Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина

Программирование на Python

Подготовка данных и немного ML. ПРАКТИКА

Андрей Скороходов

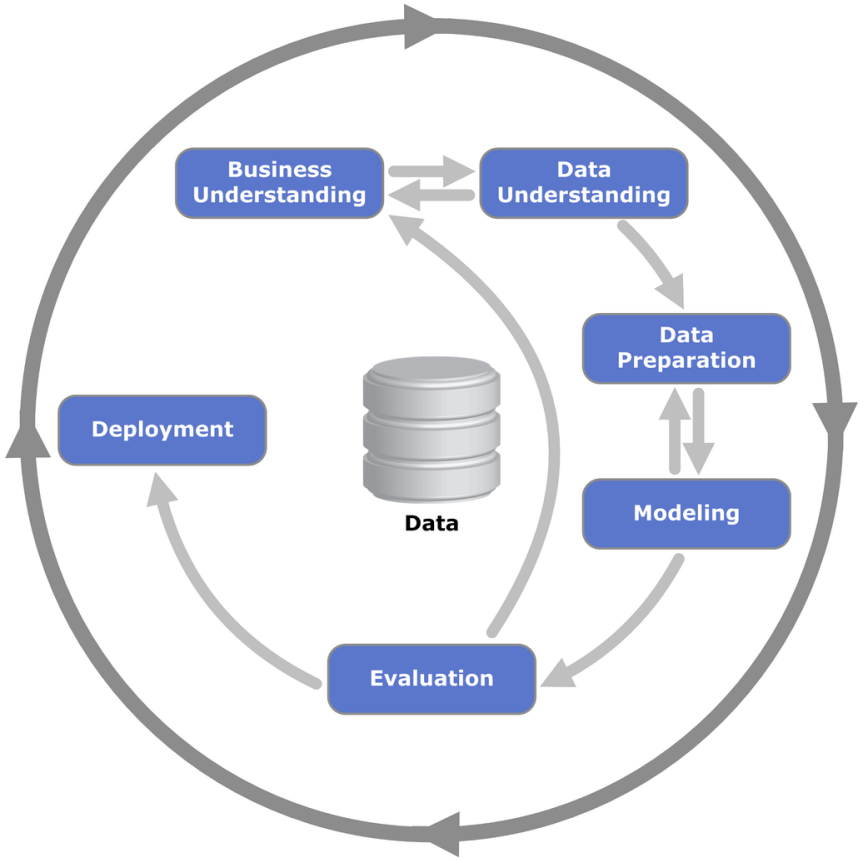
udv|group 2023

Уральский федеральный университет

Жизненный цикл модели ML

Жизненный цикл модели машинного обучения начинается с постановки задачи от бизнеса (если мы говорим о продукте) и не заканчивается даже на внедрении модели в production.

На этой практике мы будем рассматривать один из этапов – подготовку данных (Data Preparation) и немного коснемся процесса создания и обучения модели (Modeling), оценки качества ее работы (Evaluation).



https://ru.wikipedia.org/wiki/CRISP-DM

Уральский федеральный университет

Подготовка данных к машинному обучению

Подготовка данных зависит от типа данных и от планируемого метода обучения.

Но если смотреть в общих чертах, то подготовку можно свести к нескольким основным этапам:

Сбор данных

(data collection)

- БД
- Источники в сети
- Файлы
- …

Очистка данных

(data cleaning)

- Удаление или заполнение отсутствующих данных
- Удаление или корректировка ошибок
- Удаление дубликатов и нерелевантных данных
- …

Предобработка данных

(data preprocessing)

- Нормализация и стандартизация данных
- Кодирование категориальных признаков
- …

Трансформация данных

(data transformation)

- Выбор и обработка необходимых признаков
- Уменьшение размерности данных
- Создание дополнительных признаков (feature engineering)
- …

pandas

Beautiful Soup

psycopg2

PySpark

pandas

scikit learn

seaborn

matplotlib

Уральский федеральный университет

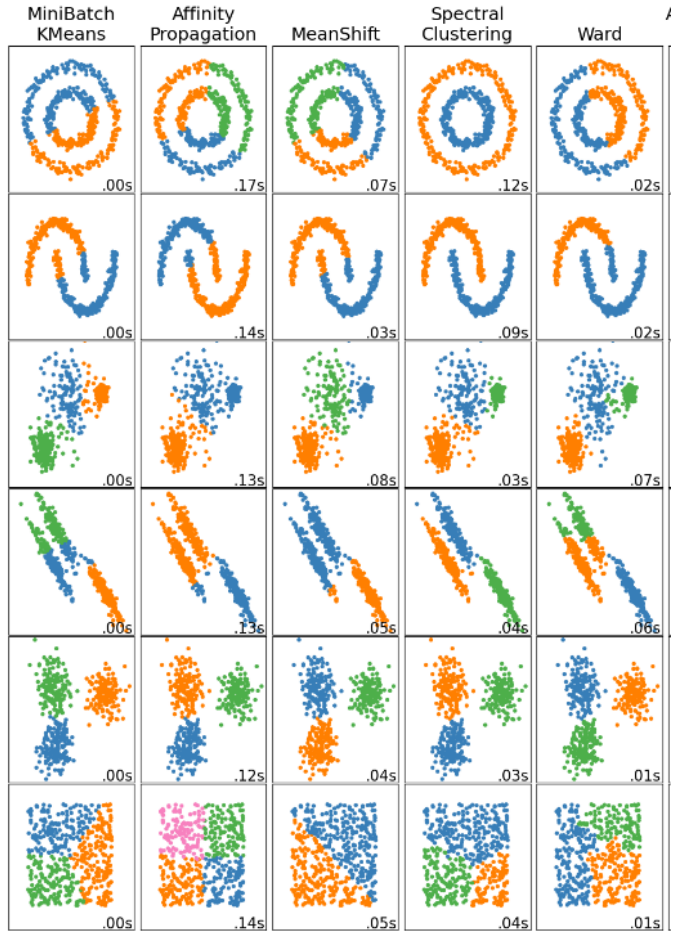
Scikit-learn

Scikit-learn – библиотека на базе NumPy, Matplotlib и SciPy, в которой реализованы операции по обработке данных, алгоритмы машинного обучения, механизмы оценки качества моделей, создания пайплайнов и т.п.

```
$pip install -U scikit-learn
import sklearn
```

Основные задачи для которых может быть использована scikit-learn:

- Обработка данных (стандартизация и нормализация, дискретизация, обработка пустых значений, векторизация, кодирование данных, …)
- Классификация данных (Logistic Regression, SVM, KNN, Random Forest, …)
- Регрессия (Linear Regression, AdaBoost, Decision Trees, …)
- Кластеризация (K-means, Agglomerative, DBSCAN, BIRCH, …)
- Уменьшение размерности (PCA, LDA, Factor Analysis, …)
- Работа с моделями (оценка, кросс-валидация, настройка гиперпараметров, сохранение, выстраивание пайплайнов, …)



https://scikit-learn.org/stable/modules/clustering.html

Уральский федеральный университет

Очистка данных

Подходы к **очистке данных** зависят от типа задачи, выбранной модели и от полноты понимания данных.

В общем случае очистка данных включает в себя:

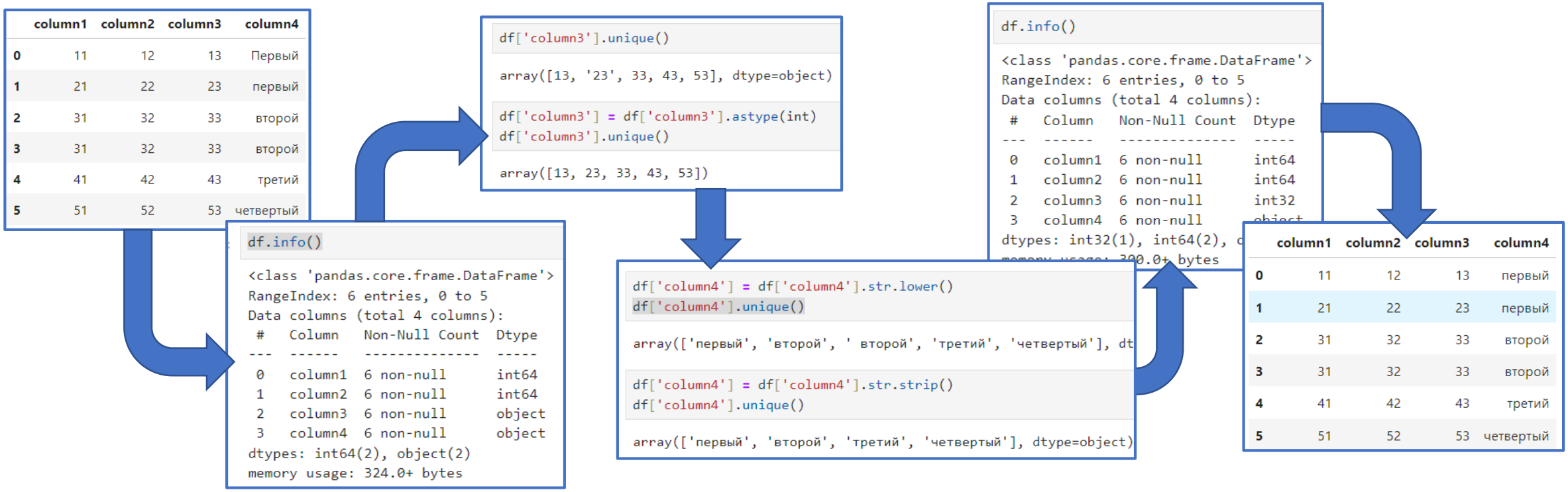
- Обработка синтаксических ошибок и ошибок преобразования типов
- Обработка дубликатов
- Обработка пропущенных значений
- Обработка выбросов

В большинстве случаев для этапа очистки данных достаточно функциональности Pandas.

В этом модуле мы не рассматриваем процесс подготовки данных для аудио или видео, только табличные данные.

Обработка ошибок типа, структуры

Реальные данные поступают из разных источников, в разное время, поэтому формат представления данных может сильно отличаться. На этапе обработки проверяется соответствие данных заданной структуре/формату.



В большинстве случаев для обработки синтаксических ошибок в данных достаточно функциональности Pandas.

Если не хватает встроенных методов, можно всегда создать свою функцию и применить ее к данным, например, через df.apply().

Для уменьшения памяти, занимаемой датафреймом, можно приводить категориальные признаки к типу category:

df.astype("category"), а также к типам данных с меньшим размером, например df.astype("int16")

Уральский федеральный университет

Обработка дубликатов

Спасибо Pandas, обработка дублирующихся значений производится до невозможности просто.

```
df
```

	column1	column2	column3	column4
0	11	12	13	первый
1	21	22	23	первый
2	31	32	33	второй
3	31	32	33	второй
4	41	42	43	третий
5	51	52	53	четвертый

```
df[df.duplicated(keep=False)]
```

	column1	column2	column3	column4
2	31	32	33	второй
3	31	32	33	второй

```
df.drop_duplicates()
```

	column1	column2	column3	column4
0	11	12	13	первый
1	21	22	23	первый
2	31	32	33	второй
4	41	42	43	третий
5	51	52	53	четвертый

Дубликаты можно искать только по указанным колонкам (параметр subset).

Можно удалять все дубликаты, можно сохранять только первый или только последний (параметр keep).

Обработка пропущенных значений

При подготовке данных почти всегда приходится лавировать между сохранением максимального количества данных, чтобы модель лучше обучилась, и удалением данных, которые будут мешать модели правильно обучаться.

Пропущенные значения – одна из основных проблем в данных. Иногда решение об оставлении или удалении приходится принимать на основании оценки работы модели, т.е. пробовать учить модель на разных вариантах.

	column1	column2	column3	column4	columnn5
0	NaN	12.0	13	первый	NaN
1	21.0	22.0	23	первый	NaN
2	31.0	NaN	33	второй	NaN
4	41.0	42.0	43	третий	NaN
5	51.0	52.0	53	четвертый	NaN

заполнить

```
df.fillna(0)
```

	column1	column2	column3	column4	columnn5
0	0.0	12.0	13	первый	0.0
1	21.0	22.0	23	первый	0.0
2	31.0	0.0	33	второй	0.0
4	41.0	42.0	43	третий	0.0
5	51.0	52.0	53	четвертый	0.0

интерполировать

```
df.interpolate(axis=0)
```

	column1	column2	column3	column4	columnn5
0	NaN	12.0	13	первый	NaN
1	21.0	22.0	23	первый	NaN
2	31.0	32.0	33	второй	NaN
4	41.0	42.0	43	третий	NaN
5	51.0	52.0	53	четвертый	NaN

удалить

```
df.dropna(how='all', axis=1)
```

	column1	column2	column3	column4
0	NaN	12.0	13	первый
1	21.0	22.0	23	первый
2	31.0	NaN	33	второй
4	41.0	42.0	43	третий
5	51.0	52.0	53	четвертый

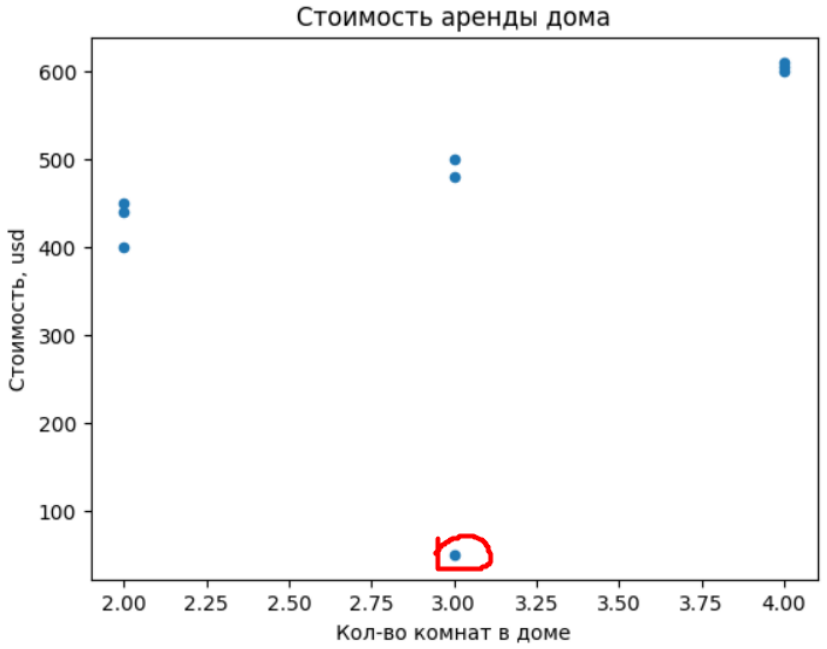
Удалять пустые значения можно при хотя бы одном пустом значении или только если все пустые (параметр how), также можно указать диапазон поиска пропусков (параметр subset) или ось (параметр axis).

Заполнять пустые значения можно константами, средними значениями или предыдущими значениями (параметр method)

Обработка выбросов

Как и при удалении пустых значений, удаление выбросов должно исходить из понимания задачи и предметной области. Некоторые методы нечувствительны к выбросам, некоторые выбросы являются следствием, например, ошибки сбора данных.

	Кол-во комнат в доме	Стоимость, usd
0	2	400
1	3	500
2	4	600
3	4	610
4	3	50
5	3	480
6	2	450
7	4	605
8	2	440



Методов выявления выбросов много:

- Использование статистики (стандартное отклонение, квантили)
- Использование методов машинного обучения
- Знания из предметной области

Обработка выбросов ведется в зависимости от используемой модели, знаний предметной области и т.п., и в целом подход похож на работу с пропущенными значениями

В большинстве случаев для обработки выбросов достаточно функциональности Pandas, если не хватает встроенных методов, можно всегда создать свою функцию и применить ее к данным, например, через df.apply().

В сложных случаях можно применять методы выявления аномалий из Sklearn (LOF, OCSVM, iForest и т.п.)

Уральский федеральный университет

Предобработка данных. Стандартизация и нормализация

Предобработка данных всегда зависит от используемой модели и чаще всего заключается в стандартизации или нормализации данных, кодировании категориальных признаков, дискретизации признаков и т.п.

Стандартизация и нормализация применяются для выравнивания диапазона (шкалы) данных.

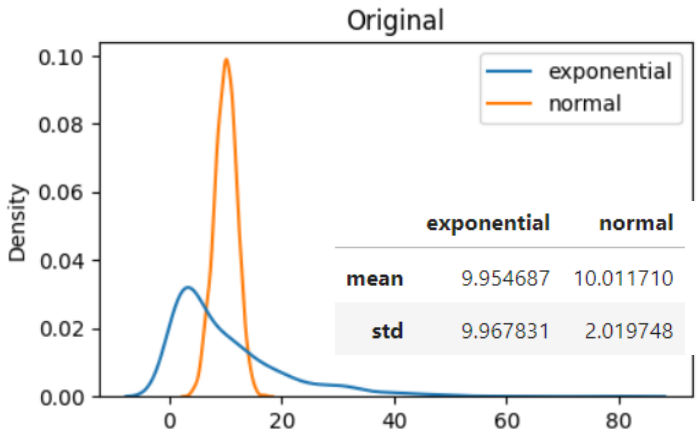
Нормализация (грубо) – преобразование данных к норме без изменения их распределения.

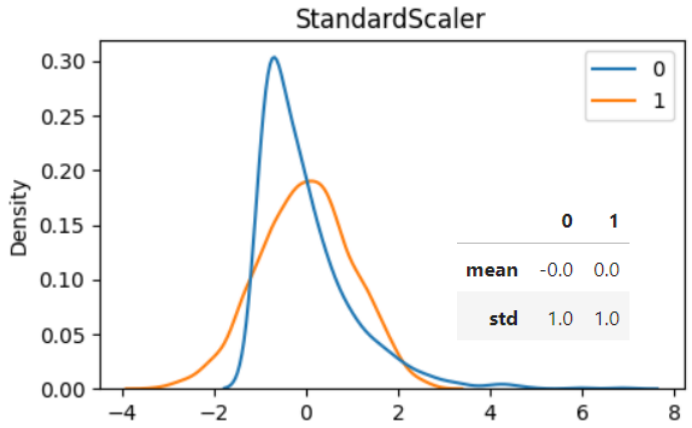
Стандартизация (также грубо) – преобразование данных к другому диапазону с изменением формы распределения.

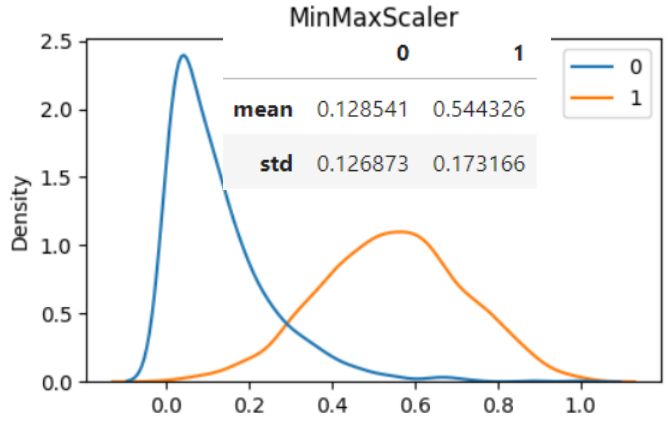
Стандартизация может сгладить эффект выбросов. В Sklearn есть различные методы стандартизации и нормализации.

```
from sklearn.preprocessing import StandardScaler
df1 = StandardScaler().fit_transform(df)
```

	exponential	normal
0	2.928654	10.066737
1	6.560796	10.417216
2	1.291698	7.767681
3	24.614723	10.884028
4	15.738040	8.275893
5	4.144890	6.057417
6	10.908205	9.031449
7	7.389212	15.006671
8	2.001612	8.230233
9	2.136691	6.872085







https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html
https://www.geeksforgeeks.org/compare-effect-of-different-scalers-on-data-with-outliers-in-scikit-learn/

Уральский федеральный университет

Кодирование признаков

Некоторые модели не умеют работать с категориальными данными в виде текста, поэтому для корректной работы данные необходимо предварительно преобразовать – закодировать.

Самые распространенные методы:

- Кодирование метки (label encoding)
- Унитарное кодирование (one-hot encoding)
- Двоичное кодирование (binary encoding)
- Бинаризация (binarizing)

Все эти и другие методы кодирования есть в Sklearn.

	sex	class	age
0	male	Third	22.0
1	female	First	38.0
2	female	Third	26.0
3	female	First	35.0
4	male	Third	35.0
6	male	First	54.0
7	male	Third	2.0
8	female	Third	27.0
9	female	Second	14.0
10	female	Third	4.0

```python
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, Binarizer

le = LabelEncoder()
ohe = OneHotEncoder(sparse_output=False).set_output(transform='pandas')
be = Binarizer(threshold=35)

df['sex_le'] = le.fit_transform(df['sex'])
df['age_be'] = be.fit_transform(df[['age']])
df = pd.concat([df, ohe.fit_transform(df[['class']]).astype(int)], axis=1

df.sort_index(axis=1)
```

Binarizer | One Hot Encoder | Label Encoder

	age	age_be	class	class_First	class_Second	class_Third	sex	sex_le
0	22.0	0.0	Third	0	0	1	male	1
1	38.0	1.0	First	1	0	0	female	0
2	26.0	0.0	Third	0	0	1	female	0
3	35.0	0.0	First	1	0	0	female	0
4	35.0	0.0	Third	0	0	1	male	1
6	54.0	1.0	First	1	0	0	male	1
7	2.0	0.0	Third	0	0	1	male	1
8	27.0	0.0	Third	0	0	1	female	0
9	14.0	0.0	Second	0	1	0	female	0
10	4.0	0.0	Third	0	0	1	female	0

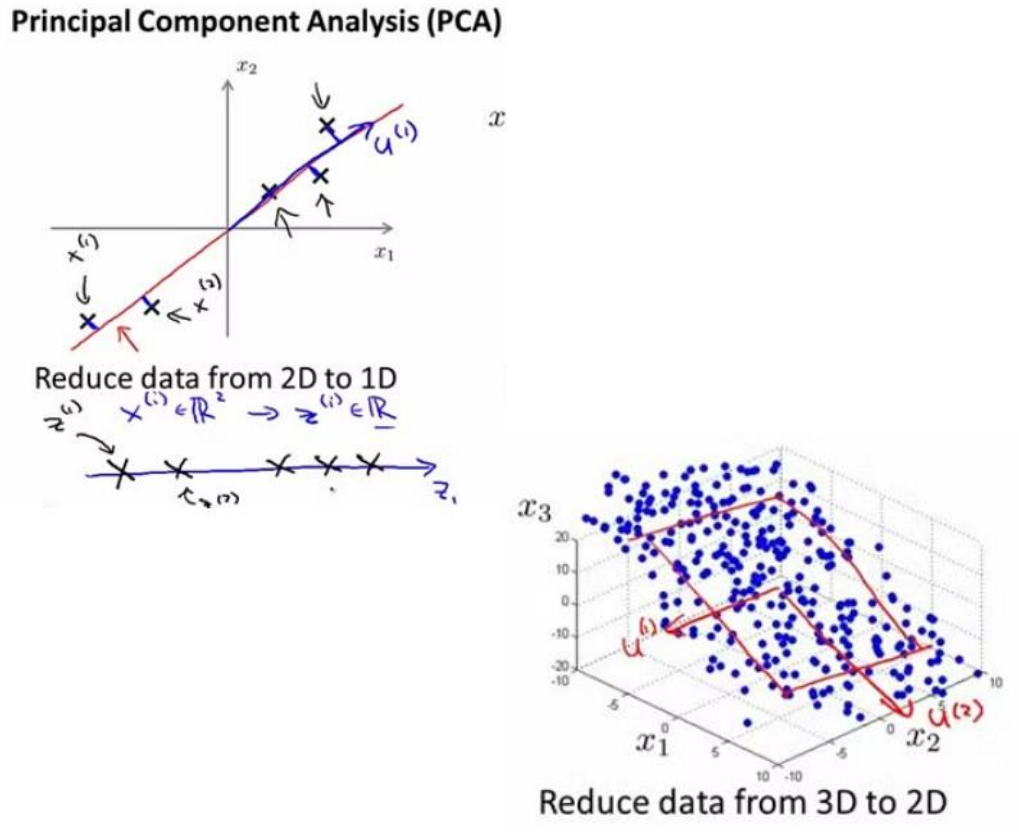
Уральский федеральный университет

Трансформация данных. Уменьшение размерности

На этапе **трансформации данных** убираются ненужные для модели признаки (уменьшение размерности, dimensionality reduction), добавляются новые признаки, сгенерированные из существующих (feature engineering). Уменьшение размерности служит для исключения малоинформативных признаков, что в свою очередь улучшает качество и ускоряют работу модели.

Для уменьшения размерности может использоваться:

- Метод главных компонент (PCA)
- t-distributed Stochastic Neighbor Embedding (t-SNE)
- Использование встроенных в некоторые модели методов feature_importance
- Корреляционный анализ
- …



https://www.projectpro.io/data-science-in-python-tutorial/principal-component-analysis-tutorial

Уральский федеральный университет

Feature engineering

Конструирование признаков или Feature Engineering – процесс создания новых признаков или трансформации существующих для улучшения работы модели.

Подходы в Feature Engineering можно грубо разделить на три группы:

- Использование знаний в предметной области
- Изучение данных (паттерны, агрегации и т.п.)
- Манипуляция с данными (комбинирование и преобразование признаков и т.п.)

Эти подходы реализуются при помощи стандартных инструментов Pandas, Sklearn, но иногда необходимо использование более сложной математики (SciPy) или применение промежуточных моделей машинного обучения.

Извлечение титула из имени

```
import re

df_train['Name_title'] = df_train['Name'].\
    apply(lambda x: re.search(',\s+(\w+)', x)[1])

df_train[['Name_title', 'Name']]
```

	Name_title	Name
0	Mr	Braund, Mr. Owen Harris
1	Mrs	Cumings, Mrs. John Bradley (Florence Briggs Th...
2	Miss	Heikkinen, Miss. Laina
3	Mrs	Futrelle, Mrs. Jacques Heath (Lily May Peel)
4	Mr	Allen, Mr. William Henry
...	...	...
886	Rev	Montvila, Rev. Juozas
887	Miss	Graham, Miss. Margaret Edith
888	Miss	Johnston, Miss. Catherine Helen "Carrie"
889	Mr	Behr, Mr. Karl Howell
890	Mr	Dooley, Mr. Patrick

Генерация синтетических признаков

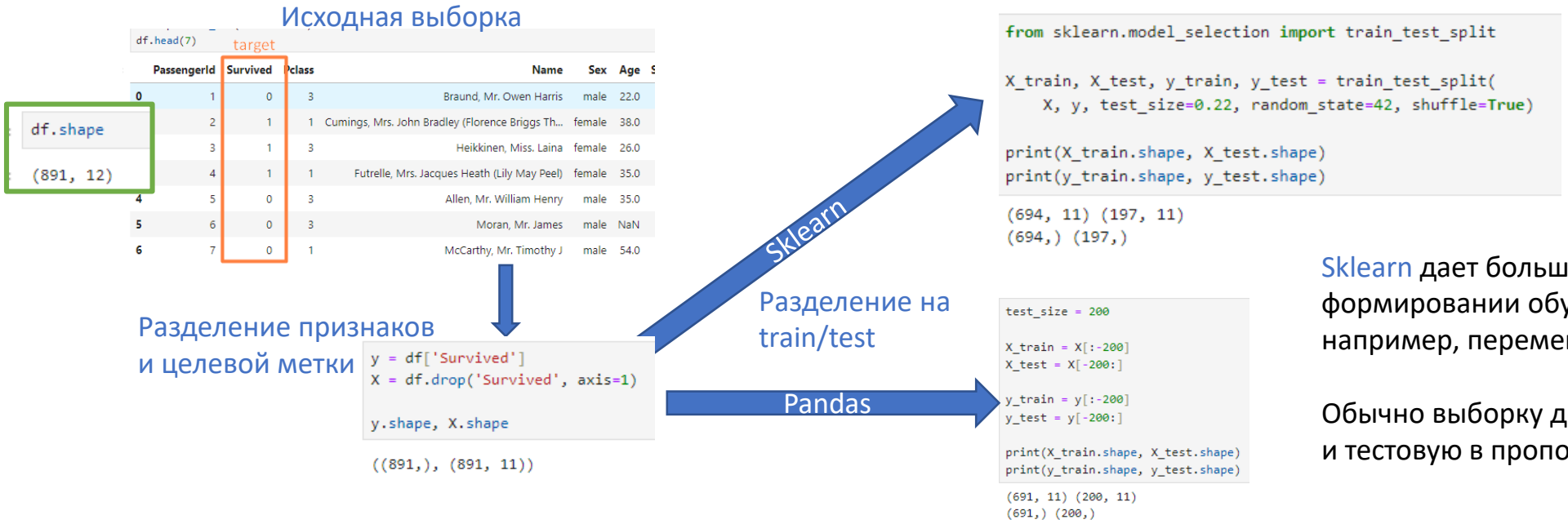
```
from sklearn.preprocessing import PolynomialFeatures

pd.DataFrame(PolynomialFeatures(2)
             .fit_transform(df_train[['Pclass', 'Age']].dropna()),
             columns = [1, 'Pclass', 'Age', 'Pclass**2', 'Age*Pclass', 'Age**2'])
```

	1	Pclass	Age	Pclass**2	Age*Pclass	Age**2
0	1.0	3.0	22.0	9.0	66.0	484.0
1	1.0	1.0	38.0	1.0	38.0	1444.0
2	1.0	3.0	26.0	9.0	78.0	676.0
3	1.0	1.0	35.0	1.0	35.0	1225.0
4	1.0	3.0	35.0	9.0	105.0	1225.0

Обучение с учителем. Разделение выборок

Когда модели обучаются на данных, то они могут настолько хорошо подстроиться под обучающую выборку, что будут показывать отличные результаты, но только на этих данных (переобучение). Но обучение модели нужно ведь для того, чтобы она смогла работать с незнакомыми данными тоже. Поэтому для проверки качества работы модели применяется разделение данных на обучающую (train) и тестовую (test) выборки. Иногда дополнительно вводится разделение тестовой выборки еще на две части для валидации (validation), чтобы на одной части настраивать гиперпараметры модели, а вторую модель видела только на этапе оценки.

Разделение выборки можно производить как стандартными методами Pandas, так и с использованием методов Sklearn.



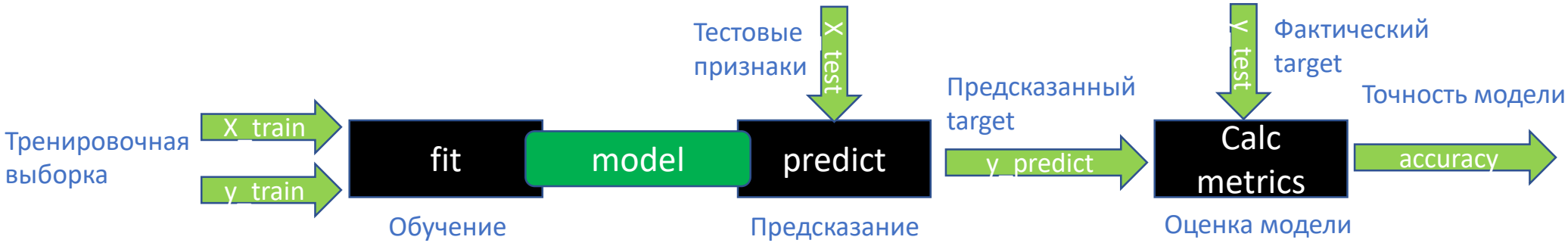
Sklearn дает больше возможностей при формировании обучающих выборок, например, перемешивание (shuffle).

Обычно выборку делят на тренировочную и тестовую в пропорциях 80/20, 70/30 и т.п.

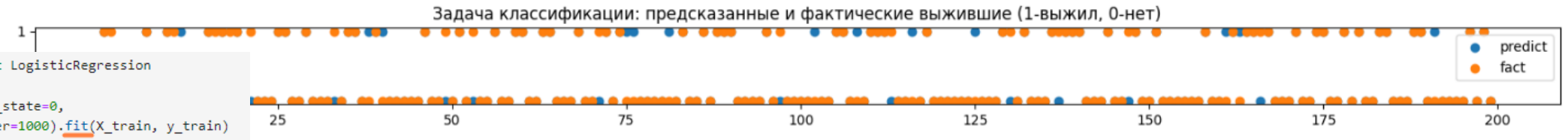
Модели обучения с учителем

Стандартные задачи **обучения с учителем** (Supervised Learning) – классификация и регрессия. Для обучения с учителем в обучающей выборке необходима целевая метка (target) – то, с чем будет сравниваться выход модели и на основании чего будет оцениваться ее работа.

С большинством моделей можно работать при помощи двух методов: fit() и predict().




```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(random_state=0,
                         max_iter=1000).fit(X_train, y_train)
y_pred = clf.predict(X_test)
```




```
from sklearn.linear_model import LinearRegression

linreg = LinearRegression().fit(X_train, y_train)
y_pred = linreg.predict(X_test)
```

Уральский федеральный университет

Метрики оценки качества

Когда мы сделали и обучили модель, всегда возникает вопрос, насколько хорошо работает модель?

Метрики качества отличаются в зависимости от типа модели, типа задачи и бизнес задачи.

В задачах классификации мы оцениваем, угадала модель фактический класс или нет.

В задачах в регрессии мы оцениваем насколько точно модель угадала фактические значения.

В задачах кластеризации необходимо оценить насколько хорошо разделены кластеры.

В задачах NLP часто оценивают схожесть текстов.

Соответственно метрики будут отличаться в зависимости от задачи.

Отдельно можно выделить бизнес-метрики, которые помогают Заказчику понять, какую пользу модель может принести бизнесу.

В этом модуле мы подробнее рассмотрим только метрики для регрессии и классификации.

Уральский федеральный университет

Метрики оценки моделей регрессии

Целевая метка (target) и предсказание (prediction) – целые или вещественные числа, соответственно нам нужно оценить разницу между предсказанием ($\widehat{y_i}$) и фактическим значением (y_i) для всех сэмплов (n) нашей выборки. Для этой цели используют различные метрики, в зависимости от данных и задачи.

- Среднеквадратичная ошибка: $\boldsymbol{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\widehat{y_i} - y_i)^2$
- Корень среднеквадратичной ошибки: $\boldsymbol{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\widehat{y_i} - y_i)^2}$
- Средняя абсолютная ошибка: $\boldsymbol{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\widehat{y_i} - y_i|$
- Коэффициент детерминации: $\boldsymbol{R^2} = 1 - \frac{\sum_{i=1}^{n}(\widehat{y_i}-y_i)^2}{\sum_{i=1}^{n}(y_i-\overline{y})^2}$
- Средняя абсолютная процентная ошибка: $\boldsymbol{MAPE} = 100\% \times \frac{1}{n}\sum_{i=1}^{n}\frac{|\widehat{y_i}-y_i|}{|y_i|}$

Выбор метрики сильно зависит от бизнес-задачи и ваших целей.

ссылка на wiki ИТМО

Уральский федеральный университет

Вычисление метрик регрессии в Sklearn

Sklearn содержит методы для вычисления практически всех распространенных метрик.

В конце концов, вы всегда можете написать свою функцию для вычисления метрик.


```python
from sklearn.metrics import (mean_squared_error, mean_absolute_error,
                             mean_absolute_percentage_error, r2_score)

print(f"MSE: {mean_squared_error(y_test, y_pred)}")
print(f"RMSE: {mean_squared_error(y_test, y_pred, squared=False)}")
print(f"MAE: {mean_absolute_error(y_test, y_pred)}")
print(f"MAPE: {mean_absolute_percentage_error(y_test, y_pred)}")
print(f"R2: {r2_score(y_test, y_pred)}")
```

```
MSE: 1074.8535329645993
RMSE: 32.784958944073715
MAE: 21.16068900067663
MAPE: 738997319933745.9
R2: 0.4269159060827572
```

Уральский федеральный университет

Метрики оценки моделей классификации

Самая простая разновидность классификации – бинарная классификация.

Целевая метка (target) и предсказание (prediction) – бинарный признак, т.е. Да/Нет (1/0, True/False).

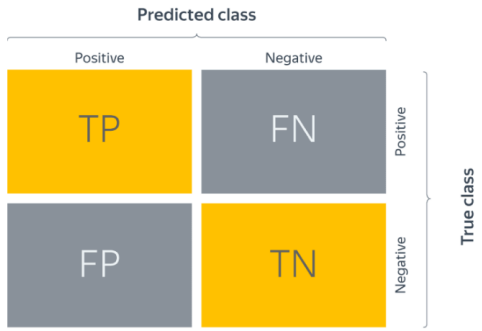
Нам необходимо оценить не только то, сколько классов угадала модель, но какой класс она угадывает лучше, в каких классах она чаще ошибается.

Принято выделять 4 типа предсказаний:

- TP – True Positive (истинно-положительный)
- TN – True Negative (истинно-отрицательный)
- FP – False Positive (ложно-положительный)
- FN – False Negative (ложно-отрицательный)

Для визуализации удобно использовать матрицу ошибок.

Матрица ошибок (Confusion Matrix)

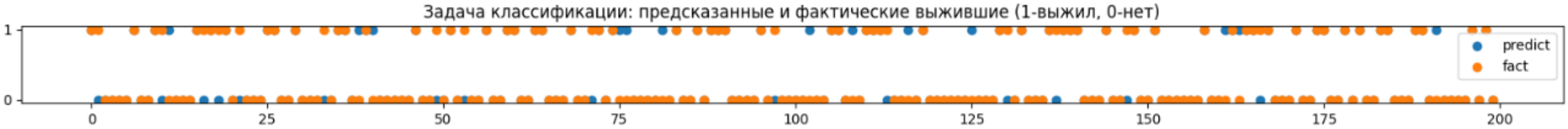


Метрики классификации – это различные отношения TP, FP, TN, FN:

- Точность: $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
- Точность: $Precision = \frac{TP}{TP+FP}$
- Полнота: $Recall = \frac{TP}{TP+FN}$
- F1-мера: $F1score = 2 \times \frac{Recall \times Precision}{Recall+Precision}$
- *ROC-AUC* – метрика, характеризующая то, насколько хорошо модель разделяет классы.

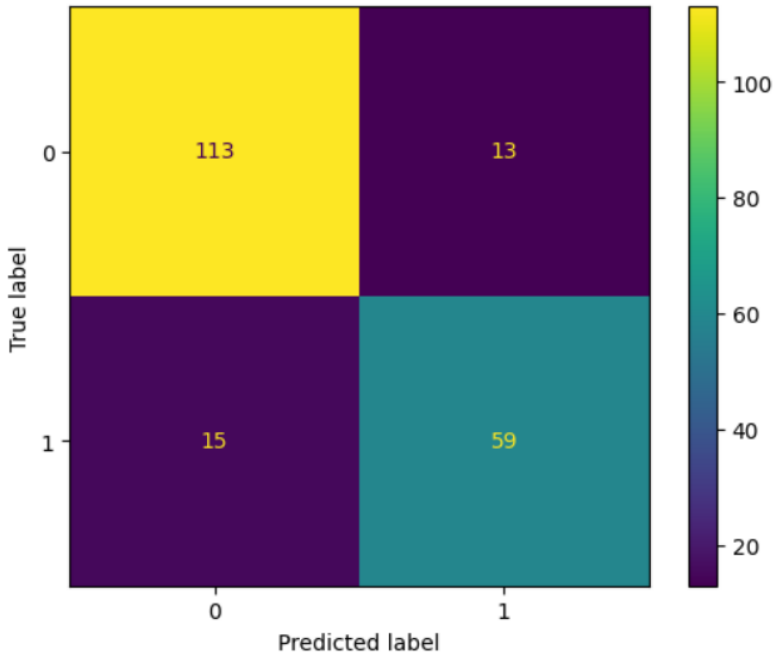
https://academy.yandex.ru/handbook/ml/article/metriki-klassifikacii-i-regressii

Вычисление метрик классификации в Sklearn



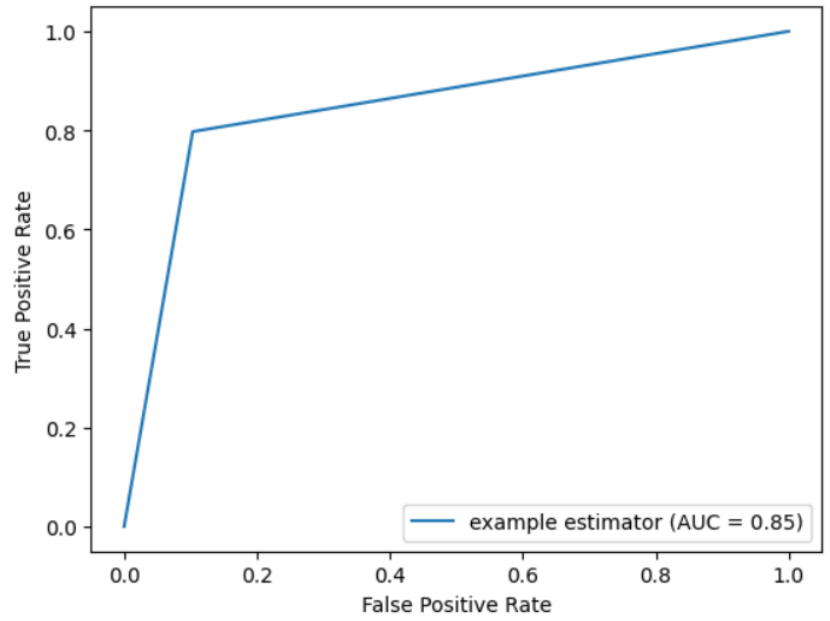

```python
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_pred, labels=clf.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=clf.classes_)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x26
```




```python
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import RocCurveDisplay
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)
display = RocCurveDisplay(fpr=fpr, tpr=tpr, roc_auc=roc_auc,
                          estimator_name='example estimator')
display.plot()
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x26059aab950>
```




```python
from sklearn.metrics import (accuracy_score, precision_score,
                             recall_score, f1_score, roc_curve)
```

```python
print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
print(f"Precision: {precision_score(y_test, y_pred):.2f}")
print(f"Recall: {recall_score(y_test, y_pred):.2f}")
print(f"F1-score: {f1_score(y_test, y_pred):.2f}")
```

```
Accuracy: 0.86
Precision: 0.82
Recall: 0.80
F1-score: 0.81
```

Уральский федеральный университет

Ну вот теперь точно все.
Удачи!