

3. Процесс решения задачи машинного обучения.

Содержание

- Виды признаков. Нормирование. Фильтрация выбросов.
- Отбор и конструирование признаков.
- Оптимизация гиперпараметров.
- Обобщенный процесс решения задачи машинного обучения.
- Развёртывание моделей ML
- Low code платформы: RapidMiner, Orange, Knime.
- Практические задачи ML

Типы признаков



Типы признаков

- Категориальные / номинальные:
 - Конечное множество
 - Семейное положение: {Холост, Женат, Разведен, Вдовец}
 - Нет отношения порядка
 - Не можем вычислить расстояние
- Порядковые:
 - Конечное упорядоченное множество
 - Рост: {Низкий, Средний, Высокий}
 - Существует отношение порядка: Низкий < Средний < Высокий
 - Не можем вычислить расстояние
- Количественные:
 - Множество действительных чисел (конечное или неограниченное)
 - Рост: {1.55, 1.68, 1.72, 1.80}
 - Существует отношение порядка: $1.55 < 1.68 < 1.72 < 1.80$
 - Расстояние вычисляется $\|1.72 - 1.55\| = \text{SQRT}(1.72^{**2} - 1.55^{**2})$

Категориальные признаки

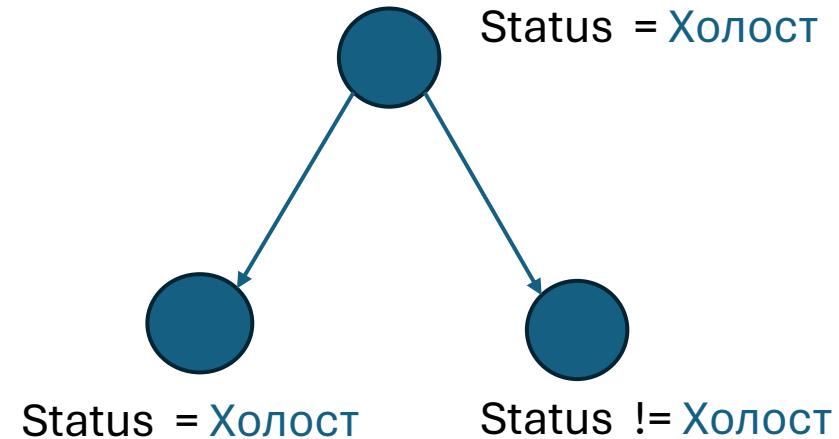
- Конечное множество
- Status: {Холост, Женат, Разведен, Вдовец}

Линейная регрессия

$$Y = a + b * \text{Status} + \dots$$

Мы можем положить Холост = 0, Женат = 1, Разведен = 2, ..., но это означает, что разведенные в 2 раза дальше от холостых, чем женатые.

Decision Tree



Категориальные признаки - кодирование

Стандартный подход: dummy (фиктивные) переменные

Income	Age	Marital Status
\$45,000	23	Single
\$48,000	25	Single
\$54,000	24	Single
\$57,000	29	Single
\$65,000	38	Married
\$69,000	36	Single
\$78,000	40	Married
\$83,000	59	Divorced
\$98,000	56	Divorced
\$104,000	64	Married
\$107,000	53	Married



Income	Age	Single	Married	Divorced
\$45,000	23	1	0	0
\$48,000	25	1	0	0
\$54,000	24	1	0	0
\$57,000	29	1	0	0
\$65,000	38	0	1	0
\$69,000	36	1	0	0
\$78,000	40	0	1	0
\$83,000	59	1	0	1
\$98,000	56	1	0	1
\$104,000	64	0	1	0
\$107,000	53	0	1	0

Порядковые признаки

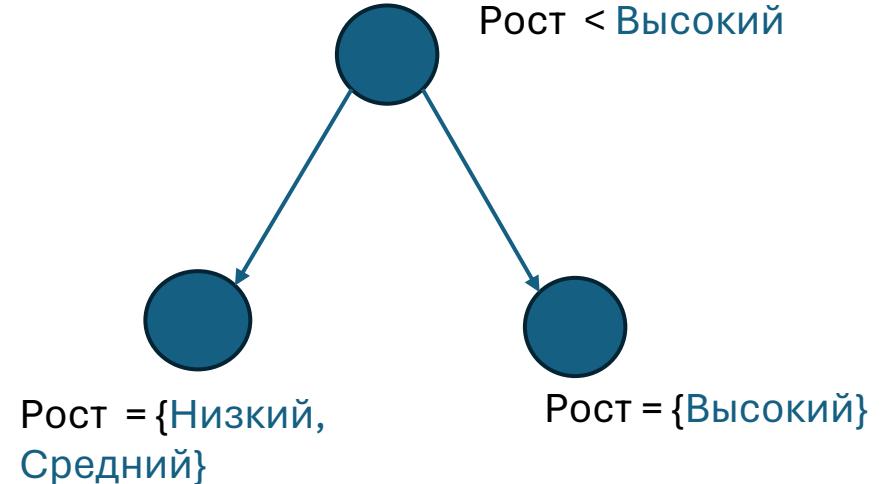
- Конечное упорядоченное множество
- Рост: {Низкий, Средний, Высокий}

Линейная регрессия

$$Y = a + b * \text{Рост} + \dots$$

Мы можем положить Низкий = 0, Средний = 1, Высокий = 2, но это означает, что расстояние высоких от низких в два раза выше, чем от средних.

Decision Tree



Порядковые признаки и регрессия

Качественная (не количественная!) оценка данных: $y = a + b * x$

Национальность	C1	C2	C3
Французы	0	0	0
Итальянцы	1	0	0
Немцы	0	1	0
Прочие	0	0	1

Фиктивное кодирование – категории изучаются относительно контрольной группы, b разница между средними экспериментальной контрольной групп

Национальность	C1	C2	C3
французы	0	0	1
итальянцы	1	0	0
немцы	0	1	0
прочие	-1	-1	-1

Кодирование влияния – нет контрольной группы, каждая группа сравнивается с общим средним.

Национальность	C1	C2
французы	+0,33	+0,50
итальянцы	+0,33	-0,50
немцы	-0,66	0

Контрастное кодирование – кодирование на основе статистических гипотез.

Нормирование признаков

Набор данных

#	Рост (м)	Вес (кг)
0	1.55	57
1	1.78	75
2	1.92	102

Расстояние между объектами

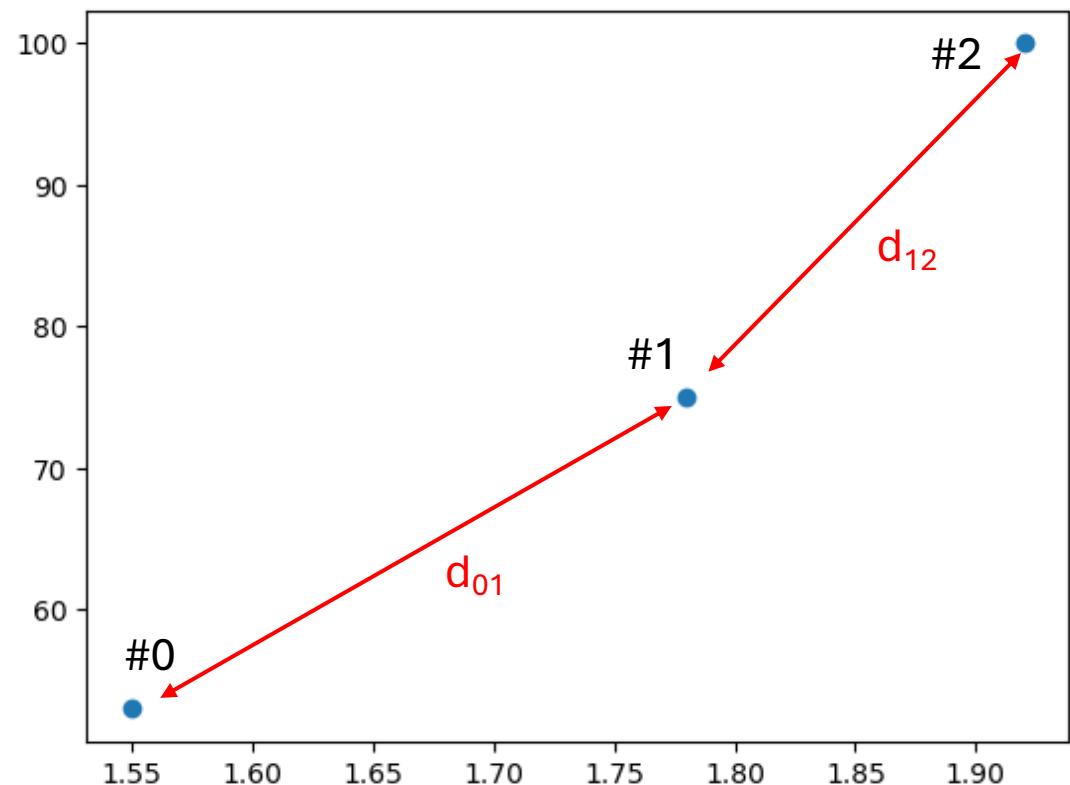
	0	1	2
0		22.0	47.0
1			25.0
2			



Объект #1 ближе всего к объекту #0

$$d_{01} < d_{12}$$

На самом деле $d_{12} < d_{01}$



Нормирование признаков

Набор данных

#	Рост (м)	Вес (кг)
0	1.55	57
1	1.78	75
2	1.92	102

Расстояния между объектами

	0	1	2
0		22.0	47.0
1			25.0
2			

Объект #1 ближе всего к объекту #0



Z-трансформация

#	Рост	Вес
0	-1.31	-1.20
1	0.20	-0.05
2	1.11	1.24

	0	1	2
0		1.89	3.45
1			1.59
2			

Объект #1 ближе всего к объекту #2

Нормирование признаков

Нормализация — это преобразование данных к неким безразмерным единицам, которая концептуально сводится к формуле $x'_j = \frac{x_j - a}{b}$. Значения x сначала смещаются на a , потом масштабируются по b .

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Минимаксная нормализация: [sklearn.preprocessing.MinMaxScaler](#)

$$x' = \frac{x - \bar{x}}{\sigma_x}$$

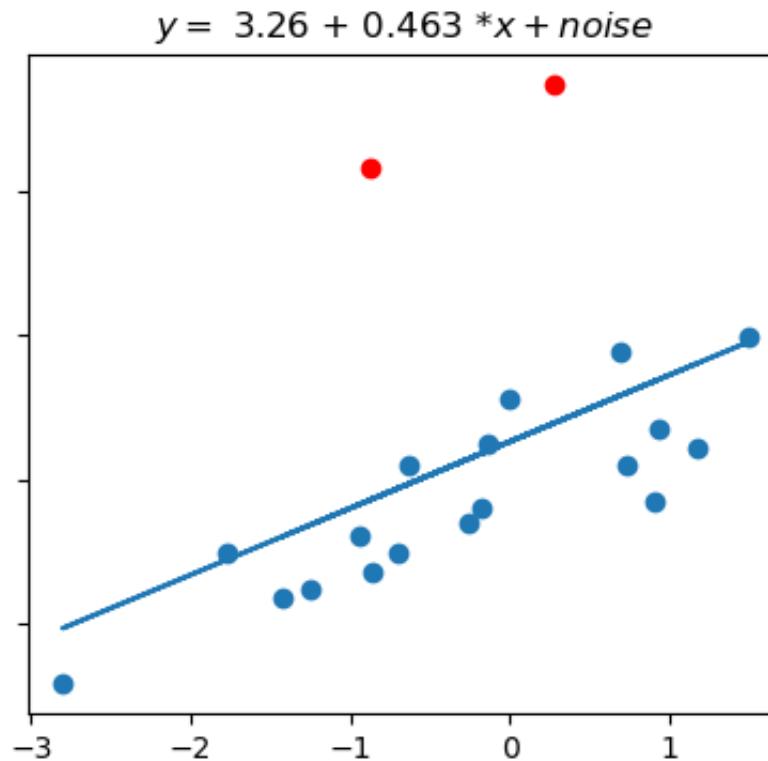
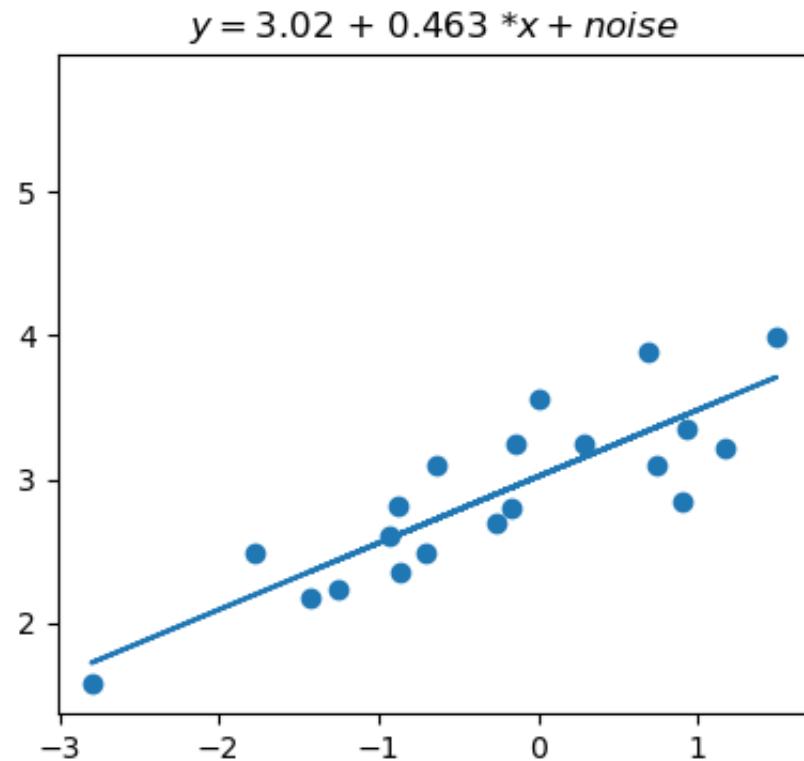
Z - нормализация: [sklearn.preprocessing.StandardScaler](#)

```
from sklearn.preprocessing import StandardScaler

data = [[0, 0], [0, 0], [1, 1], [1, 1]]

scaler = StandardScaler()
scaler.fit(data)
data_transformed = scaler.transform(data)
```

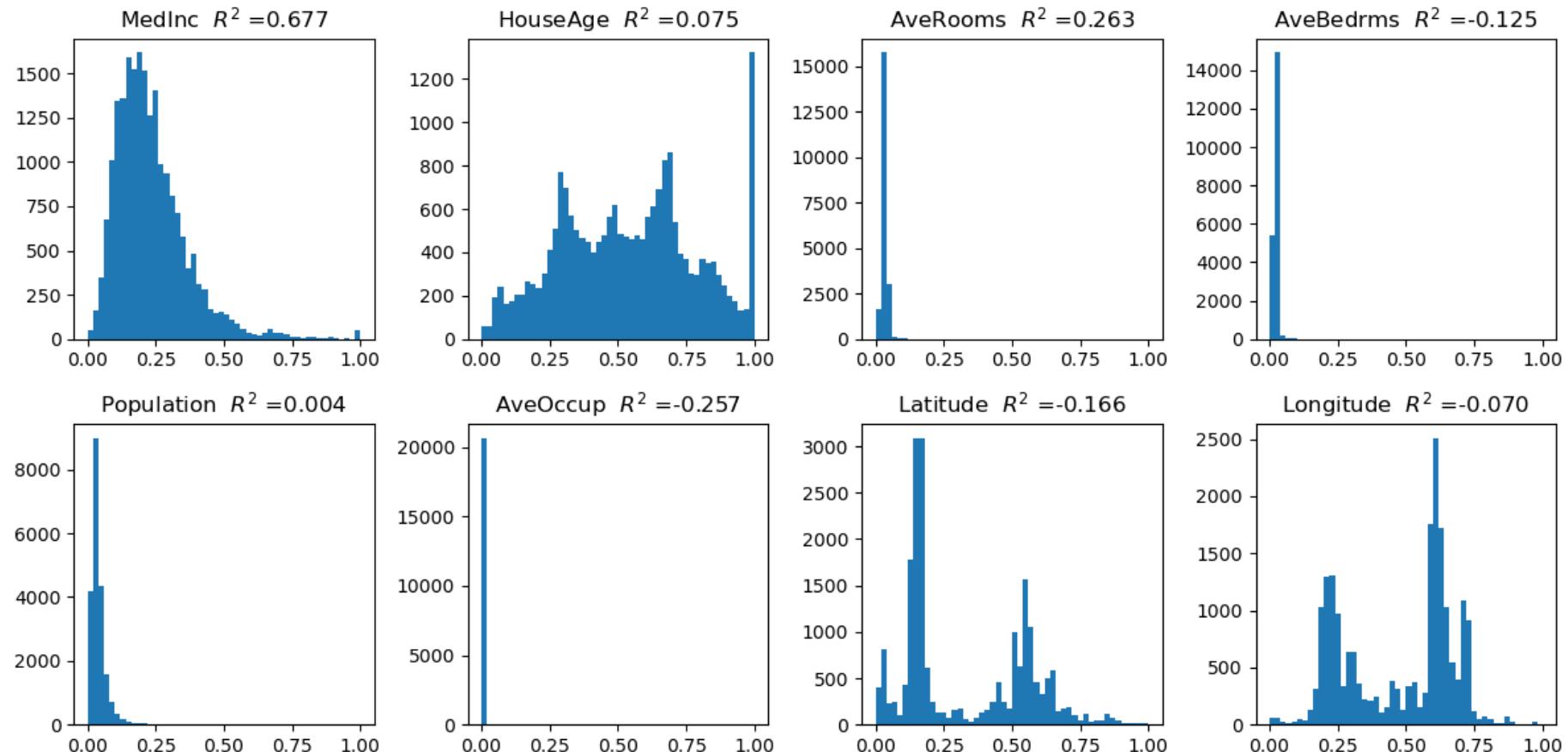
Выбросы (outliers)



True distribution: $y = 3.0 + 0.5 * x + noise$

Как понять, есть ли выбросы в данных?

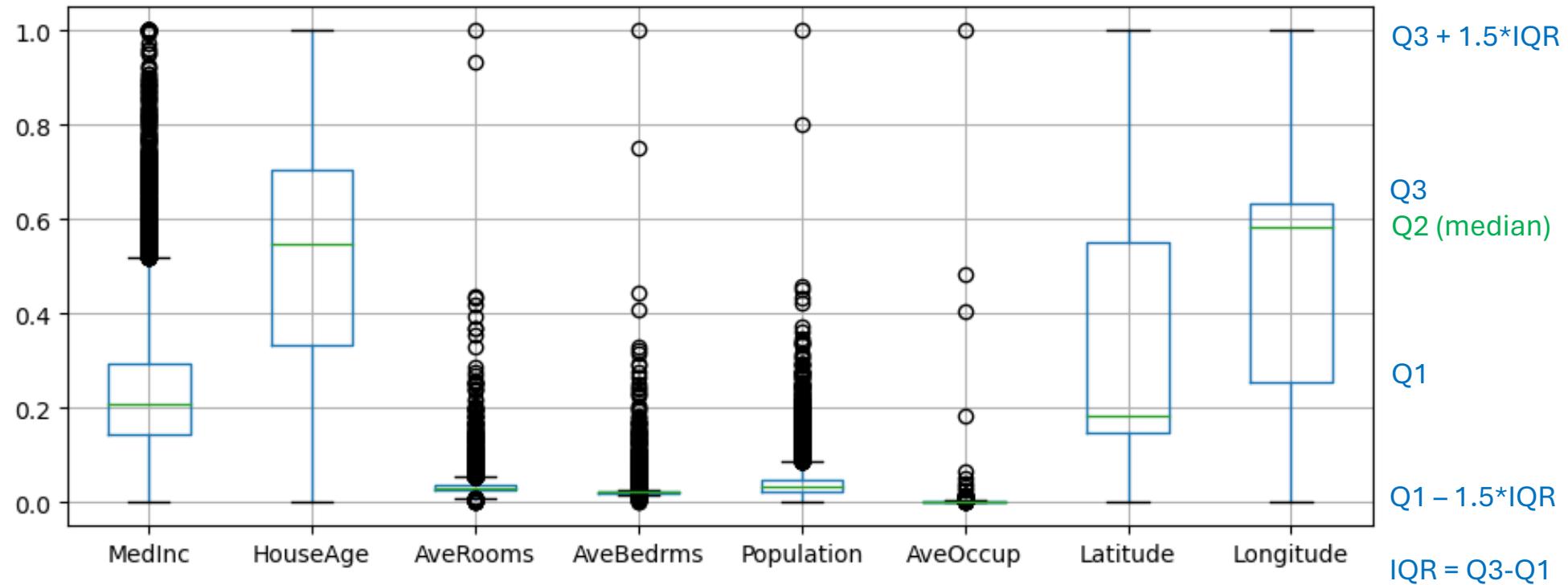
California Housing
Гистограммы
признаков



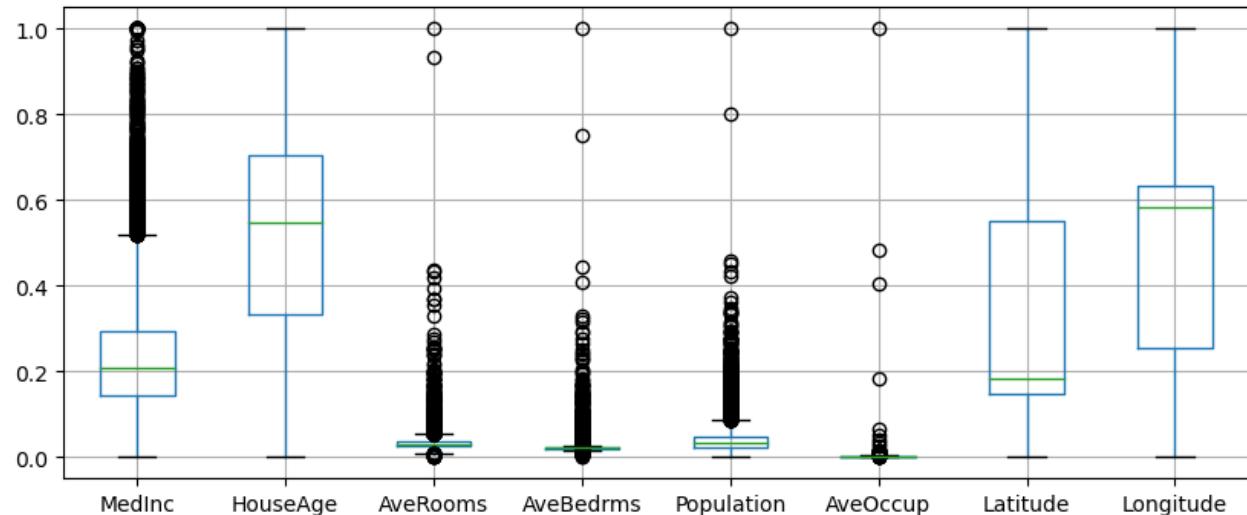
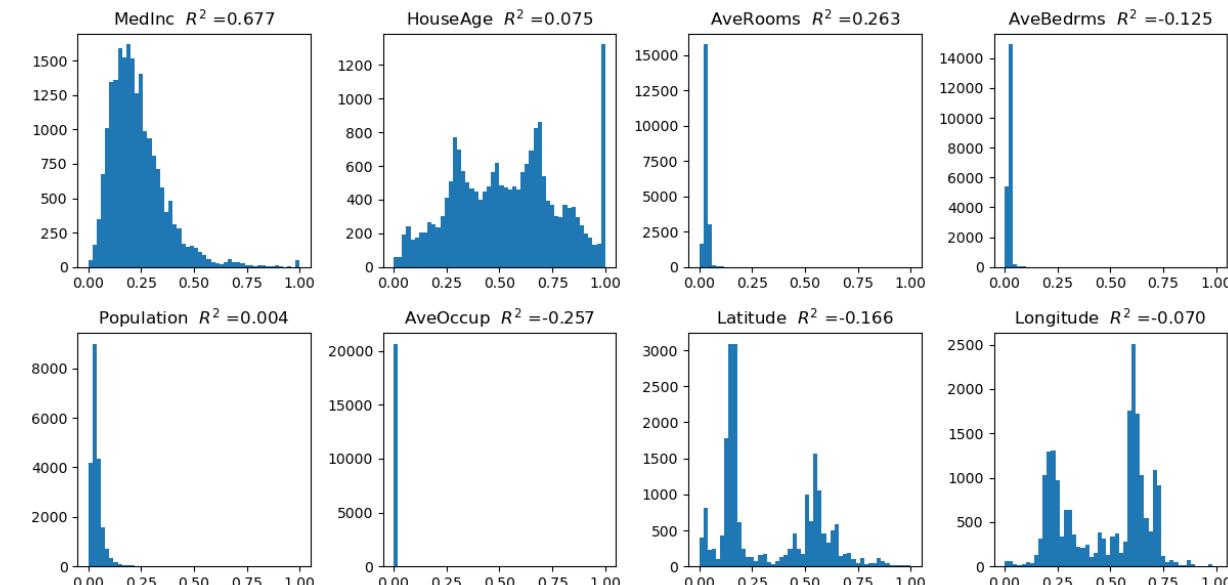
Как понять, есть ли выбросы в данных?

California Housing

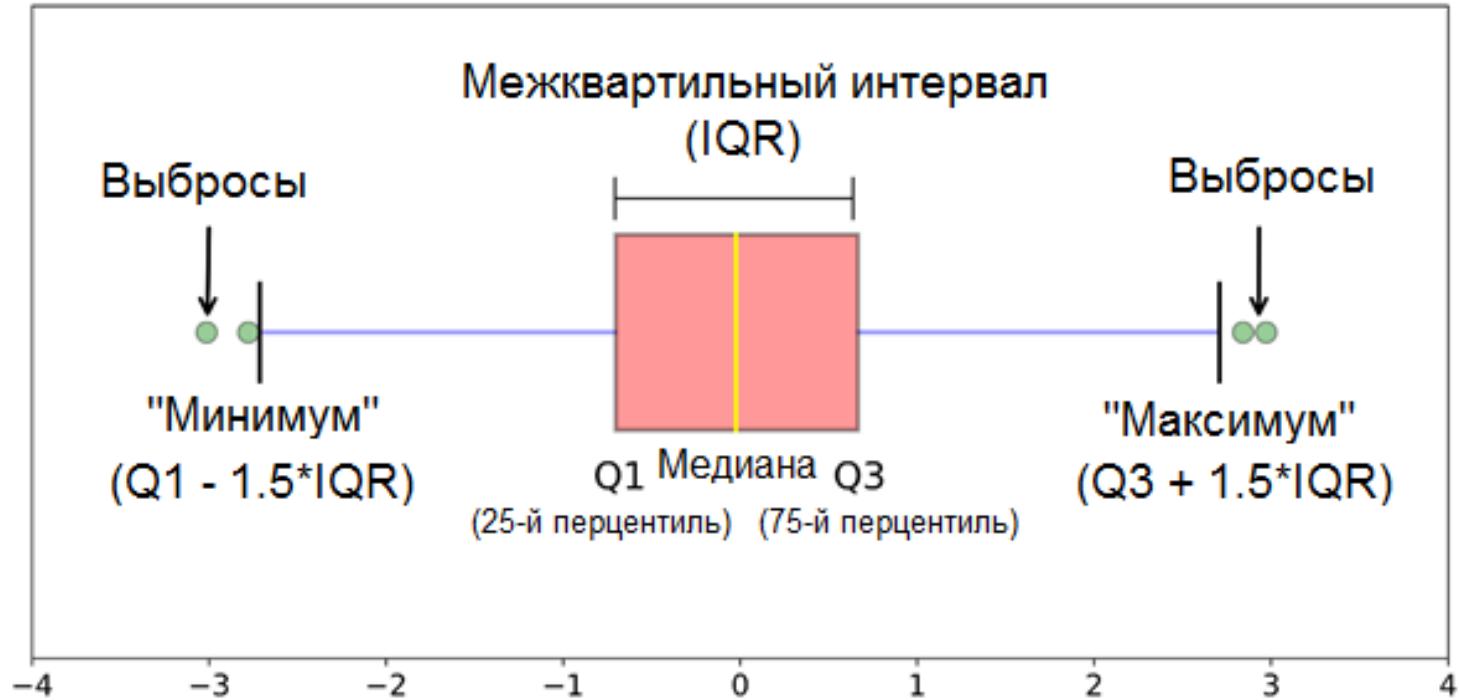
Ящик с усами
(box-and-whisker)



Посмотрим на диаграммы вместе

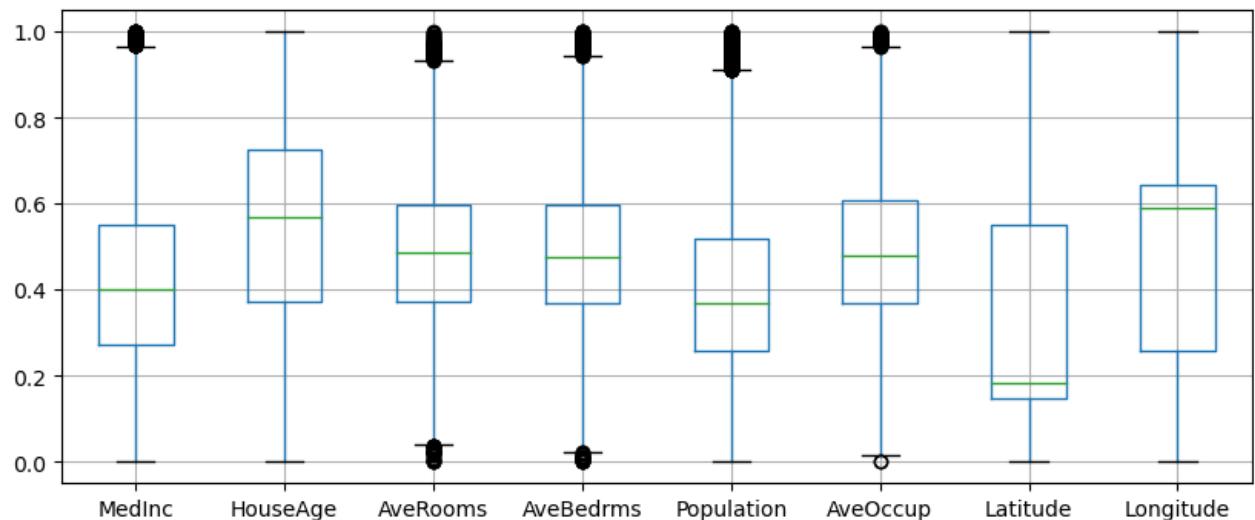
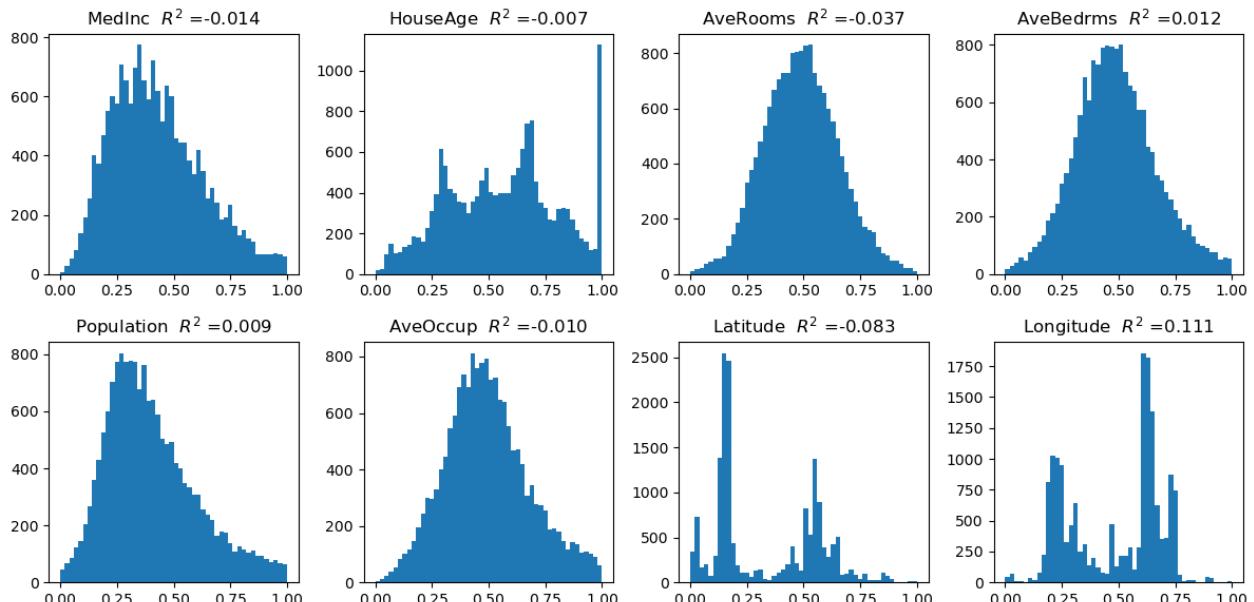


Фильтрация выбросов



```
def find_outliers_IQR(df):
    q1 = df.quantile(0.25)
    q3 = df.quantile(0.75)
    IQR = q3 - q1
    outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]
    outliers = outliers.isna()
    return outliers
```

Удаляем выбросы



Обучаемся на очищенных данных ($c_{\text{v}} = 5$)

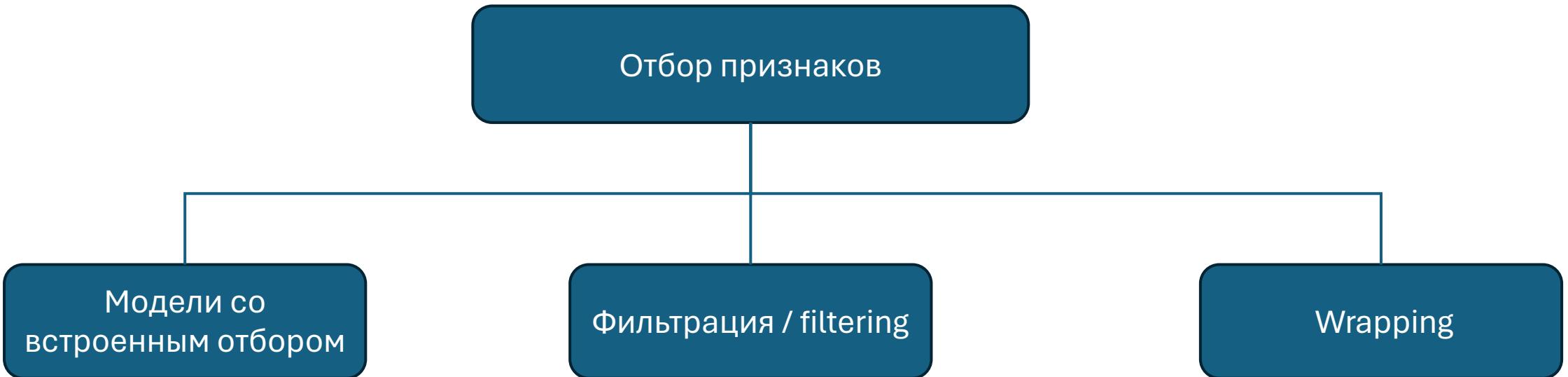
Данные с выбросами (лекция 2)

	Decision Tree	Random Forest	Gradient Boosting
MSE	0.609 (0.064)	0.432 (0.068)	0.439 (0.113)

Очищенные данные

	Decision Tree	Random Forest	Gradient Boosting
MSE	0.596 (0.081)	0.416 (0.053)	0.405 (0.034)

Отбор и конструирование признаков



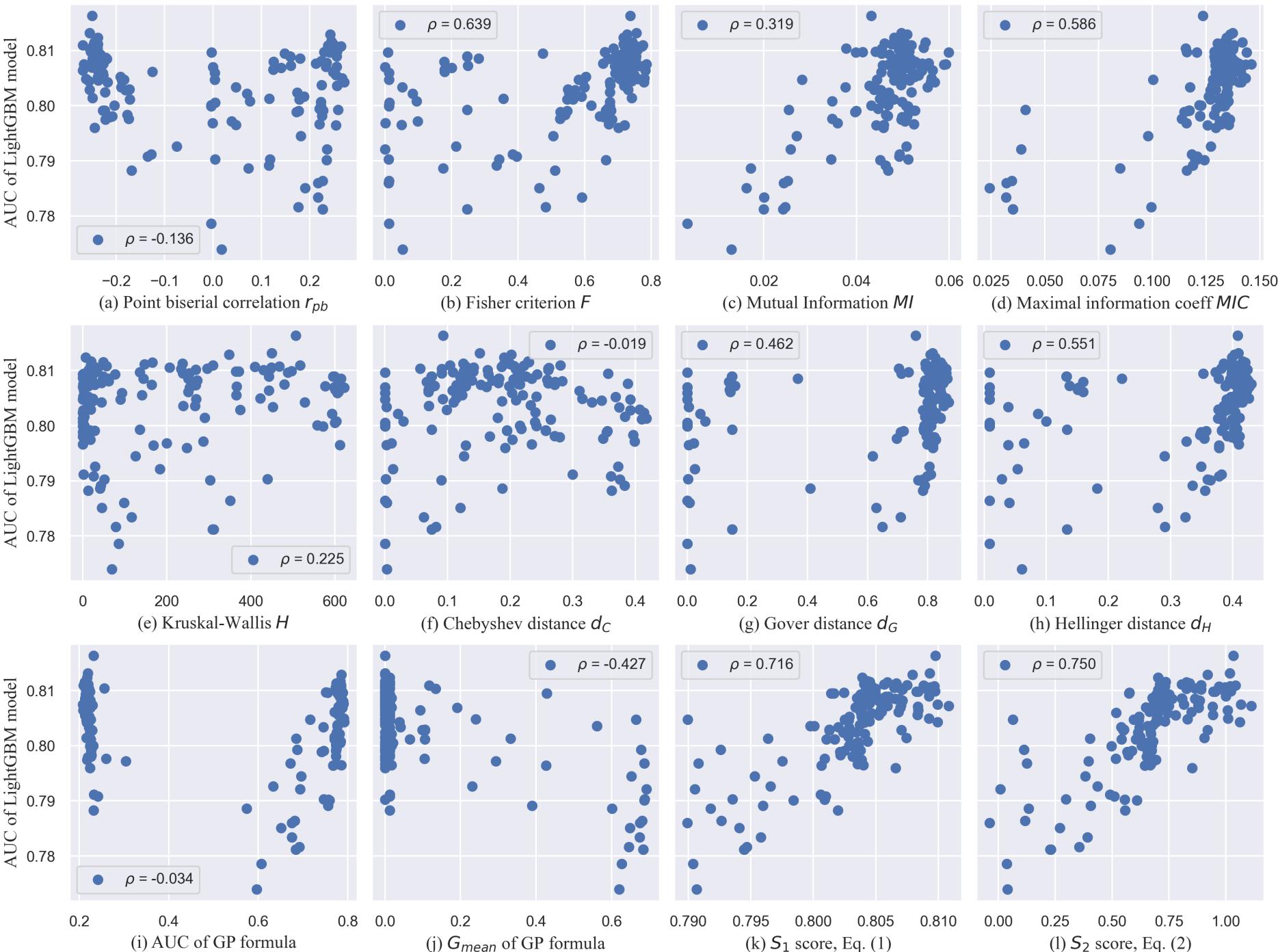
$$J_{LASSO} = \sum_i (y_i - \hat{y})^2 + \lambda ||w||$$

$$J_{RIDGE} = \sum_i (y_i - \hat{y})^2 + \lambda w^2$$

Признак оценивается с помощью метрики, как правило, измеряющей его связь с зависимой переменной (y).

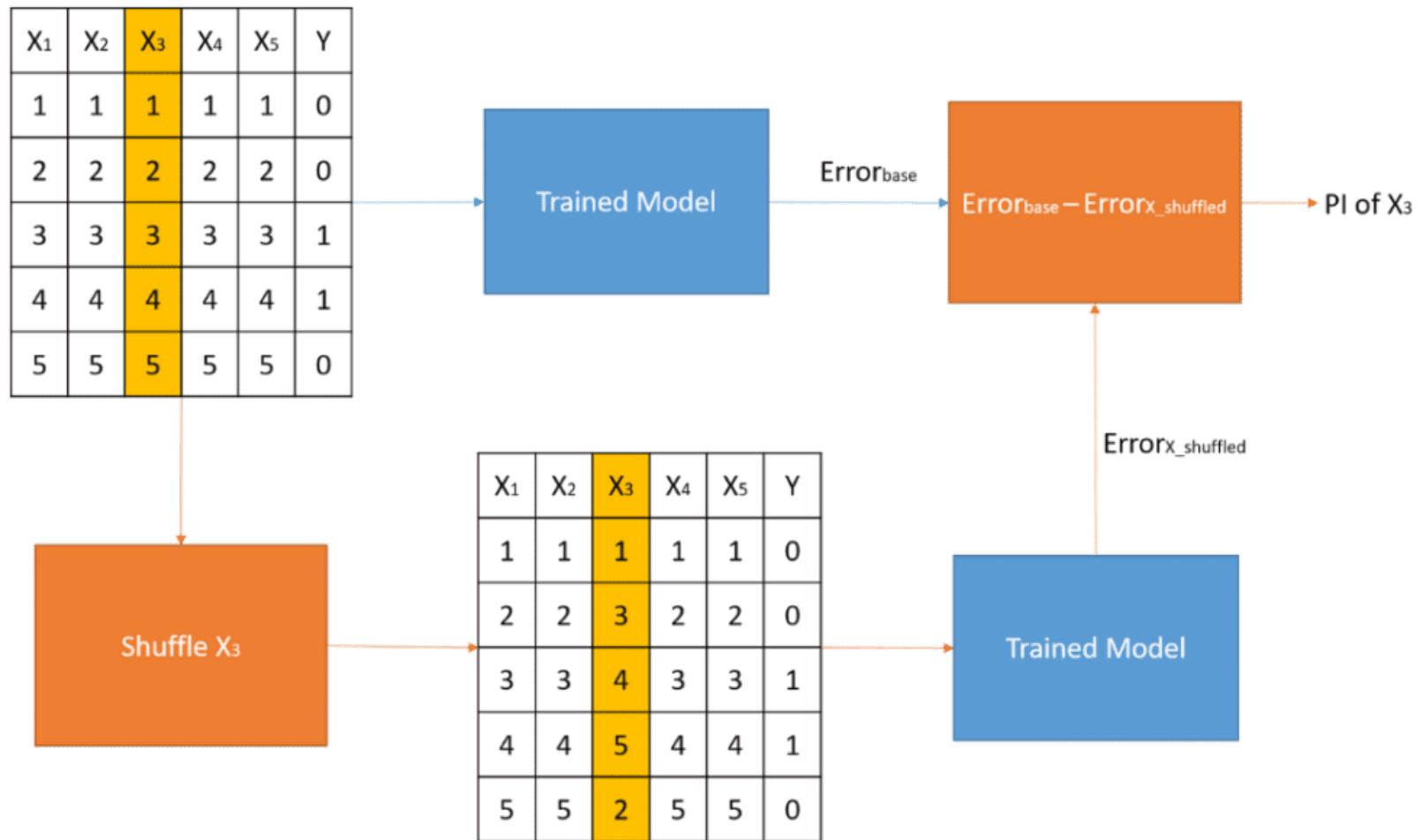
Модель обучается с признаком и без, оценивается потеря точности.

Filtering

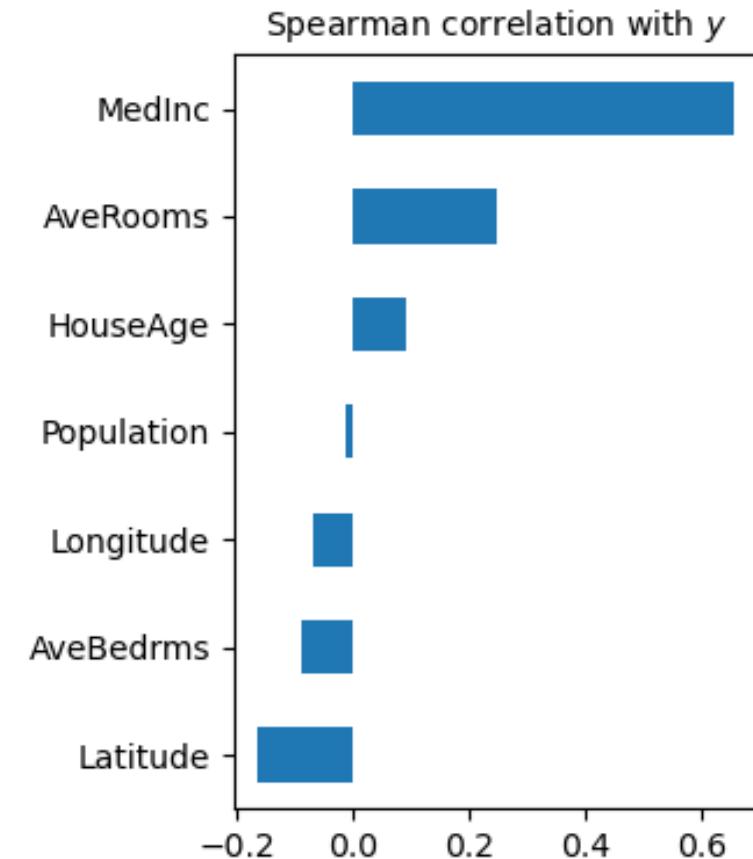
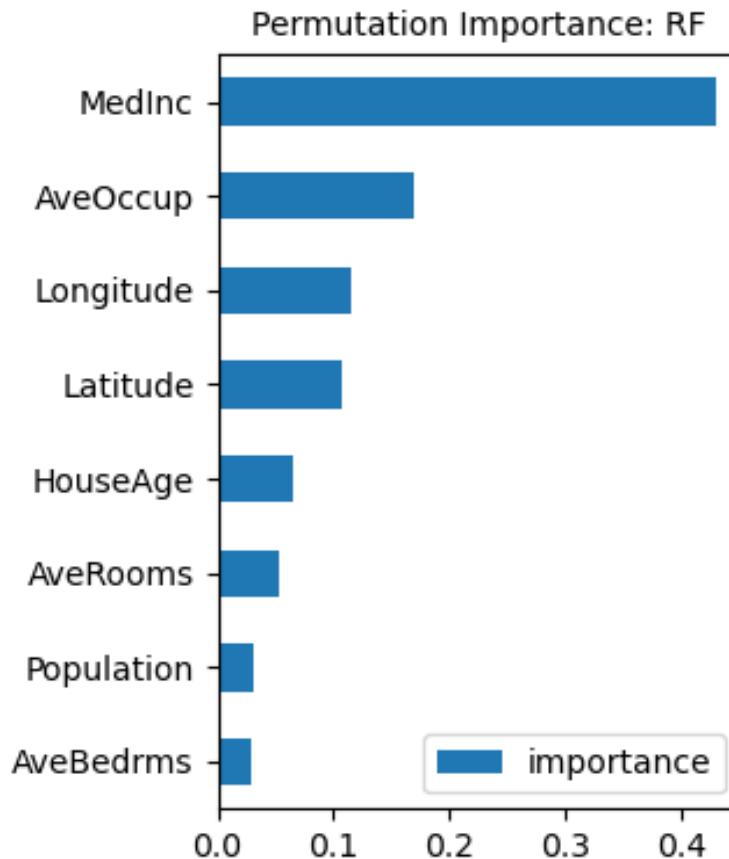
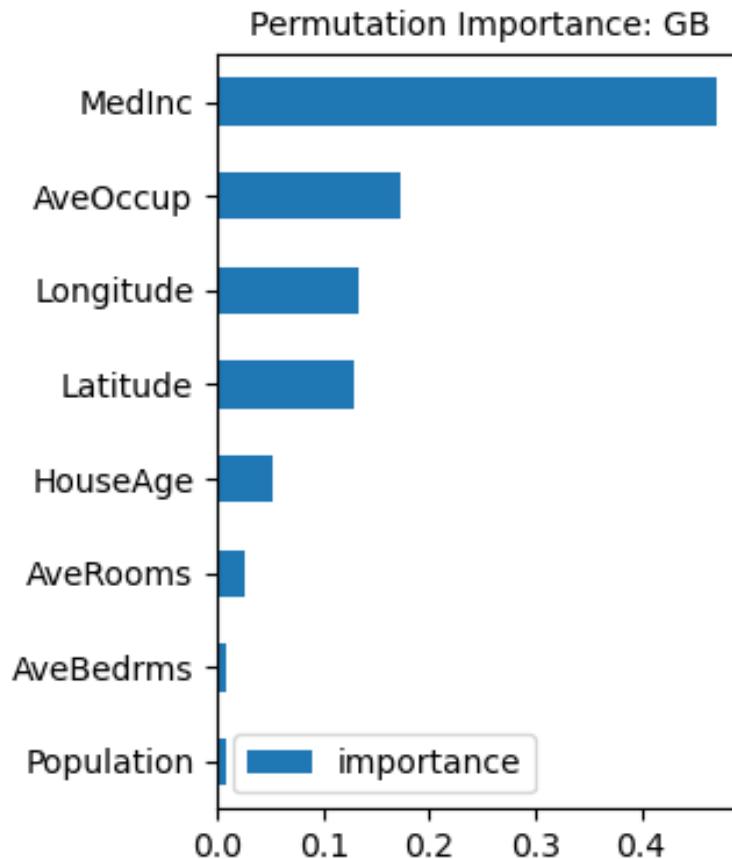


Source: Zelenkov, Y. (2024) Firm failure prediction using genetic programming generated features. (*in press*).

Важность признаков: Permutation Importance

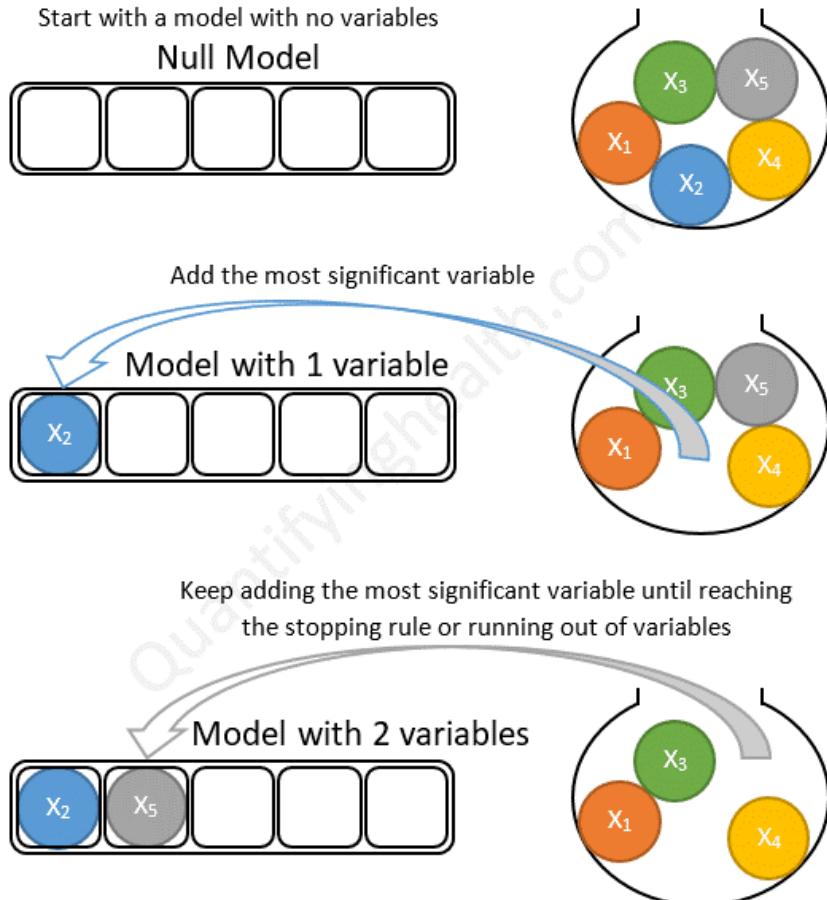


Важность признаков

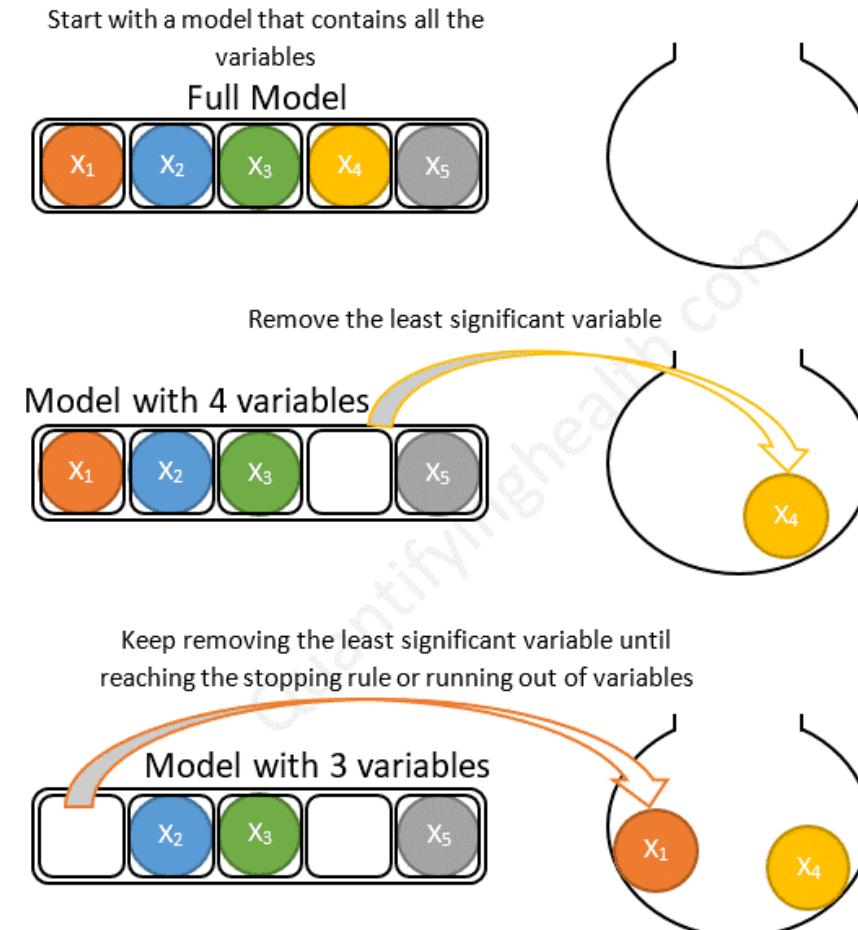


Forward and Backward Feature Selection

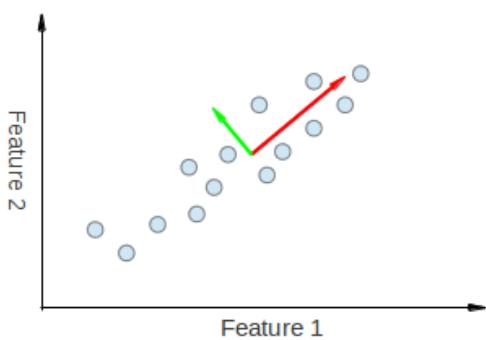
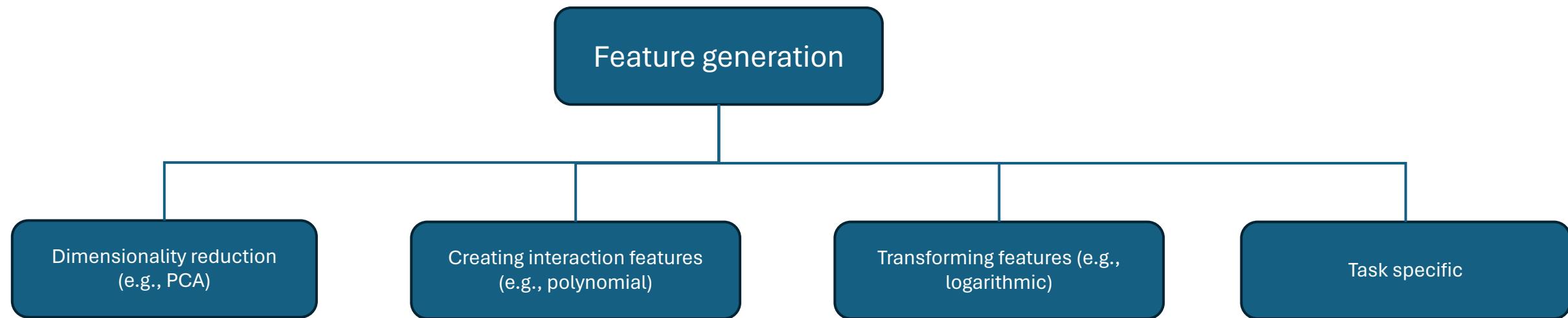
Forward stepwise selection example with 5 variables:



Backward stepwise selection example with 5 variables:



Генерация признаков



$$\begin{aligned} \mathcal{D} &= \{x_1, x_2\} \\ &\downarrow \\ \mathcal{D}' &= \{x_1, x_2, x_1 x_2, x_1^2, x_2^2\} \end{aligned}$$

$$x' = \log(x + 1)$$

Локомотив - Урал	2:0
Спартак - Крылья Советов	3:0
Ахмат - Ростов	0:0
Зенит - Пари НН	1:0
Факел - Динамо	1:1

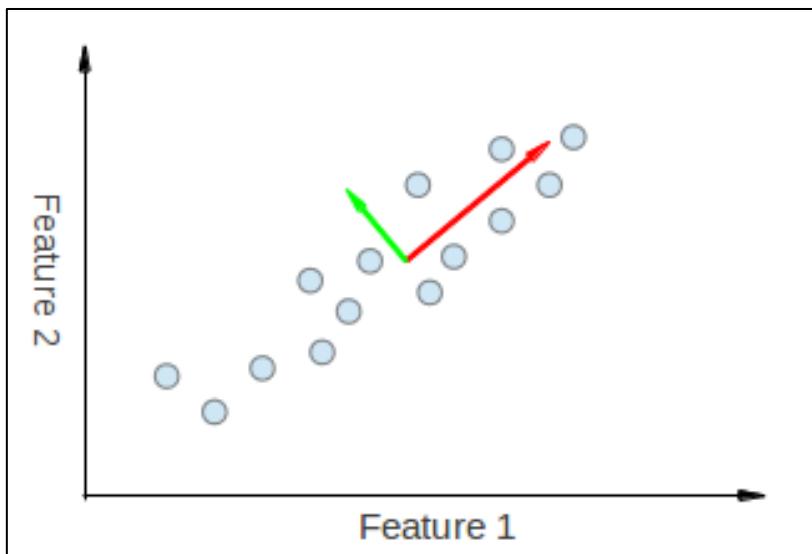


Клуб	Рейтинг
Зенит	4.32
Спартак	3.98
Локомотив	3.12

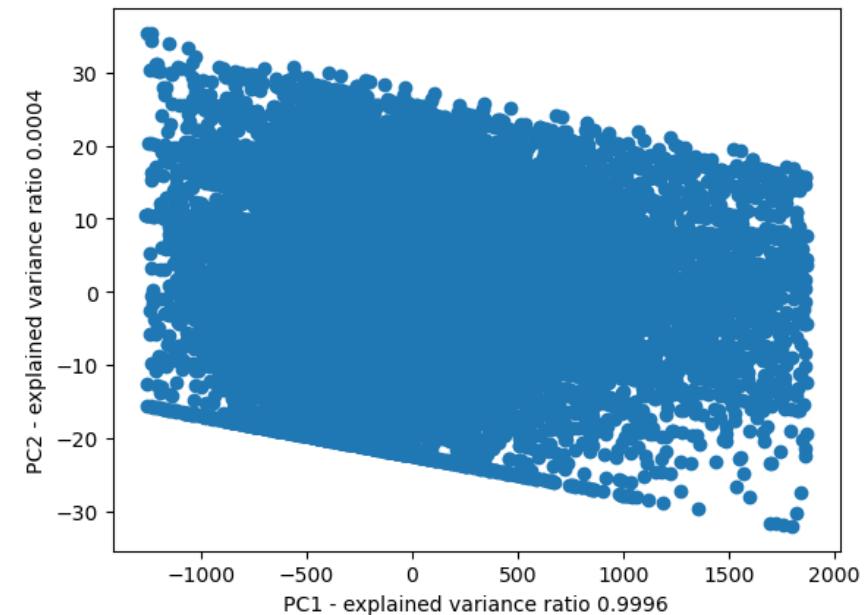
Метод главных компонент (PCA)

```
from sklearn.decomposition import PCA  
  
pca = PCA(n_components = 2)  
pca.fit(X_clean)  
X_pca = pca.transform(X_clean)
```

Principle of transformation



Two first PC of California Housing dataset



Полиномиальные признаки

```
from sklearn.preprocessing import PolynomialFeatures  
  
poly = PolynomialFeatures(degree = 2)  
poly.fit(data)  
data_ext = poly.transform(data)
```

Создает новое множество признаков, состоящее из всех полиномиальных комбинаций со степенью меньше или равной заданной.

Например, если входная выборка двумерна и имеет вид $[a, b]$, то полиномиальные признаки степени 2 будут иметь вид $[1, a, b, a^2, ab, b^2]$.

Очищенные данные (8 признаков)

	Decision Tree	Random Forest	Gradient Boosting
MSE	0.596 (0.081)	0.416 (0.053)	0.405 (0.034)

Данные после PCA трансформации (6 признаков)

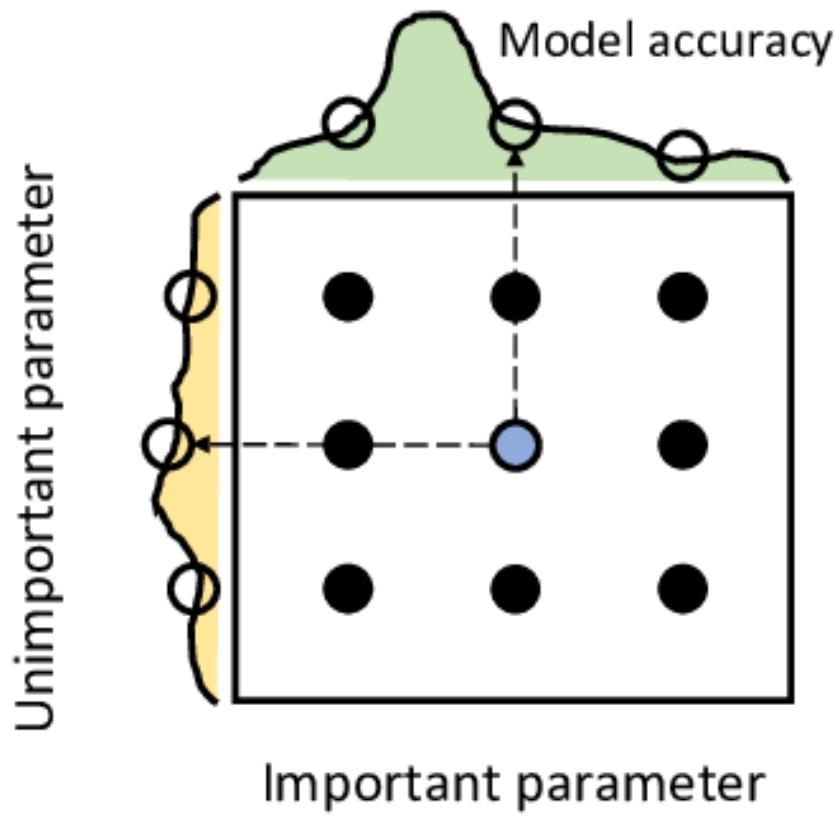
	Decision Tree	Random Forest	Gradient Boosting
MSE	0.526 (0.052)	0.448 (0.038)	0.434 (0.044)

Данные с полиномиальными признаками (45 признаков)

	Decision Tree	Random Forest	Gradient Boosting
MSE	0.599 (0.085)	0.403 (0.048)	0.419 (0.093)

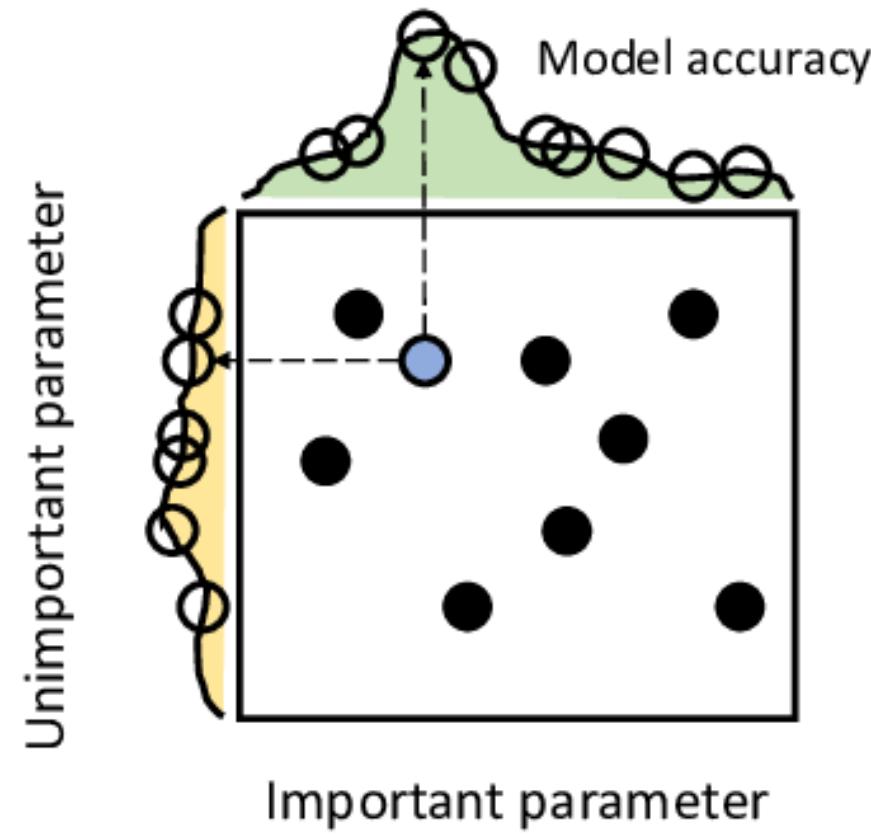
Оптимизация гиперпараметров

Grid Search



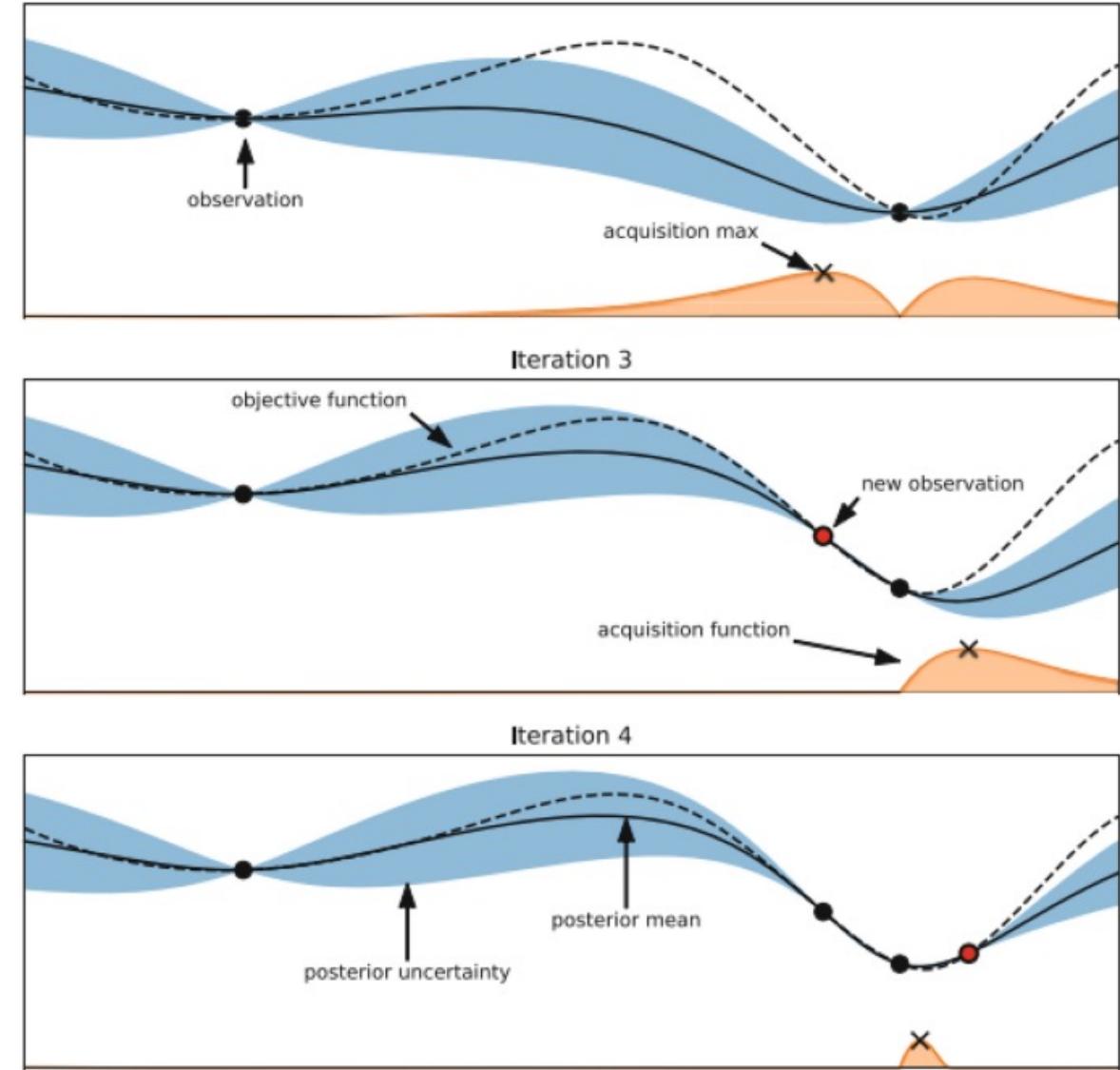
(a)

Random Search

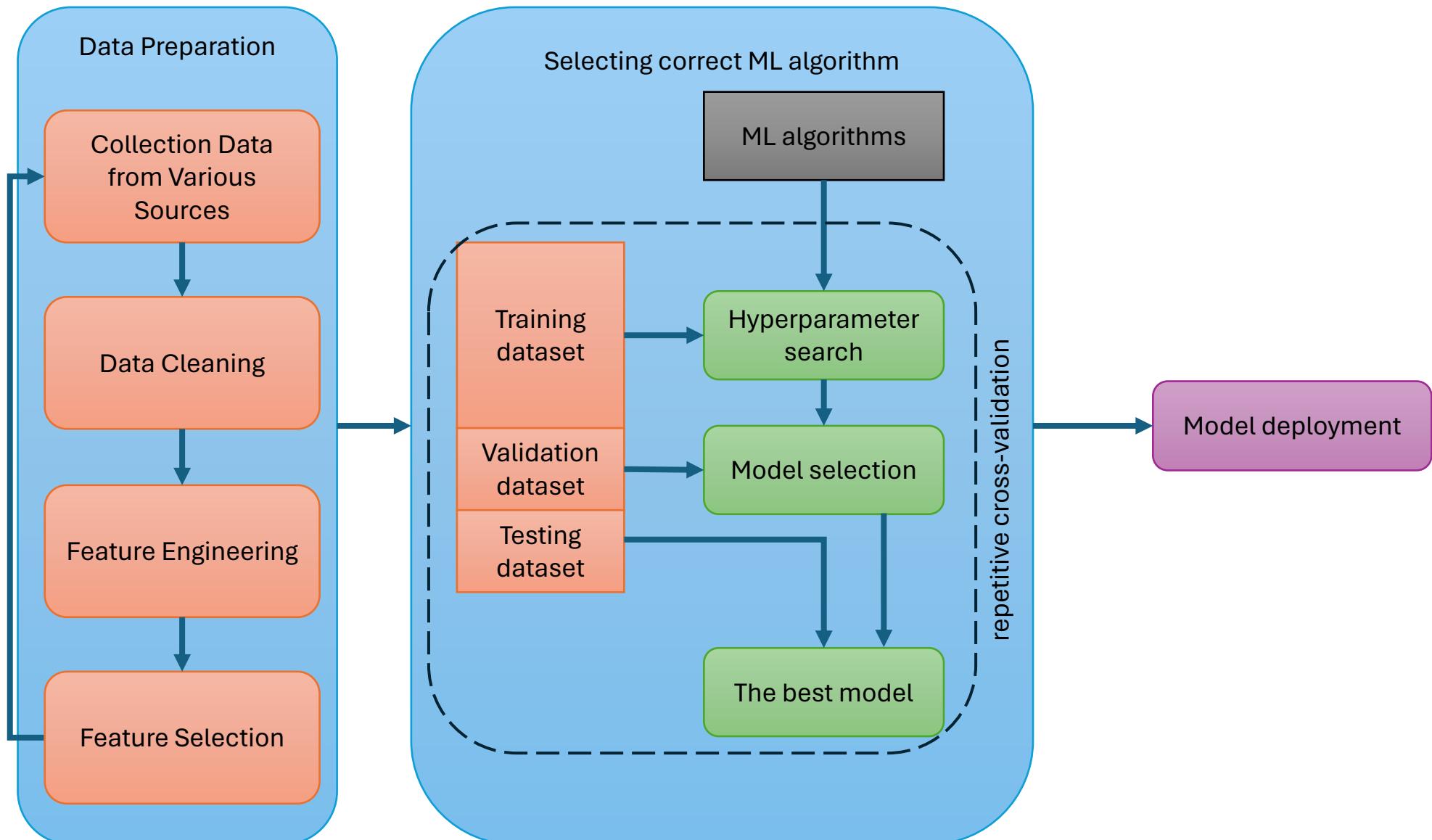


(b)

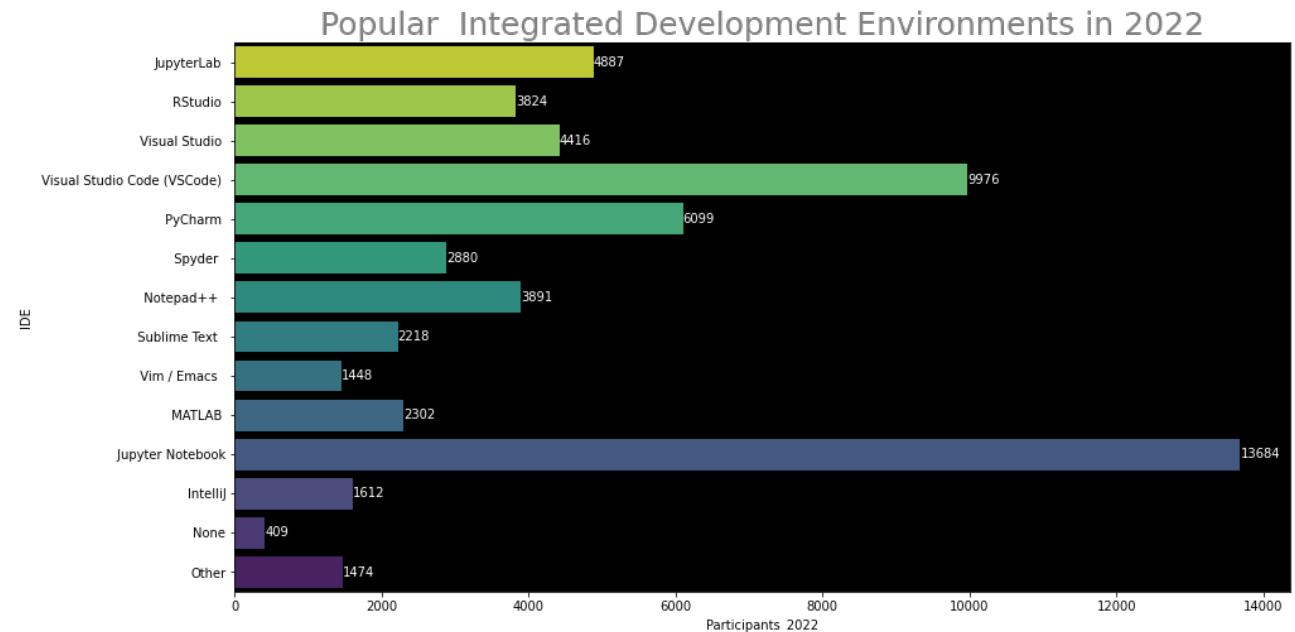
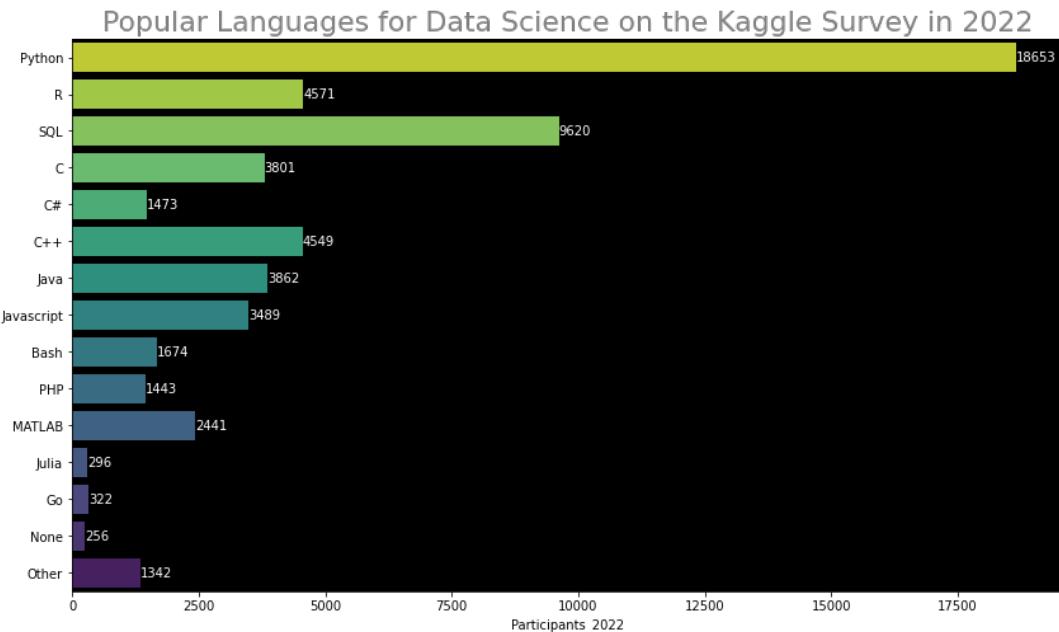
Байесовская оптимизация



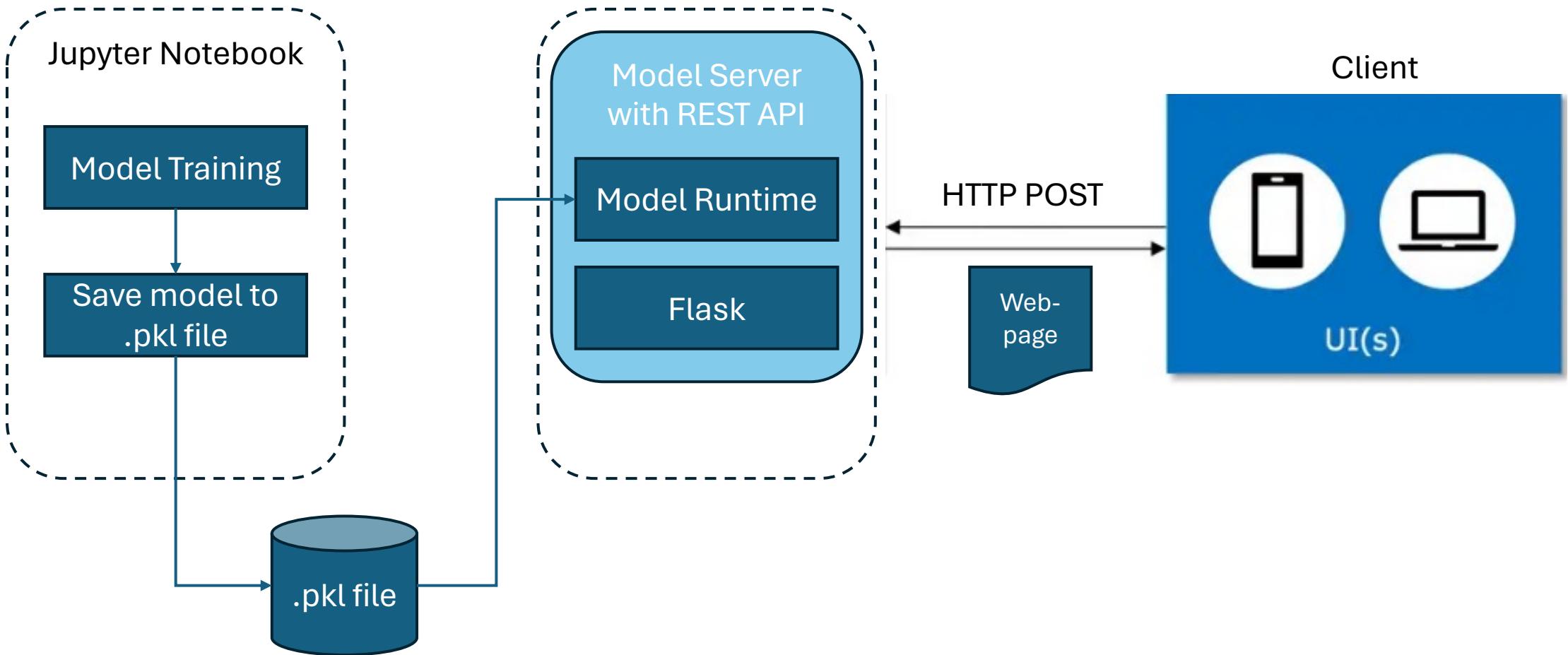
Обобщенный процесс решения задачи ML



Инструменты разработки (Kaggle 2022 survey)

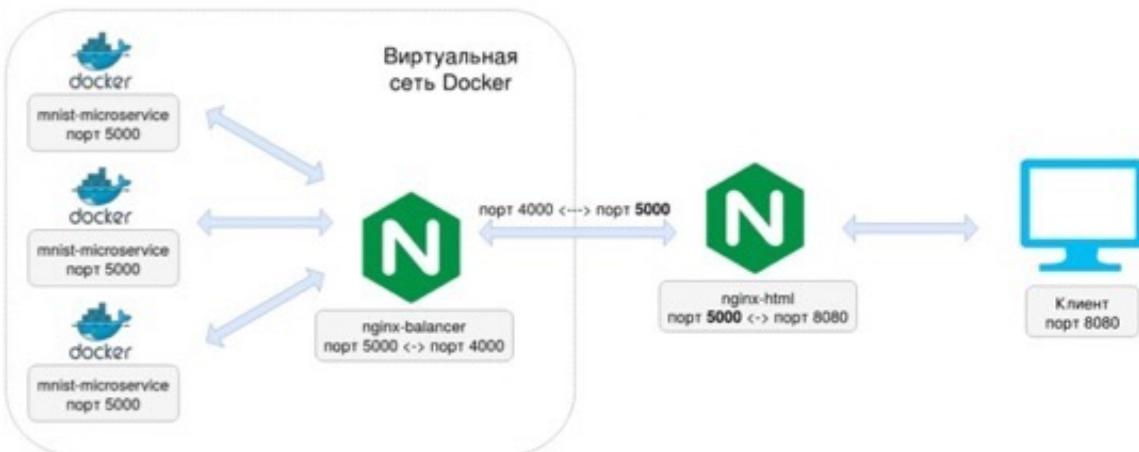


Развертывание ML модели в production



Микросервисы и балансировка нагрузки

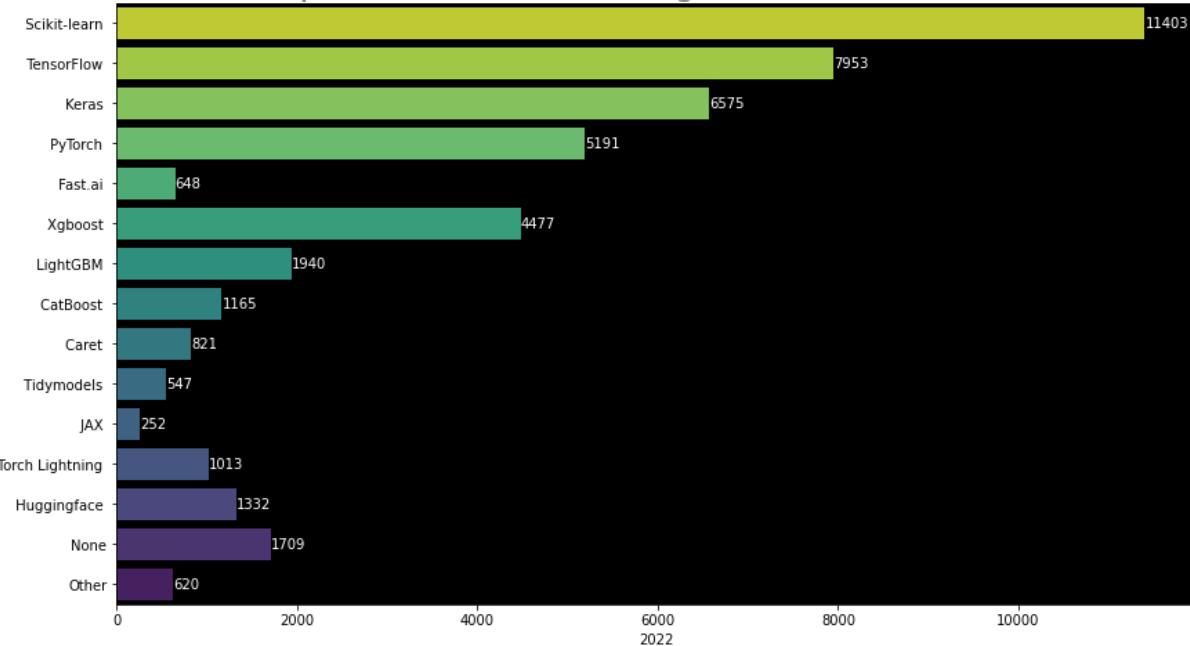
Popular Cloud Computing Platforms on the Kaggle Survey in 2022



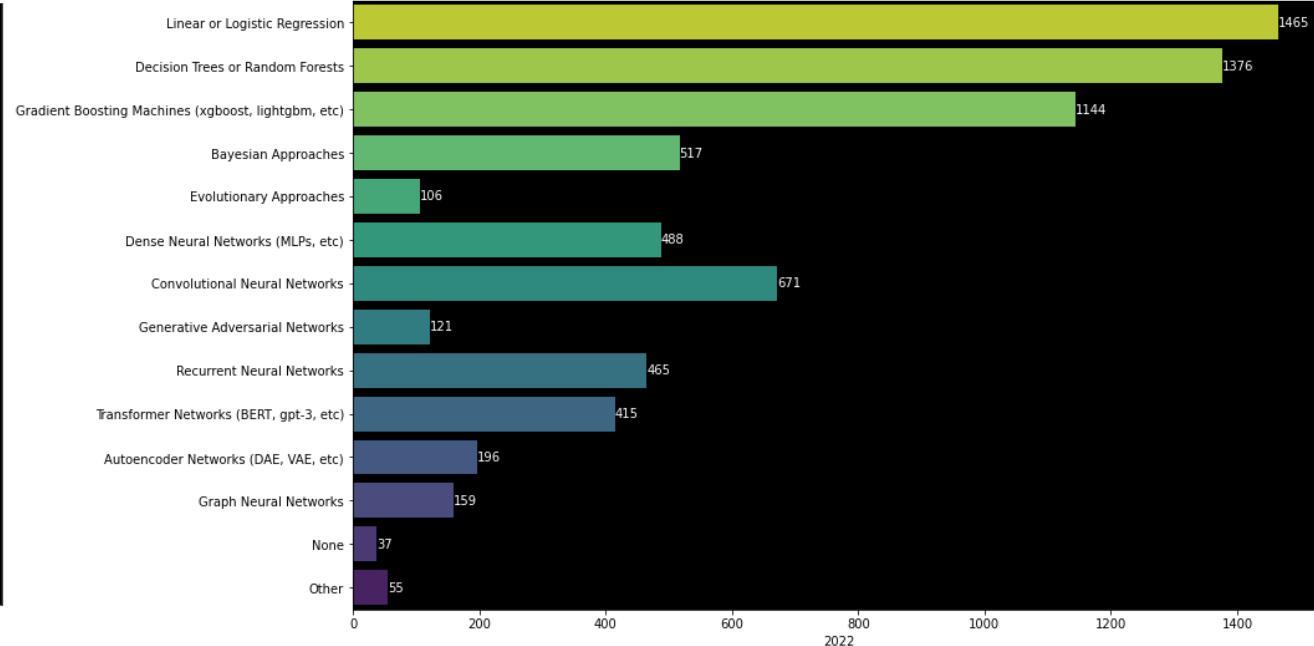
	clouds	Users	Users%	Rank
0	Amazon Web Services (AWS)	2346	28.750000	1.000000
1	Microsoft Azure	1416	17.352941	3.000000
2	Google Cloud Platform (GCP)	2056	25.196078	2.000000
3	IBM Cloud / Red Hat	287	3.517157	5.000000
4	Oracle Cloud	230	2.818627	6.000000
5	SAP Cloud	107	1.311275	9.000000
6	VMware Cloud	155	1.899510	8.000000
7	Alibaba Cloud	76	0.931373	10.000000
8	Tencent Cloud	56	0.686275	11.000000
9	Huawei Cloud	47	0.575980	12.000000
10	None	1167	14.301471	4.000000
11	Other	217	2.659314	7.000000

Библиотеки и модели (Kaggle 2022 survey)

Popular Machine learning frameworks in 2022



ML algorithms are being used by the Data Scientist in 2022



Low code платформы



Orange

<https://orangedatamining.com>



Weka

<https://www.cs.waikato.ac.nz/ml/weka/>



rapidminer

<https://rapidminer.com>



Open for Innovation

KNIME

<https://www.knime.com>



Untitled

The screenshot shows the Orange data mining software interface. On the left, there's a sidebar with a search bar and categories: Data, Transform, Visualize, and Model (which is highlighted). Below these are icons for various machine learning and data processing components. A central workspace displays a workflow diagram:

- A **Data Table** node is connected to a **File** node via a "Data" arrow.
- The **File** node is connected to a **Learner** node via a "Data" arrow.
- The **Learner** node is connected to an **Evaluation Results** node via a "Learner" arrow.
- The **Evaluation Results** node is connected to a **Confusion Matrix** node via an "Evaluation Results" arrow.
- The **Confusion Matrix** node is connected to a **Data Table (1)** node via a "Selected Data → Data" arrow.

Annotations with arrows explain specific parts of the workflow:

- An arrow points to the **Data Table** node with the text: "Choose class-labeled dataset. Say, "iris.tab" from documentation datasets."
- An arrow points to the **Data Table (1)** node with the text: "Select a cell in confusion matrix to obtain related data instances. Here we examine them in the spreadsheet."
- An arrow points to the **Test & Score** section of the **Evaluation Results** node with the text: "Cross-validation takes place here. Double click to see the performance scores."
- An arrow points to the **Confusion Matrix** node with the text: "Use for additional analysis of cross-validation results."
- An arrow points to the **Learner** node with the text: "It's always a good idea to check out the data first."
- A green arrow points to the **Random Forest Classification** icon with the text: "Several learners can be scored in cross-validation at the same time."

At the bottom of the workspace, there are two small icons: a downward arrow inside a cube and an upward arrow inside a cube.

Select a widget to show its description.
See [workflow examples](#), [YouTube tutorials](#), or open the [welcome screen](#).

Toolbar icons: magnifying glass, hash, T, pencil, double vertical bars, double horizontal bars, question mark.

The screenshot displays the RapidMiner studio interface with several open windows:

- Data Table**: Shows the "iris" dataset with 150 instances, 4 features, and 3 target values.
- Test & Score**: Configured for Cross-validation (10 folds, Stratified) and compares models by Area under ROC. Results table:

	Logistic Regres...	Random Forest...	SVM Learner
Logistic Regression	0.693	0.387	
Random Forest Learner	0.307		0.266
SVM Learner	0.613	0.734	

- Confusion Matrix**: Displays the performance of the Random Forest Learner with the following matrix:

		Predicted			
		Iris-setosa	Iris-versicolor	Iris-virginica	
Actual	Iris-setosa	50	0	0	50
	Iris-versicolor	0	47	3	50
Iris-virginica	0	2	48	50	
	Σ	50	49	51	150

- Workflow Diagram**: A process flow starting from a "File" node (containing the "iris" dataset) through a "Data Table" node, then branching into three learner nodes: "Logistic Regression", "Random Forest Classification", and "SVM". The outputs of these learners feed into an "Evaluation Results" node, which then connects to a "Test & Score" node. A tooltip for the "Test & Score" node states: "Cross-validation takes place here. Double click to see the performance scores."
- Random Forest Classification Settings**: Set to "Random Forest Learner" with 10 trees, no attribute selection, replicable training, and balanced class distribution.

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose RandomCommittee -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.RandomTree -- -K 0 -M 1.0 -V 0.001 -S 1

Test options

Use training set
 Supplied test set Set...
 Cross-validation Folds 10
 Percentage split % 66
More options...

(Nom) class

Start Stop

Result list (right-click for options)

15:47:02 - meta.RandomCommittee

Classifier output

```
| | | | a6 = true
| | | | | a1 = false : c1 (2/0)
| | | | | a1 = true : c0 (1/0)

Size of the tree : 69

Time taken to build model: 0.01 seconds

== Stratified cross-validation ==
== Summary ==

Correctly Classified Instances      81      81      %
Incorrectly Classified Instances   19      19      %
Kappa statistic                   0.5406
Mean absolute error               0.2267
Root mean squared error           0.3612
Relative absolute error            50.3218 %
Root relative squared error       76.1479 %
Total Number of Instances          100

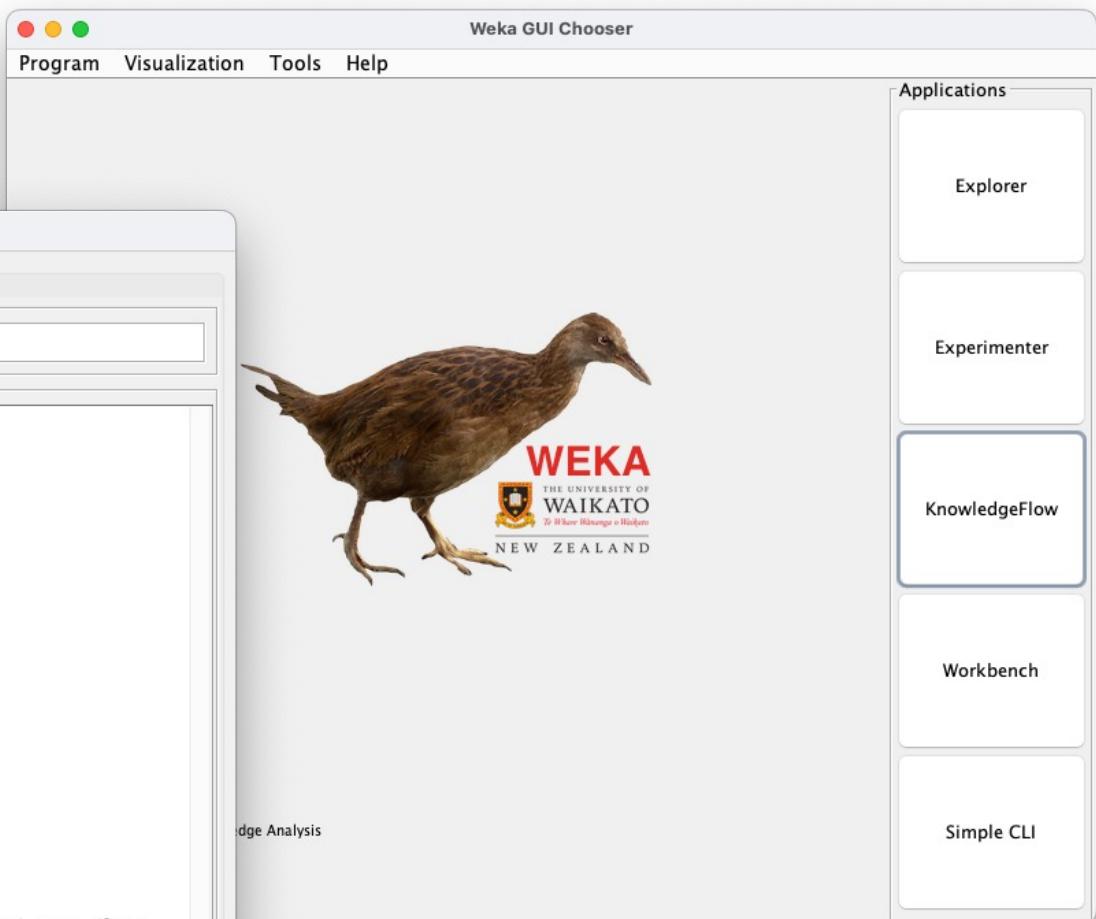
== Detailed Accuracy By Class ==

      TP Rate  FP Rate  Precision  Recall  F-Measure  MCC    ROC Area  PRC Area  Class
      0,939    0,441    0,805    0,939    0,867    0,561    0,891    0,937    c0
      0,559    0,061    0,826    0,559    0,667    0,561    0,891    0,753    c1
Weighted Avg.    0,810    0,312    0,812    0,810    0,799    0,561    0,891    0,874

== Confusion Matrix ==

  a  b  <-- classified as
62  4  |  a = c0
15 19  |  b = c1
```

Status OK Log x 0



//Local Repository/Dzyuba/08_Border_cleaning* – RapidMiner Studio Free 10.3.001 @ MacBook-Air2020.local

Views: Design Results Turbo Prep Auto Model **Interactive Analysis**

Find data, operators...etc All Studio

Process

Process >

Process

```
graph LR; A[Retrieve 07_New_Fi...] --> B[Select Attributes]; B --> C[Normalize]; C --> D[Generate Attributes]; D --> E[Generate Aggregation]; E --> F[Generate Attributes ...]; F --> G[Filter Examples]; G --> H[Select Attributes 2]; H --> I[Clustering]; I --> J[Execute Python]; J --> K[ ]; K --> L[ ]; L --> M[ ]; M --> N[ ]; N --> O[ ]; O --> P[ ]; P --> Q[ ]; Q --> R[ ]; R --> S[ ]; S --> T[ ]; T --> U[ ]; U --> V[ ]; V --> W[ ]; W --> X[ ]; X --> Y[ ]; Y --> Z[ ]; Z --> AA[ ]; AA --> BB[ ]; BB --> CC[ ]; CC --> DD[ ]; DD --> EE[ ]; EE --> FF[ ]; FF --> GG[ ]; GG --> HH[ ]; HH --> II[ ]; II --> JJ[ ]; JJ --> KK[ ]; KK --> LL[ ]; LL --> MM[ ]; MM --> NN[ ]; NN --> OO[ ]; OO --> PP[ ]; PP --> QQ[ ]; QQ --> RR[ ]; RR --> TT[ ]; TT --> YY[ ]; YY --> ZZ[ ]; ZZ --> AA[ ]
```

Repository

- + Import Data
- Training Resources (connected)
- Samples
- Community Samples (connected)
- Keras Samples
- Time Series Extension Samples
- Local Repository (Legacy)
- Temporary Repository (Legacy)
- DB (Legacy)

Operators

Search for Operators

- Data Access (59)
- Blending (81)
- Cleansing (28)
- Modeling (167)
- Scoring (13)
- Validation (30)
- Utility (86)
- Extensions (460)

Parameters

Process

- logverbosity init
- logfile
- resultfile
- random seed 2001
- send mail never

Help

Process

RapidMiner Studio Core

Synopsis

The root operator which is the outer most operator of every process.

Description

Each process must contain exactly one operator of this class, and it must be the root operator of the process. This operator provides a set of parameters that are of global relevance to the process like logging and initialization parameters of the random number generator.

Parameters

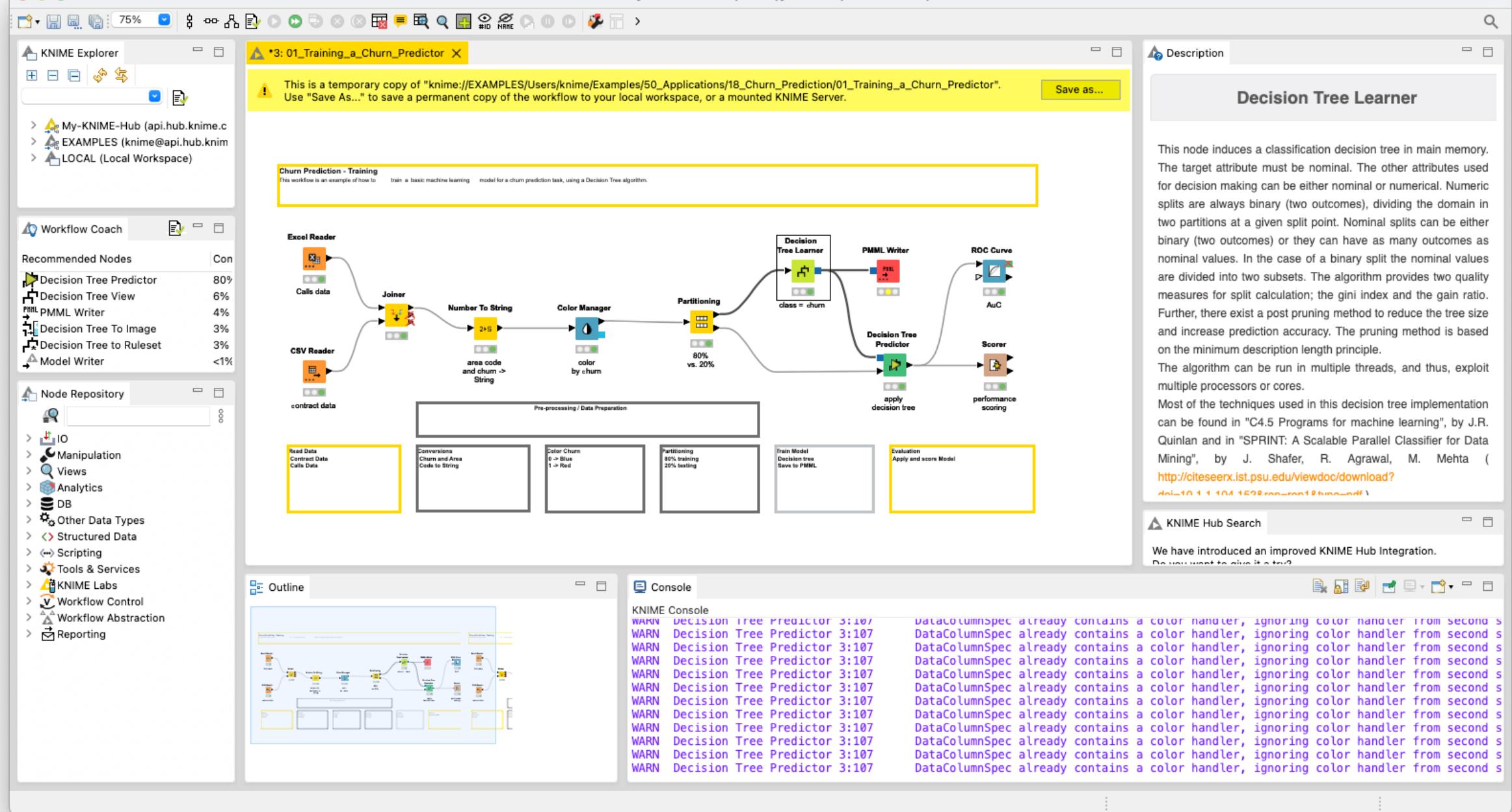
logverbosity (optional)

Log verbosity level.

Recommended Operators

- Multiply 29%
- Apply Model 28%
- Join 25%
- Aggregate 24%

Get more operators from the Marketplace

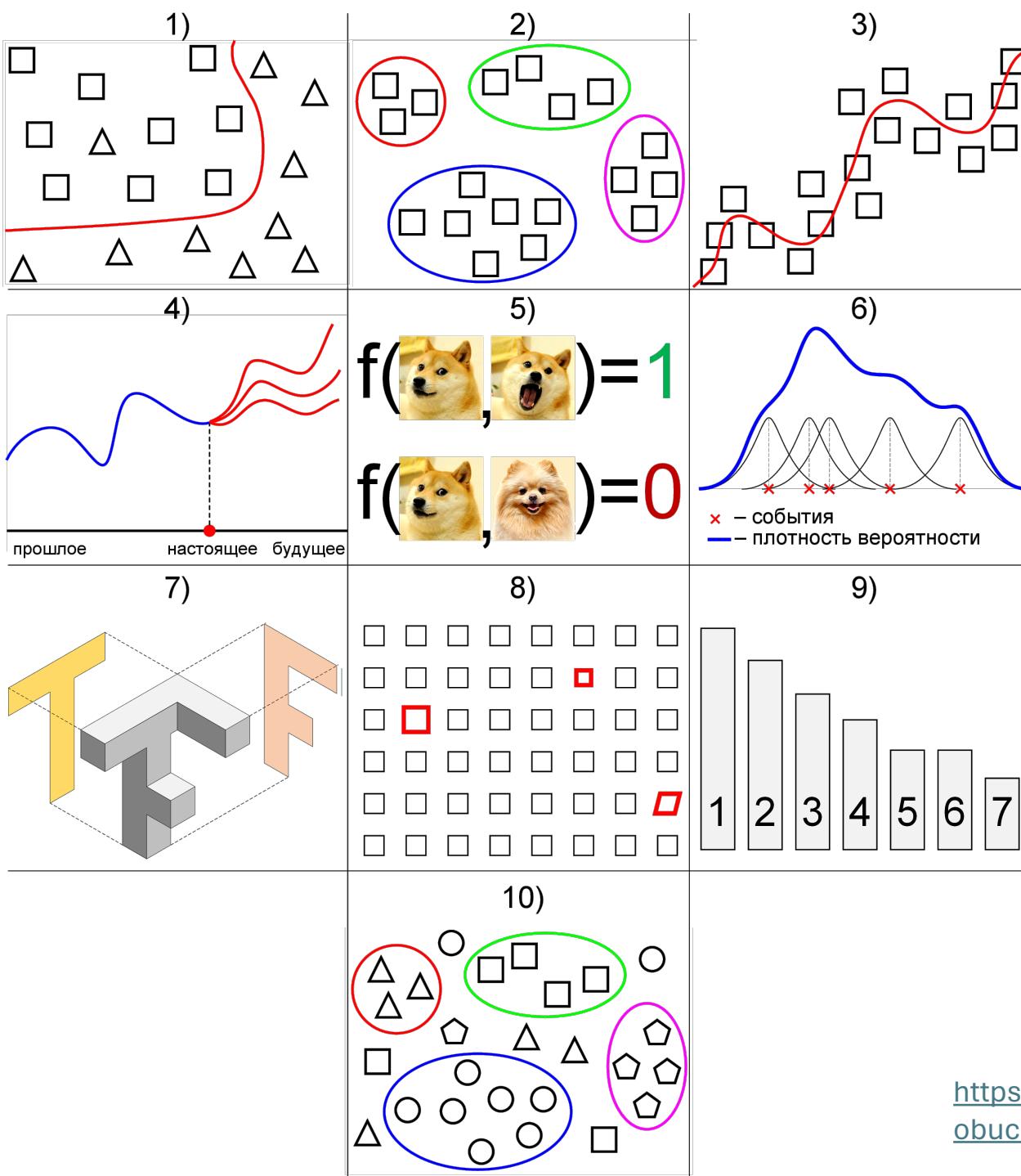


Сравнение

- Orange
 - Python / [open source](#) – очень легкий и быстрый
 - К библиотекам Orange можно обращаться из Python
 - Относительно малый набор операторов / Есть community
 - Deployment – как обычная модель Python
- Weka
 - Java / [open source](#)
 - Самый сложный инструмент
 - Deployment: as separate Java executables ?
- Rapid Miner
 - Java (умеет выполнять скрипты на Python) / **коммерческий**
 - Rapid Miner Studio Free (10,000 записей, 1 процессор)
 - Рассчитан на мало подготовленного пользователя, есть AutoML
 - Относительно малый набор операторов / Есть community
 - Deployment: Rapid Miner AI Hub - \$ 54,000 per user per month
 - **Заблокирован для РФ**
- Knime
 - Java (умеет выполнять скрипты на Python) / **коммерческий**
 - Knime Analytics Platform - бесплатно
 - Очень большой набор операторов / Большое community
 - Deployment: Knime Community Hub - от €35,000 в год



Практические задачи ML



Типы задач ML:

- 1) классификация;
- 2) кластеризация;
- 3) регрессия;
- 4) прогнозирование;
- 5) идентификация;
- 6) восстановление плотности распределения вероятности по набору данных;
- 7) понижение размерности;
- 8) одноклассовая классификация и выявление новизны;
- 9) построение ранговых зависимостей;
- 10) добыча данных.

Приложения ML

- Медицина:
 - Диагностика по симптомам
 - Диагностика по изображениям
 - ...
- Финансы:
 - Кредитный scoring
 - Мошеннические транзакции
 - ...
- Страхование:
 - Идентификация мошенников
 - ...
- HR:
 - Идентификация недовольных сотрудников
 - ...
- Производство:
 - Идентификация брака
 - Предсказание параметров продукта до испытаний
 - ...
- Сервис / Ритейл:
 - Отток клиентов
 - ...
- Фильтрация e-mail

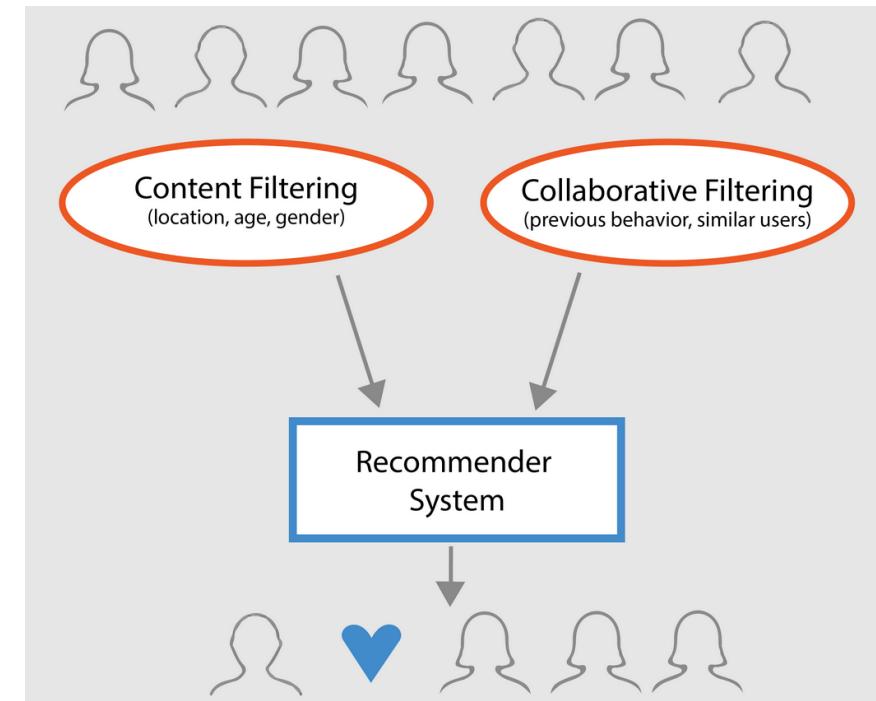
Приложения ML



Рекомендательные системы

	Book 1	Book 2	Book 3	Book 4	Book 5
User A	thumb up	thumb down	thumb up		thumb up
User B		thumb up		thumb down	thumb down
User C	thumb up	thumb up	thumb down		
User D		thumb up	?		thumb down

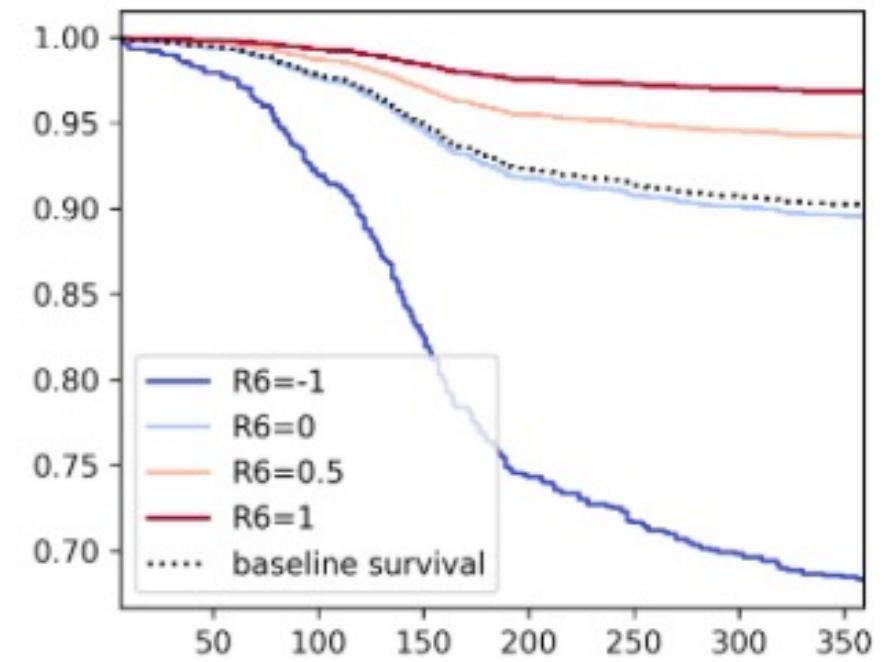
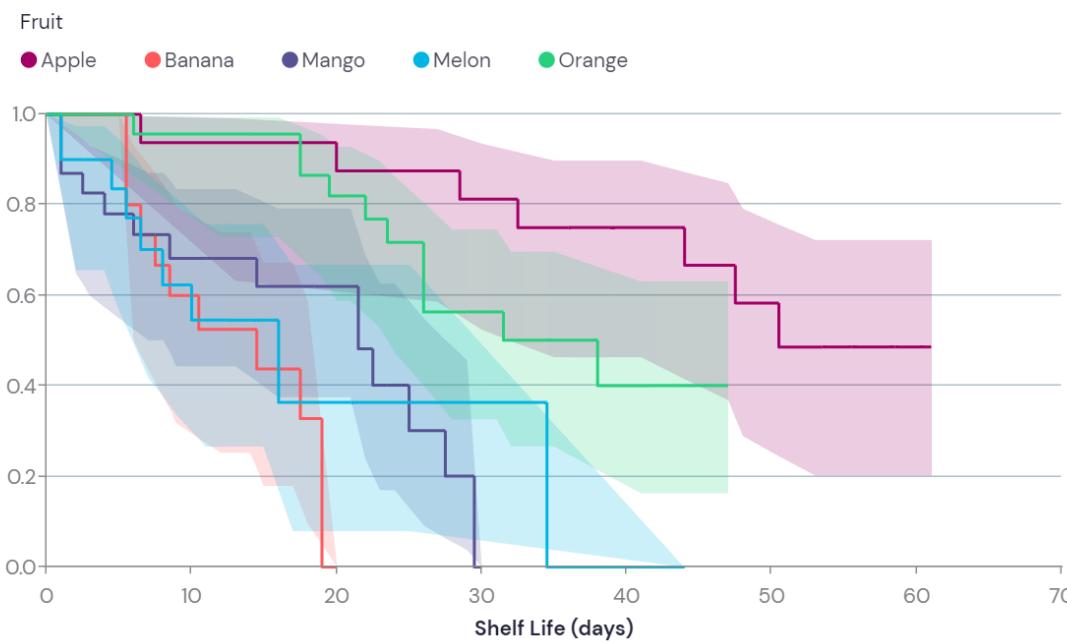
Похожие
пользователи



Похожие
продукты

Анализ выживаемости / Survival Analysis

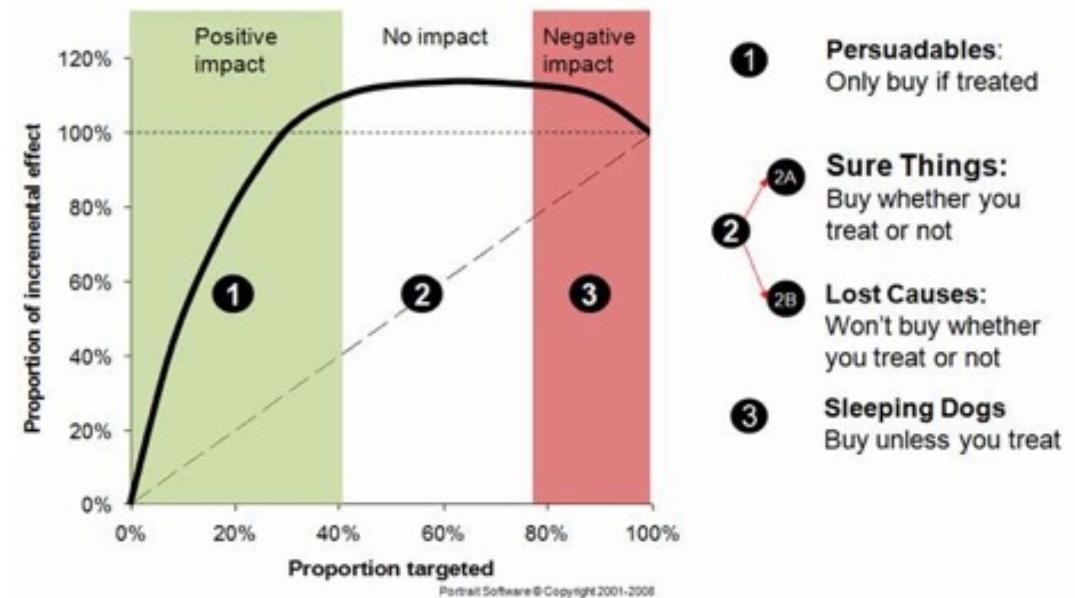
^Kaplan Meier



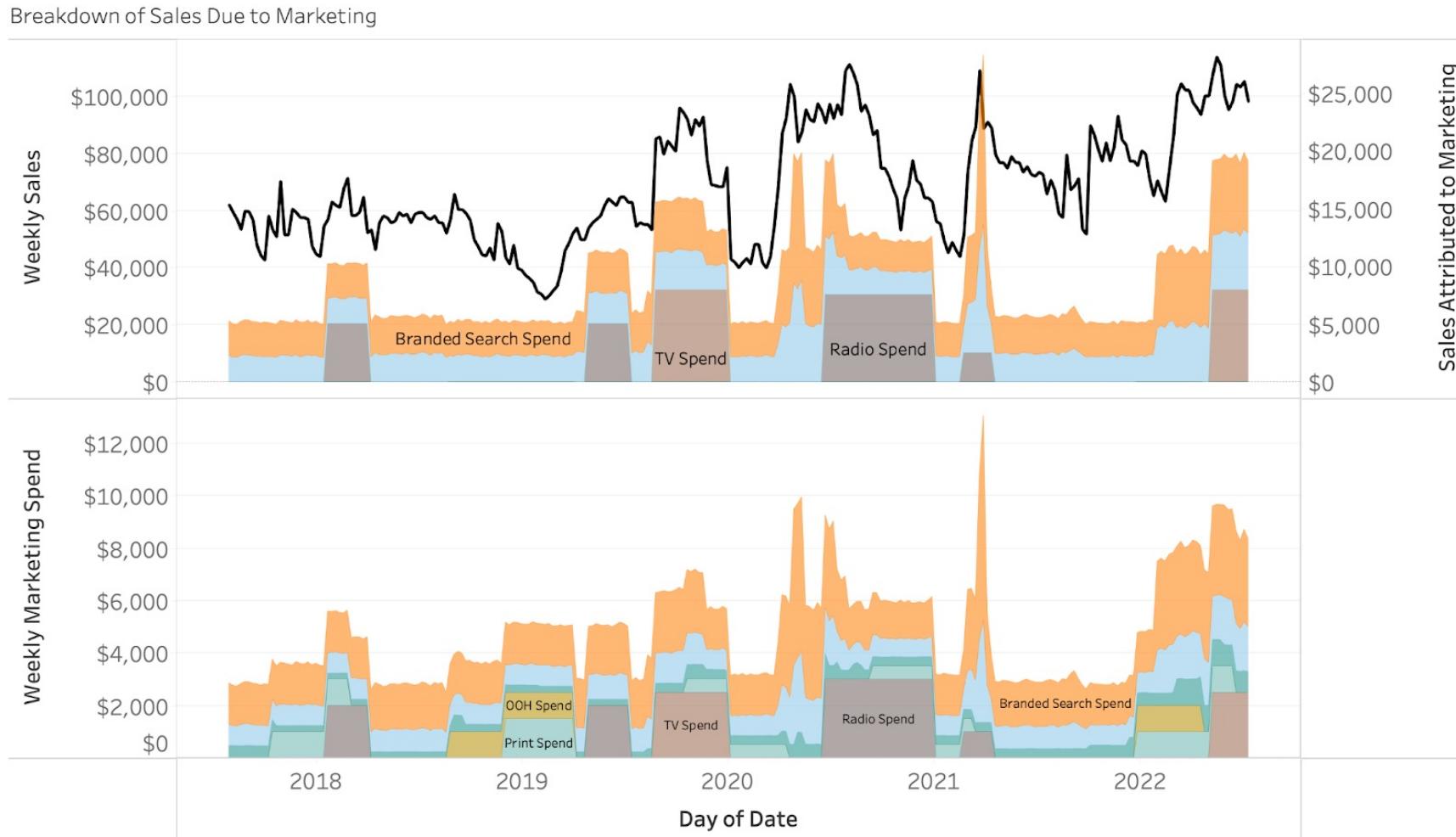
Source: Zelenkov Y. (2020) Bankruptcy Prediction Using Survival Analysis Technique, in: *Proceedings of 2020 IEEE 22nd Conference on Business Informatics*. 141-149

Uplift modeling

RENEW IF TREATED	Yes	Persuadables	Sure Things
			
No	Lost Causes	Sleeping Dogs	 
	No	Yes	
RENEW IF NOT TREATED			



Marketing Mix Modeling



Cost Sensitive Learning

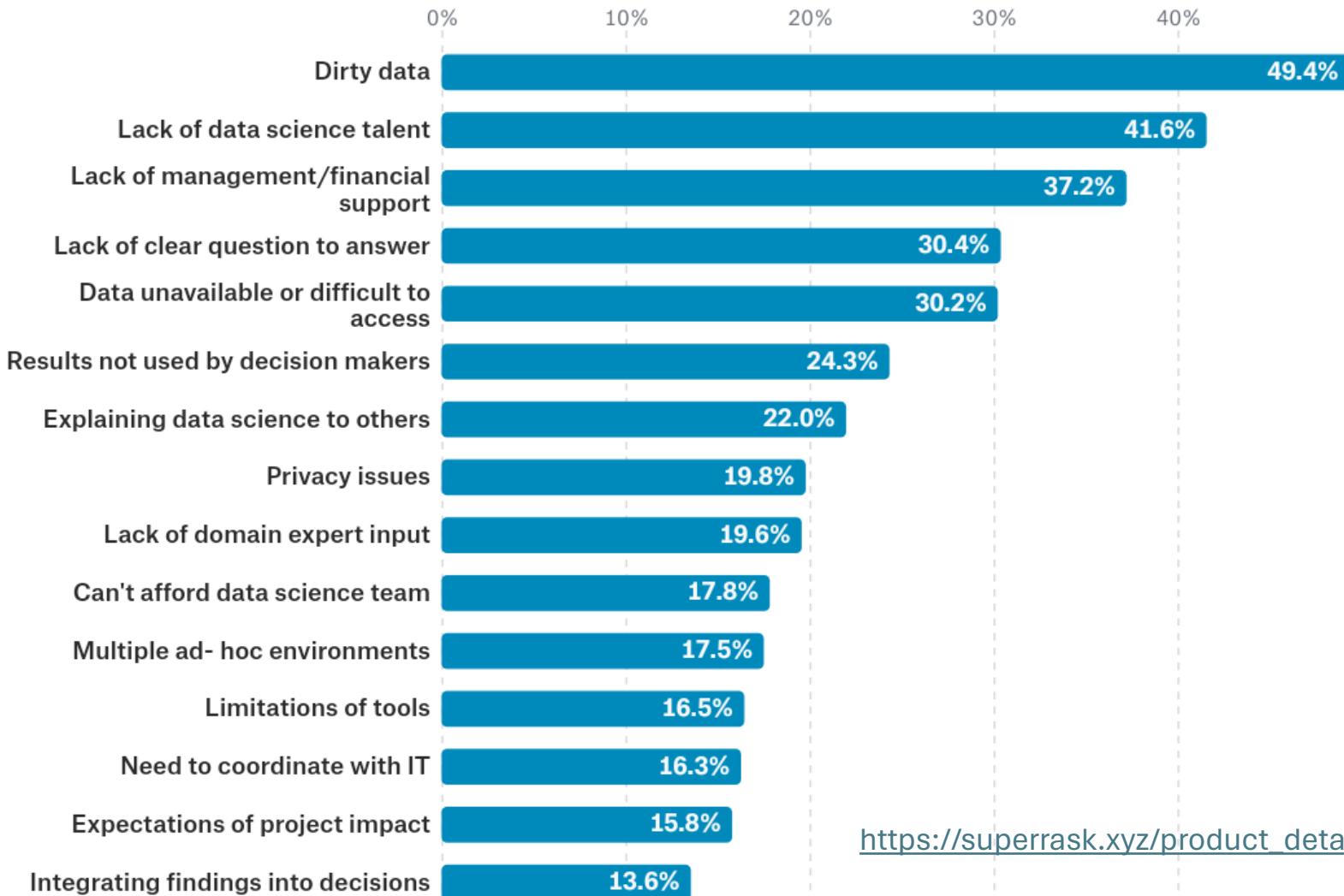
Cost Matrix for Credit Scoring

	Positive	Negative
Positive	TP, C = 0	FN, C = C_{FN}
Negative	FP, C = C_{FP}	TN, C = 0

$$C_{FN} > C_{FP}$$

$$\begin{aligned} C(y_i, f(z_i)) = \frac{1}{4} & \left[(1 + y_i) \left((1 - f(z_i))C_{FN_i} + (1 + f(z_i))C_{TP_i} \right) \right. \\ & \left. + (1 - y_i) \left((1 + f(z_i))C_{FP_i} + (1 - f(z_i))C_{TN_i} \right) \right]. \end{aligned}$$

Что препятствует внедрению AI/ML?



https://superrask.xyz/product_details/37338224.html

Вопросы?