Лекция 7:
Линейные модели регрессии, МНК и отбор признаков

Параметрические линейные модели

- Линейная модель представляет собой важный пример параметрической модели для регрессии:

$$a(x_1, \ldots, x_p) = E(Y | X_1 = x_1, \ldots, X_p = x_p)$$

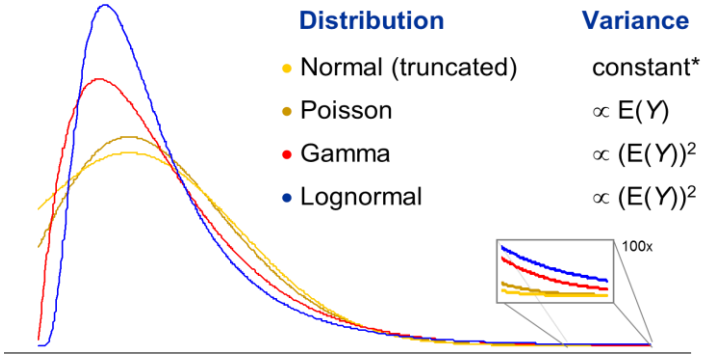
- Она определяется:
  - Линейным уравнением $a(x) = w_0 + \sum_{j=1}^{p} x_j w_j + \varepsilon$
  - *p + 1* параметром w_i, которые нужно найти по обучающей выборке
  - Минимизируя квадратичную функцию потерь

$$Q(w, Z) = \frac{1}{l} \sum_{i=1}^{l} (y_i - a(\bar{x}_i, w))^2 = \frac{1}{l} \sum_{i=1}^{l} \left(y_i - w_0 - \sum_{j=1}^{p} x_{ij} w_j \right)^2$$

- Должны быть выполнены предположения:
  - Шум распределен по закону $\varepsilon \sim N(0, \sigma^2)$
  - Наблюдения в выборке независимы и одинаково распределены (i.i.d)
  - Мы «угадали» с линейным уравнением и признаковым пространством

Другие функции потерь в регрессиях

- Используются и другие функции потерь
  - На основе экспоненциального распределения в обобщенных линейных моделях (будут позже в курсе):



Normal: $Q(\mathbf{w}) = \frac{1}{l}\sum (y - a(\mathbf{w}))^2$

Poisson: $Q(\mathbf{w}) = \frac{2}{l}\sum [y \ln(y/a(\mathbf{w})) - (y - a(\mathbf{w}))]$

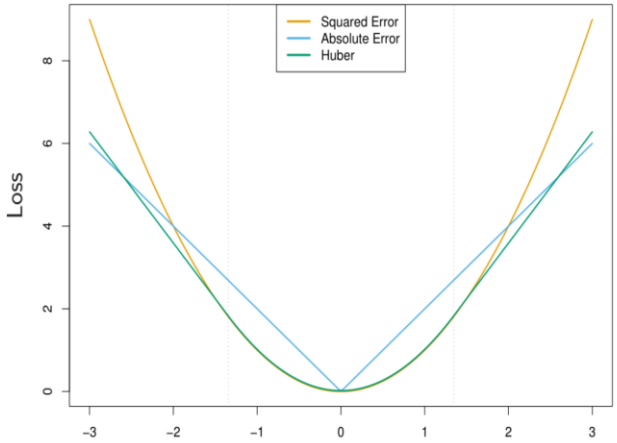
Gamma: $Q(\mathbf{w}) = \frac{2}{l}\sum [-\ln(y/a(\mathbf{w})) + (y - a(\mathbf{w}))/a(\mathbf{w})]$

  - Робастные (устойчивые к выбросам):

$$a^*(x) = \arg\min_{a(x)} E(|a(x) - y||x) \Rightarrow$$

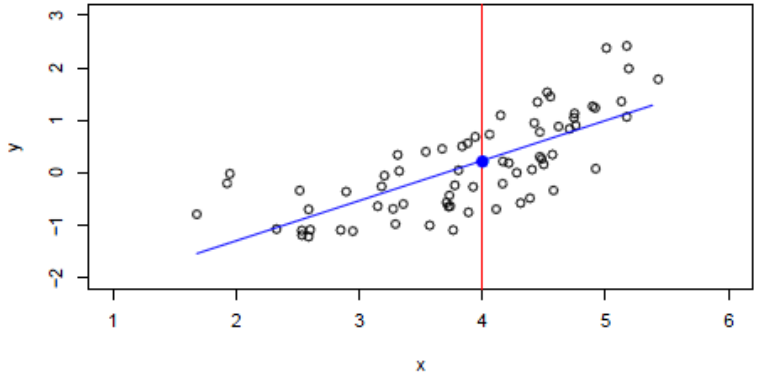
$$\Rightarrow a^*(x) = median(y|x)$$

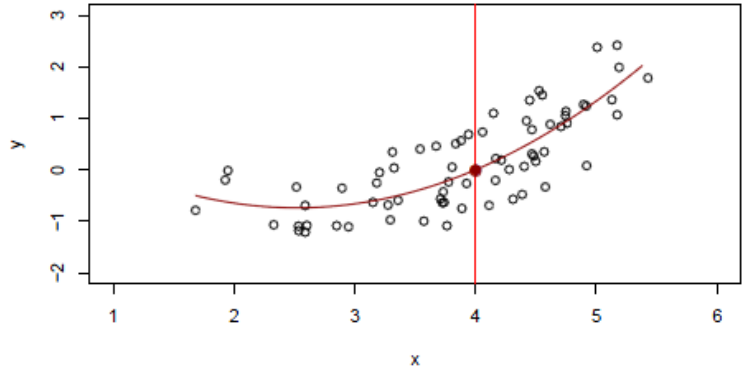
  - Но мы пока сосредоточимся на **квадратичной функции потерь**



Параметрические линейные модели

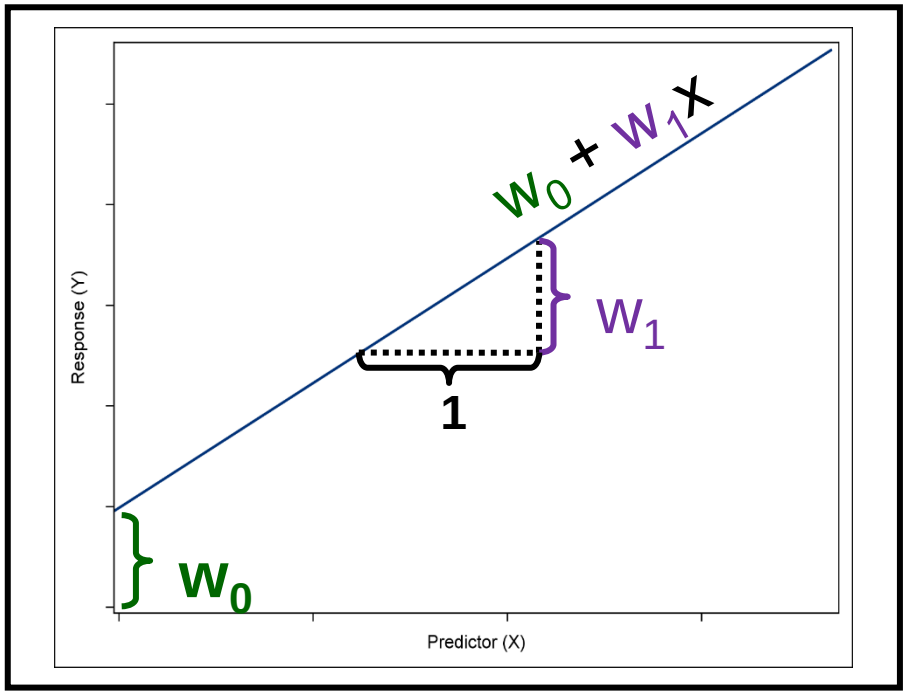
- Линейные модели *почти никогда* не показывают высокую точность, но служат хорошей и интерпретируемой аппроксимацией неизвестной истинной зависимости.

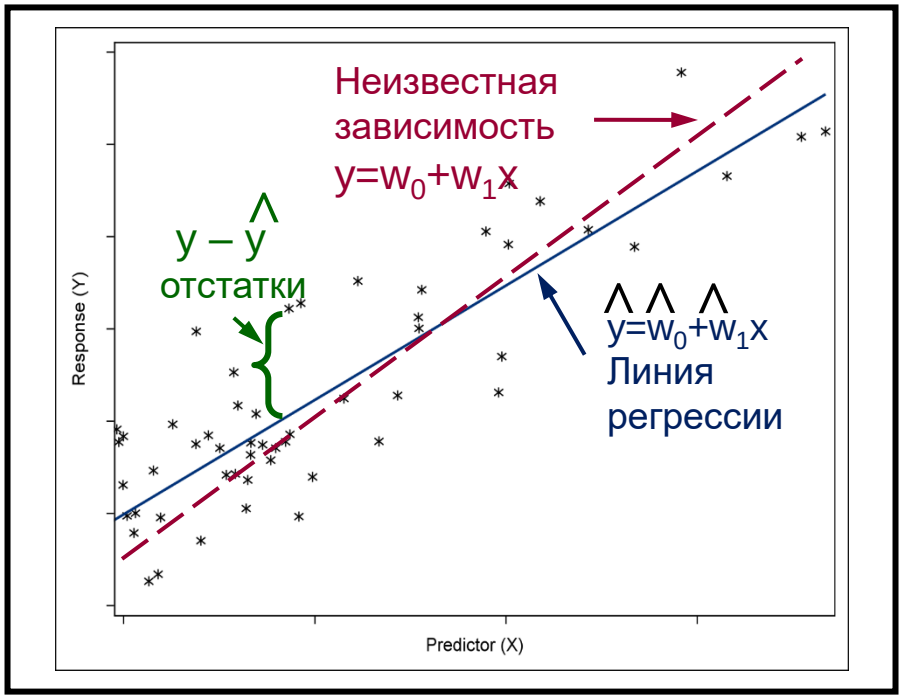




- Цель регрессионного анализа:
  - Определение наличия связи между переменными и характера этой связи (подбор уравнения)
  - Предсказание значения зависимой переменной с помощью независимойых
  - Определение вклада отдельных независимых переменных в вариацию зависимой

Одномерная линейная регрессия

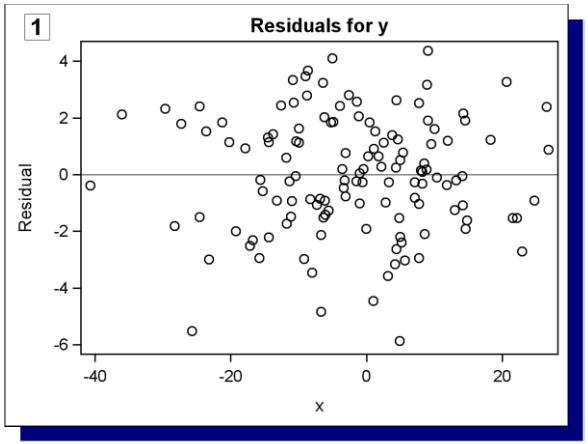




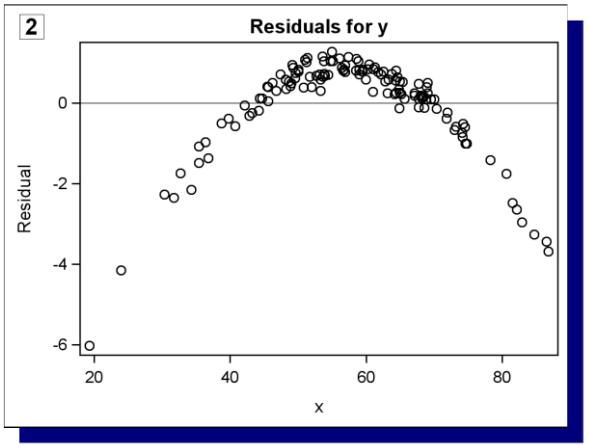
Предположения:

- Независимость наблюдений
- Выбранное уравнение регрессии (например, линейное) соответсвует истинной зависимости в данных
- Нормальность ошибки (с константной дисперсией по всем наблюдениям)

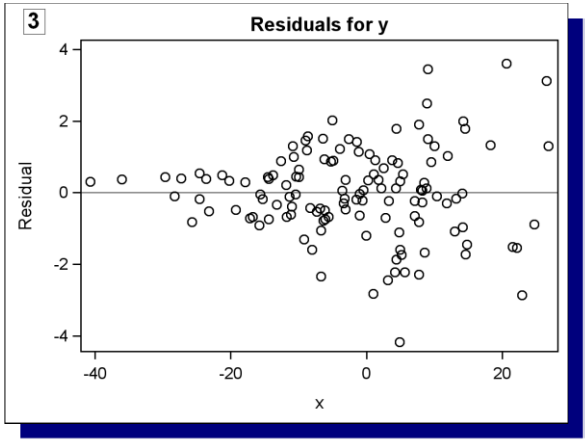
Графики остатков



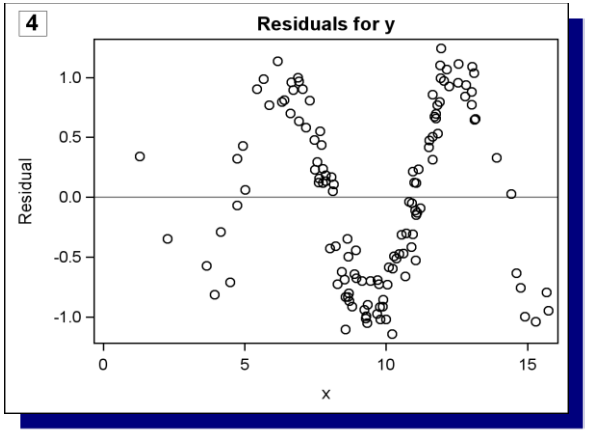
норма



Нелинейная зависимость



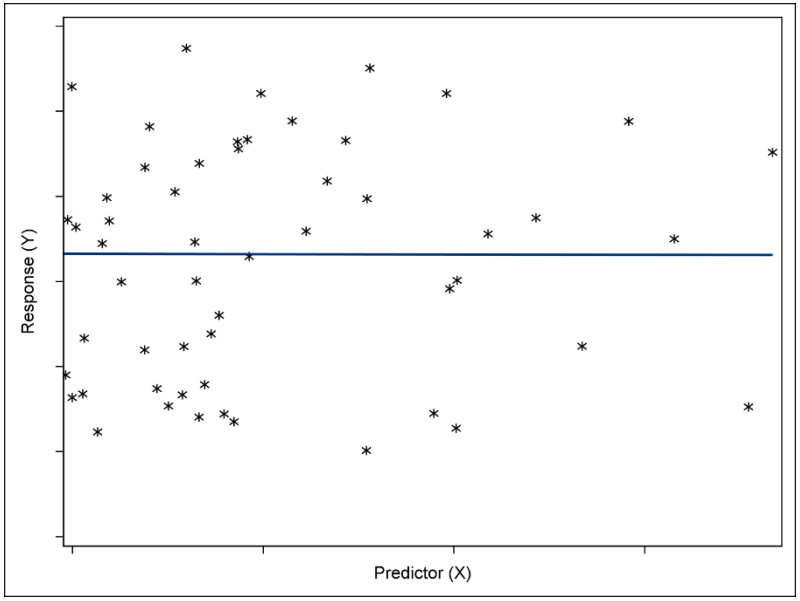
Гетероскедастичность

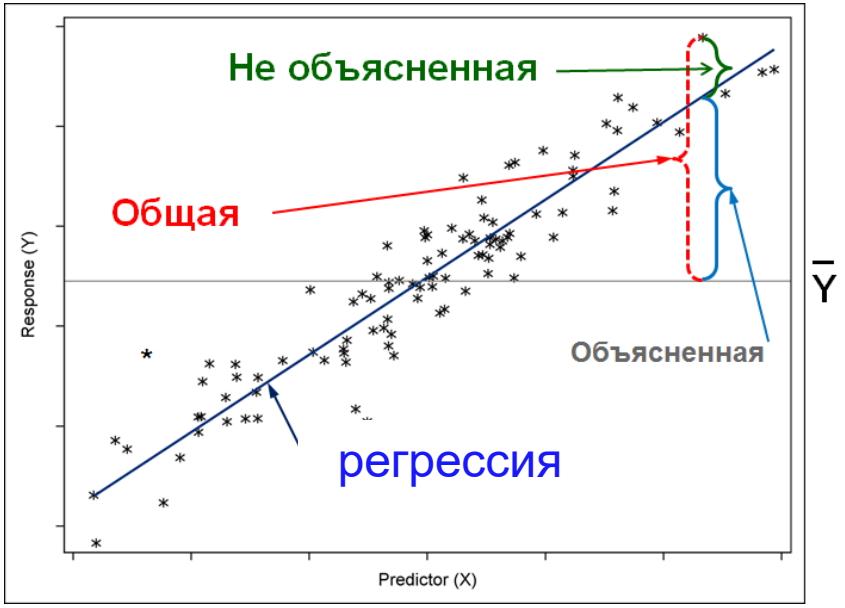


Зависимость наблюдений

Для проверки предположений регрессионной модели полезно построить графики зависимости остатков от прогноза

Объясненная и необъясненная дисперсия





- Необъясненная дисперсия отклика - сумма квадратов остатков (невязок): $RSS = \sum_l \left(y_i - a(x_i)\right)^2$
- Общая дисперсия отклика: $TSS = \sum_l (y_i - E(y))^2$
- Объясненная дисперсия отклика – разность общей и необъясненной: $MSS = TSS - RSS = \sum_l \left(E(y) - a(x_i)\right)^2$

Линейная регрессия с точки зрения статистики

- Стандартная ошибка невязок $RSE = \sqrt{\frac{1}{n-2} RSS}$
- Коэффициент детерминации *R-квадрат* или доля объясненной дисперсии:

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$$

- Критерий Фишера для проверки базовой гипотезы (что отклик не зависит от предикторов):

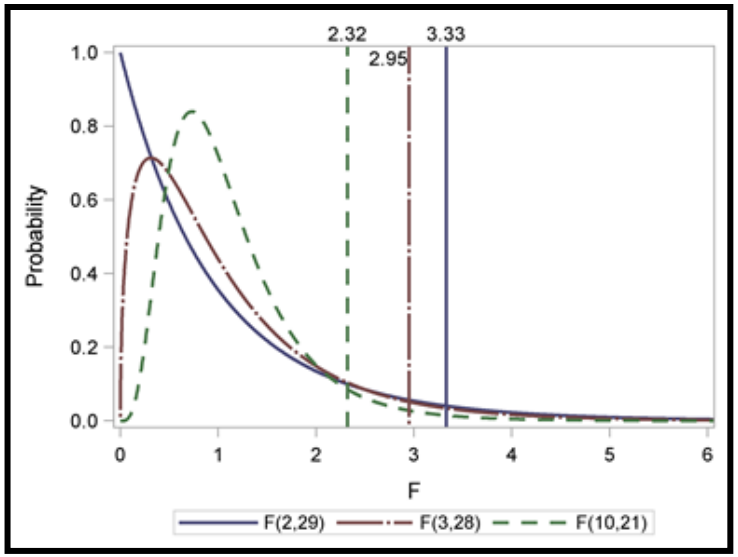
$$\mathrm{F} = \frac{(TSS-RSS)/p}{RSS/(l-p-1)} \sim F_{p,l-p-1}$$

Нулевая гипотеза:

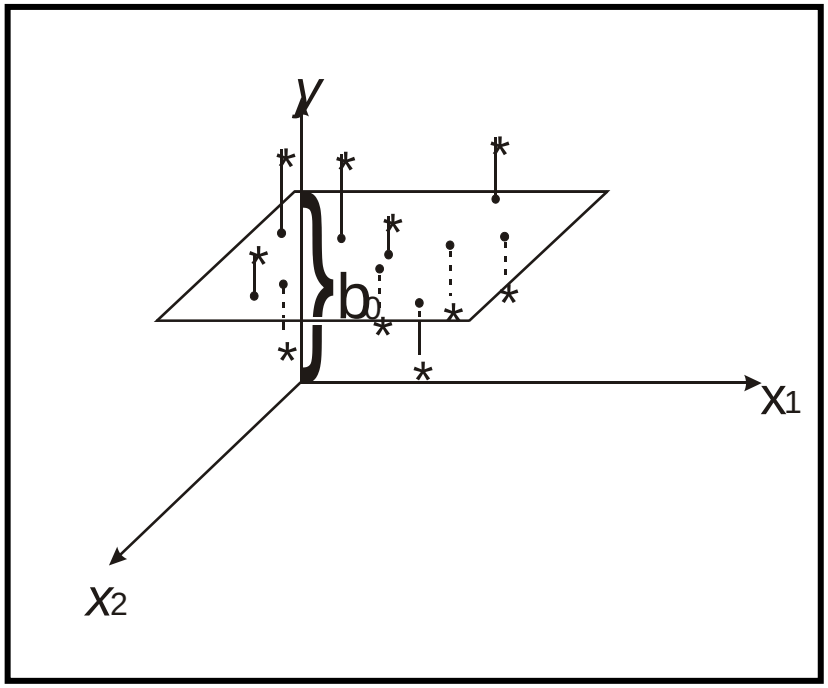
- все $w_1=0$

Альтернативная гипотеза:

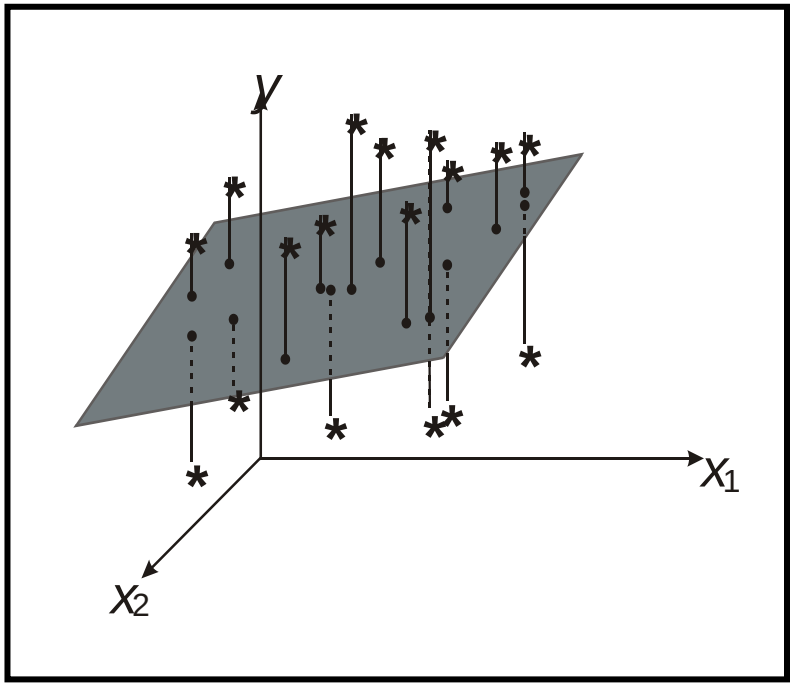
- существует $w_i \neq 0$



Множественная линейная регрессия

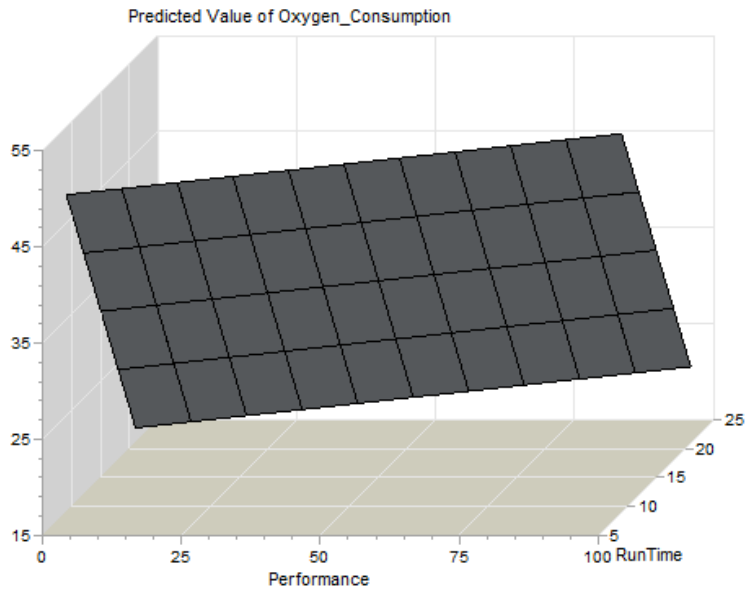


p-value для статистики Фишера больше заданного уровня значимости (например, 5%), тогда принимают базовую гипотезу, что **нет зависимости**



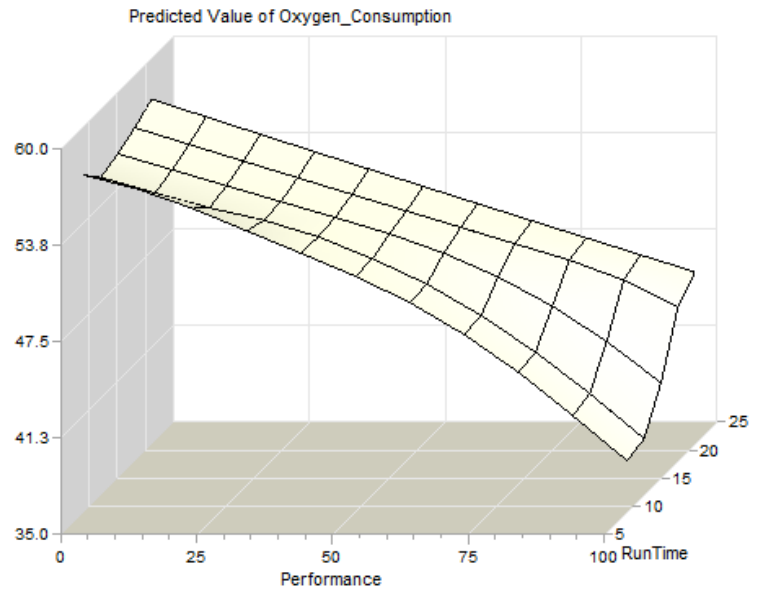
p-value для статистики Фишера меньше заданного уровня значимости (например, 5%), тогда **не** принимают базовую гипотезу, предполагая, что **зависимость есть**

Множественная линейная регрессия



Линейная модель с линейными эффектами

$$a(x) = w_0 + \sum_{j=1}^{p} x_j w_j + \varepsilon$$



Линейная модель с нелинейными эффектами, например

$$a(x) = w_0 + \sum_{j=1}^{p} x_j w_j + \sum_{j,i=1} x_i x_j w_{ij} + \varepsilon$$

или пример аддитивной

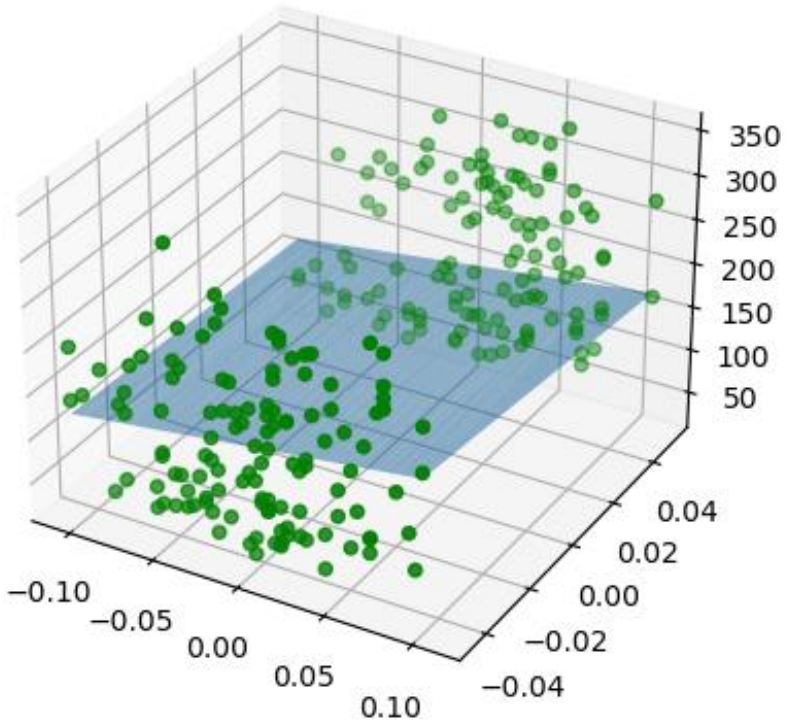
$$a(x) = w_0 + \sum_{j=1}^{p} f(x_j) w_j + \varepsilon$$

Пример (Python)

```
from sklearn.linear_model import LinearRegression
```

```
X_2d = X[:, :2]
X_2d_test = X_test[:, :2]
```

```
regr = LinearRegression()
regr.fit(X_2d, y)
pass
```




```
ax = plt.subplot(projection='3d')
ax.scatter(X_2d_test[:, 0], X_2d_test[:, 1], y_test, color="green")
ax.plot_trisurf(X_2d_test[:, 0], X_2d_test[:, 1], regr.predict(X_2d_test), alpha=0.5)
```

Метод наименьших квадратов для линейной регрессии

- Оценка ошибки - сумма регрессионных остатков (эмпирический риск с квадратичной функцией потерь):

$$RSS(w) = \sum_{i=1}^{l}(y_i - a(\bar{x}_i))^2 = \sum_{i=1}^{l}(y_i - w_0 - \sum_{j=1}^{p} x_{ij}w_j)^2$$

- В матричной форме: $RSS(w) = (y - Xw)^T(y - Xw)$ - квадратичная функция с p+1 параметрами, где w – вектор искомых параметров, X-матрица данных (строки – наблюдения, колонки - признаки), y – вектор известных откликов
- Найдем и приравняем нулю производную по вектору параметров, получим решение:

$$\nabla_w RSS(w) = -2X^T(y - Xw) = 0 \Rightarrow w = (X^T X)^{-1}X^T y$$

- Поскольку целевая функция выпуклая и матрица вторых производных имеет вид: $\nabla_w^2 RSS(w) = -2X^T X$ ⇒решение единственное и прогноз отклика можно найти по формуле:

$$\hat{y} = Xw = X\left(X^T X\right)^{-1}X^T y = Hy$$

Метод наименьших квадратов для линейной регрессии

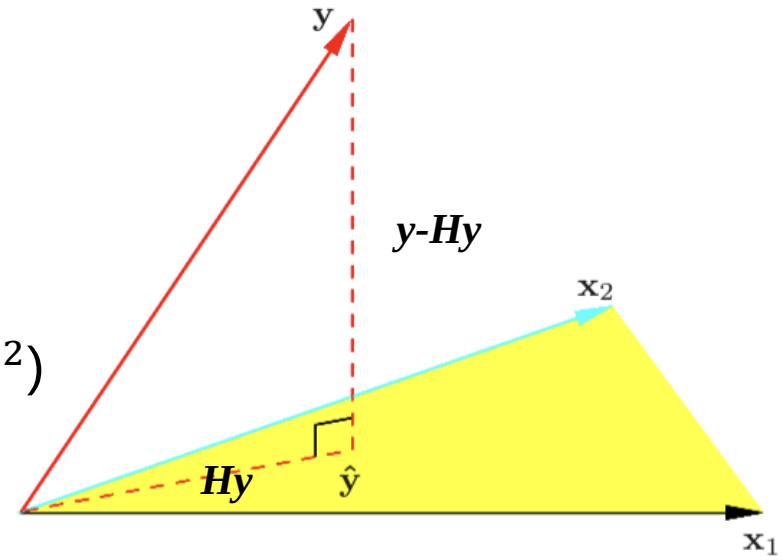
- $H = X\left(X^T X\right)^{-1} X^T$ - проекционная матрица (иногда называют «калькой» с английского – матрица «шляпы»), проецирует отклик на линейную оболочку столбцов матрицы данных (признаковое пространство)
- Можно оценить дисперсию прогноза и дисперсию оценок коэффициентов и соответствующих доверительных интервалов:

$$\hat{\sigma}^2 = \frac{RSS}{l-p-1},\ Var(w) = \left(X^T X\right)^{-1}\hat{\sigma}^2,$$

$$SE(w_i) = \hat{\sigma}\sqrt{diag\left((X^T X)^{-1}\right)_i}$$

- Предполагая $w \sim N(w^{true}, \left(X^T X\right)^{-1}\hat{\sigma}^2)$

получим 95% интервал $w_i \pm 2SE(w_i)$

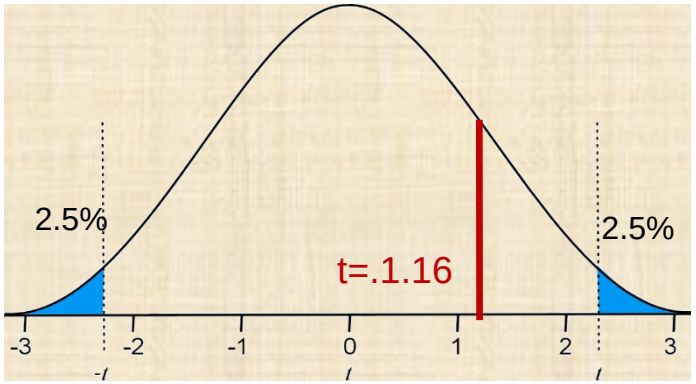


Проверка важности предикторов в МНК

- Для оценки важности предиктора можно проверить нулевую гипотезу (о равенстве нулю коэфициента), вычисляя *t-статистику* (критерий студента): $t_i = w_i / SE(w_i)$

Она будет иметь распределение Стьюдента с $l - 2$ степенями свободы

Можно вычислить вероятность наблюдения значения статистики, большего или равного |t|, это *p*-value.



$$\widehat{\texttt{sales}} = \beta_0 + \beta_1 \times \texttt{TV}$$

	Coefficient	Std. Error	t-statistic	p-value
Intercept	7.0325	0.4578	15.36	< 0.0001
TV	0.0475	0.0027	17.67	< 0.0001

Бинарные признаки

Пример: исследовать различия в балансе кредитных карт между мужчинами и женщинами, не учитывая другие переменные.

Создается новая переменная

$$x_i = \begin{cases} 1 & \text{if } i\text{th person is female} \\ 0 & \text{if } i\text{th person is male} \end{cases}$$

Итоговая модель имеет вид:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{if } i\text{th person is female} \\ \beta_0 + \epsilon_i & \text{if } i\text{th person is male.} \end{cases}$$

Интерпретация:

	Coefficient	Std. Error	t-statistic	p-value
`Intercept`	509.80	33.13	15.389	< 0.0001
`gender[Female]`	19.73	46.05	0.429	0.6690

Категориальные признаки с двумя и более значениями

- Для признаков с несколькими возможными значениями создаются дополнительные фиктивные переменные. Например, для переменной ethnicity :

$$x_{i1} = \begin{cases} 1 & \text{if } i\text{th person is Asian} \\ 0 & \text{if } i\text{th person is not Asian,} \end{cases}$$

а вторая:

$$x_{i2} = \begin{cases} 1 & \text{if } i\text{th person is Caucasian} \\ 0 & \text{if } i\text{th person is not Caucasian.} \end{cases}$$

- Тогда обе эти переменные могут быть использованы в формуле регрессии и модель будет иметь вид

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{if } i\text{th person is Asian} \\ \beta_0 + \beta_2 + \epsilon_i & \text{if } i\text{th person is Caucasian} \\ \beta_0 + \epsilon_i & \text{if } i\text{th person is AA.} \end{cases}$$

Категориальные признаки с двумя и более значениями

- Число фиктивных переменных будет на единицу меньше, чем количество возможных различных значений, есть специальное *базовое значение*.

Level	D_A	D_B	D_C	D_D	D_E	D_F	D_G	D_H	D_I
A	1	0	0	0	0	0	0	0	0
B	0	1	0	0	0	0	0	0	0
C	0	0	1	0	0	0	0	0	0
D	0	0	0	1	0	0	0	0	0
E	0	0	0	0	1	0	0	0	0
F	0	0	0	0	0	1	0	0	0
G	0	0	0	0	0	0	1	0	0
H	0	0	0	0	0	0	0	1	0
I	0	0	0	0	0	0	0	0	1

	Coefficient	Std. Error	t-statistic	p-value
Intercept	531.00	46.32	11.464	< 0.0001
ethnicity[Asian]	-18.69	65.02	-0.287	0.7740
ethnicity[Caucasian]	-12.50	56.68	-0.221	0.8260

Кодирование категориальных переменных по отклику

- Основная идея:
  - Отобразить категориальную переменную на числовую шкалу так, чтобы на новой шкале «рядом» были значения категориальных переменных, на которых отклик ведет себя одинаково, например:

$$X_i^{new} = E(y|\ X^{old} = Val_i),$$

где X^{old} - категориальная переменная,

Val_i - одно из значений X^{old},

$X^{new} \in \mathbb{R}$ - новая числовая переменная.

id	job	job_mean	target
1	Doctor	0,50	1
2	Doctor	0,50	0
3	Doctor	0,50	1
4	Doctor	0,50	0
5	Teacher	1	1
6	Teacher	1	1
7	Engineer	0,50	0
8	Engineer	0,50	1
9	Waiter	1	1
10	Driver	0	0

- Зачем?
  - Упрощает модель, сохраняя информацию
  - Уменьшает потенциальные корреляции с другими признаками (они часто возникают при one hot кодировании нескольких категориальных переменных)
  - Уменьшает возможность переобучения (т.к. модель проще)
  - Увеличивает стабильность модели (т.к. нет «редких уровней)

Интерпретация коэффициентов регрессии

- Идеальный сценарий: предикторы не коррелированы:
  - Каждый коэффициент можно оценить и тестировать отдельно.
  - Интерпретации такие как «*единичное изменение предиктора связано с w_j –ым изменением в значении отклика, тогда как все остальные переменные остаются фиксированными*»
  - Для корректной оценки влияния предиктора на отклик нужно либо стандартизировать коэффициенты, либо нормировать признаковое пространство
- Корреляции между переменными вызывают проблемы:
  - Дисперсия всех коэффициентов имеет тенденцию к увеличению
  - Интерпретации становятся непредсказуемыми - когда предиктор меняется, зависимые с ним тоже меняется.

«По сути, все модели ошибочны, но некоторые из них полезны»

George Box

Расширение линейной модели

Удаление предположения аддитивности: *взаимодействующие* переменные и *нелинейность* в уравнении

Взаимодействующие переменны:

- Раньше мы предполагали, что влияние предикторов на отклик независимо, например:

$$\widehat{\texttt{sales}} = \beta_0 + \beta_1 \times \texttt{TV} + \beta_2 \times \texttt{radio} + \beta_3 \times \texttt{newspaper}$$

- Модель с взаимосвязями имеет вид

$$\begin{aligned} \texttt{sales} &= \beta_0 + \beta_1 \times \texttt{TV} + \beta_2 \times \texttt{radio} + \beta_3 \times (\texttt{radio} \times \texttt{TV}) + \epsilon \\ &= \beta_0 + (\beta_1 + \beta_3 \times \texttt{radio}) \times \texttt{TV} + \beta_2 \times \texttt{radio} + \epsilon. \end{aligned}$$

Интерпретация примера

- Результаты в этой таблице показывают, что взаимосвязи важны.
- Величина *p*-value для члена TV*radio, отражающего взаимосвязь, мала, что свидетельствует о наличии достоверных доказательств для гипотезы $w_{tv*radio} \neq 0$.
- R^2 для модели с учетом взаимосвязей составляет 96.8%, по сравнению с моделью, которая прогнозирует значение sales, используя значения TV и radio без учета взаимосвязей между ними 89.7%
- Это означает, что (96.8 – 89.7)/(100 – 89.7) = 69% дисперсии для sales, которая остается после построения аддитивной модели, объясняется эффектом, отражающим взаимосвязь.

	Coefficient	Std. Error	t-statistic	p-value
Intercept	6.7502	0.248	27.23	< 0.0001
TV	0.0191	0.002	12.70	< 0.0001
radio	0.0289	0.009	3.24	0.0014
TV×radio	0.0011	0.000	20.73	< 0.0001

Пример

	fixed acidity	volatile acidity	citric acid	sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	target_quality
4893	6.2	0.21	0.29	1.6	0.039	24.0	92.0	0.99114	3.27	0.50	11.2	6
4894	6.6	0.32	0.36	8.0	0.047	57.0	168.0	0.99490	3.15	0.46	9.6	5
4895	6.5	0.24	0.19	1.2	0.041	30.0	111.0	0.99254	2.99	0.46	9.4	6
4896	5.5	0.29	0.30	1.1	0.022	20.0	110.0	0.98869	3.34	0.38	12.8	7
4897	6.0	0.21	0.38	0.8	0.020	22.0	98.0	0.98941	3.26	0.32	11.8	6

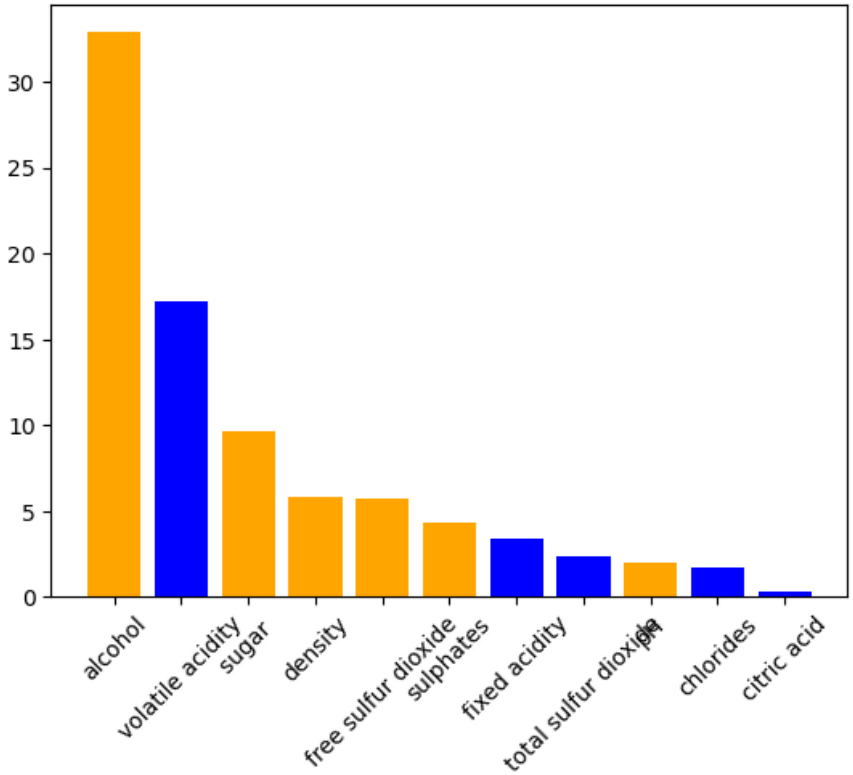
```python
import statsmodels.api as sm

y = df["target_quality"]
X = df[set(df.columns) - {"target_quality"}]

results = sm.OLS(y, X).fit()
results.summary()
```

Dep. Variable:	target_quality	**R-squared (uncentered):**	0.984
Model:	OLS	**Adj. R-squared (uncentered):**	0.984
Method:	Least Squares	**F-statistic:**	2.707e+04
Date:	Sun, 05 Mar 2023	**Prob (F-statistic):**	0.00
Time:	08:08:45	**Log-Likelihood:**	-5575.5
No. Observations:	4898	**AIC:**	1.117e+04
Df Residuals:	4887	**BIC:**	1.124e+04
Df Model:	11		

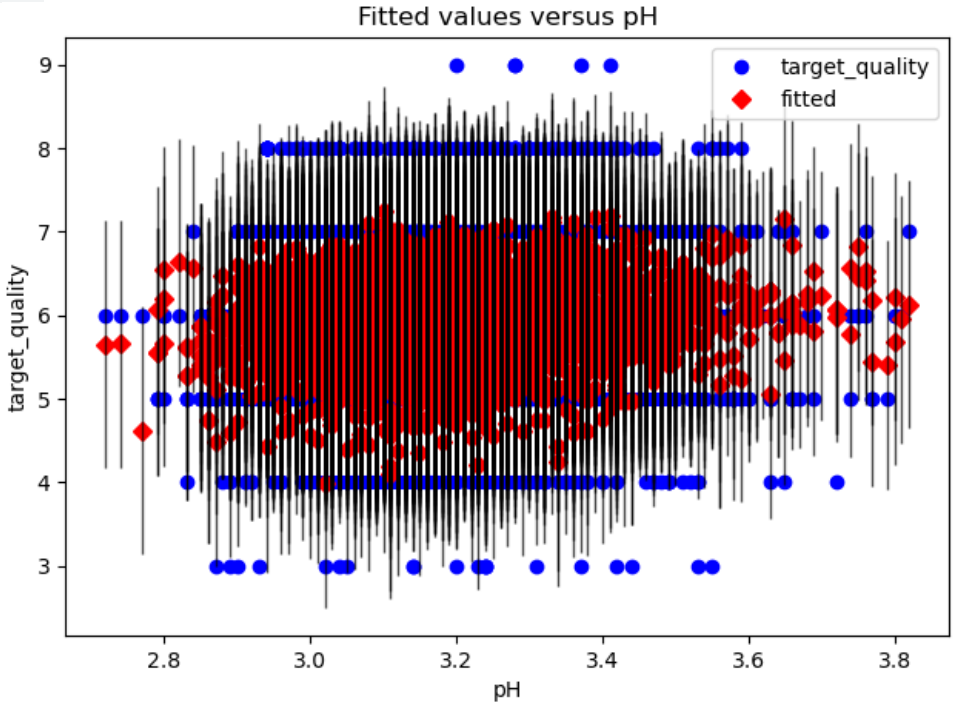
	coef	std err	t	P>|t|	[0.025	0.975]
sugar	0.0250	0.003	9.642	0.000	0.020	0.030
pH	0.1684	0.084	2.014	0.044	0.005	0.332
citric acid	-0.0293	0.096	-0.305	0.760	-0.218	0.159
volatile acidity	-1.9585	0.114	-17.196	0.000	-2.182	-1.735
density	2.0420	0.353	5.780	0.000	1.349	2.735
alcohol	0.3656	0.011	32.880	0.000	0.344	0.387
free sulfur dioxide	0.0048	0.001	5.710	0.000	0.003	0.006
sulphates	0.4165	0.097	4.279	0.000	0.226	0.607
chlorides	-0.9426	0.543	-1.736	0.083	-2.007	0.122
total sulfur dioxide	-0.0009	0.000	-2.352	0.019	-0.002	-0.000
fixed acidity	-0.0506	0.015	-3.356	0.001	-0.080	-0.021

Пример

```
res = results.tvalues
sign = (res > 0).map({True:"orange", False:"blue"})
res = abs(res).sort_values()[::-1]
sign = sign[res.index].values
plt.bar(res.index, res.values, color=sign)
plt.xticks(rotation=45)
plt.show()
```




```
fig = sm.graphics.plot_fit(results, "pH")
fig.tight_layout(pad=1.0)
```



Пример с formula

```
import statsmodels.formula.api as smf
model1 = smf.ols(formula='target_quality ~ alcohol * pH', data=df)

res1 = model1.fit()
res1.summary()
```

model1

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	16.9769	2.194	7.739	0.000	12.676	21.277
alcohol	-1.1383	0.207	-5.491	0.000	-1.545	-0.732
pH	-4.5215	0.691	-6.547	0.000	-5.875	-3.168
alcohol:pH	0.4557	0.065	6.990	0.000	0.328	0.583

Dep. Variable:	target_quality	**R-squared:**	0.200
Model:	OLS	**Adj. R-squared:**	0.199
Method:	Least Squares	**F-statistic:**	407.6
No. Observations:	4898	**AIC:**	1.162e+04
Df Residuals:	4894	**BIC:**	1.165e+04
Df Model:	3		

```
model2 = smf.ols(formula='target_quality ~ alcohol + pH', data=df)

res2 = model2.fit()
res2.summary()
```

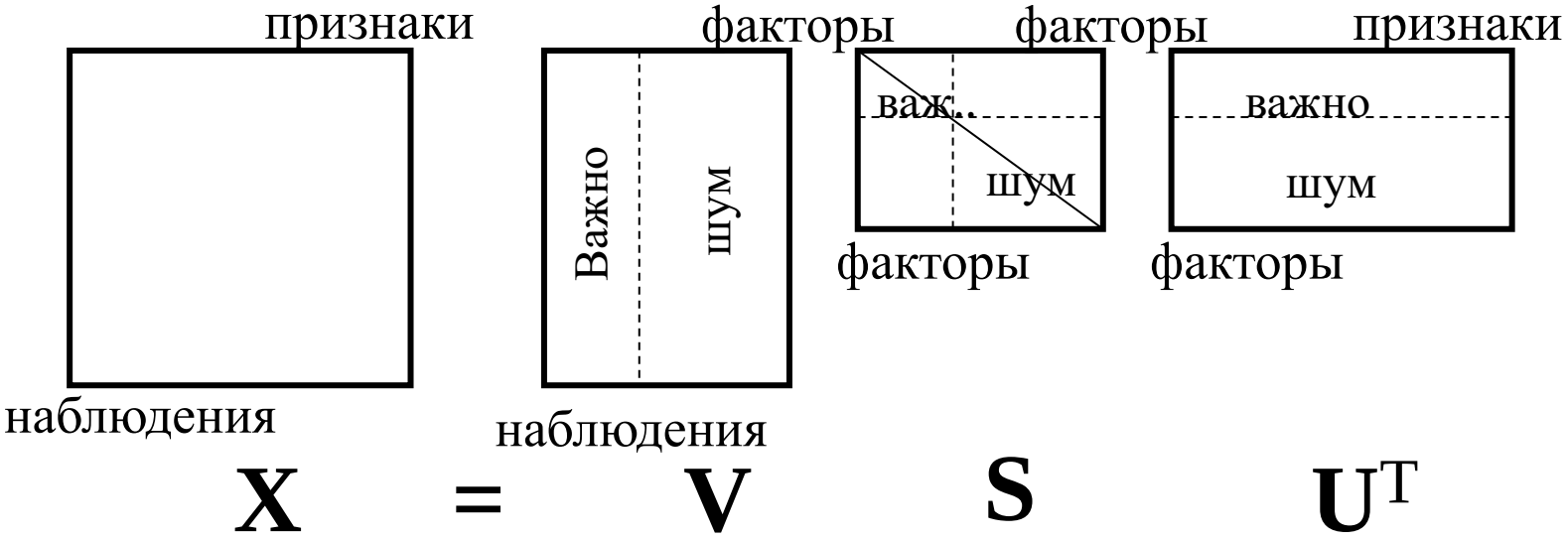
model2

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.7422	0.250	6.966	0.000	1.252	2.233
alcohol	0.3093	0.009	33.207	0.000	0.291	0.328
pH	0.2770	0.076	3.649	0.000	0.128	0.426

Dep. Variable:	target_quality	**R-squared:**	0.192
Model:	OLS	**Adj. R-squared:**	0.192
Method:	Least Squares	**F-statistic:**	581.3
No. Observations:	4898	**AIC:**	1.167e+04
Df Residuals:	4895	**BIC:**	1.169e+04
Df Model:	2		

Вычисление МНК

- Обычно на основе матричных разложений QR, SVD и др.
- Рассмотрим SVD:



- По умолчанию число факторов равно числу признаков
- Орт. матрица $U^T U = I$ правых сингулярных векторов, с.в. $X^T X$
- Орт. матрица $VV^T = I$ левых сингулярных векторов, с.в. XX^T
- $S = diag(\sqrt{\lambda_1},\ldots,\sqrt{\lambda_p})$ – с.зн. XX^T и $X^T X$

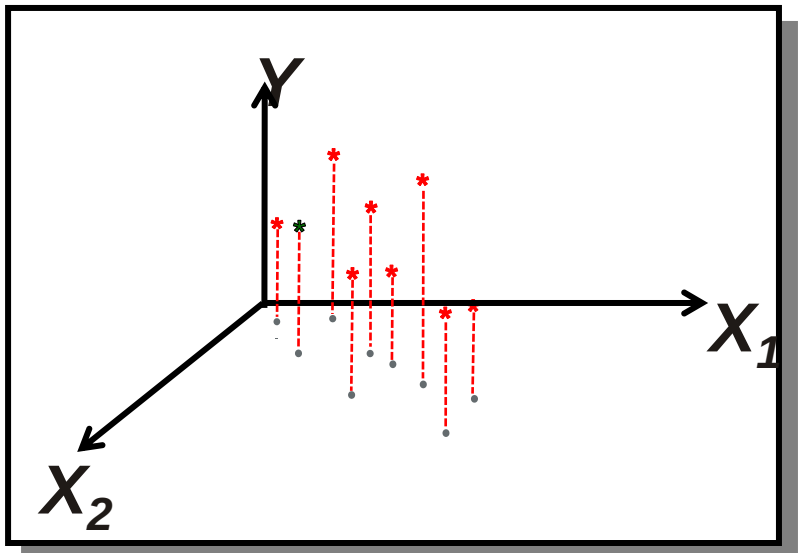
Вычисление МНК

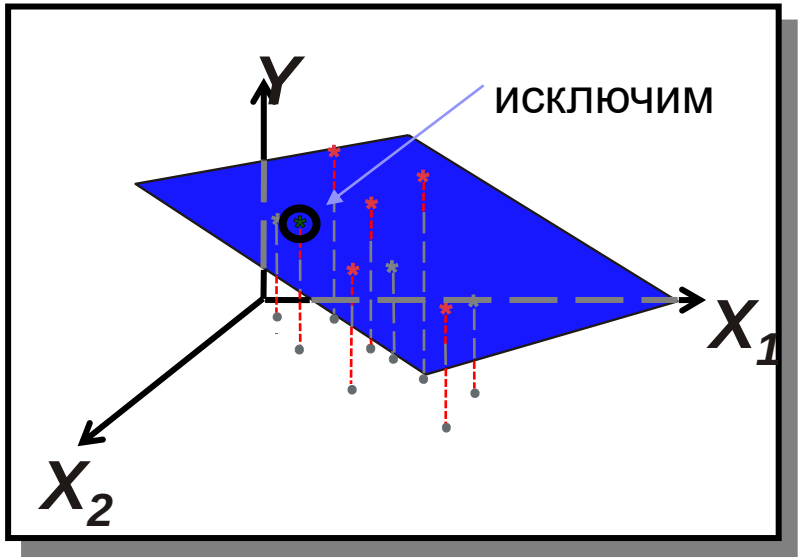
- Псевдообратная матрия $X^+ = \left(X^T X\right)^{-1} X^T$:

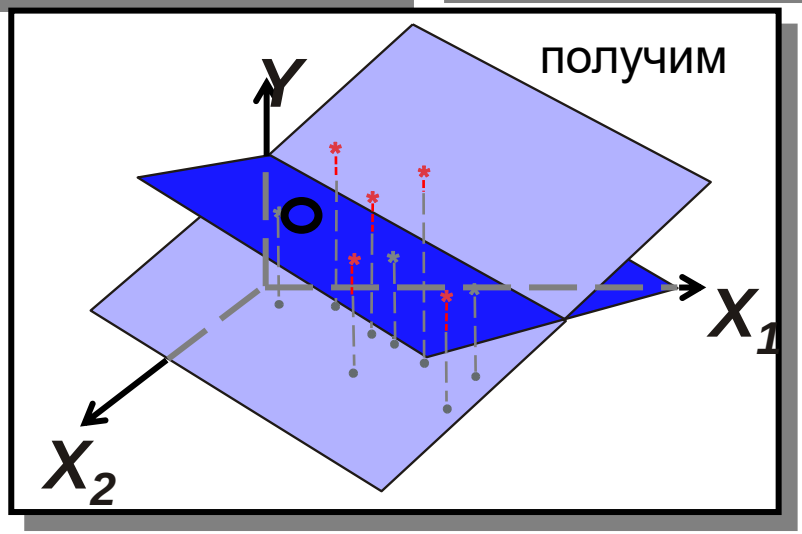
$$X^+ = \left(USV^T VSU^T\right)^{-1} USV^T = US^{-1}V^T = \sum_i \frac{1}{\sqrt{\lambda_i}} u_i v_i^T$$

- Решение: $w = X^+ y = \sum_i \frac{1}{\sqrt{\lambda_i}} u_i (v_i^T y)$
- Если есть корреляции между признаками в X, тогда:
  - $X^T X$ плохо обусловлена (велико $\lambda_{max}/\lambda_{min}$)
  - Увеличивается погрешность вычисления решения w и норма $\left||w|\right|^2$
  - Решение неустойчиво и плохо интерпретируемо
  - Возникает переобучение
  - Портятся статистики с оценкой значимости переменных
  - Увеличивается вариативность оценки параметров и как следствие ошибка
  - Есть тенденция к неограниченному росту коэффициентов при завис. призн.

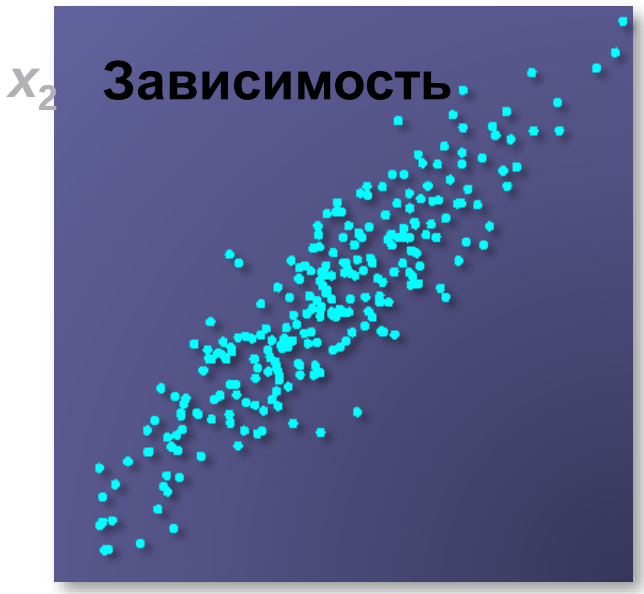
Иллюстрация неустойчивости при мультиколлинеарности

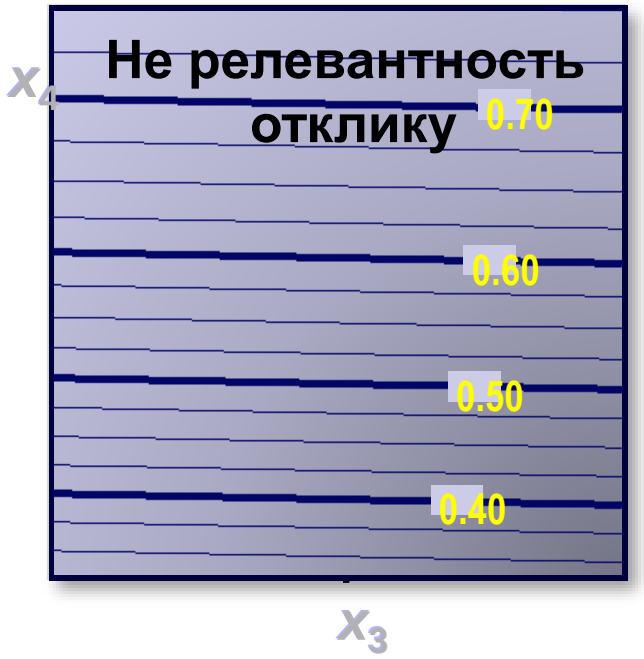






Проблемы входных переменных для МНК

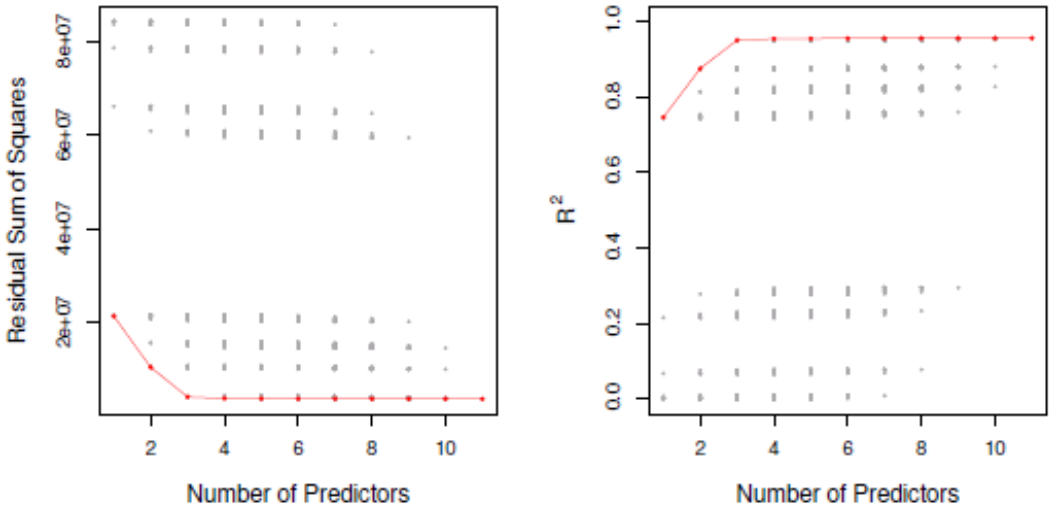




- Что делать?
  - Отбирать признаки (пошаговые методы, прямой, обратный, комбинированные)
  - Использовать регуляризацию (штраф за сложность, гребневая регрессия, LASSO, Elastic Net)
  - Преобразовывать исходное пространство признаком (регрессия главных компонент и PLS)

Полный перебор переменных

- Наиболее очевидный подход называется регрессия *всех подмножеств* или регрессия *наилучших подмножеств* (МНК для всех комбинаций и выбор лучшего варианта по некторому критерию)



- На практие – не всегда применимо для значительного числа переменных, экспоненциально растет число проверяемых моделей, поэтому жадный пошаговый перебор

Особенности пошаговых методов отбора

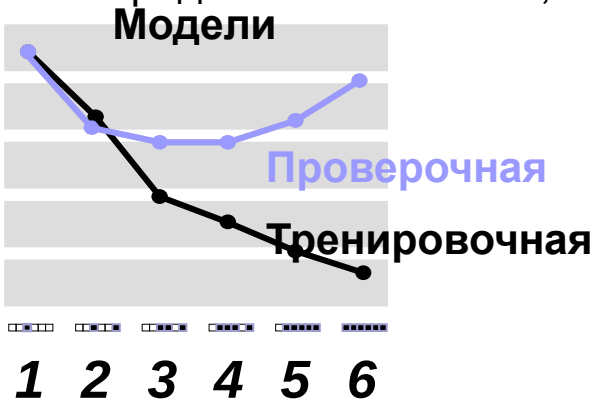
- Жадные алгоритмы:
  - не находят глобально лучшую модель даже с точки зрения эмпирического риска на тренировочном набор
- Необходимо определить 3 правила:
  - Правило для выбора следующего шага (выбор переменной для добавления и/или удаления)
  - Правило для останова (когда дальнейшее изменение модели не целесообразно)
  - Правило выбора лучшей модели из семейства (если не совпадает с правилом останова)

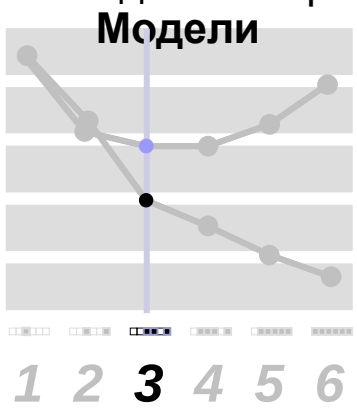
Правила на основе «лучшего» эмпирического риска на тренировочном наборе

- Суть правила:
  - наибольшее улучшение эмпирического риска для шага добавления переменных в модель и наименьшее ухудшение эмпирического риска для удаления переменных из модели
  - правило для останова можно задать, определив пороги: минимальный допустимый «прирост» при добавлении переменных или максимально допустимое «ухудшение» модели при удалении переменной
  - для линейной регрессии можно использовать выбор по RSS, коэффициенту детерминации, корреляции с остатками и т.д.
- Достоинство
  - простота и скорость расчета
- Недостатки:
  - не учитывается сложность модели, сложные всегда лучше на тренировочном наборе, значит не подходит для выбора лучшей модели
  - тяжело определить порог останова – нет общей шкалы, у каждой задачи будет своя, в зависимости от дисперсии отклика
  - не оценивается обобщающая способность получаемой модели

На основе лучшей обобщающей способности

- Суть правила:
  - Использование валидационной выборки, кросс-валидации или бутстреппинга для сравнения моделей
- Достоинство
  - Самый «правильный» метод с точки зрения теории, т.к. оценивается обобщающая способность получаемой модели
- Недостатки:
  - Большая вычислительная сложность, поэтому редко используется для выбора шага и определения останова, но часто для выбора лучше модели





Правила на основе статистического подхода – тест Стьюдента

- Суть правила:
  - Используем статистическую оценку важности переменной на основе критерия Стьюдента
  - При выборе предиктора-кандидата на удаление выбираем переменную с максимальным p-value для t-статистики
  - При выборе предиктора-кандидата на добавление выбираем с минимальным p-value для t-статистики
- Достоинство
  - Пороги для останова задаются для p-value на шкале [0,1] не зависимо от разброса отклика
- Недостатки:
  - Не учитывается сложность модели
  - Не оценивается обобщающая способность получаемой модели

Правила на основе статистического подхода – тест Фишера, тест Уальда

- Суть правила:
  - Оцениваем не отдельные переменные, а модели «до» и «после» шага добавления/удаления, например, по Уальду:

$$W = \frac{\left(RSS_{short} - RSS_{long}\right)}{RSS_{long}/l}, \sim \chi^2_{(p_{long} - p_{short})}$$

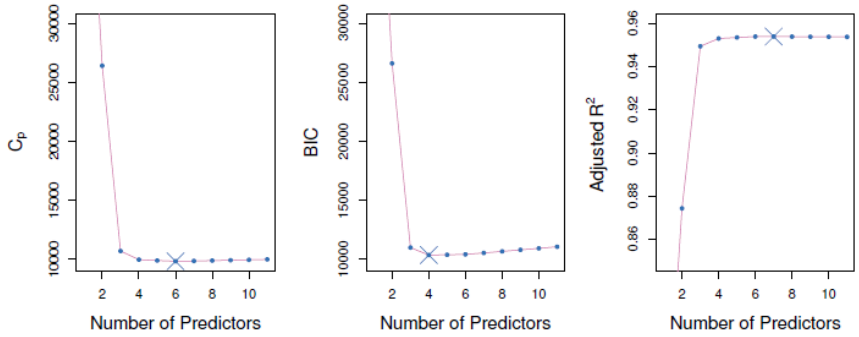
  - или сравниваем с самой плохой моделью (без предикторов) с помощью критерия Фишера:

$$\mathrm{F} = \frac{(TSS - RSS)/p}{RSS/(l - p - 1)} \sim F_{p,l-p-1}$$

- Достоинства:
  - Пороги для останова задаются для p-value на шкале [0,1]
  - Учитывается сложность модели
- Недостатки:
  - Напрямую не оценивается обобщающая способность модели

Правила на основе информационных критериев C_p, AIC, BIC и скорректированного R^2

- Суть правила:
  - Корректируют ошибку обучения с учетом сложности модели, и могут быть использованы для выбора лучшего шага, останова и лучшей среди множества моделей с различным числом предикторов



- Достоинства:
  - Учитывается сложность модели
  - Считается, что приблизительно оценивается обобщающая способность модели
- Недостатки:
  - Если используем в качестве правила останова, то прекращаем отбор, если критерий перестает улучшаться

C_p и AIC

- *Mallow* C_p :

$$C_p = \frac{1}{l}(RSS + 2p\sigma^2)$$

  - где p – число степеней свободы модели, обобщенное значение числа используемых параметров
  - σ^2 - оценка дисперсии ошибки ε, связанной с каждым измерением отклика.

- Критерий *AIC*, определяемый для более широкого класса моделей, рассчитывается методом максимального правдоподобия:

$$\mathrm{AIC} = -2loglik + 2p$$

  - В случае линейной модели с гауссовскими ошибками, максимальное правдоподобие и наименьшие квадраты - это одно и то же, т.е. C_p и AIC эквивалентны.

BIC и скорректированный R^2

- Байесовский информационный критерий:

$$BIC = \frac{1}{l}(RSS + \mathbf{log}(\boldsymbol{l})\, p\sigma^2)$$

- Аналогично *AIC* и C_p, но *BIC*, увеличивает штраф за сложность в $\log(l)$ раз, и при $l > 7$ сильнее штрафует модели, и приводит к выбору модели меньшего размера.
- Для модели МНК с *d* переменными скорректированная R^2 :

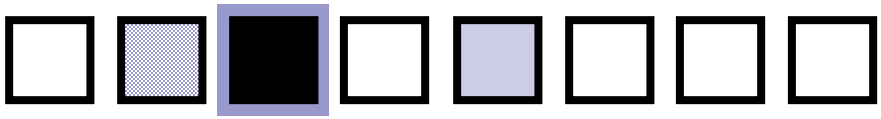
$$Adj\ R^2 = 1 - \frac{RSS(\boldsymbol{l-1})}{(\boldsymbol{l-p-1})TSS}$$

- В отличие от C_p, AIC и BIC, *большое* значение скорректированного R^2 соответствует модели с небольшой ошибкой тестирования.
- В отличие от статистики R^2, скорректированная R^2 статистика наказывает за включение «ненужных» переменных в модель.

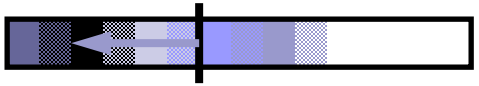
Прямой отбор

- Начинаем с *нулевой модели* содержащей только константу.
- Цикл:
  - Рассматриваем все варианты добавления одной из еще не добавленных переменных
  - Выбираем тот вариант, что соответствует выбранному правилу отбора (например, наименьшее значение RSS, или наименьшее p-value для критерия Фишера, Уальда или Стьюдента, наибольше уменьшение информационного критерия и т.д.)
  - Проверяем правило останова (по порогам для статистических критериев и правил на основе эмпирического риска или по ухудшению информационного критерия)
  - Если правило останова сработало, то выходим из цикла, иначе добавляем переменную в модель, переобучаем ее и переходим на следующую итерацию

Input *p*-value

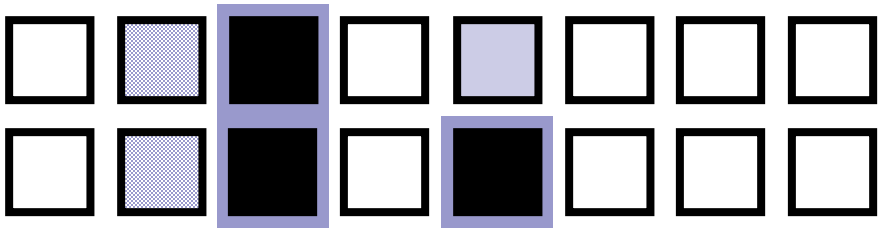


Entry Cutoff

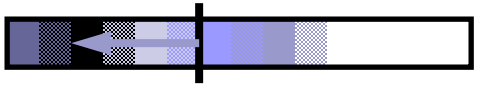


...

Input *p*-value

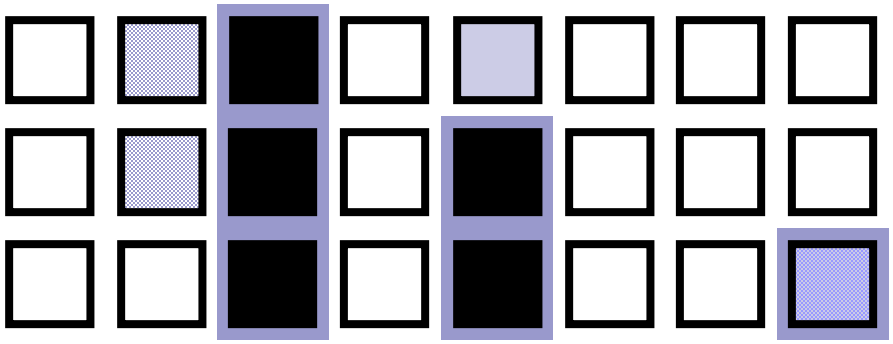


Entry Cutoff

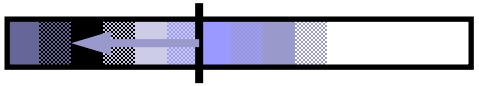


...

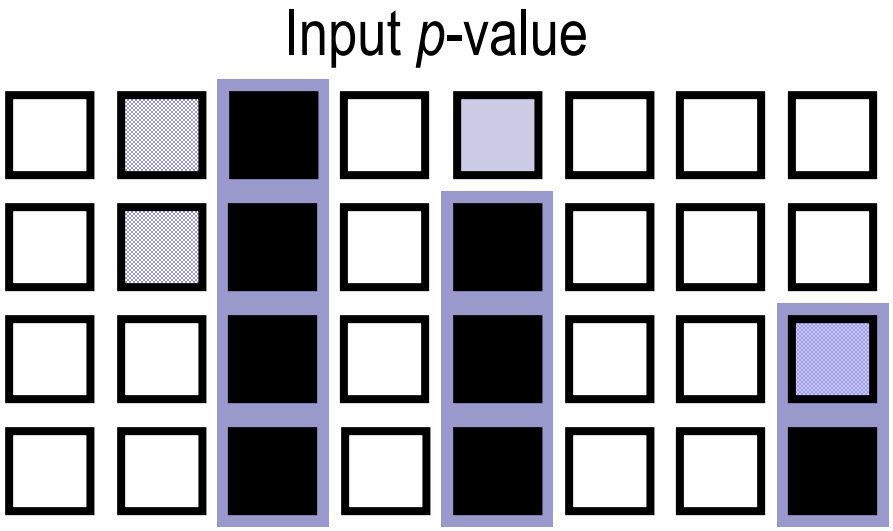
Input *p*-value

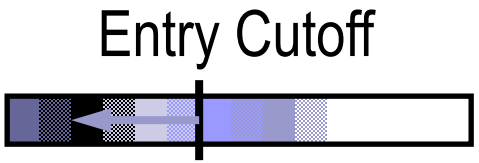


Entry Cutoff

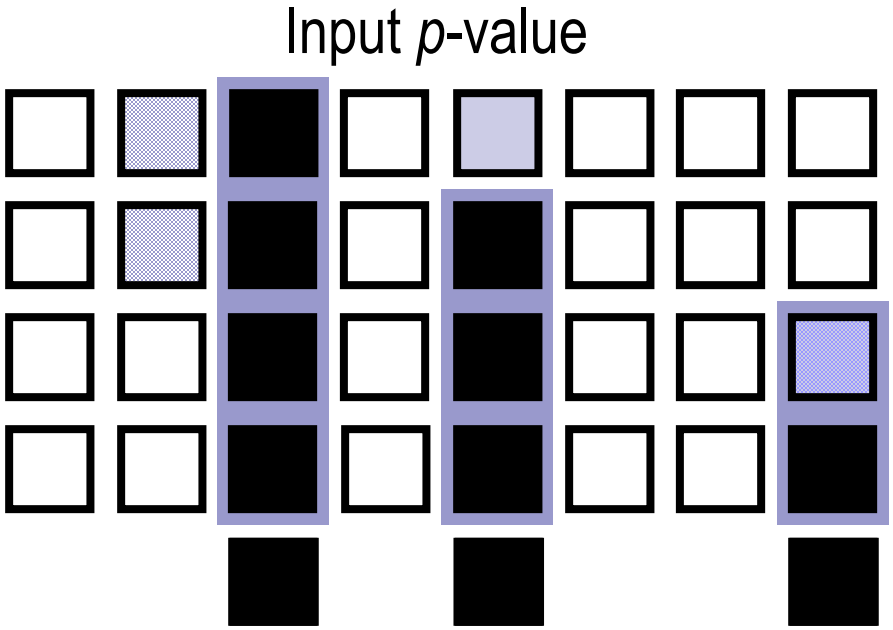


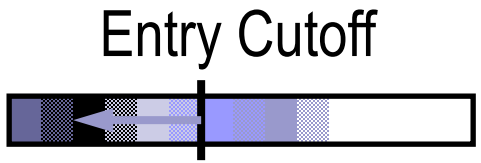
...


Input *p*-value


Entry Cutoff

...

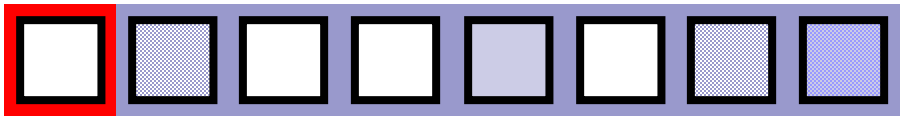

Input *p*-value


Entry Cutoff

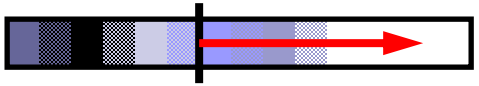
Обратный отбор

- Начинаем с *полной модели,* содержащей все переменный.
- Цикл:
  - Рассматриваем все варианты удаление одной из оставшихся переменных
  - Выбираем тот вариант, что соответствует выбранному правилу отбора (например, наибольшее значение RSS, или наибольшее p-value для критерия Фишера, Уальда или Стьюдента, наибольше уменьшение информационного критерия и т.д.)
  - Проверяем правило останова (по порогам для статистических критериев и правил на основе эмпирического риска или по ухудшению информационного критерия)
  - Если правило останова сработало, то выходим из цикла, иначе удаляем выбранную переменную из модели, перестраиваем ее и переходим на следующую итерацию

Input *p*-value

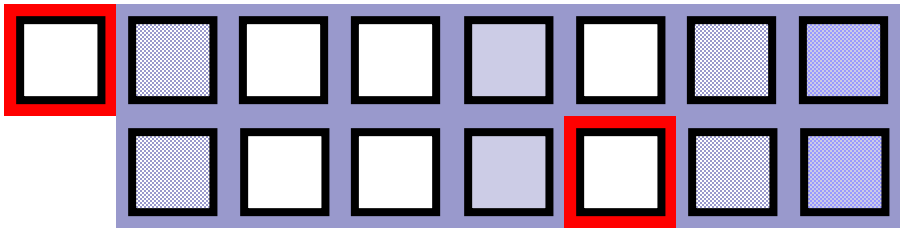


Stay Cutoff

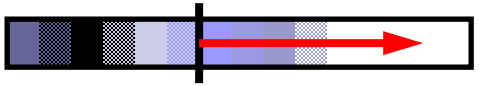


...

Input *p*-value

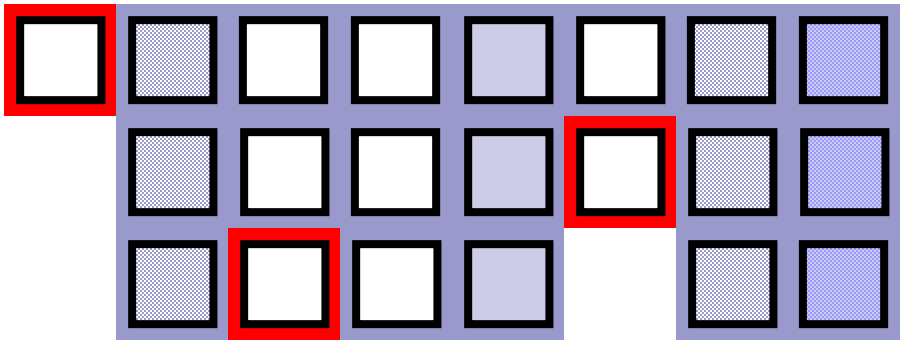


Stay Cutoff

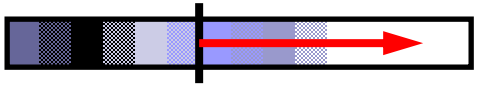


...

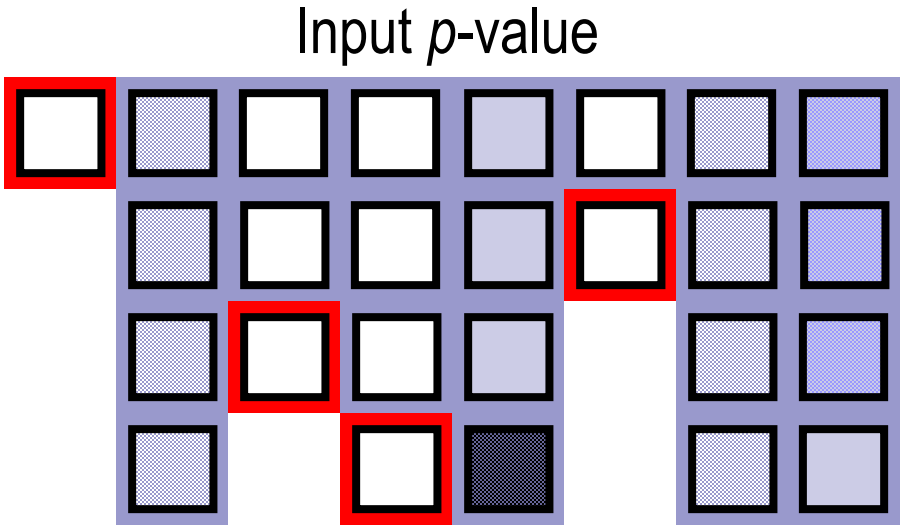
Input *p*-value

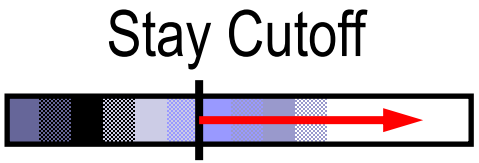


Stay Cutoff

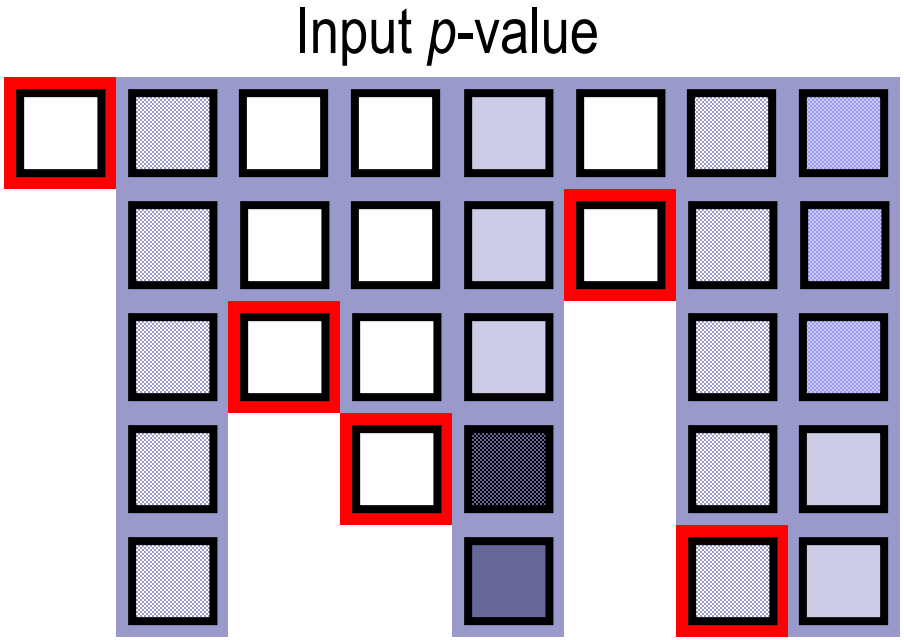


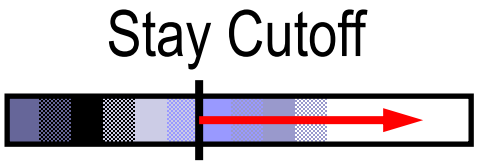
...


Input *p*-value

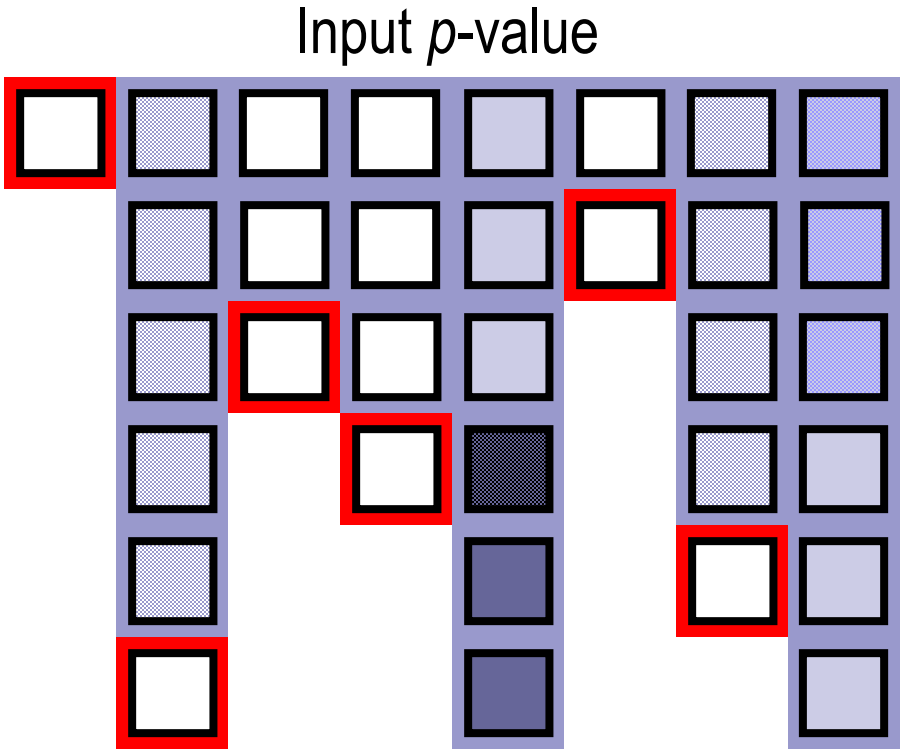

Stay Cutoff

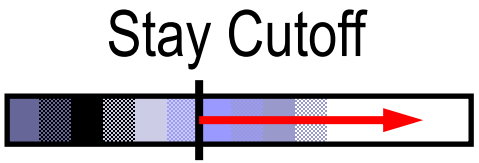
...


Input *p*-value

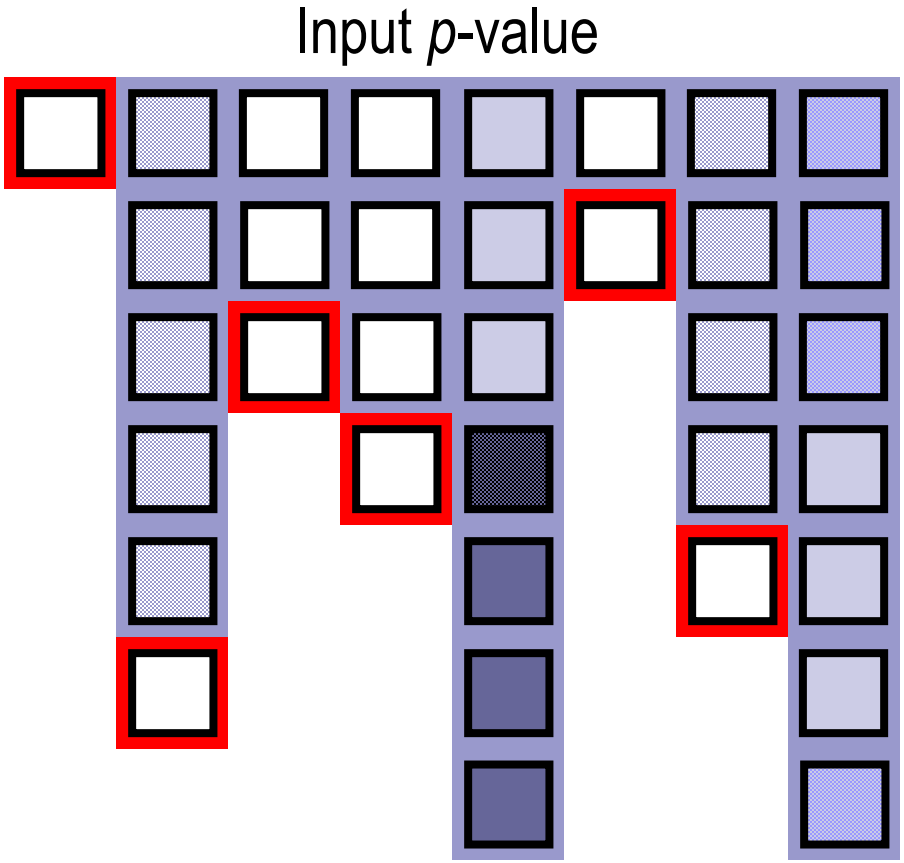

Stay Cutoff

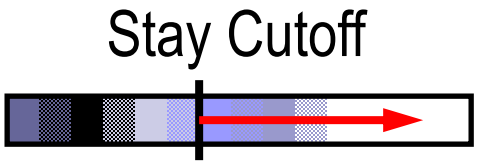
...


Input *p*-value

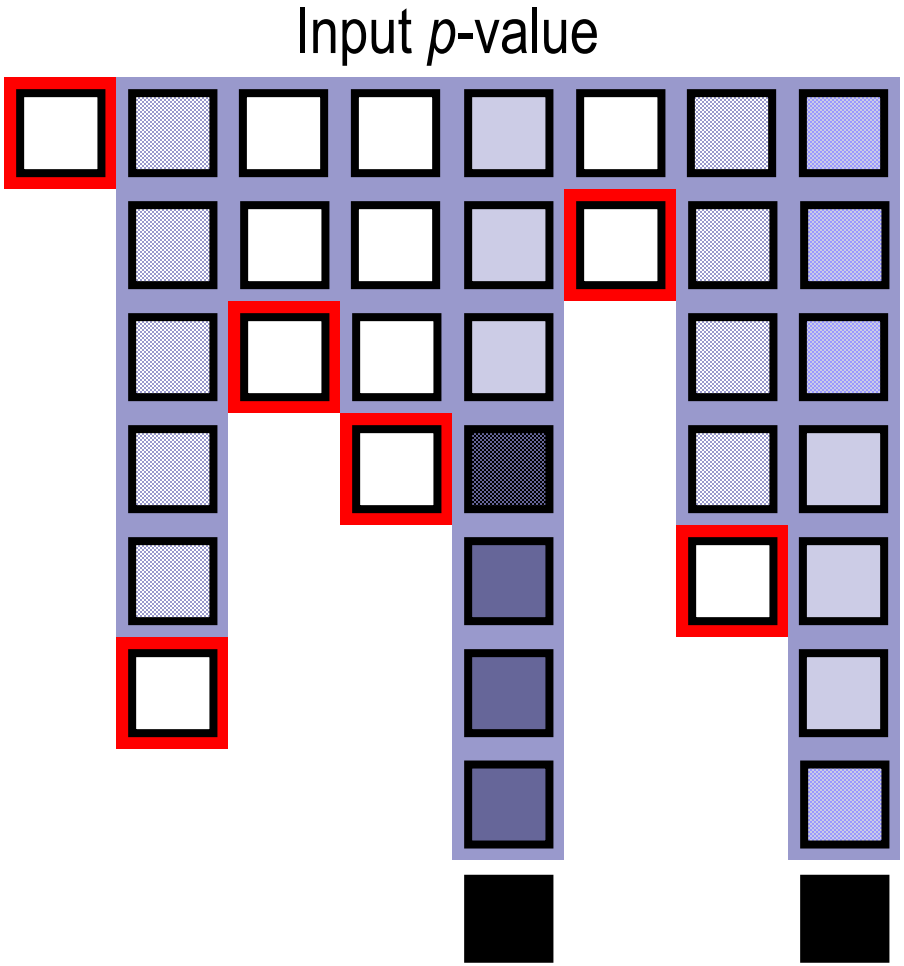

Stay Cutoff

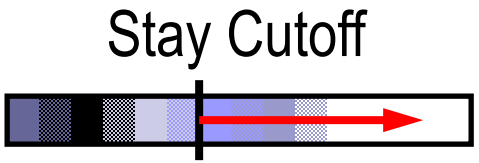
...


Input *p*-value


Stay Cutoff

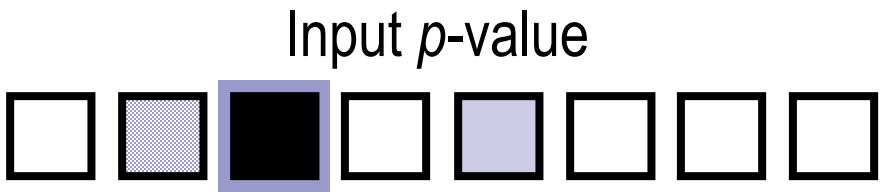
...

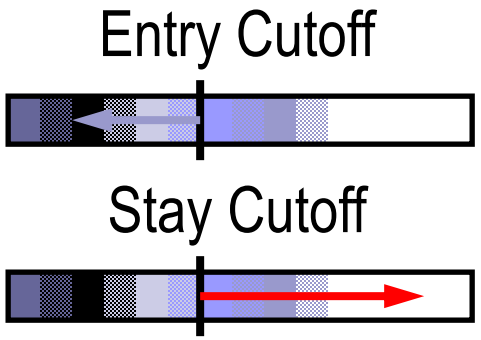

Input *p*-value


Stay Cutoff

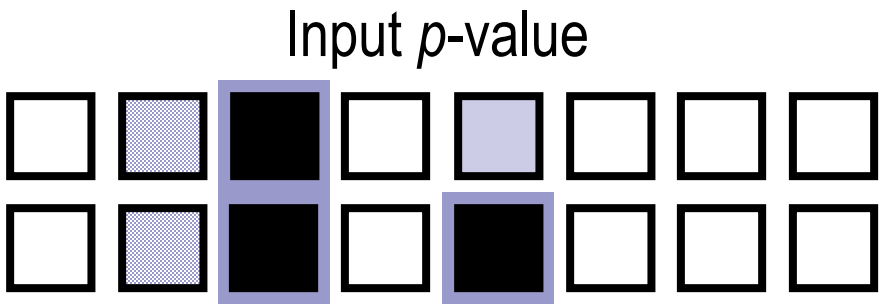
Комбинированный отбор

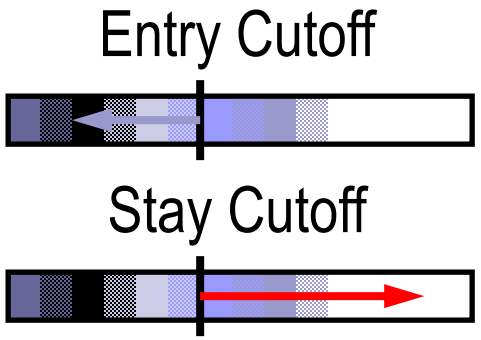
- Прямой и обратный методы – жадные (как и комбинированный) с не сильно гибким перебором, пытаемся улучшить
- Комбинированный метод (более гибкий перебор):
  - Пытаемся сделать шаг вперед (сначала из нулевой модели)
  - Затем пытаемся сделать шаг назад
  - Продолжаем 1 и 2 до тех пор, пока не сработает правило останова и для добавления, и для удаления переменных или пока не попали в «цикл» (последоватльное добавление и удаление одной и той же переменной)
- Есть варианты дополнительного «свопа»:
  - не просто пытаемся добавить, а потом удалить, а сначала ищем лучшую пару на замену (одну не добавленную с одной добавленной), если не находим, то обычный комбинированный шаг.


Input *p*-value

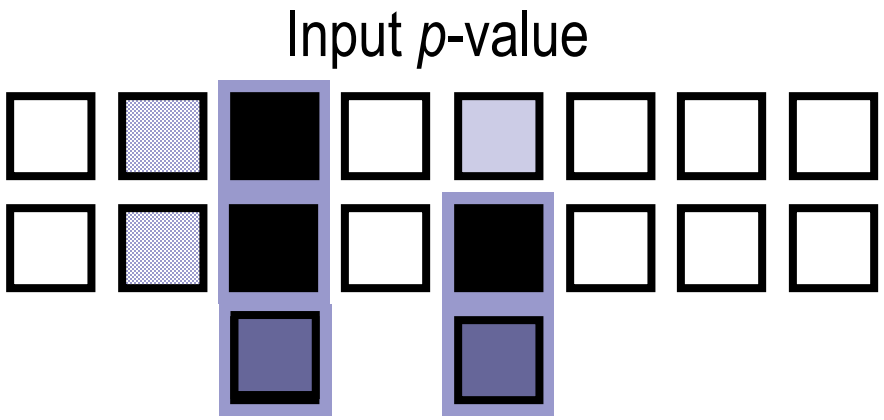

Entry Cutoff
Stay Cutoff

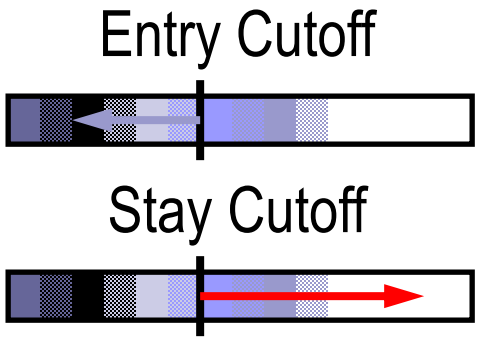
...


Input *p*-value

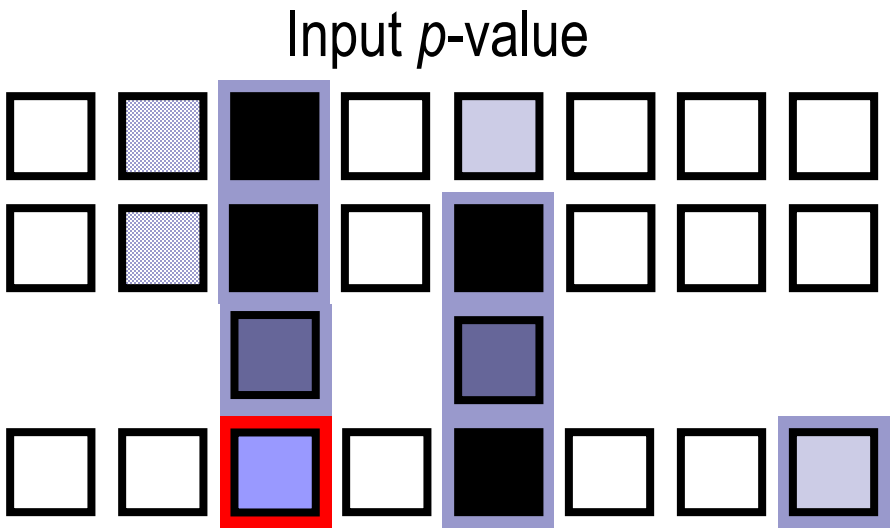

Entry Cutoff
Stay Cutoff

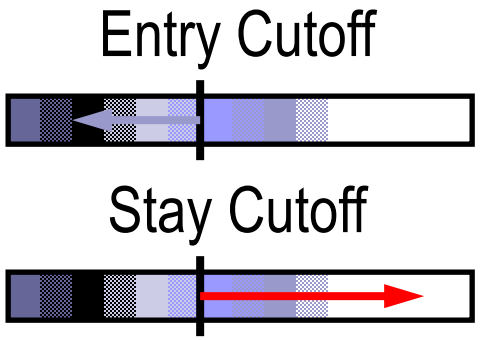
...


Input *p*-value

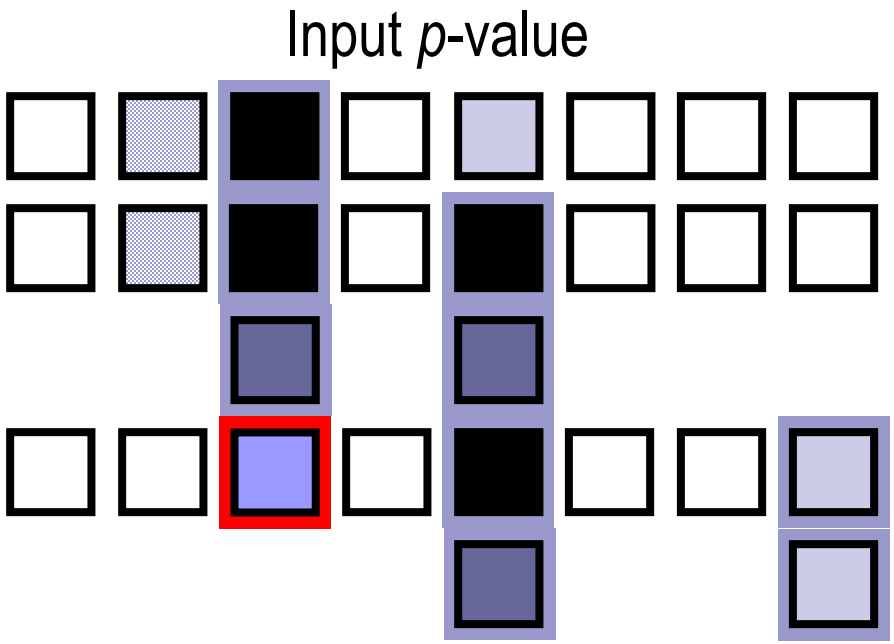

Entry Cutoff
Stay Cutoff

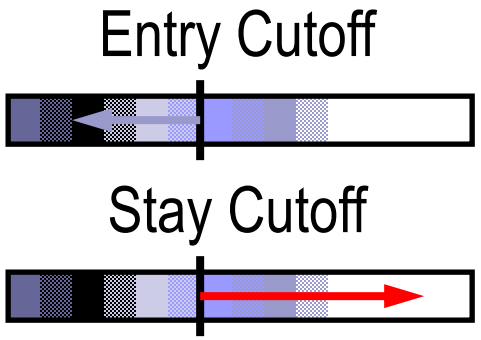
...


Input *p*-value

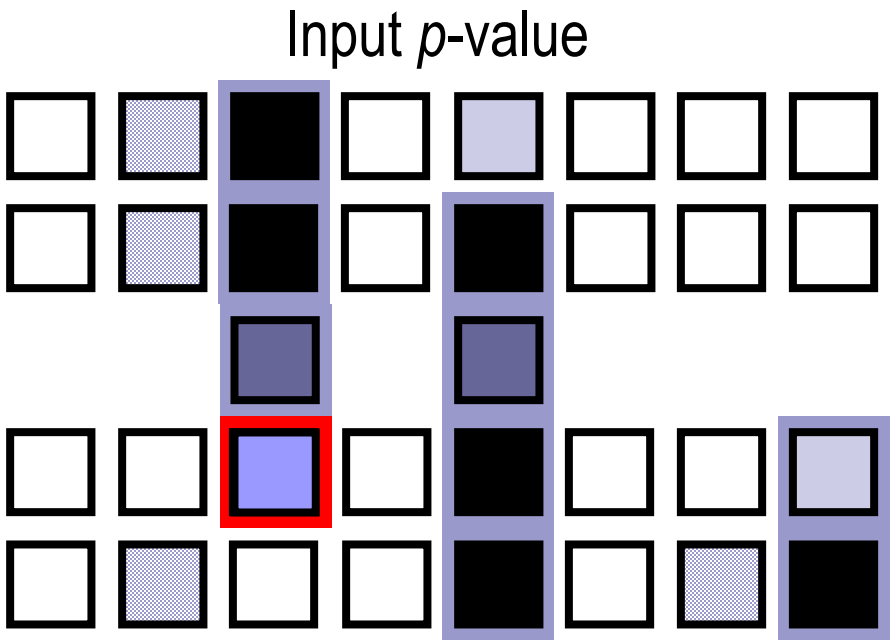

Entry Cutoff
Stay Cutoff

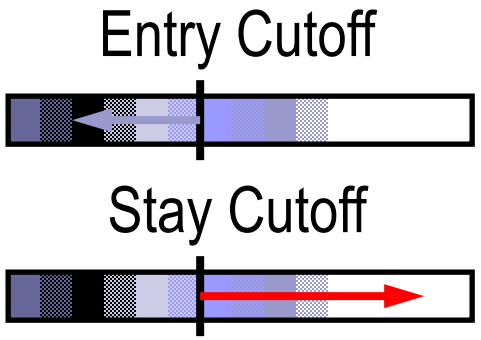
...


Input *p*-value

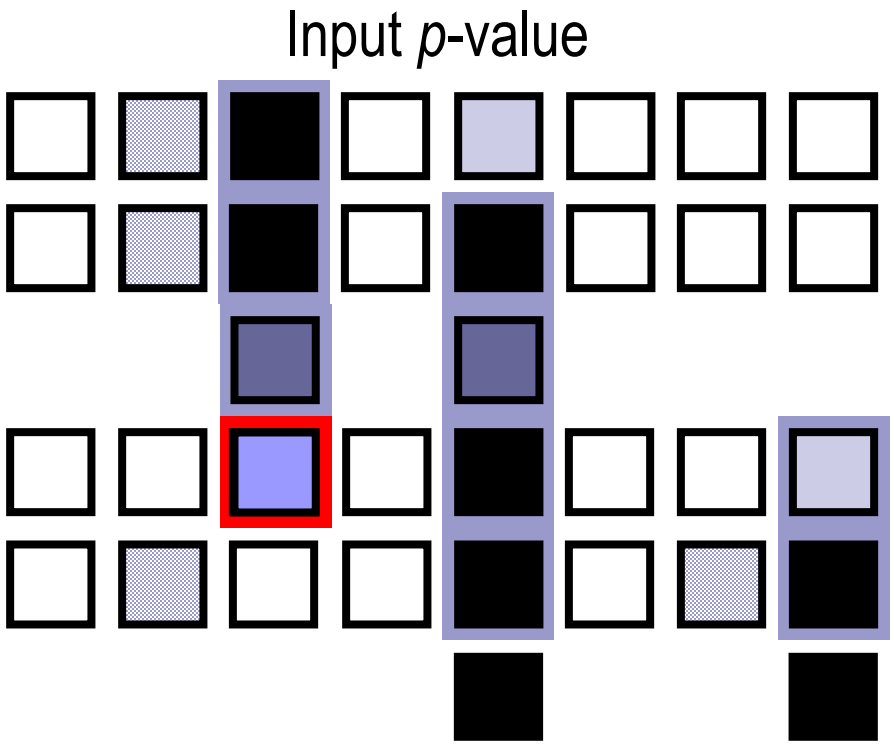

Entry Cutoff
Stay Cutoff

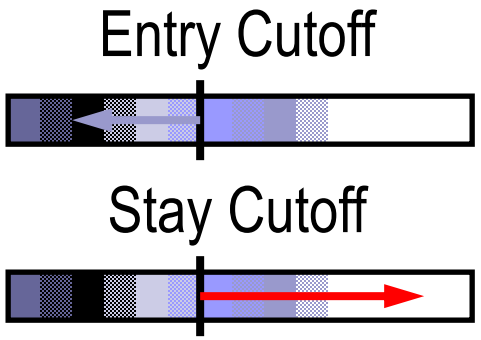
...


Input *p*-value


Entry Cutoff
Stay Cutoff

...


Input *p*-value


Entry Cutoff
Stay Cutoff

Пример прямой R^2

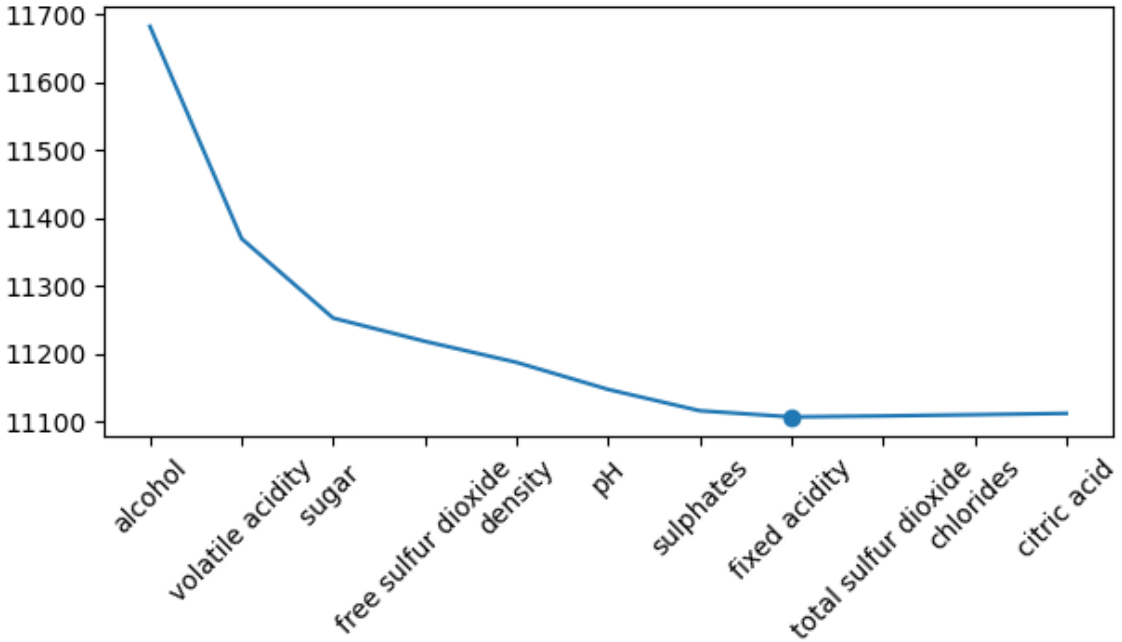
```python
def next_possible_feature(X, y, current_features,
                          by="rsquared", asc=False):
    feat_dict = {'feat': [], by:[]}
    for col in X.columns:
        if col not in (current_features):
            sub_X = X[current_features + [col]]
            sub_X = sm.add_constant(sub_X)
            model = sm.OLS(y, sub_X).fit()
            metric = getattr(model, by)
            feat_dict['feat'].append(col)
            feat_dict[by].append(metric)
    feat_df = pd.DataFrame(feat_dict)
    feat_df = feat_df.sort_values(by=[by],ascending=asc)
    best = feat_df.iloc[0].to_dict()
    return (best['feat'], best[by])
```

```python
# r^2
curr_feats = []
for i in range(len(X.columns)):
    print("="*10, "iteration:", i, "="*10)
    col, val = next_possible_feature(X, y, curr_feats,
                                     "rsquared", False)
    print("+", col, "->", val)
    curr_feats.append(col)
```

```
========== iteration: 0 ==========
+ alcohol -> 0.18972533274925663
========== iteration: 1 ==========
+ volatile acidity -> 0.2402311847533
========== iteration: 2 ==========
+ sugar  -> 0.25852615806597845
========== iteration: 3 ==========
+ free sulfur dioxide -> 0.2639942210
========== iteration: 4 ==========
+ density -> 0.2689515645655992
========== iteration: 5 ==========
+ pH -> 0.2751821150463122
========== iteration: 6 ==========
+ sulphates -> 0.2801195827165033
========== iteration: 7 ==========
+ fixed acidity -> 0.2817519637249508
========== iteration: 8 ==========
+ total sulfur dioxide -> 0.281835337
========== iteration: 9 ==========
+ chlorides -> 0.28186254437932134
========== iteration: 10 ==========
+ citric acid -> 0.2818703641332855
```

Пример прямой AIC

```
# AIC
curr_feats = []
vals = []
for i in range(len(X.columns)):
    col, val = next_possible_feature(X, y, curr_feats,
                                     "aic", True)
    curr_feats.append(col)
    vals.append(val)

ind_min = np.argmin(vals)
```



Пример SequentialFeatureSelector

```
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn import linear_model

lm = linear_model.LinearRegression()
sfs = SequentialFeatureSelector(lm,
        n_features_to_select=4,
        direction="forward",
        cv=3)
sfs.fit(X, y)
```

```
sfs.get_support()
```

```
array([False, False,  True, False,  True, False, False,  True,  True,
       False, False])
```

```
sfs.get_feature_names_out()
```

```
array(['alcohol', 'volatile acidity', 'sulphates', 'sugar '], dtype=object)
```

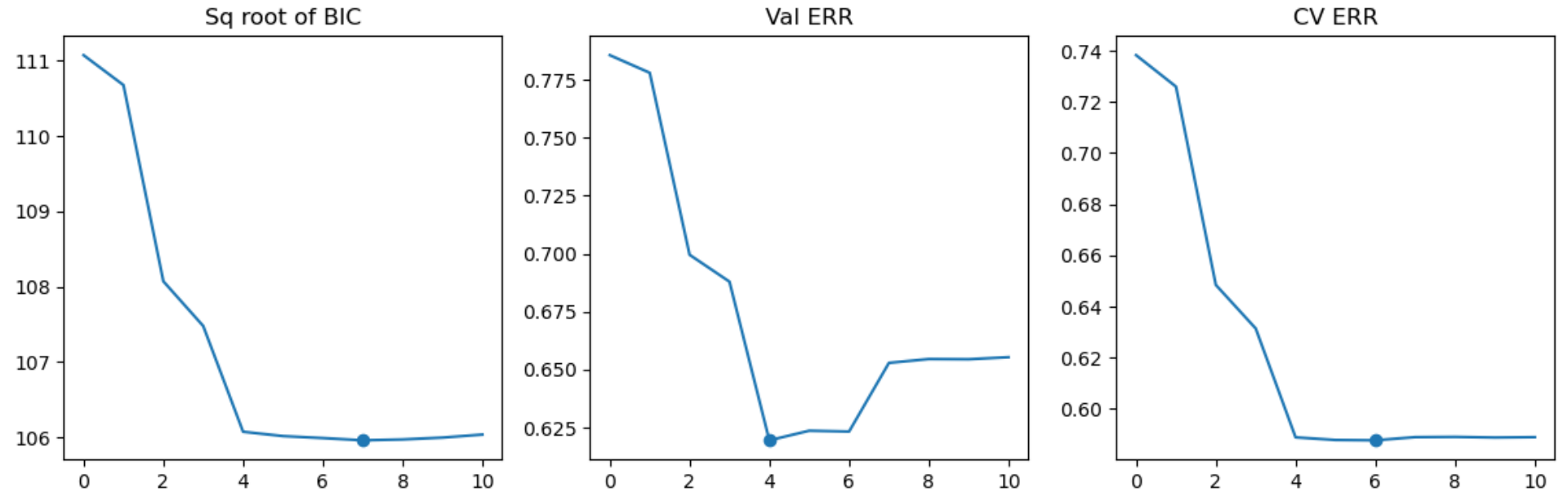
Пример

```python
# BIC
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import mean_squared_error

def get_full_bic(X, y):
    model = sm.OLS(y, X).fit()
    return model.bic

def get_folds_error(X, y, folds):
    kf = StratifiedKFold(n_splits=folds)
    res_folds = []
    for tr_ind, tst_ind in kf.split(X, y):
        model = sm.OLS(y[tr_ind],
                       X.iloc[tr_ind]).fit()
        y_pred = model.predict(X.iloc[tst_ind])
        error = mean_squared_error(y[tst_ind], y_pred)
        res_folds.append(error)
    return np.mean(res_folds)
```

```python
full_bic = []
val_err = []
cv_err = []

range_feats = range(len(curr_feats))
for i in range_feats:
    sub_X = X[curr_feats[:i+1]]
    full_bic.append(get_full_bic(sub_X, y))
    val_err.append(get_folds_error(sub_X, y, 2))
    cv_err.append(get_folds_error(sub_X, y, 5))
```

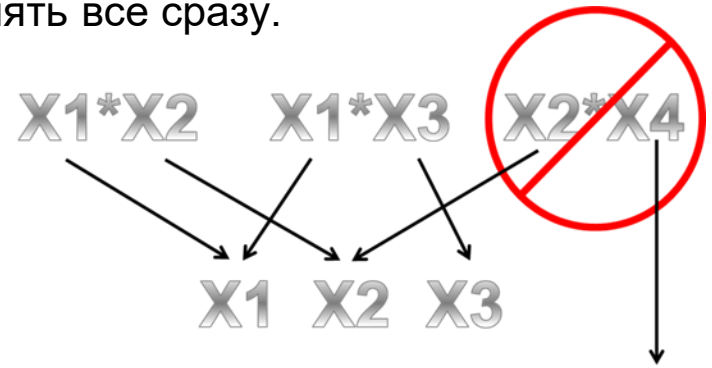
Пример

```
fig, axes = plt.subplots(ncols=3, figsize=(14, 4))

metr = [np.sqrt(full_bic), val_err, cv_err]
names = ["Sq root of BIC", "Val ERR", "CV ERR"]
for i, (l, n) in enumerate(zip(metr, names)):
    ind_min = np.argmin(l)
    axes[i].set_title(n)
    axes[i].plot(range_feats, l)
    axes[i].scatter(range_feats[ind_min], l[ind_min])
plt.show()
```



Иерархия признаков

- Иногда может иметь место ситуация:
  - член регрессионного уравнения, отражающий взаимодействие переменных является важным, но связанные с ним основные эффекты не являются важными
- Принцип иерархии:
  - *Если мы включаем взаимосвязь в модель, мы должны также включать базовые признаки, даже если значения p-value, связанные с их коэффициентами, не показывают значимость.*
  - Влияет на правило выбора в пошаговых методах, например, не добавлять/удалять без основных эффектов, или наоборот, добавлять/удалять все сразу.



Мотивация – улучшение интерпретируемости

Валидация, кросс-валидация, бутстреппинг

- Имеет преимущество при выборе лучшей модели по сравнению с AIC, BIC, C_p и скорректированным R^2, т.к. обеспечивает непосредственную оценку тестовой ошибки, и *не требует оценки дисперсии ошибки*
- Также могут быть использованы в более широком диапазоне задач выбора модели, даже в случаях, когда трудно точно определить число степеней свободы модели (например, для непараметрических методов) или трудно оценить дисперсию ошибок