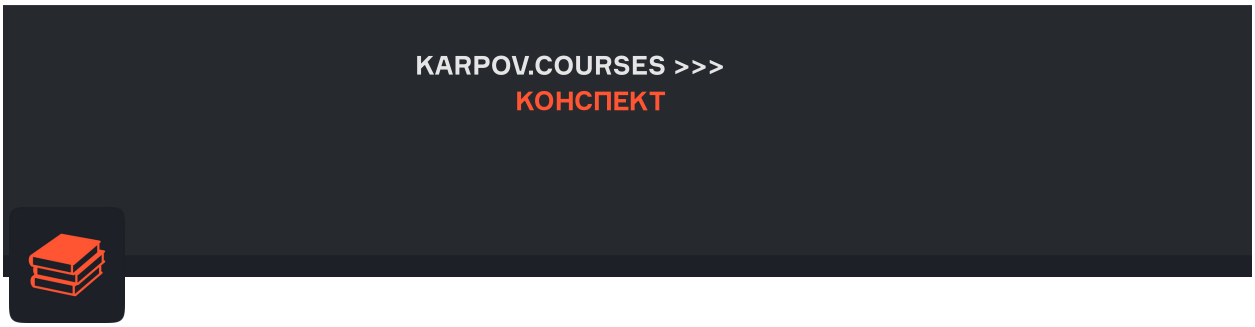


Конспект > 7 урок > Методы отбора признаков

>Оглавление

>Отбор признаков

Отбор признаков - это сечение ненужных и оценка важности признаков при помощи алгоритмов машинного обучения.

Зачем отбирать признаки?

- Иногда плохие, зависимые признаки ведут к мультиколлинеарности.

- Признаков может быть много. Лучше отбросить ненужные признаки для сокращения времени обучения.
- Нам может быть необходимо выделить лучшие признаки и понять, насколько нам их будет хватать в будущем. То есть, упростить модель, тем самым повысив интерпретируемость.
- Признаки могут собираться из платных источников

>EDA

Разведочный анализ данных (EDA или Exploratory Data Analysis) – предварительное исследование данных с целью определения его основных характеристик, взаимосвязей между признаками, а также сужения набора методов, используемых для создания моделей.

Подразумевает собой использование различных графиков и статистических инструментов.

Обладает рядом преимуществ:

- Способ достаточно наглядный. С помощью визуализации можно легко понять что происходит внутри данных.
- Данный признак достаточно дешевый в плане вычислительных мощностей. Не нужно строить сложные модели, на нужна только **matplotlib** и данные.

И недостатков:

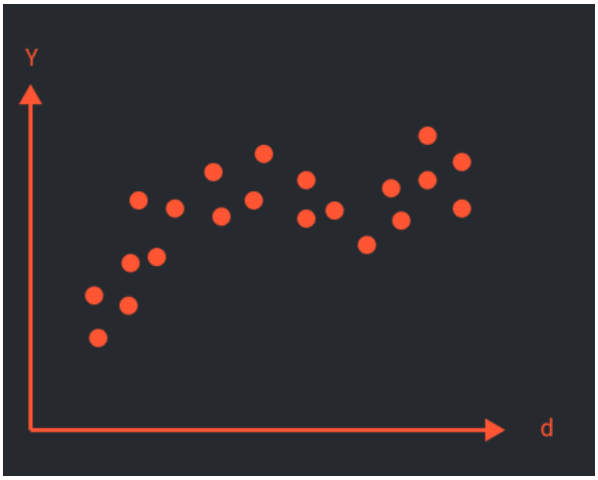
- Метод абсолютно наивный: при построении модели надежда только на сильную экспертизу и хорошую интуицию.
- Не получаем никаких строгих численных оценок о важности признаков. При использовании метода полагаемся только на визуализацию и умение ее правильно читать

Больше про визуализацию в matplotlib

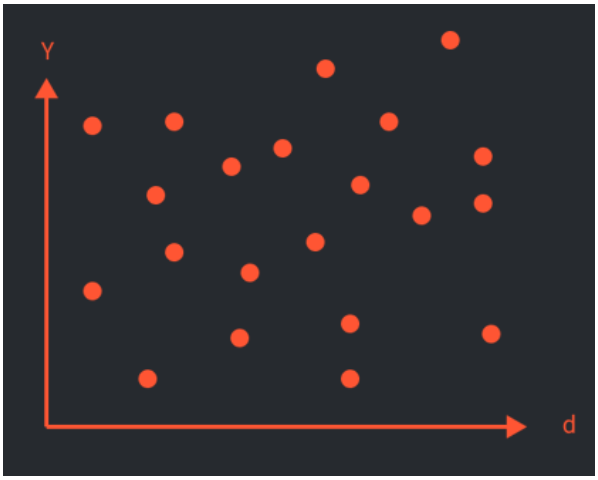
ScatterPlot

Точечная диаграмма или диаграмма рассеяния - это лучший способ визуализировать отношения между двумя переменными.

Выберем признак δ и изобразим все точки (d, y). Можно ли разглядеть зависимость?



1. Берём фичу



2. Не берём фичу

1. Построенная диаграмма рассеяния на 1 графике предполагает корреляцию между признаками. Признак δ как-то связан с таргетной переменной, значит его можно использовать для предсказаний.
2. Точки на 2 графике разбросаны достаточно хаотично. Кажется, сильной взаимосвязи между признаком δ и таргетной переменной совсем нет. Значит, этот признак можно спокойно убрать из выборки.

В python можно визуализировать с помощью метода `scatter` библиотеки `matplotlib`

Документация

BarChart

Столбчатая диаграмма - хорошо подходит для того, чтобы сравнить долю разных категорий

Выберем признак δ. Если он категориальный, то для каждой категории можно изобразить столбик, высота которого будет отображать среднее значения таргета в этой категории.

1. Берём фичу

2. Не берём фичу

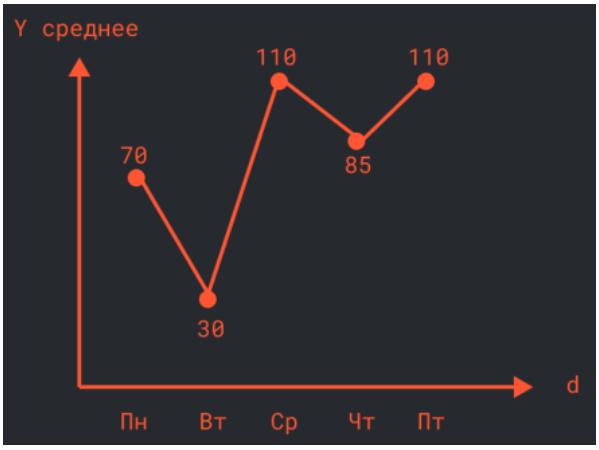
1. На 1 графике можно заметить, что переходя из одной категории в другую можно заметить различное распределение таргетов. Кажется, из построенной на 1 графике диаграммы можно сделать вывод, что категория нам важна.
2. На 2 рисунке столбики примерно одинаковы, т.е признаки совпадают. Значит, можно сказать, что эта информация в датасете избыточна и ее можно выбросить.

В python можно визуализировать с помощью метода `bar` библиотеки `matplotlib`

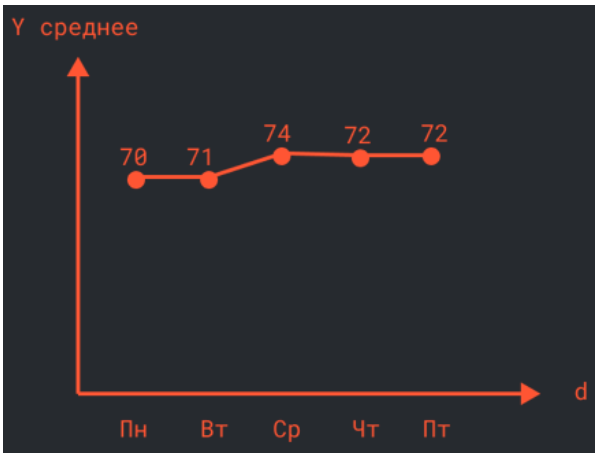
Документация

LineChart

Линейный график - диаграмма, которая отображает информацию в виде линии. Стандартный способ показать изменения данных во времени.



1. Берём фичу



2. Не берём фичу

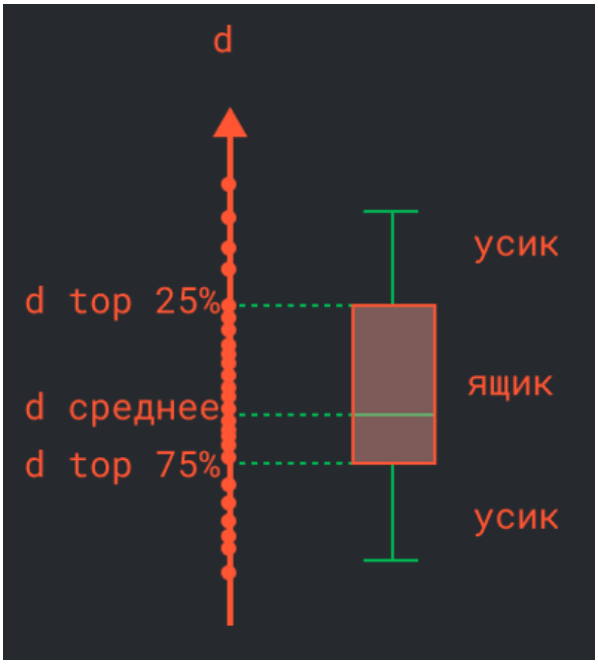
На первом графике, в отличии от второго, можно заметить, что значение таргетной переменной меняется в зависимости от дня недели.

В python можно визуализировать с помощью метода `lineplot` библиотеки `seaborn`

Документация

BoxPlot

Ящики с усами – компактное описание того, какие в среднем значения принимает переменная, с какими разбросом и частотой.

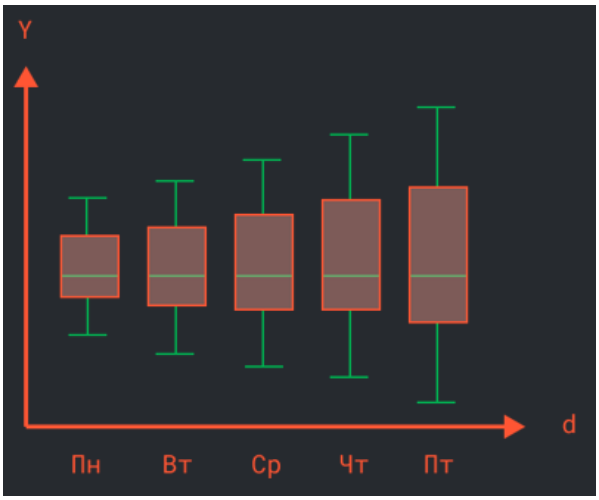


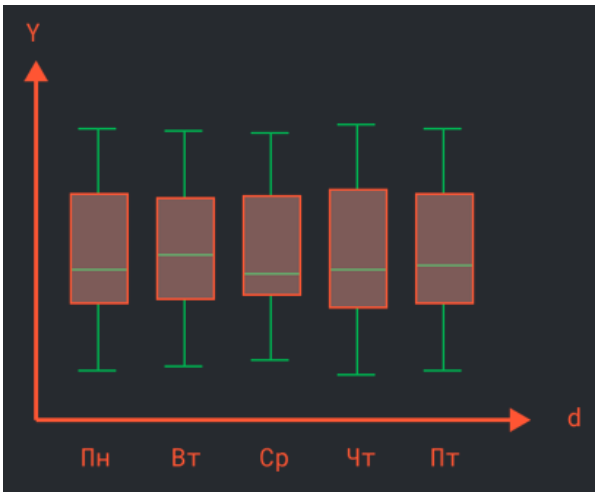
Как его нарисовать? Необходимо вычислить несколько элементов:

- Отсортируем все значения
- Замерим значение признака, который стоит в топ 25%
- Посчитаем среднее значение выборки
- Посчитает топ 75% обектов
- Нарисуем ящик, между 25% и 75%, отмерим среднее значение(50%)
- Нарисуем усики, равные 1.5 длинны ящика.

Ящики с усами хорошо подходит, чтобы сравнивать распределение таргетной переменной между категориями/группами данных.

У нас есть категориальная фича(день недели). Для каждой категории соберём множество таргетов и нарисуем ящики с усами для полученных наборов





На первом графике среднее значение между таргетами совпадает. И если бы мы рисовали лайн-чарт, то могли бы сделать вывод, что признак не важен и его нужно выкинуть. Бокс-плот же показывает более полную картину, не только среднее, но и разброс значений, кол-во выбросов и др. метрики. Ящики с усами от категории к категории разные, значит, можно сделать вывод, что категории как-то могут влиять на таргет.

На втором графике можно заметить, что средние значения отличаются, но сами ящики и их усики практически идентичны. Ящики с усами от категории к категории не сильно изменяются, значит, можно сделать вывод, что категории не очень влияют на распределение таргета.

В python можно визуализировать с помощью метода `boxplot` библиотеки `matplotlib`

Документация

Подробнее о бокс-плотах

>Встроенные алгоритмы

Встроенные методы — это методы, которые одновременно строят модель и фильтруют признаки. Например, для фильтрации части признаков можно использовать регуляризацию Lasso, которая склонна обнулять некоторые признаки у модели линейной регрессии. В регуляризации есть параметр λ, с помощью которого можно влиять на количество нужных признаков. Из минусов – регуляризатор может случайно убить хорошие признаки. Зато это встроенный метод, и в дополнительных исследованиях отсутствует необходимость, т.е мы ничего не рисуем, ничего не считаем просто используем модель Lasso регуляризации, получаем и модель и отбор признаков. Очень удобно.

Помимо Lasso регуляризации есть и другие встроенные методы. Например, отбор признаков методом случайного леса.

Встроенные методы обладают рядом преимуществ:

- Способ достаточно быстрый, т.к работает под капотом.
- По вычислительным мощностям не требуем доп. нагрузку. Используем только те, что зашиты в самой модели.

И недостатков:

- Ограничены заданной функциональной формой. То есть, например, для задачи линейной регрессии мы ограничены только линейной зависимостью между признаком и таргетом.
- В случае регуляризации – всегда есть шанс удалить признак, который давал бы значимую прибавку на Кросс-Валидации

В python можно построить Lasso регрессию с помощью метода `Lasso` из библиотеки `sklearn.linear_model`

Документация

>Метод обёртки

Основная идея метода заключается в поиске всевозможных признаков и оценки их качества путем прогона через модель

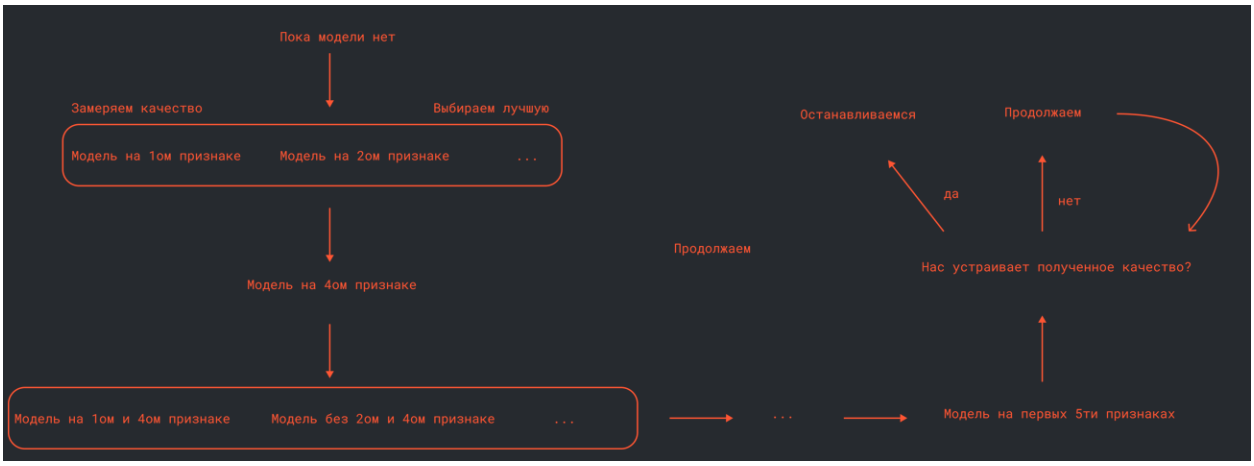
Если мы не знаем на каких признаках лучше обучать модель, а на каких нет, то давайте просто переберём все варианты и сравним модели между собой на кросс-валидации. Можно использовать любой алгоритм обучения

Представим, у нас есть набор признаков для квартиры $\left[m^2, ..., R_{\text{метро}}\right]$. Можем построить много моделей на всех подмножествах, на пример, линейную регрессию относительно площади квартиры

Но есть небольшой минус данного метода: таких комбинаций может быть гигантское количество. Например, если в множестве 20 признаков, то различных комбинаций моделей будет более миллиона. Задача построить миллион моделей, измерить их качество на кросс-валидации и выбрать из них лучшую кажется невыполнимой. Надо как-то модифицировать метод:

Метод прямого отбора

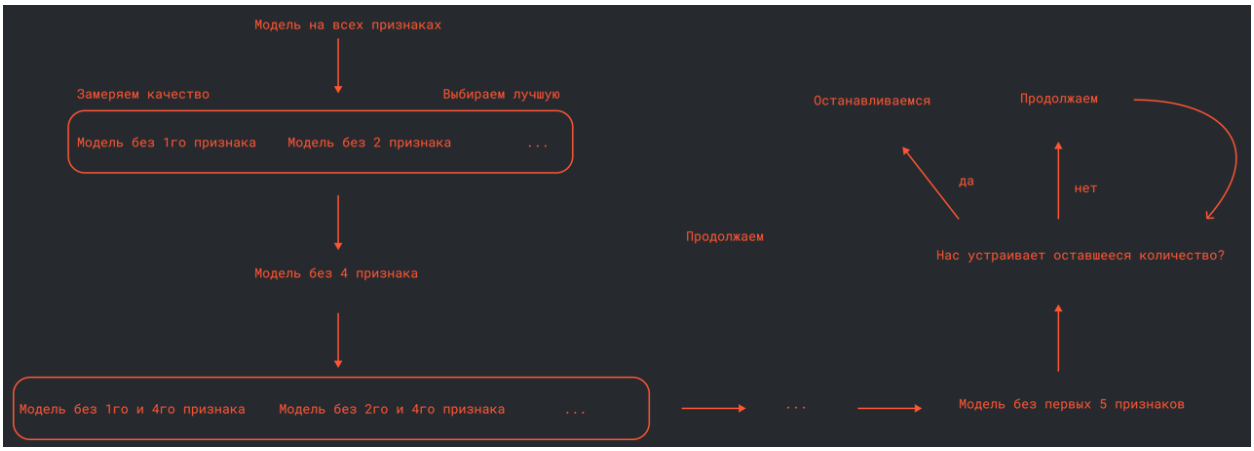
Представим, у нас есть набор признаков для квартиры $\left[m^2, ..., R_{\text{метро}}\right]$. Из списка переменных выберем ту, которая имеет лучшее качество. После чего замерим, при добавлении какого признака получили лучшую прибавку в качестве(тоже перебором). Продолжим добавлять признаки, прока не получим модель с удовлетворяющим нас качеством. То есть, на каком-то моменте у нас будет построена модель, например, на 5 признаках. Мы замерим её качество и сможем быть уверены, что больше никаких признаков включать не надо.



Метод обратного отбора.

Представим, у нас есть набор признаков для квартиры $\left[m^2, ..., R_{\text{метро}}\right]$. Обучим модель на всех имеющихся переменных. Далее поймем при выбрасывании какого одного признака модель не теряет в качестве. Теперь опять, замерим качество всех моделей, которые получаются из предыдущей выкидыванием каждого

признака. Посмотрим, где качество упадет слабее всего. И таким образом продолжим цикл до тех пор, пока не получим нужное количество признаков.



Методы обёртки обладают рядом преимуществ:

- Получаем более полную картину относительно различных подмножеств наших признаков
- Не требуем обработки данных (например, масштабирование)

И недостатков:

- Долго и очень дорого
- А жадные алгоритмы всегда оставляют чувство недосказанности(”вдруг, стоило выкинуть другой признак...”)
- Склонны к переобучению

>Метод фильтрации

Основная идея заключается в том, чтобы внимательно посмотреть на данные и посчитать какую-то характеристику (статистические метрики, например).

Как и в EDA, в этом методе мы не строим модели, здесь мы делаем акцент на инструменты статистики и оцениваем, например:

- Как вещественные показатели связаны с таргетом.
- Как вещественные показатели связаны друг с другом.

- Какое влияние категории имеют на тергетную переменную.

Вещественные признаки: корреляция

Коэффициент корреляции в своём статистическом смысле обозначает силу и характер взаимосвязи между двумя переменными. Взаимосвязь может быть положительной (когда одна переменная растёт, другая тоже растёт), отрицательной (когда одна переменная растёт, другая уменьшается), либо совсем отсутствовать. В учебниках формулу часто можно встретить в таком виде:

$$corr_{xy} = \frac{\Sigma(x_i - \bar{x}) \times (y_i - \bar{y})}{\sqrt{\Sigma(x_i - \bar{x})^2 \times \Sigma(y_i - \bar{y})^2}}$$

X и Y это количественных величины. $\bar{X}$ и $\bar{Y}$ среднее по выборке для каждой из величин

Это называется коэффициентом корреляции Пирсона.

График корреляции называется диаграммой рассеяния, scatterplot(уже сталкивались в EDA). В Python его можно нарисовать следующим образом:

```
sns.set(style='whitegrid', rc={'figure.figsize' : (10,5)})

sns.scatterplot(x = 'x_column', y = 'y_column', data = df)
plt.title('Корреляция переменных X и Y')
plt.xlabel('Переменная X')
plt.ylabel('Переменная Y')
```

Вещественные признаки: дисперсия

Дисперсия (variance) – средний квадрат отклонений индивидуальных значений признака от их средней величины. То есть, дисперсия оценивает разброс переменной, показывает как в среднем ее значения отклоняются от среднего. Формула дисперсии:

$D = \frac{\sum(x_i - \overline{X})^2}{n}$

X количественная величина. $\bar{X}$ среднее значение признака. n размер выборки

Как эта характеристика может помочь при отборе признаков? Если дисперсия мала(допустим, равна нулю), то значения признака практически не меняются.

Такие признаки от объекта к объекту не меняются, поэтому их стоит убрать из выборки.

Посчитать в python:

```
import pandas as pd
df.A.var()

import numpy as np
np.var(df.A)
```

Методы фильтрации обладают рядом преимуществ:

- Дешевые с точки зрения времени и вычислительных мощностей. Мы не строим модели, а просто смотрим на данные и считаем какие-то характеристики.

И недостатков:

- Требуют работы с данными.
- Зачастую используемые “критерии” не улавливают сложные взаимосвязи между данными, а ограничиваются только линейными