КУРС «Технологии интеллектуального анализа данных»

***

Тема «Регрессионные модели»

Лекция

bkiselev@yandex.ru

Регрессионные модели в Python

Простая (одномерная) линейная регрессия (модель с одним предиктором) аппроксимируется всем известной со школы функцией прямой линии y=kx+b с той лишь разницей, что теперь нужно добавить случайную ошибку ϵ:

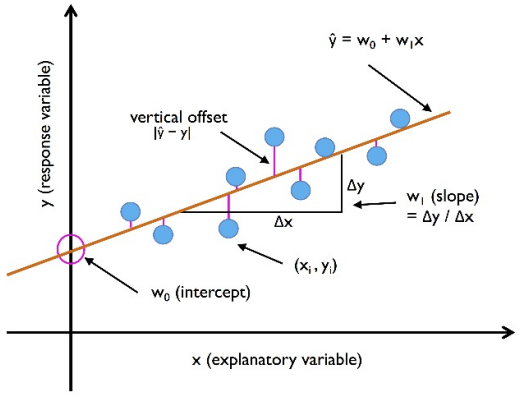
$$y_i=\beta_0+\beta_1 x_i+\epsilon_i$$

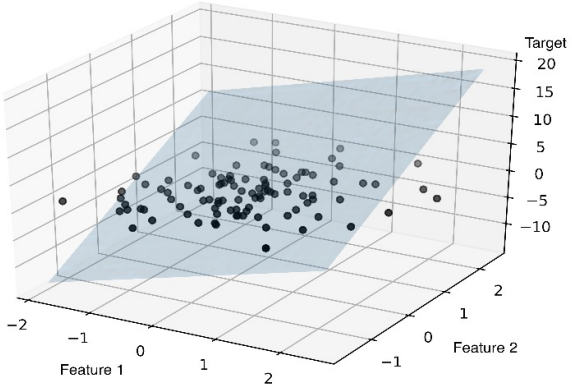
y_i — зависимая переменная (отклик)

x_i — известная константа (значение объясняющей переменной, измеренной в i-ом эксперименте)

β_0, β_1 — параметры модели (свободный член и угловой коэффициент).

ϵ_i — случайная ошибка





Линейная модель очень похожа на модель SVM. Однако в SVM гиперплоскость играет роль границы принятия решения: она используется для отделения двух групп данных друг от друга. Как следствие, она должна проходить **как можно дальше от каждой группы**.

В линейной регрессии, напротив, гиперплоскость проводится так, чтобы оказаться как можно ближе ко всем обучающим образцам.

Обучение линейной регрессии
Метод наименьших квадратов

Чаще всего линейная регрессия обучается с использованием среднеквадратичной ошибки. В этом случае получаем задачу оптимизации **функции стоимости** (эмпирического риска) (считаем, что среди признаков есть константный, и поэтому свободный коэффициент не нужен):

$$\mathcal{L}\left(X, \vec{y}, \vec{w}\right) = \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - \vec{x}_i^T\vec{w}_i\right)^2 = \frac{1}{2n}\left|\vec{y} - X\vec{w}\right|_2^2 = \frac{1}{2n}\left(\vec{y} - X\vec{w}\right)^T\left(\vec{y} - X\vec{w}\right)$$

Если продифференцировать данный функционал по вектору $\{w_i\}$, приравнять к нулю и решить уравнение, то получим явную формулу для решения:

$$\vec{w} = \left(X^TX\right)^{-1}X^T\vec{y}$$

Оценка МНК является лучшей оценкой параметров модели, среди всех линейных и несмещенных оценок, то есть обладающей наименьшей дисперсией (из теоремы Гаусса — Маркова).

Матричные производные

$$\frac{\partial}{\partial x}x^Ta = a$$

$$\frac{\partial}{\partial x}x^TAx = \left(A + A^T\right)x$$

$$\frac{\partial}{\partial A}x^TAy = xy^T$$

$$\frac{\partial}{\partial x}A^{-1} = -A^{-1}\frac{\partial A}{\partial x}A^{-1}$$

Вычислим производную функции стоимости:

$$\frac{\partial\mathcal{L}}{\partial\vec{w}} = \frac{\partial}{\partial\vec{w}}\frac{1}{2n}\left(\vec{y}^T\vec{y} - 2\vec{y}^TX\vec{w} + \vec{w}^TX^TX\vec{w}\right) = \frac{1}{2n}\left(-2X^T\vec{y} + 2X^TX\vec{w}\right)$$

Приравняем к нулю и найдем решение в явном виде:

$$\frac{\partial\mathcal{L}}{\partial\vec{w}} = 0 \quad \Leftrightarrow \quad \begin{aligned} &\frac{1}{2n}\left(-2X^T\vec{y} + 2X^TX\vec{w}\right) = 0 \\ &-X^T\vec{y} + X^TX\vec{w} = 0 \\ &X^TX\vec{w} = X^T\vec{y} \\ &\boxed{\vec{w} = \left(X^TX\right)^{-1}X^T\vec{y}} \end{aligned}$$

Ограничения линейной регрессии

Линейность: зависимая переменная может линейно аппроксимировать независимые переменные

Нормальность распределения Y и ε

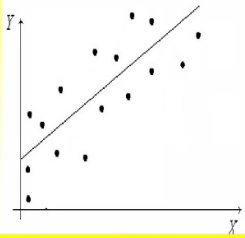
Отсутствие избытка влиятельных наблюдений

Гомоскедастичность распределения остатков

Отсутсвие мультиколлинеарности

Гомоскедастичность означает "одинаковый разброс".

Типичный вид облака точек в модели с гомоскедастичным и остатками



Остатки регрессии - это разности между наблюдаемыми значениями и значениями, предсказанными изучаемой регрессионной моделью.

Мультиколлинеарность — корреляция независимых переменных, которая затрудняет оценку и анализ общего результата. Когда независимые переменные коррелируют друг с другом, говорят о возникновении мультиколлинеарности.

$$Y = a + b_1 x_1 + b_2 x_2 + b_3 x_3$$

где Y - общая величина расходов на питание;
x_1 - заработная плата;
x_2 - доход, получаемый вне работы;
x_3 - совокупный доход.

Housing Dataset

Название этого набора данных - просто **бостон** . Он имеет две протозадачи: **nox** , в которой необходимо спрогнозировать уровень закиси азота; и **цена** , по которой следует прогнозировать среднюю стоимость дома

Происхождение: Данные о жилищном строительстве в Бостоне произошли от **природы** .

Применение: Этот набор данных можно использовать для **оценки**

Количество случаев: Набор данных содержит всего **506** случаев.

Последовательность: Порядок случаев **случайный** .

Замечание: Переменная № 14, похоже, подверглась цензуре на уровне 50,00 (что соответствует средней цене 50 000 долларов); На цензуру указывает тот факт, что самая высокая средняя цена, составляющая ровно 50 000 долларов, указана в 16 случаях, в то время как в 15 случаях цены составляют от 40 000 до 50 000 долларов с округлением до ближайшей сотни. Харрисон и Рубинфельд не упоминают никакой цензуры.

Переменные

В каждом случае набора данных есть **14** атрибутов. Они есть:

1. CRIM - уровень преступности на душу населения по городам
2. ZN - доля земли под жилую застройку, зонированная под участки более 25 000 кв. Футов.
3. INDUS - доля акров, не относящихся к розничной торговле, на город.
4. CHAS - фиктивная переменная реки Чарльз (1, если участок ограничивает реку; 0 в противном случае)
5. NOX - концентрация оксидов азота (частей на 10 миллионов)
6. RM - среднее количество комнат в доме
7. AGE - доля занимаемых владельцами единиц, построенных до 1940 г.
8. DIS - взвешенные расстояния до пяти бостонских центров занятости
9. RAD - индекс доступности радиальных автомобильных дорог
10. TAX - полная ставка налога на имущество за 10 000 долларов США.
11. PTRATIO - соотношение учеников и учителей по городам
12. B - 1000 (Bk - 0,63) ^ 2, где Bk - доля черных по городам
13. LSTAT -% более низкого статуса населения
14. MEDV - Средняя стоимость домов, занимаемых владельцами, в 1000 долларов

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
#%matplotlib inline
#%config InlineBackend.figure_format = 'svg'

from sklearn.datasets import load_boston
boston = load_boston()
print(boston.DESCR)
boston_df = pd.DataFrame(boston.data, columns=boston.feature_names)
boston_df['MEDV'] = boston.target
print(boston_df.head())
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

Предварительная обработка данных

Мы подсчитываем количество пропущенных значений для каждого параметра, используя isnull()

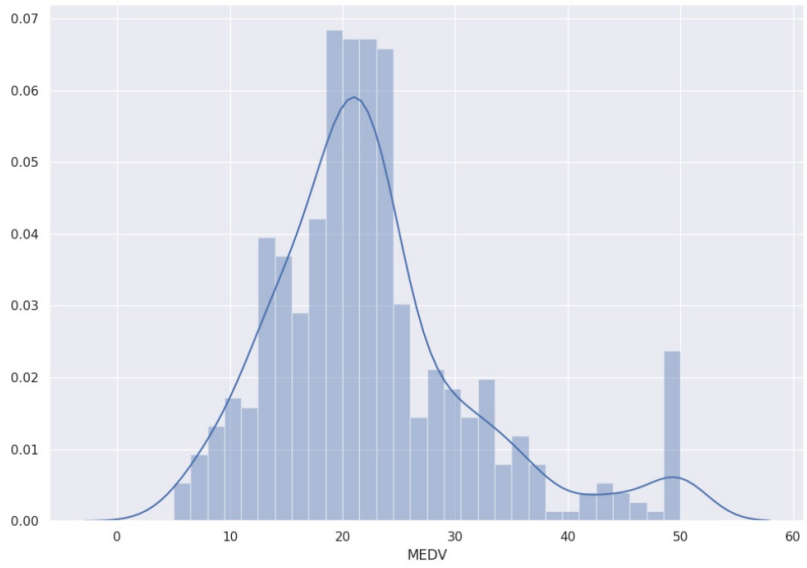
```
print(boston_df.isnull().sum())
```

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

в этом наборе данных нет пропущенных значений

построим график распределения целевой переменной MEDV

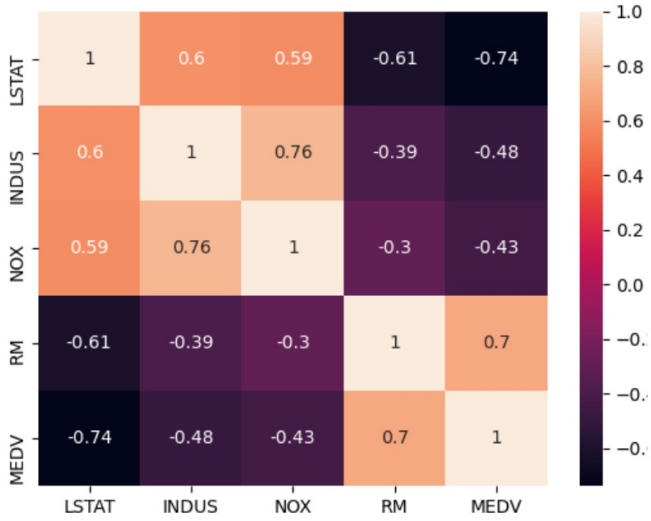
```
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.distplot(boston_df['MEDV'], bins=30)
plt.show()
```



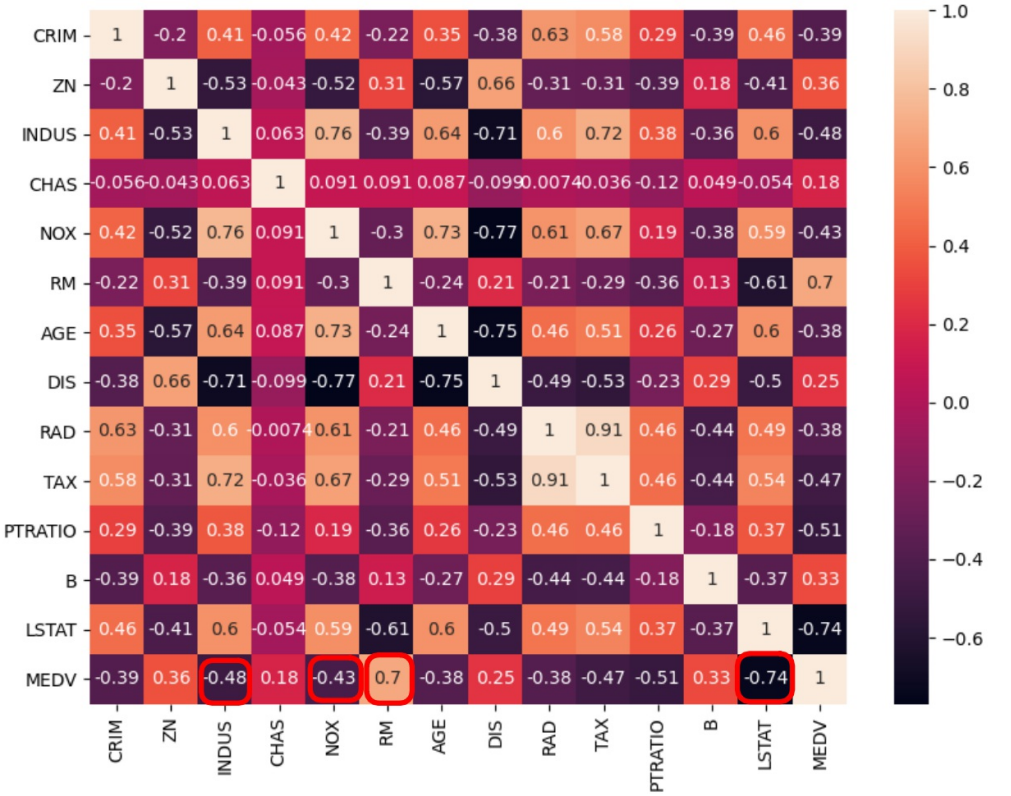
Мы видим, что значения MEDV распределены нормально с небольшими выбросами.

Как всегда начинаем изучение нового набора данных с разглядывания графиков. Визуализируем матрицу корреляций и ввиде тепловой карты:

```
cols = ['LSTAT', 'INDUS', 'NOX', 'RM', 'MEDV']
hm = sns.heatmap(boston_df[cols].corr(),
                 cbar=True,
                 annot=True)
plt.show()
```




```
cols = ['CRIM','ZN','INDUS','CHAS','NOX','RM','AGE','DIS','RAD','TAX','PTRATIO','B','LSTAT','MEDV']
hm = sns.heatmap(boston_df[cols].corr(),
                 cbar=True,
                 annot=True)
plt.show()
```



Корреляция — степень связи между 2-мя или несколькими независимыми явлениями.

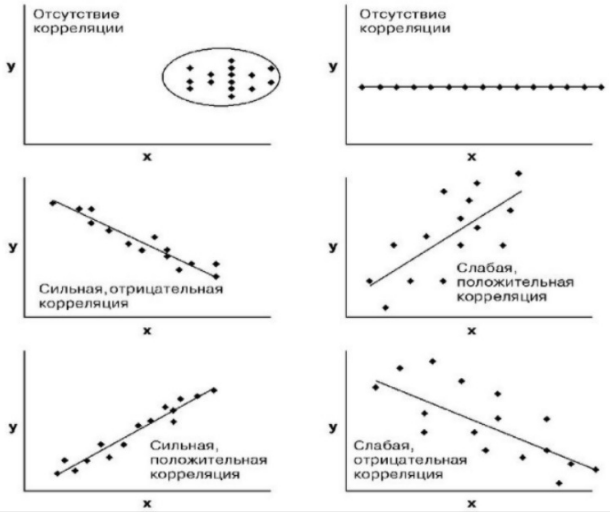
Положительная корреляция (прямая) возникает при одновременном изменении 2-х переменных величин в одинаковых направлениях (в положительном или отрицательном). Например, взаимосвязь между количеством пользователей, приходящих на сайт из поисковой выдачи и нагрузкой на сервер: чем больше пользователей, тем больше нагрузка.

Корреляция отрицательна (обратная), если изменение одной величины приводит противоположному изменению другой. Например, с увеличением налоговой нагрузки на компании уменьшается их прибыль. Чем больше налогов, тем меньше денег на развитие.

При значении КК равном 1, следует понимать, что при каждом изменении 1-й переменной происходит эквивалентное изменение 2-й переменной в том же направлении.

Если значение КК равно -1, то при каждом изменении происходит эквивалентное изменение второй переменной в противоположном направлении.

Чем ближе корреляция к -1 или 1, тем сильнее связь между переменными. При нулевом значении (или близким к 0) значимая связь между 2-мя переменными отсутствует или очень минимальна.



Коэффициент корреляции колеблется от -1 до 1. Если значение близко к 1, это означает, что существует сильная положительная корреляция между двумя переменными. Когда он близок к -1, переменные имеют сильную отрицательную корреляцию.

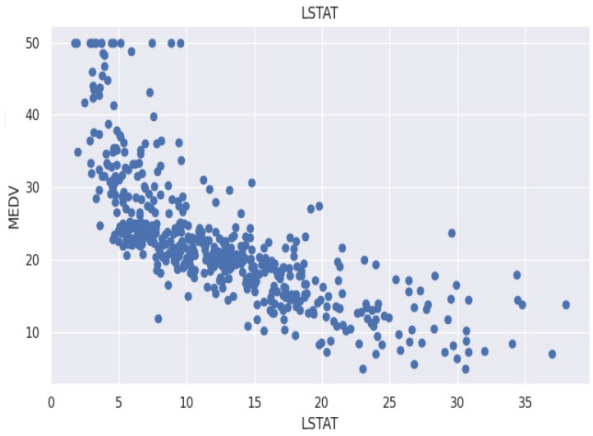
Чтобы соответствовать модели линейной регрессии, мы выбираем те показатели, которые имеют высокую корреляцию с нашей целевой переменной MEDV. Посмотрев на матрицу корреляции, мы можем увидеть, что RM имеет сильную положительную корреляцию с MEDV(0,7), а LSTAT имеет высокую отрицательную корреляцию с MEDV(-0,74).

Важным моментом при выборе параметров для модели линейной регрессии является проверка **мультиколинейности**. RAD, TAX имеет соотношение 0,91. Эти пары признаков сильно коррелированы друг с другом. Мы не должны выбирать обе эти функции вместе для обучения модели. То же самое касается особенностей DIS и AGE, которые имеют корреляцию -0.75.

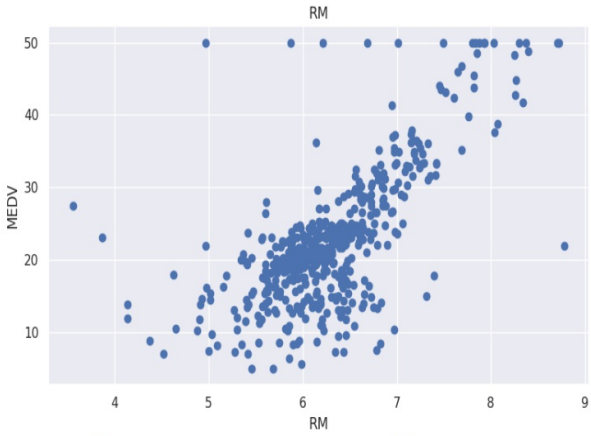
Исходя из вышеперечисленных наблюдений выделим RM и LSTAT как параметры. Используя диаграмму рассеяния, давайте посмотрим, как эти параметры меняются с MEDV.

```
plt.figure(figsize=(20, 5))
features = ['LSTAT', 'RM']
target = boston_df['MEDV']

for i, col in enumerate(features):
    plt.subplot(1, len(features) , i+1)
    x = boston_df[col]
    y = target
    plt.scatter(x, y, marker='o')
    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('MEDV')
plt.show()
```



Цены имеют тенденцию к снижению с увеличением LSTAT. Хотя это не похоже на то, чтобы следовать точно по линейной линии.



Цены растут по мере линейного увеличения стоимости RM. Есть несколько выбросов, и данные, похоже, ограничены 50.

Подготовка данных для обучения модели

```
# Объединим LSTAT и RM столбцы
X = pd.DataFrame(np.c_[boston_df['LSTAT'], boston_df['RM']], columns = ['LSTAT','RM'])
Y = boston_df['MEDV']
# Разделение данных на обучающие и тестовые наборы
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=5)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(404, 2)
(102, 2)
(404,)
(102,)
```

Мы разбиваем данные на обучающие и тестовые наборы. Мы обучаем модель на 80% образцов и тестируем на оставшихся 20%. *Мы делаем это, чтобы оценить производительность модели на невидимых данных* . Для разделения данных мы используем train_test_split функцию, предоставляемую библиотекой scikit-learn. Наконец, мы печатаем размеры нашего обучающего и тестового набора, чтобы проверить, правильно ли произошло разделение.

Обучение и тестирование модели

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
lin_model = LinearRegression()
lin_model.fit(X_train, Y_train)

# model evaluation for training set
from sklearn.metrics import mean_absolute_error,
mean_squared_error, median_absolute_error, r2_score

y_train_predict = lin_model.predict(X_train)
rmse = (np.sqrt(mean_squared_error(Y_train,
y_train_predict)))
r2 = r2_score(Y_train, y_train_predict)

print("The model performance for training set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
print("\n")

# model evaluation for testing set
y_test_predict = lin_model.predict(X_test)
rmse = (np.sqrt(mean_squared_error(Y_test,
y_test_predict)))
r2 = r2_score(Y_test, y_test_predict)

print("The model performance for testing set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
```

```
The model performance for training set
--------------------------------------
RMSE is 5.6371293350711955
R2 score is 0.6300745149331701

The model performance for testing set
--------------------------------------
RMSE is 5.13740078470291
R2 score is 0.6628996975186954
```

Проверка качества модели

metrics.explained_variance_score(y_true,…) Объясненная функция оценки регрессии дисперсии
metrics.max_error(y_true, y_pred) Метрика max_error вычисляет максимальную остаточную ошибку.
metrics.mean_absolute_error(y_true, y_pred, *) Потеря регрессии средней абсолютной ошибки
metrics.mean_squared_error(y_true, y_pred, *) **Среднеквадратичная ошибка регрессии потерь**
metrics.mean_squared_log_error(y_true, y_pred, *) Среднеквадратичная ошибка регрессии логарифмической ошибки
metrics.median_absolute_error(y_true, y_pred, *) Медианная потеря регрессии абсолютной ошибки
metrics.r2_score(y_true, y_pred, * [,…]) **R ^ 2 (коэффициент детерминации) функция оценки регрессии.**
metrics.mean_poisson_deviance(y_true, y_pred, *) Потеря регрессии среднего отклонения Пуассона.
metrics.mean_gamma_deviance(y_true, y_pred, *) Потеря регрессии среднего гамма-отклонения.
metrics.mean_tweedie_deviance(y_true, y_pred, *) Средняя потеря регрессии отклонения Твиди.

https://scikit-learn.org/stable/modules/classes.html#regression-metrics

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=0)
slr = LinearRegression()

slr.fit(X_train, y_train)
y_train_pred = slr.predict(X_train)
y_test_pred = slr.predict(X_test)

from sklearn.metrics import mean_absolute_error,
mean_squared_error, median_absolute_error, r2_score

print('MSE train: {:.3f}, test: {:.3f}'.format(
        mean_squared_error(y_train, y_train_pred),
        mean_squared_error(y_test, y_test_pred)))
print('R^2 train: {:.3f}, test: {:.3f}'.format(
        r2_score(y_train, y_train_pred),
        r2_score(y_test, y_test_pred)))
```

```
MSE train: 37.934, test: 39.817
R^2 train: 0.552, test: 0.522
```

Поскольку в нашей модели несколько независимых переменных, мы не можем отобразить их зависимость на двумерном пространстве, но можем нанести на график связь между остатками модели и предсказанными значениями, что также поможет нам диагностировать качество модели. Это называется **Residuals plot**. С его помощью мы можем увидеть нелинейность и выбросы, проверить случайность распределения ошибки.

```
plt.scatter(y_train_pred,  y_train_pred - y_train,
            c='blue', marker='o', label='Training data')
plt.scatter(y_test_pred,  y_test_pred - y_test,
            c='lightgreen', marker='s', label='Test data')
plt.xlabel('Predicted values')
plt.ylabel('Residuals')
plt.legend(loc='upper left')
plt.hlines(y=0, xmin=-10, xmax=50, lw=2, color='red')
plt.xlim([-10, 50])
plt.tight_layout()
plt.show()
```

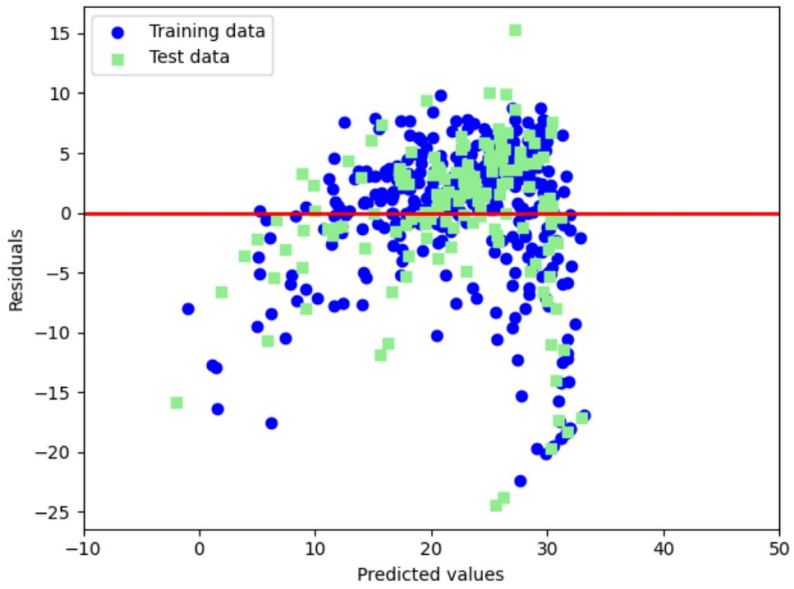
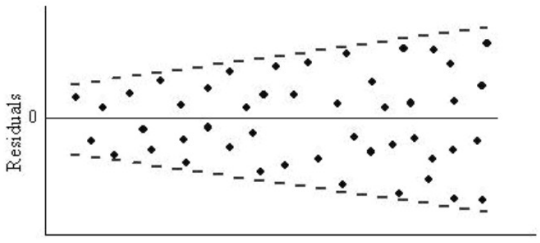


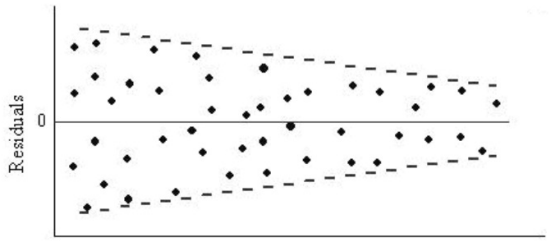
График распределения остатков, могут быть как положительны так и отрицательны. На нем мы должны наблюдать случайно распределение остатков с нулевым средним значением.

Структуры шаблонов графиков остатков не только помогают проверить достоверность регрессионной модели, но также могут дать подсказки о том, как ее улучшить. Например, криволинейный узор на графике **остаточных и независимых** значений предполагает, что в подгоночную модель следует ввести член более высокого порядка.

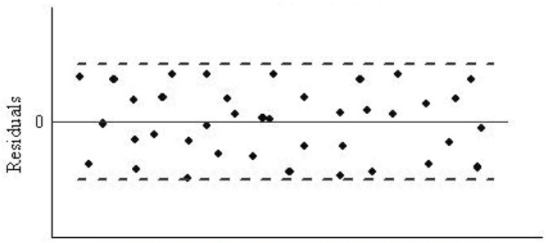
График остатков, имеющий тенденцию к увеличению, предполагает, что дисперсия ошибок увеличивается с увеличением независимой переменной; в то время как распределение, которое показывает тенденцию к уменьшению, указывает, что дисперсия ошибки уменьшается с независимой переменной. Ни одно из этих распределений не является образцом постоянной дисперсии. Следовательно, они указывают на то, что предположение о постоянной дисперсии вряд ли будет верным, а регрессия не является хорошей. С другой стороны, шаблон с горизонтальной полосой предполагает, что дисперсия остатков постоянна.



тенденция к увеличению



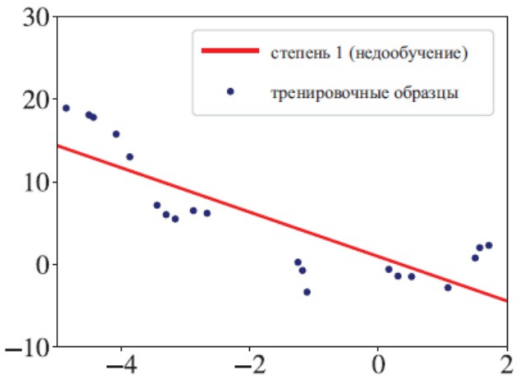
тенденция к уменьшению



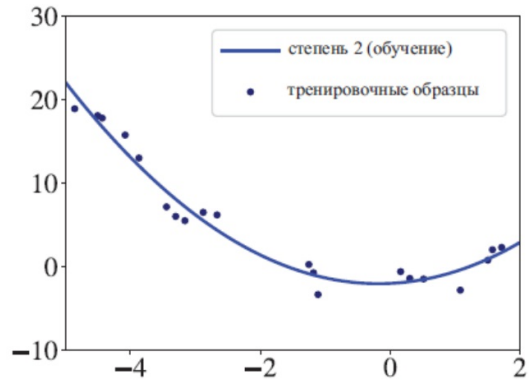
дисперсия остатков постоянна

Примеры недообучения (линейная модель), хорошего обучения (квадратичная модель) и переобучения (полином степени 15)

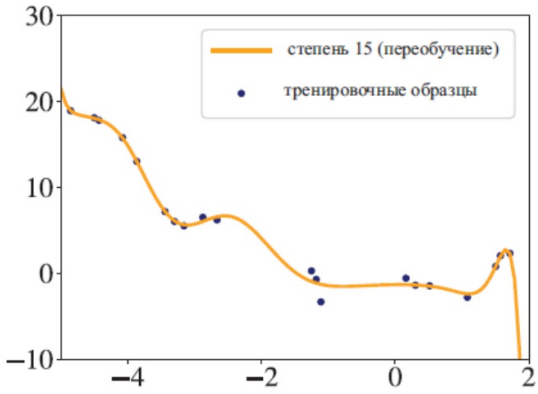
Начнем с простой истины: машины не учатся. Типичное машинное обучение заключается в поиске математической формулы, которая при применении к набору входных данных (называемых обучающими данными) даст желаемые результаты.



Недообучение



Хорошее обучение



Переобучение

Вот некоторые методы борьбы с переобучением:

1.
Попробовать более простую модель (линейную регрессию вместо полиномиальной, метод опорных векторов (SVM) с линейным ядром вместо радиальных базисных функций (RBF), нейронную сеть с меньшим числом слоев/узлов).

2.
Уменьшить размерность данных в наборе данных (например, с помощью одного из методов уменьшения размерности, рассматриваемых в главе 9).

3.
Добавить больше обучающих данных, если возможно.

4.
Применить регуляризацию к модели.

Регуляризация — один из самых широко используемых приемов в борьбе с переобучением.

Регуляризация — это собирательный термин, охватывающий методы, позволяющие алгоритмам обучения строить менее сложные модели. На практике это часто приводит к небольшому увеличению смещения, но значительно уменьшает дисперсию. Эта проблема известна в литературе как дилемма смещения-дисперсии.

На практике наиболее широко используются L1- и L2-регуляризация. Идея довольно проста. Чтобы создать регуляризованную модель, нужно модифицировать целевую функцию, добавив штрафной член, значение которого увеличивается с ростом сложности модели.