Визуализация данных

Peer Review

Визуализация данных - первый шаг в решении практически любой задачи анализа данных, в частности, при участии в соревновании. Визуализация помогает решить, какую предобработку данных нужно провести, какие методы лучше использовать для предсказания, какие признаки добавить. В этом Peer Review мы будем работать с данными kaggle-соревнования (https://www.kaggle.com/c/prudential-life-insurance-assessment), завершившегося зимой 2016 года.

Задача соревнования: по характеристикам человека и его заявки на медицинскую страховку предсказать степень риска, характерную для данной страховки.

Чем будем заниматься мы:

- рассматривать данные;
- пробовать разные методы визуализации;
- анализировать графики и пытаться сделать выводы, полезные для следующих этапов решения задачи.

Целевой признак (степень риска) в задаче номинальный: целые числа от 1 до 8, а метрика качества, предложенная организаторами соревнования, оценивает степень согласованности двух рейтингов. Поэтому задачу можно решать и методами классификации, и методами регрессии (в последнем случае придется округлять предсказания). Это стоит учитывать при анализе результатов визуализации.

Мы будем визуализировать только обучающую выборку. Часто при решении конкурсов полезно визуализировать также тестовую выборку (на которой нужно сделать предсказание), чтобы убедиться, что данные идентичны.

Часть кода, необходимого для построения графиков, уже написана; для другой части даны инструкции, по которым вам нужно написать скрипт; такие ячейки помечены # Код X, X - число. Кроме того, вам необходимо проанализировать графики, ответив на вопросы.

Внимание! Мы будем заниматься именно визуализацией - построение изображением. Пожалуйста, отвечайте на вопросы именно **по изображениям**. Если вы считаете, что картинка слишком маленькая, и пытаетесь найти ответ на вопрос другими способами, сопоставляйте его с изображением, потому что в рекомендациях к проверке именно такие ответы. Если вы считаете, что вопрос не корректен, пишите об этом в *специальном* треде **"Peer Review по визуализации: формулировки вопросов."** на форуме. Если вы (после проверки других работ) считаете, что рекомендация к проверке не корректна, пишите об этом **в обратной связи к заданию**.

Визуализацию в python часто делают с помощью библиотеки seaborn. Установить ее можно командой pip install seaborn. Если вы по каким-то причинам не можете установить библиотеку, вам придется строить некоторые графики самостоятельно или использовать их аналоги в pyplot. Pandas также использует seaborn, чтобы строить графики.

In [ ]:
```
import numpy as np
import pandas
from matplotlib import pyplot as plt
import seaborn
%matplotlib inline
```

Считываем данные:

In [ ]:
```
data = pandas.read_csv("train.csv", na_values="NaN")
```

In [ ]:
```
data.head()
```

Часто в соревнованиях смысл признаков не известен, в нашем соревновании это не совсем так. Вы можете посмотреть описание признаков на странице с данными (https://www.kaggle.com/c/prudential-life-insurance-assessment/data). Резюме: значение известно только для нескольких признаков, для остальных известна только группа, к которой этот признак принадлежит (например, медицинские данные) и тип признака: вещественный, целочисленный или категориальный. При этом неясно, можно ли считать категориальные признаки упорядоченными.

Создадим три списка признаков, соответствующие их группам: вещественные, целочисленные и категориальные (эти списки даны на странице соревнования). Уберем признак Id, так как он не несет смысловой нагрузки.

Если бы типы признаков были не даны, нам бы пришлось вручную просматривать все 128 признаков, чтобы понять, какие значения они принимают.

In [ ]:
```
real_features = ["Product_Info_4", "Ins_Age", "Ht", "Wt", "BMI", "Employment
                 "Employment_Info_4", "Employment_Info_6",
                 "Insurance_History_5", "Family_Hist_2", "Family_Hist_3", "F
                 "Family_Hist_5"]
discrete_features = ["Medical_History_1", "Medical_History_10", "Medical_His
                     "Medical_History_24", "Medical_History_32"]
cat_features =
data.columns.drop(real_features).drop(discrete_features).drop(["Id", "Respon
```

Первичный анализ

Есть набор стандартных приемов визуализации, которые нужно попробовать, когда вы начинаете работу с набором данных; к ним относятся построение гистограмм признаков (histogram, density estimation), л статистик, оценка зависимости целевого признака от остальных (boxplot, scatterplot, violinplot), визуализация пар признаков (как правило, scatterplot). Конкретный вид графика, который вам подходит, зависит от типа признаков, хороший обзор приведен в разделе Plotting functions туториала seaborn (https://stanford.edu/~mwaskom/software/seaborn/tutorial.html).

Сначала рассмотрим числовые признаки, затем - категориальные.

Выведем статистики вещественных и целочисленных признаков:

In [ ]:
```
data[real_features].describe()
```

In [ ]:
```
data[discrete_features].describe()
```

In [ ]:
```
data.shape
```

Ответьте на вопросы (Блок 1):

- Есть ли пропуски в данных? Перечислите вещественные и целочисленные признаки, которые известны для всех объектов.
  - Ответ:
- Перечислите вещественные и целочисленные признаки, значение которых известно менее, чем для половины объектов.
  - Ответ:
- Одинаков ли масштаб вещественных признаков? Масштаб целочисленных признаков? *
  - Ответ:

* Будем считать, что масштаб двух признаков одинаков, если их минимумы отличаются не более, чем в 2 раза, и аналогично с максимумами.

Мы видим, что в датасете есть признаки, которые не известны почти для всех объектов. Кроме того, легко проверить, что в выборке нет объектов, для которых известны все признаки, даже если рассматривать только вещественные. Значит, при решении задачи нужно было бы использовать более сложные методы обработки пропусков, чем удаление объектов.

Постройте гистограммы вещественных и целочисленных признаков.

Вместо того, чтобы в цикле по признакам строить отдельно каждую гистограмму, стоит воспользоваться методом hist датафрейма. Рекомендуется отдельно вывести гистограммы вещественных и целочисленных признаков. Установите размер изображения (20, 20) для первой группы признаков и (10, 10) для второй, bins=100.

In [ ]:
```
# Код 1. Постройте гистограммы.
```

Ответьте на вопросы (Блок 2):

- Есть ли константные признаки среди вещественных и целочисленных? Признак является константным, если множество его значений состоит из единственного элемента.
  - Ответ:
- Предположим, что мы составили выборку, в которую вошли все и только вещественные признаки. Можно ли ее описать многомерным нормальным распределением? *
  - Ответ:

* Плотность многомерного нормального распределения во всех проекциях на отдельные переменные должна быть симметрична, куполообразна, а значит, унимодальна.

Среди вещественных есть признаки с очень большим перекосом в сторону какого-то одного значения (например, Employmennt_Info_4), возможно, их стоило бы прологарифмировать при решении задачи. Кроме того, есть признаки со сложной структурой гистограммы. Распределения целочисленных признаков (относящихся к группе медицинских) имеют схожую структуру.

В целом данные разнородны, и описать их одним вероятностным распределением непросто.

Теперь визуализируем признаки попарно.

Построим scatterplot для пар вещественных признаков. Для этого в seaborn есть функция pairplot. Исключим признаки, распределение которых не похоже на колокол (хотя бы скошенный и неровный), тем самым мы уменьшим размер таблицы пар. На диагоналях таблицы будут стоять оценки распределений признаков (гистограммы или восстановленные плотности, diag_kind="hist" или "kde"). Если указать параметр hue = дискретный целевой признак, то разные его значения будут отображаться разными цветами.

In [ ]:
```
seaborn.pairplot(data[real_features+["Response"]].drop(
        ["Employment_Info_4", "Employment_Info_6", "Insurance_History_5", "I
        axis=1),
        hue="Response", diag_kind="kde")
```

Классы накрывают друг друга, и графики не очень информативны. Но они позволяют ответить на некоторые вопросы о взаимоотношениях признаков и соотношений признаков и классов.

Ответьте на вопросы (Блок 3):

- Есть ли пары признаков, у которых облака точек разных классов отличаются? Иными словами, есть ли графики, на которых видны облака разных классов, а не только последнего нарисованного?
  - Ответ:
- Есть ли объекты-выбросы, то есть такие точки, которые стоят далеко от общего облака точек?
  - Ответ:

В целом облака точек выглядят достаточно компактно (в неформальном понимании этого слова), их можно приблизить искривленным кругом.

Теперь постройте такие же графики для целочисленных признаков (никакие признаки удалять не нужно, потому что таких признаков и так немного).

In [ ]:
```
# Код 2. Постройте pairplot для целочисленных признаков
```

Графики выглядят еще менее информативно. Заметна тенденция, что пары признаков сконцентрированы либо на сторона квадрата [0, 240] x [0, 240], либо на его диагонали, то есть признаки как-то связаны.

Посмотрим на корреляции признаков, чтобы узнать, не нужно ли удалять какие-то признаки перед (гипотетическим) построением модели.

Для визуализации матрицы попарных корреляций удобно использовать функцию seaborn.heatmap, она автоматически подпишет признаки на осях и покажет colorbar. Мы вычисляем корреляции только между вещественными признаками, для целочисленных признаков корреляции вычисляется по другим формулам.

In [ ]:
```
seaborn.heatmap(data[real_features].corr(), square=True)
```

Ответьте на вопрос (Блок 4):

- Есть ли пара (несовпадающих) признаков, корреляция между которыми больше 0.9 (можно проверить программно)? Если есть, то выпишите эту пару. Напоминаем, что матрица корреляций симметрична, и дважды выписывать одну пару признаков не нужно.
  - Ответ:

Перейдем к визуализации категориальных признаков.

Посчитаем количество значений для каждого признака.

Строим много графиков вручную (а не с помощью готовой функции, как с hist или pairplot). Для этого создаем новую pyplot-фигуру, указываем, сколько графиков на ней будет, задаем размер; параметр sharey говорит не подписывать все оси отдельно, а подписать их только один раз слева; здесь это уместно, потому что масштабы всех счетчиков одни и те же.

В цикле по всем категориальным признакам строим countplot признака с помощью seaborn. Указываем параметр data, какой признак визуализировать, а также передаем ссылку на конкретную ячейку таблицы, в которой нужно изобразить график. Признаков всего 108, поэтому последние две ячейки таблицы графиков размером 11 x 10 останутся пустыми.

In [ ]:
```
fig, axes = plt.subplots(11, 10, figsize=(20, 20), sharey=True)
for i in range(len(cat_features)):
    seaborn.countplot(x=cat_features[i], data=data, ax=axes[i / 10, i % 10])
```

Ответьте на вопросы (Блок 5):

- Есть ли среди категориальных признаков константные?
  - Ответ:
- Есть ли признаки с количеством возможных категорий (число значений признака) больше 5?
  - Ответ:

У нас есть много признаков с сильными перекосами в сторону отдельного признака. Учитывая, что среди категориальных признаков много медицинских показателей, можно предположить, что именно эти признаки сильно влияют на увеличение риска (целевой признак). Проверить это предположение можно, построив такие же countplot с разбивкой каждого значения дополнительно по классам (то есть у нас будет несколько групп столбиков, и в каждой группе их будет 8). Это можно сделать, указав параметр hue в этой функции аналогично тому, как мы это делали выше.

Постройте графики countplot для признаков 'Medical_Keyword_23', 'Medical_Keyword_39', 'Medical_Keyword_45' (признаки выбраны случайно) с разбивкой по классам.

In [ ]:
```
# Код 3. Постройте countplot
```

Ответьте на вопрос (Блок 6):

- Наличие каких из этих трех факторов сильно повышает риск? Будем считать, что наличие признака сильно повышает риск, если количество человек, имеющих этот признак (то есть он для них равен 1) и отнесенных к категории риска 8, больше, чем такая же величина для любой другой категории риска.

Наконец, посмотрим на распределение целевого признака, чтобы узнать, сбалансированы ли классы:

In [ ]:
```
seaborn.countplot(data.Response)
```

В категорию 8 люди попадают чаще, чем в другие категории.

Визуализация с помощью понижения размерности

Далее можно воспользоваться средствами понижения размерности. Для задачи с дискретным целевым признаком это позволит понять, какие классы хорошо разделяются, а какие - нет.

Такие методы строят матрицу попарных расстояний между объектами, которая в случае, когда объектов много, будет занимать много памяти. Кроме того, отображать много точек на scatter plot (а именно его используют для визуализации результата понижения размерности) неудобно. Поэтому мы перемешаем выборку (и далее будем использовать ее) и выберем првые 1000 объектов для понижения размерности. Вы можете попробовать взять больше объектов.

In [ ]:
```
from sklearn.utils import shuffle
from sklearn.preprocessing import scale
```

In [ ]:
```
sdata = shuffle(data, random_state=321)
# del data  # удалите неперемешанные данные, если не хватает оперативной па
```

Методы sklearn не принимают матрицы с пропусками (nan). Чтобы избежать этой проблемы, не будем рассматривать признаки, которые имеют много пропусков (последние четыре в списке вещественных признаков). Кроме того, ограничимся рассмотрением вещественных признаков.

В следующей ячейке мы отбираем нужные признаки, затем находим объекты, у которых все признаки известны (нет пропусков в выбранных признаках), а затем создаем отдельно матрицу объекты-признаки для работы методов понижения размерности и отдельно вектор правильных ответов на этих объектах: data_subset и response_subset. Кроме того, мы дополнительно стандартизуем нашу маленькую выборку, потому что методы понижения размерности очень чувствительны к разномасштабным данным (это отдельно указано в туториале (http://scikit-learn.org/stable/modules/manifold.html#multidimensional-scaling) в sklearn)

In [ ]:
```
subset_l  = 1000
selected_features = real_features[:-4]
objects_with_nan = sdata.index[np.any(np.isnan(sdata[selected_features].valu
data_subset = scale(sdata[selected_features].drop(objects_with_nan, axis=0)
response_subset = sdata["Response"].drop(objects_with_nan, axis=0)[:subset_
```

Будем строить визуализацию методами, разобранными на лекции: t-SNE и MDS.

In [ ]:
```
from sklearn.manifold import TSNE
import matplotlib.cm as cm # импортируем цветовые схемы, чтобы рисовать гра
```

Методы понижения размерности имеют такой же интерфейс, как классификаторы и регрессоры. Для построения визуализации t-SNE нужно создать объект класса и вызвать его метод fit_transform, который вернет матрицу размера число объектов x новая размерность; по умолчанию новая размерность равна 2. Выполните эти действия и запишите результат работы метода в переменную tsne_representation. Рекомендуем ознакомиться с параметрами метода на странице документации TSNE (http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html#sklearn.manifold.TSNE). Зафиксируйте случайное приближение random_state=321, чтобы результат визуализации у всех был одинаковый.

In [ ]:
```
# Код 4. Присвойте переменной tsne_representation результат понижения размер
# с параметрами по умолчанию
```

Визуализируем полученное представление. Для этого создадим набор цветов по количеству классов, а затем в цикле по классам будем отображать представления точек, относящихся к этому классу. Будем указывать параметр alpha=0.5, чтобы сделать точки полупрозрачными, это лучше в ситуации, когда точки накладываются.

In [ ]:
```
colors = cm.rainbow(np.linspace(0, 1, len(set(response_subset))))
for y, c in zip(set(data.Response), colors):
    plt.scatter(tsne_representation[response_subset.values==y, 0],
                tsne_representation[response_subset.values==y, 1], c=c, alpl
plt.legend()
```

Теперь сделаем то же с MDS.

In [ ]:
```
from sklearn.manifold import MDS
from sklearn.metrics.pairwise import pairwise_distances
```

In [ ]:
```
# Код 5. Присвойте переменной MDS_transformed результат понижения размернос
# с параметрами по умолчанию
# Не забудьте зафиксировать random_state=321
```

In [ ]:
```
colors = cm.rainbow(np.linspace(0, 1, len(set(response_subset))))
for y, c in zip(set(response_subset), colors):
    plt.scatter(MDS_transformed[response_subset.values==y, 0],
                MDS_transformed[response_subset.values==y, 1],
                c=c, alpha=0.5, label=str(y))
plt.legend()
#plt.xlim(-5, 5)   # масса точек концентрируется в этом масштабе
#plt.ylim(-5, 5)   # рекомендуем сначала отобразить визуализацию целиком,
# а затем раскомментировать эти строки.
```

Для построения представления можно пробовать разные метрики, их список доступен в документации scipy (http://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.spatial.distance.pdist.html) (потому что sklearn использует метрики scipy). Вы можете попробовать использовать разные метрики в t-SNE и MDS и смотреть получающиеся визуализации, по мнению автора задания наиболее информативной выглядит визуализация MDS с косинусной метрикой ("cosine").

В t_SNE метрику можно указать при создании объекта класса TSNE, в MDS это реализуется несколько сложнее. Нужно указать dissimilarity="precomputed", а в fit_transform подать не матрицу объектов, а матрицу попарных расстояний между объектами. Создать ее можно с помощью функции pairwise_distances с параметрами: матрица объектов, метрика.

In [ ]:
```
# Код 6. Присвойте переменной MDS_transformed_cos результат понижения разме
# с косинусной метрикой
```

In [ ]:
```
colors = cm.rainbow(np.linspace(0, 1, len(set(response_subset))))
for y, c in zip(set(response_subset), colors):
    plt.scatter(MDS_transformed_cos[response_subset.values[:subset_l]==y, 0]
                MDS_transformed_cos[response_subset.values[:subset_l]==y, 1]
                c=c, alpha=0.5, label=str(y))
plt.legend()
```

Ответьте на вопросы (Блок 7): Будем нумеровать визуализации от 1 до 3: tSNE, MDS по умолчанию, MDS с косинусной метрикой.

- Есть ли визуализация, на которой классы хорошо разделяются? Классы хорошо разделяются, если между облаками точек разных классов можно проводить границы. Если есть, укажите хотя бы одну такую визуализацию.
  - Ответ:
- Есть ли визуализации, по которым видно, что объекты с наименьшим риском (1 и 2) отличаются от объектов с высоким риском (7 и 8)? Будем считать, что объекты классов A и B отличаются, если можно выделить область плоскости, в которой в целом больше точек класса A, и можно выделить область, в которой больше точек класса B. Если есть, укажите хотя бы одну такую визуализацию.
  - Ответ:
- Есть ли визуализации, на которым видны отдельно стоящие (от общего облака) точки? Если есть, укажите хотя бы одну такую визуализацию.
  - Ответ:

Визуализации разными методами достаточно сильно отличаются, но на каждой можно условно выделить направление, вдоль которого происходит увеличение риска от 1 до 8 (то есть точки постепенно меняют цвет). Если бы это не выполнялось, можно было бы предположить, что задача не решаема.

Поиск аномалий

Воспользуемся методами поиска аномалий, рассмотренными в лекции.

One class SVM

Для простоты выберем вещественные признаки, которые известны для всех объектов: Product_Info_4, Ins_Age, Ht, Wt, BMI. Оставим только те, которые относятся к человеку, то есть не будем рассматривать Product_Info_4. Получится, что мы отбираем нестандартных по комбинации вес/рост/возраст людей.

In [ ]:
```
from sklearn import svm
```

In [ ]:
```
person_features = ["Ins_Age", "Ht", "Wt", "BMI"]
```

Два ключевых параметра OC_SVM - gamma и nu. Первый влияет на то, как хорошо граница будет приближать данные, второй - сколько точек нужно относить к выбросам. Вы можете попробовать разные значения, в том числе, значения по умолчанию, и убедиться, что при них алгоритм работает не очень адекватно. Запустите следующую ячейку, на ее выполнение может понадобиться некоторое время.

In [ ]:
```
svm_ = svm.OneClassSVM(gamma=10, nu=0.01)
svm_.fit(sdata[person_features])
```

In [ ]:
```
labels = svm_.predict(sdata[person_features])
```

In [ ]:
```
(labels==1).mean()
```

У вас должно получиться, что около 98% объектов не признаются выбросами.

Попытаемся визуализировать, какие объекты отнесены к шумовым. Для этого нужно построить scatter-графики для каждой пары признаков в person_features (всего 6 графиков). На каждом графике нужно отдельно отобразить точки с labels==1 и labels==-1.

Создайте pyplot-фигуру с 6 графиками: 2 x 3, укажите размер фигуры (12, 8). Затем в цикле по **парам признаков** из person_features отобразите scatter-графики точек (сделайте точки с разными labels разных цветов: синие, c="blue", - обычные точки, у которых labels==1, красные, c="red", - шумовые, у которых labels==-1.) Для построения графика можно пользоваться командой axes[...].scatter(...), вместо ... - ваш код. Функция scatter принимает две вектора одинаковой длины, абсциссы и ординаты точек, и дополнительные параметры, например, цвет с и коэффициент прозрачности точек alpha. Подпишите оси названиями признаков, это можно сделать с помощью команды axes[...].set_xlabel(...) или axes[...].ylabel(...).

In [ ]:
```
# Код 7. Постройте 6 графиков
```

Вы должны увидеть, что, во-первых, облака синие точки заполняют почти все облако точек, а во-вторых, красные точки находятся на границе или далеко от облака (интуитивно понятно, что это аномальные объекты).

Непараметрическое восстановление плотности.

Искать аномалии можно, анализируя непараметрически восстановленную плотность одномерного распределения. Так можно заметить, что какие-то объекты выбиваются из общей тенденции.

В seaborn это удобно делать с помощью метода distplot. Посмотрев на гистограммы в начале ноутбука, выберем для ближайшего рассмотрения признаки BMI, Employment_Info_1, Medical_History_32.

Примените метод distplot к этим трем признакам, указав bins=50. Рекомендуется удалять значения nan с помощью метода датафрейма .dropna().

In [ ]:
```
# Код 8. Восстановите плотности трех для трех указанных признаков
```

На трех графиках видно, что в конце множества значений признака есть скачок, и это явление очень похоже на аномалию.

Заключение

Мы познакомились с методами визуализации и отбора шумовых объектов и посмотрели на примере, с чего стоит начинать работу с новым датасетом. Как и в других методах анализа данных, в методах визуализации приходится подбирать параметры, но здесь эта задача осложняется отсутствием автоматически вычисляемого критерия качества визуализации, и подбирать параметры приходится вручную.