш подчинённый обучил модель и получил качество классификации 99%. Какие бы вопросы про данные и про модель вы бы задали, чтобы убедиться, что модель полезная? Можно считать, что программных багов нет.

Качество классификации в 99% выглядит очень хорошим. Естественно, точная интерпретация такой оценки зависит от используемой метрики. Если из контекста задачи неясно какая метрика была использована, то следует задать вопрос об этом вплоть до просьбы показать формулу или фрагмент кода, при помощи которой/которого производилось вычисление метрики. Далее, для ухода от рассмотрения специфики различных метрик (recall, precision, F_1 , AUC и т.п.), предположим, что речь идёт о доле объектов с верно распознанными классами для них (accuracy).

Итак, качество классификации в 99% выглядит настолько хорошим, что стоит перепроверить, что полученная модель не является настолько специфичной, что её потом будет нецелесообразно использовать на практике. Другими словами, что нет случая переобучения, смещения оценки, при котором модель будет малоценной для новых объектов. Для такой проверки можно задать следующие вопросы:

1. Как распределены объекты по классам? Как удостоверились, что классификатор действительно работает (выдаёт разные значения)? Здесь, необходимо понять, что нет выраженного дисбаланса между классами, когда подавляющее большинство объектов относится к одному классу, а сам классификатор всегда выдаёт значение только одного класса. Другими словами, стоит провести проверку насколько полученная модель лучше dumb-модели. Также, стоит отметить, что в этом случае, несмотря на высокое качество классификации, такое решение "в лоб" может иметь высокую стоимость потерь для оставшегося 1% случаев, делая модель экономически неэффективной.
2. Каким образом была проведена оценка модели? Здесь, необходимо проверить, что оба множества объектов, используемые для построения модели (тренировки) и её последующей оценки (тестирования), были взяты таким образом, что каждое множество достаточно хорошо представляет собой множество всех возможных объектов (генеральную совокупность). Следует подтолкнуть обучающего к тому, что нужно было использовать техники train/test split, k-fold cross-validation, если они не были использованы.

💡 Что можно добавить в материал:

- пояснения по тому, что метрика accuracy - неполна, есть разные метрики оценки качества классификатора: recall, precision, F_1 и т.п.; некоторые из них специфичны, а другие были сконструированы, как интегральные.

3. Практическая задача

Задание:

Напишите функцию `one_hot_encode()`, которая принимает на вход объект `pandas.DataFrame` и название столбца, и выполняет one-hot-кодирование этого столбца. Гарантируется, что переданный столбец категориальный. Функция должна возвращать новую таблицу (`pandas.DataFrame`), в которой отсутствует старый столбец, а новые добавлены в конец. Использовать готовые функции для решения этой задачи (например, `pandas.get_dummies()`) не разрешается.

Код функции и её тесты оформлены в виде Jupyter-ноутбука и доступны для просмотра по ссылке https://nbviewer.jupyter.org/github/ddonetskov/misc/blob/master/yp_ds/one_hot_encoding.ipynb#.

! Примечание: реализованная функция не покрывает всех возможных практических случаев, а именно:

- если есть пропущенные данные в указанном столбце, то функция возвращает исключение `NotImplementedError`, т.к. логика обработки пропущенных значений может различаться в зависимости от задачи,
- кодирование выполняется для всех значений в столбце, никакое значение не удаляется, что может порождать проблему с коллинеарностью для соответствующих моделей (например, для линейной регрессии).

4. Урок

Тема урока:

Что делает функция `pandas.concat()`? В каких случаях её используют?

В библиотеке `pandas` есть функция `concat`. Её название - это сокращение от английского `concatenate`, что означает "соединять, сцеплять". Действительно, данная функция используется для соединения двух и более дата фреймов в один. Соединение можно производить как по столбцам и тогда получаемый дата фрейм будет состоять из исходных дата фреймов, „поставленных“ друг на друга (Рис. 1a), так и по строкам (индексам) и тогда получаемый дата фрейм будет состоять из исходных, „пристыкованных“ боком друг к другу (Рис. 1b). При этом, в получаемом дата фрейме могут появиться совершенно новые элементы, порождённые разницей столбцов или индексов исходных дата фреймов. Данным элементам по умолчанию будет присвоено `NaN`-значение.

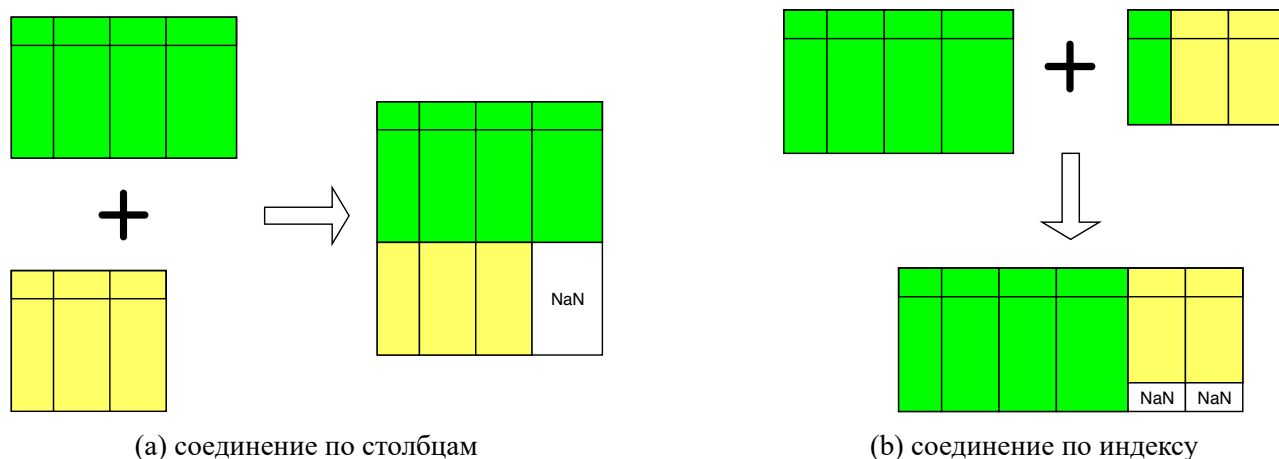


Рис. 1: Соединение двух дата фреймов

Рассмотрим на примере оба этих случая. В качестве набора данных будем использовать датасет World Happiness Report для 2015-2017, предоставленный ООН, и который доступен по ссылке <https://www.kaggle.com/unsdsn/world-happiness>.

Допустим, вы - исследователь счастья на планете и решили включить в свой анализ и данные из вышеуказанного датасета. Затруднение в том, что на каждый год приходится свой собственный файл, а вам хочется иметь данные в едином дата фрейме. Функция `concat` поможет здесь. Например, следующим образом можно объединить данные за 2015 и 2016 годы в один итоговый дата фрейм `whr`:

```

whr_2015 = pd.read_csv('data/2015.csv', sep=',')
whr_2016 = pd.read_csv('data/2016.csv', sep=',')

whr_2015['year'] = 2015
whr_2016['year'] = 2016

whr = pd.concat([whr_2015, whr_2016], axis=0, sort=False)

```

Полный код с пояснениями размещён по ссылке https://nbviewer.jupyter.org/github/ddonetskov/misc/blob/master/yp_ds/pandas_concat.ipynb, см. раздел 'Vertical Concatenation'. Обратите внимание!, что те столбцы, которые присутствуют не во всех соединяемых дата фреймах по умолчанию всё равно попадут в итоговый дата фрейм, а элементы, для которых данный столбец ранее не существовал, получают NaN-значения. Таким образом, функция concat придерживается принципа сохранения данных при объединении. ⚙ Задание: а) найдите такие столбцы для приведённого примера, б) добавьте в пример и файл 2017.csv (сложность объединения его с данными по другим двум годам состоит в том, что для него столбцы при том же смысловом значении получили другие именованья, поэтому их предварительно следует переименовать, привести к общему виду).

Приведённый выше пример можно продолжить для случая объединения по индексу. Например, вы решили остановиться на анализе данных только 2017 года, но хотели бы дополнить их статистикой по странам. Тогда, можно воспользоваться датасетом по ссылке <https://www.kaggle.com/fernandol/countries-of-the-world> и присоединить его данные к данным World Happiness Report для 2017 года. См. полный код в разделе 'Horizontal Concatenation' в том же ноутбуке. ⚙ Задание: изучите опцию ignore_index в документации, попробуйте выполнить объединение дата фреймов при ignore_index=True, и пояснить, почему итоговый результат получится другим?

```

whr_2017 = pd.read_csv('data/2017.csv', sep=',')
countries = pd.read_csv('data/countries of the world.csv', sep=',')

# removing spaces around the country names so they would match with whr_2017
countries['Country'] = countries['Country'].str.strip()

# setting the index which will be used to combine two data frame 'horizontally'
whr_2017 = whr_2017.set_index('Country')
countries = countries.set_index('Country')

whr_2017_ext = pd.concat([whr_2017, countries], axis=1, sort=False,
    ↪ ignore_index=False, join='inner')

```

! Примечания:

1. При объединении необходимо отслеживать, что имена общих колонок или значения индекса общих строк - совпадают, иначе полученный результат будет выглядеть, как объединение объектов без общих колонок или значений индекса (данные не объединятся по одним и тем же колонкам или строкам).
2. Функция concat также может использоваться и для Series.
3. Функция concat может использоваться для объединения произвольного количества дата фреймов, не только двух, но и трех, четырех и т.д.
4. Для объединения двух датафреймов по индексу можно использовать функции join и merge класса pandas.DataFrame. Они могут быть более удобными в силу своей специализации.

5. Упражнение (кейс)

Тема:

Классификация для несбалансированных классов. Ошибки первого и второго рода, ROC-кривая, AUC-ROC.

Разбираемый случай должен быть сконструирован таким образом, чтобы обучающий смог получить понимание, почему а) оценка качества классификатора только по ассигасу не является полной, а иногда и проигрышной, б) существуют и другие оценки качества классификатора, включая интегральные. Для этого в качестве бизнес-задачи можно выбрать такую, что для ошибок и первого, и второго рода есть стоимость потерь, а при помощи анализа ROC-кривой следует выбрать наиболее оптимальное, с точки зрения минимизации суммы потерь, значение порога срабатывания классификатора.

Например, можно рассмотреть процесс в поликлинике, когда при помощи бинарного классификатора определяют стоит ли отправить пациента на дальнейшую диагностику, не очень безопасную. При этом стоимость потерь (иски, компенсации и т.п.) при диагностике здорового пациента (по сути, бесполезной диагностики) составляет C_1 , а стоимость потерь (иски, компенсации и т.п.) при не отправлении больного пациента (по сути, пропущенной диагностики) составляет C_2 .

Тогда, датасет может быть сконструирован следующим образом:

- объект - это пациент с физиологическими и прочими признаками, целевым признаком (не болен/болен),
- бОльшая часть случаев отмечена, как "не болен т.к. допустим что заболевание - редкое,
- граница между больными и небольными пациентами - размытая, в том смысле, что классификатор, насколько он не был бы хорош, всегда будет делать ошибки и первого, и второго рода.

Далее, в задании обучающий знакомится с понятиями ошибок, получает представление о том, что ассигасу может достигать высоких значений даже при выдачи классификатором всегда "не болен" (dumb или константный классификатор), знакомится с recall/precision/FPR, строит ROC-кривую, вычисляет стоимость потерь в зависимости от выбранного порога срабатывания классификатора. Было бы очень полезно выполнить данное упражнение для разных классификаторов, чтобы сравнить их по метриками качеств, особенно, по ROC-кривой.

💡 Что можно улучшить. Для того, чтобы разбираемый случай (кейс) получился запоминающимся, можно сделать его анекдотичным, указав, что речь идёт о хомячках, которые заболевают тем, что, например, в колесе бегут не вперёд, а назад. Можно взять каких-нибудь литературных персонажей, например, демонов Максвелла из НИИЧАВО, которых нужно заменить классификатором по определению впускать ли посетителя в Институт или нет.

А. Прочее

1. Темы и задачи, приведённые выше, могут потребовать бОльшего обсуждения, т.к. они достаточно обширны. В рамках выполнения тестового задания я придерживался принципа минимальной достаточности, но старался показать потенциал развития обучения по той или иной теме.
2. Материал, который предназначен для студентов, лучше оформить не в виде документа подобному текущему, а выложить на сайт, который поддерживает режим совместного редактирования (вики и т.п.). При этом должно выполняться критическое требования - "движок" сайта должен поддерживать оформление математических формул, как это реализовано, например, на Stack Exchange. Но, поскольку у меня нет доступа к платформе, используемой Яндекс.Практикум, то данный материал был пока оформлен в виде текущего PDF-документа (с использованием системы вёрстки L^AT_EX).