

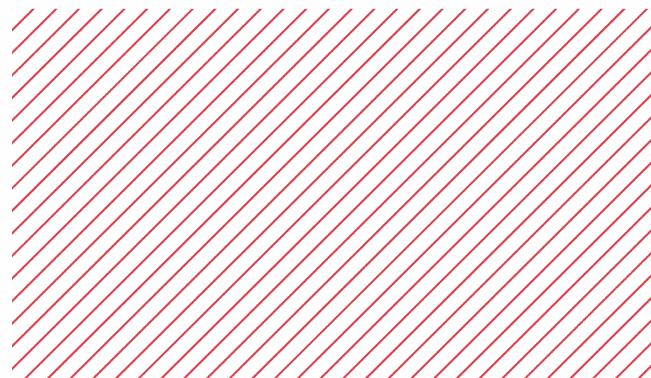
академия
больших
данных



mail.ru
group

Production Python code

Михаил Марюфич, MLE



Q&A

Результаты

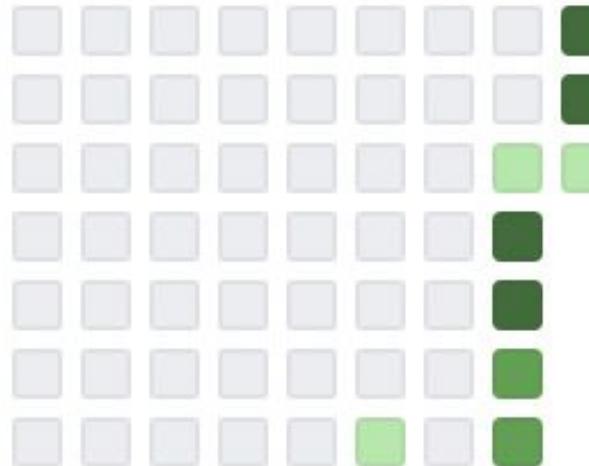
№	00.04.2021 0.27.20	Клещов Андрей Владимирович
132	06.04.2021 10:19:10	Латыпов Ильяс Дамирович
133	06.04.2021 11:32:17	Махмутова Лилия Равильевна
134	06.04.2021 12:44:59	Иванкин Михаил
135	06.04.2021 13:22:36	Ермаков Владимир Николаевич
136	06.04.2021 13:44:16	Горностаев Дмитрий Александрович
137	06.04.2021 14:27:34	Шевчук Алексей
138		
139	06.04.2021 15:45:10	Мурзина Александра Ивановна
140	06.04.2021 17:04:08	Солиев Музаффар Анваржон угли
141	06.04.2021 18:07:22	Юнусов Егор Викторович
142	06.04.2021 18:16:17	Опалёв Роман Сергеевич
143	06.04.2021 18:47:13	Войтешонок Вячеслав Сергеевич
144		

Результаты



Reviewed 76 pull requests in 75 repositories

Feb Mar



Less More



Ошибки

- 1) Создать репозиторий не в зоне <https://github.com/made-ml-in-prod-2021>
- 2) Замерджить без approve
- 3) Написать в readme что-то вроде ololo kek
- 4) Добавиться в форму и сразу после этого написать в чатик

Вопрос

Добрый день! Хотел задать вам небольшой вопрос по первой лекции.

Не совсем понял, чем отличаются ML-инженеры от DS (на ваших слайдах они были упомянуты через /, создалось впечатление, что они являются синонимами в некотором роде).

Я всегда представлял, что ML-инженер -- это Software-инженер, специализирующийся на разработке ML-сервисов или ML-подмодулей программ. Т.е он проводит весь цикл подготовки данных, экспериментов с моделями, но затем реализует это в виде продакшен кода, а не ограничивается добавлением модельки в условный Model Registry, о котором вы рассказывали. Так ли это?

Data Scientist/ML Engineer



- Разработку модели
- За то, что модель готова к эксплуатации
- Развертывание модели +-
- Оценка качества модели онлайн-оффлайн
- Итеративное улучшение моделей

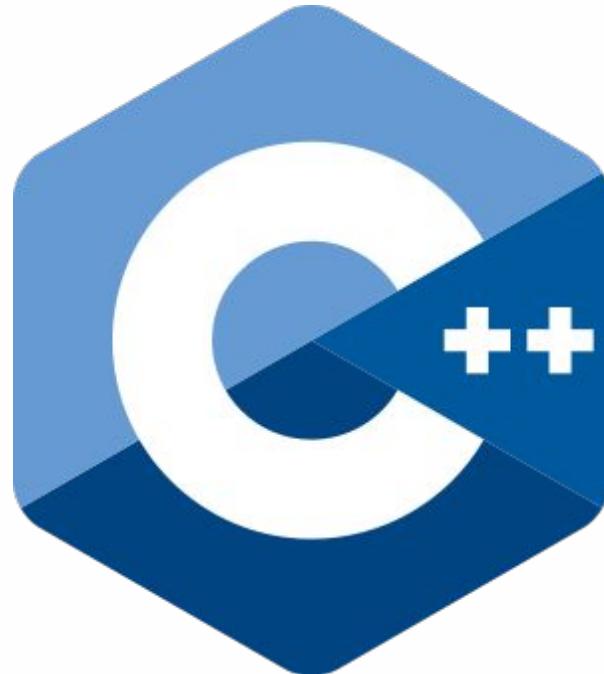
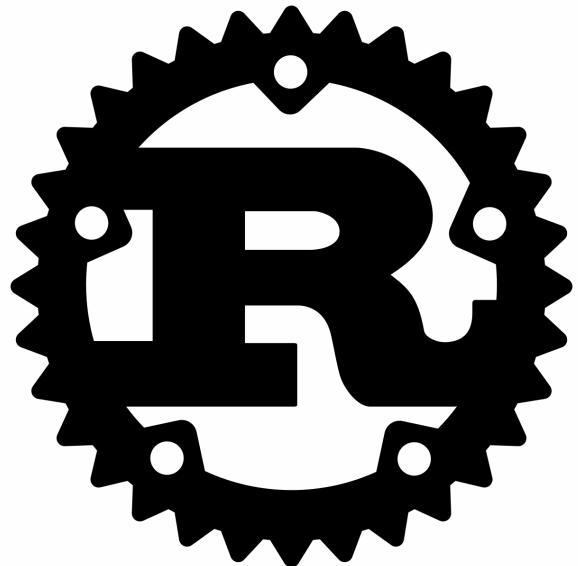
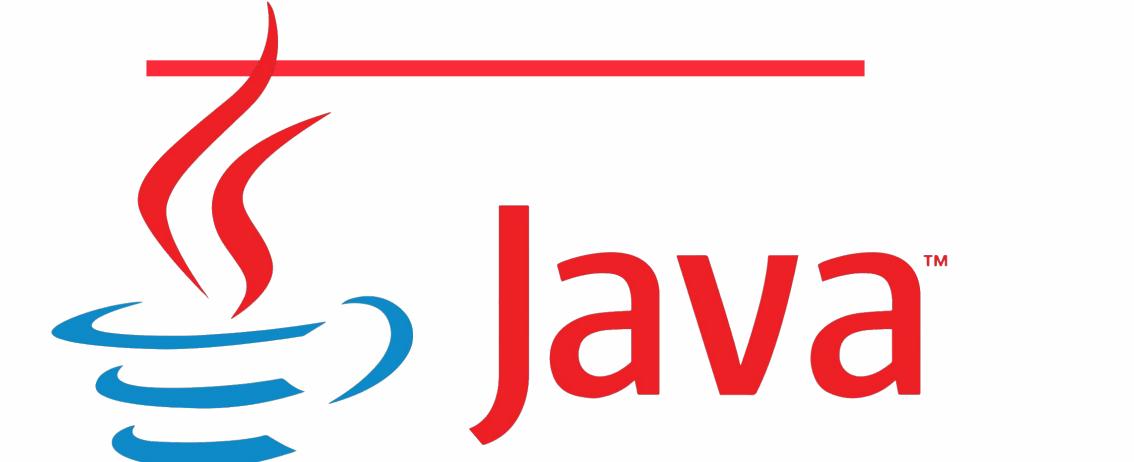


Содержание занятия

- Почему, как и зачем DS используют python?
- Советы по читаемому коду на python
- Организация и дистрибуция кода на python
- Виртуальные окружения и менеджеры зависимостей
- Демо проект с регрессией

Какие языки мы используем
для production?

Языки для production



Python

- Динамическая типизация
- Интерпретируемый



Python

- Динамическая типизация
- Интерпретируемый
- Огромное количество готовых пакетов
- В ML стандарт de facto



XGBoost

P Y T R C H



TensorFlow





Jupyter



Jupyter Notebook

jupyter BOT CLASSIFICATION MODEL -- BEST ATTEMPT Last Checkpoint: 8 минут назад (unsaved changes)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [1]: `import data_processing`

In [2]: `import pandas as pd`

In [3]: `data = pd.read_csv("data/behaviour_public_2018-2019_filtered_raw.csv", "\t")
features = data_processing.generate_features(data)
target = data_processing.get_target(data)
model = BotClassificationPipeline()
model.fit(features, target)`

```
NameError Traceback (most recent call last)
<ipython-input-3-c940d1691b65> in <module>
      2 features = data_processing.generate_features(data)
      3 target = data_processing.get_target(data)
----> 4 model = BotClassificationPipeline()
      5 model.fit(features, target)
```

NameError: name 'BotClassificationPipeline' is not defined

Jupyter Notebook

The screenshot shows a Jupyter Notebook interface with a red header bar. The title bar reads "jupyter lgb-and-cat (автосохранение)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar has icons for file operations and a progress bar set at 0%. The status bar shows "Value" and a scale from 0 to 1000.

In the main area, a cell labeled "B [23]:" contains Python code for training a CatBoost classifier. The code defines training and validation datasets, sets up the classifier with parameters like learning rate, iterations, depth, and eval metric, and fits the model. It also includes a section for tuning all features and a final section for 10000 iterations.

```
train_data = Pool(data=X_train,
                   label=y_train,
                   cat_features=cat_features)

val_data = Pool(data=X_val,
                 label=y_val,
                 cat_features=cat_features)

model2 = CatBoostClassifier(learning_rate=0.15,
                           iterations=500,
                           depth=4,
                           eval_metric="AUC",
                           task_type="GPU",
                           bagging_temperature=0.3,
                           # random_strength=1.5
                           # 12_leaf_reg=20
                           )

model2.fit(train_data, eval_set=val_data)
# preds_class = model.predict(train_data)

# all features no tuning
# 999: learn: 0.7731863 test: 0.7352514 best: 0.7352514 (999) total: 29.6s remaining: 0us
#2000 iter
# 1999: learn: 0.7873610 test: 0.7358354 best: 0.7358915 (1806) total: 57.1s remaining: 0us

# 999: learn: 0.7689705 test: 0.7360432 best: 0.7360432 (999) total: 28.6s remaining: 0us

#10000 iter
# 9999: learn: 0.8155870 test: 0.7375954 best: 0.7377543 (8579) total: 4m 56s remaining: 0us
# 9999: learn: 0.8177055 test: 0.7381214 best: 0.7382481 (7166) total: 4m 47s remaining: 0us
# 9999: learn: 0.8318836 test: 0.7400163 best: 0.7405080 (6557) total: 5m 17s remaining: 0us 0.59522

465: learn: 0.7571411 test: 0.7608129 best: 0.7608129 (465) total: 9.57s remaining: 664ms
466: learn: 0.7571480 test: 0.7608409 best: 0.7608409 (466) total: 9.39s remaining: 663ms
467: learn: 0.7571760 test: 0.7608415 best: 0.7608415 (467) total: 9.42s remaining: 644ms
468: learn: 0.7572022 test: 0.7609403 best: 0.7609403 (468) total: 9.43s remaining: 624ms
469: learn: 0.7572228 test: 0.7608956 best: 0.7609403 (468) total: 9.45s remaining: 603ms
470: learn: 0.7572533 test: 0.7609246 best: 0.7609403 (468) total: 9.47s remaining: 583ms
471: learn: 0.7573009 test: 0.7609762 best: 0.7609762 (471) total: 9.48s remaining: 563ms
472: learn: 0.7573085 test: 0.7610192 best: 0.7610192 (472) total: 9.5s remaining: 542ms
473: learn: 0.7573488 test: 0.7610533 best: 0.7610533 (473) total: 9.52s remaining: 522ms
```

Jupyter Notebook

The screenshot shows a Jupyter Notebook interface with a red header bar. The title bar reads "jupyter lgb-and-cat (автосохранение)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. A toolbar below the menu has icons for file operations like Open, Save, and Run, along with a Python logo icon and a Logout button. A status bar at the bottom right shows "Не доверять" (Do not trust) and "Python 3".

Out[4]:

	card_id	target	delivery_type	addr_region_reg	addr_region_fact	channel_name	channel_name_2	channel_name_modified_2018	sas_limit_after_003_amt	s
0	cid_10620	1	cat_1	107	107	cat_0	cat_3	cat_0	1	
1	cid_105724	0	cat_1	9	9	cat_2	cat_5	cat_2	3	
2	cid_101410	1	cat_1	109	109	cat_0	cat_3	cat_0	1	
3	cid_38961	0	cat_1	66	66	cat_0	cat_3	cat_0	3	
4	cid_57462	0	cat_1	16	16	cat_0	cat_3	cat_0	0	

5 rows x 92 columns

B []:

```
# d = data.groupby("addr_region_reg")['inquiry_1_week'].mean().to_dict()
# data['inquiry_1_week_mean'] = data.addr_region_reg.apply(lambda x: d[x])

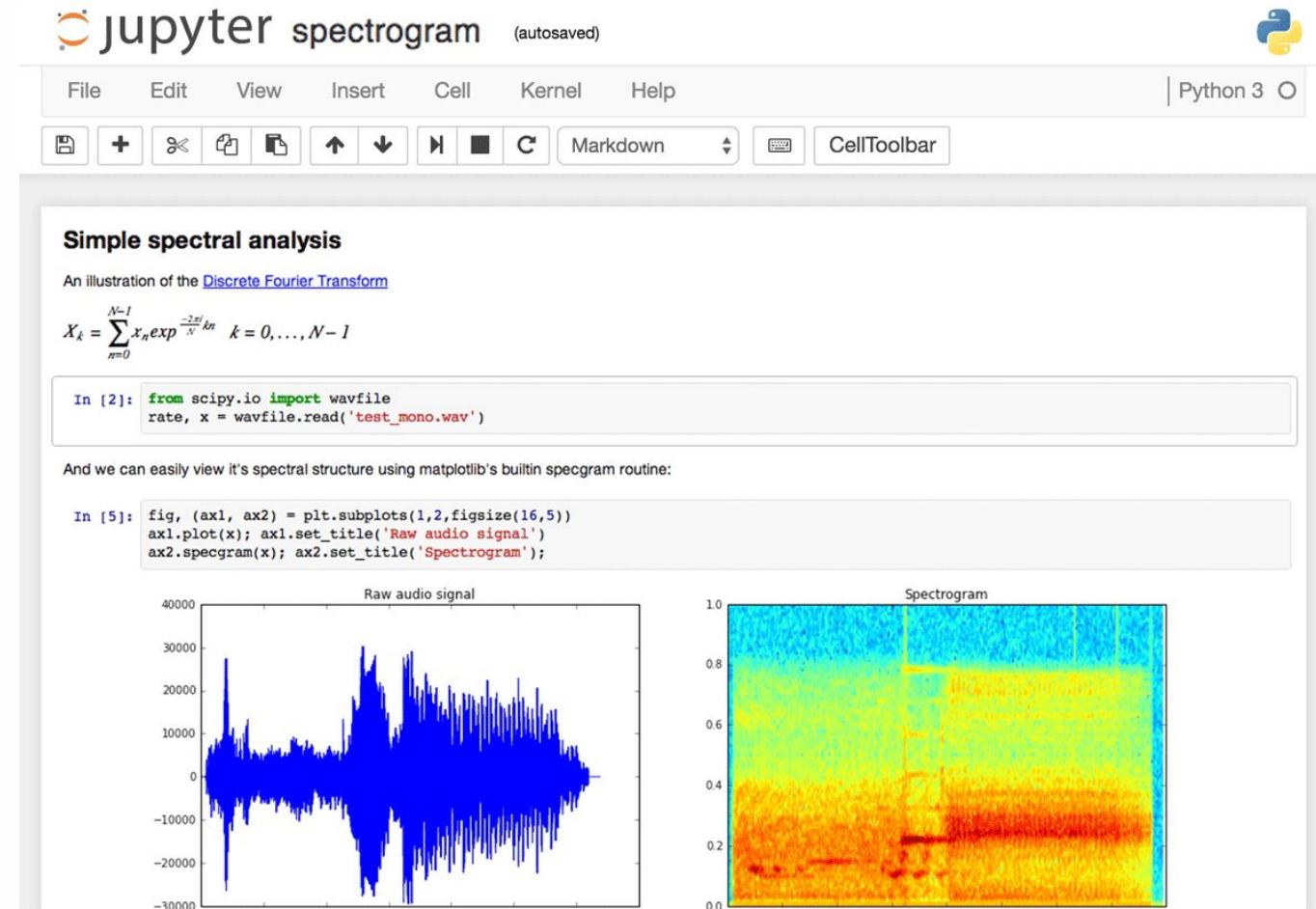
# data["inquiry_old_count"] = data.ttl_inquiries - data.inquiry_12_month
data["ttl_inquiries_with_current"] = data.ttl_inquiries.fillna(0) + data.inquiry_recent_period.fillna(0)
data["inquiry_recent_period_perc"] = data.inquiry_recent_period.fillna(0) / (data.ttl_inquiries_with_current.fillna(0)+data["loans_period"])
data["loans_period"] = data.last_loan_date.fillna(0)-data.first_loan_date.fillna(0)
# data["loans_close_percent"] = (data.loans_main_borrower+data.loans_active)/(data.loans_active+0.001)
data["loans_close_times"] = data.loans_main_borrower.fillna(0) - data.loans_active.fillna(0)
# data["sas_limit_last_amt_delta"] = data.sas_limit_after_003_amt.fillna(0)==data.sas_limit_last_amt.fillna(0)

data["region_same"] = data.addr_region_reg.fillna(0)==data.addr_region_fact.fillna(0)
# data["region_same2"] = data.addr_region_reg==data.app_addr_region_reg
# data["region_same3"] = data.addr_region_fact==data.app_addr_region_fact

data['age'] = data.cltnt_birth_year.fillna(0)*365-data.first_loan_date.fillna(0)
data['age2'] = data.cltnt_birth_year.fillna(0)*365-data.last_loan_date.fillna(0)
data['f1'] = data.cltnt_income_month_avg_net_amt.fillna(0)-data.sas_limit_after_003_amt.fillna(0)
data['f2'] = data.loans_main_borrower.fillna(0)/(data.loans_period.fillna(0)+0.001)
data['f3'] = data.loans_active.fillna(0)/(data.loans_main_borrower.fillna(0)+0.001)
data["feature_20_21_27"] = data.feature_20.fillna(0)+data.feature_21.fillna(0)+data.feature_27.fillna(0)
data["feature_22_23_24_25_28_12_10_13"] = data.feature_13.fillna(0)+data.feature_10.fillna(0)+ data.feature_12.fillna(0)
```

Jupyter Notebook - достоинства

- Можно строить графики
- Удобно делать отчеты
- Удобно изучать новые пакеты
- Прост в установке
- Легко развернуть на удаленном сервере



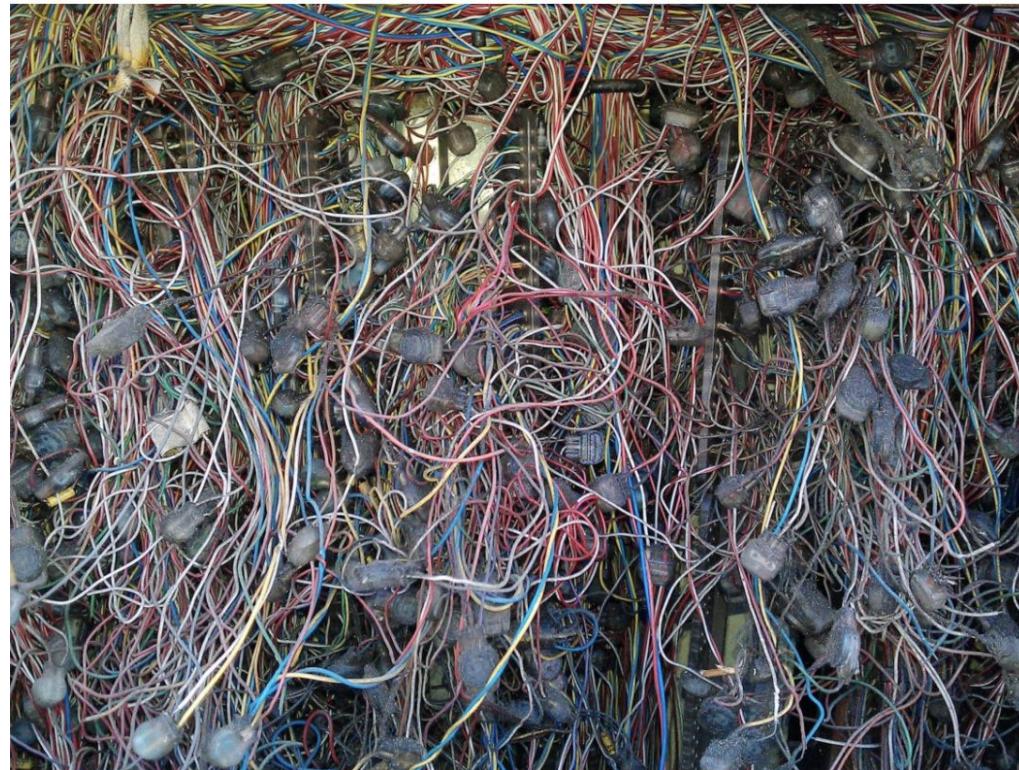
Jupyter Notebook - недостатки

- Проблемы с версионированием в гите
- Слабая IDE
- Сложно тестировать код
- Позволяет выполнять ячейки в произвольном порядке
-
- Плохой пользовательский опыт при запуске длительных вычислений

5 reasons why jupyter notebooks suck



Alexander Mueller Mar 24, 2018 · 3 min read ★



How it feels like managing jupyter notebooks (Complexity ©
<https://www.flickr.com/photos/bitterjug/7670055210>)

Еще про ноутбуки

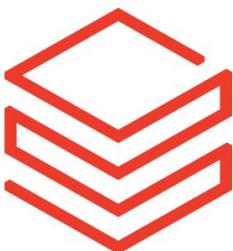
Яндекс Облако

Yandex
DataSphere



Amazon
SageMaker

aws



databricks

Еще про ноутбуки

JetBrains DataSpell

A new IDE by JetBrains for data science. Join the preview!



Native notebook experience

A redesigned user interface based on the popular Jupyter Notebook experience, including successful standards such as physical input cells, inline outputs, and edit/command modes.

Smart coding assistance

Combine the advantages of Jupyter Notebooks with the benefits of code completion, error checking, debugging capabilities, and much more.

A screenshot of the JetBrains DataSpell IDE. The interface is dark-themed. On the left, there's a sidebar with a 'Workspace' tree view containing files like 'nbestimate_2.ipynb', 'estimate.ipynb', and 'LICENSE'. Below it is a 'Structure' panel showing a tree for 'Estimate of Public Jupyter Notebooks' with nodes for 'Data Collection', 'Assumptions', 'Raw Hits', 'Change', 'Prediction', and 'Milestones'. The main area shows a Jupyter notebook titled 'estimate.ipynb' with code cells. Cell 28 contains: 'combined_df = pd.concat([cleaned_hits_df.reset_index(drop=True).rename(columns={'hits': 'y'}), forecast_df], axis=1)'. Cell 29 contains: 'rows = [] cols = {'y': 'actual', 'yhat_upper': 'optimistic', 'yhat': 'predicted', 'yhat_lower': 'conservative'} for i in range(1, 11): milestone = i * 1e6 row = {'milestone': milestone}'. Cell 30 contains: 'pd.DataFrame(rows, columns=['milestone']+list(cols.keys())).rename(columns=cols)'. Below the code is a table titled 'Out 30' with four rows of data:

	milestone	actual	optimistic	predicted	conservative
0	1000000.0	2017-06-19	2017-06-01	2017-06-06	2017-06-11
1	2000000.0	2018-04-04	2018-03-30	2018-04-02	2018-04-04
2	3000000.0	2018-11-01	2018-10-12	2018-10-14	2018-10-16
3	4000000.0	2019-03-27	2019-03-26	2019-03-28	2019-03-29

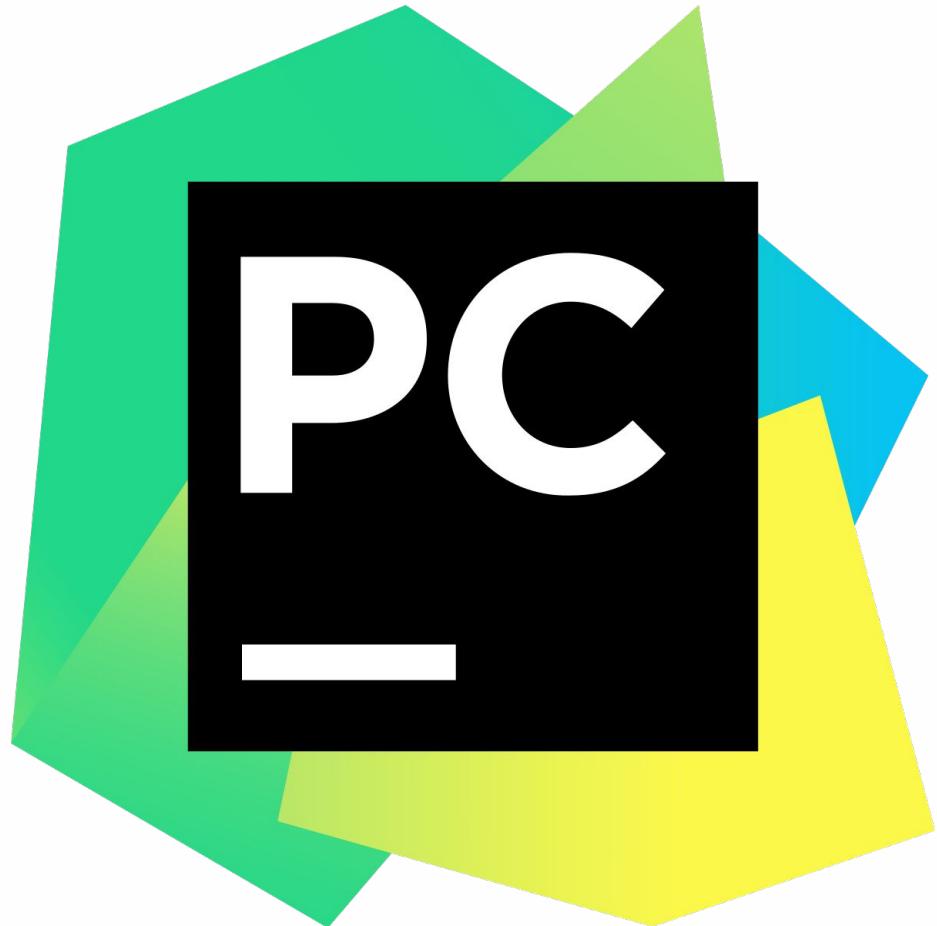
The bottom status bar shows 'Terminal Python Packages R Console Python Console Jupyter Server started at http://localhost:8888 // Open in Browser (6 minutes ago)' and '184:1 LF UTF-8 4 spaces Python 3.7 (base)'.

Еще про ноутбуки

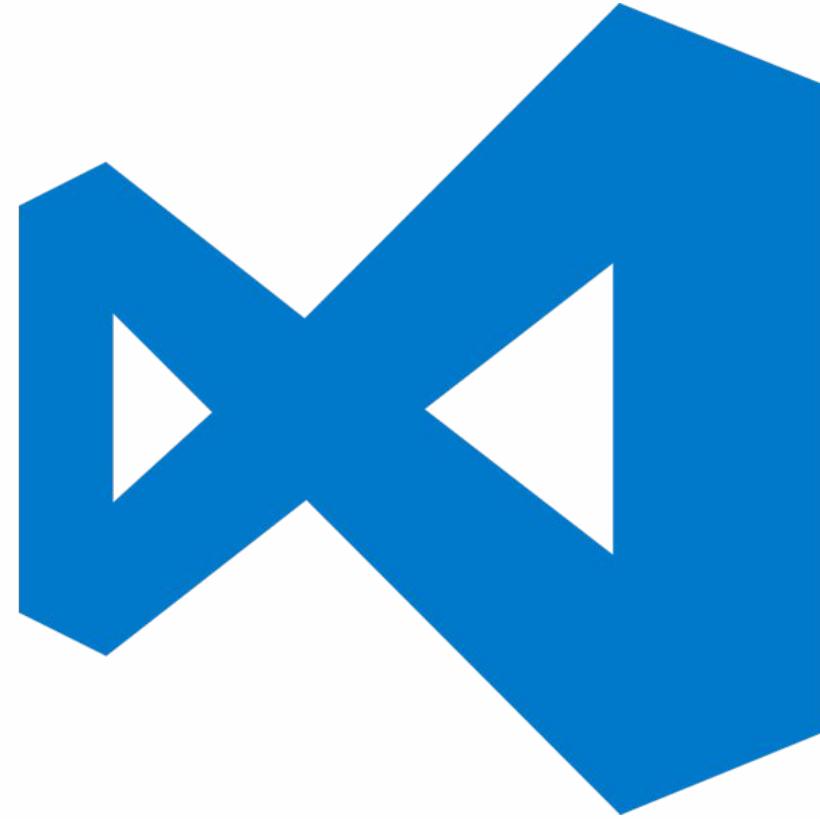
NETFLIX



В чем же нам тогда
программировать?



PyCharm



VS CODE



Характеристики продакшн кода

Качественный продакшн код должен обладать следующими характеристиками (top-5):

- Читаемый
- Эффективный
- Модульный
- Содержащий тесты и прописанные exceptions
- Задокументированный
-

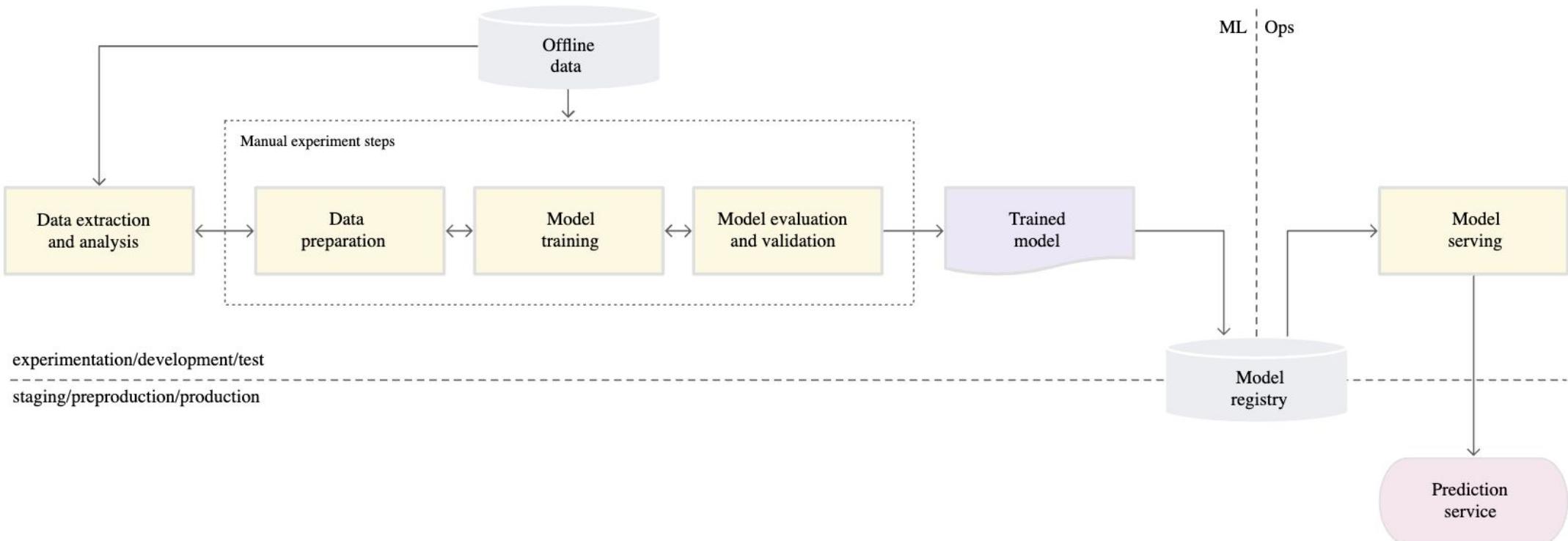
Характеристики продакшн проекта:

- **Версионирование**
- **Удобная структура**
- **Явные зависимости**

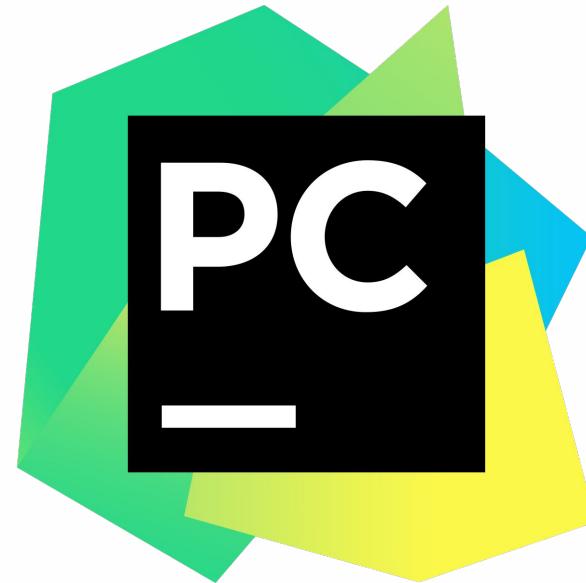
MLOPS Level 0



DS делает модели



Типичная картина



Код построения модели

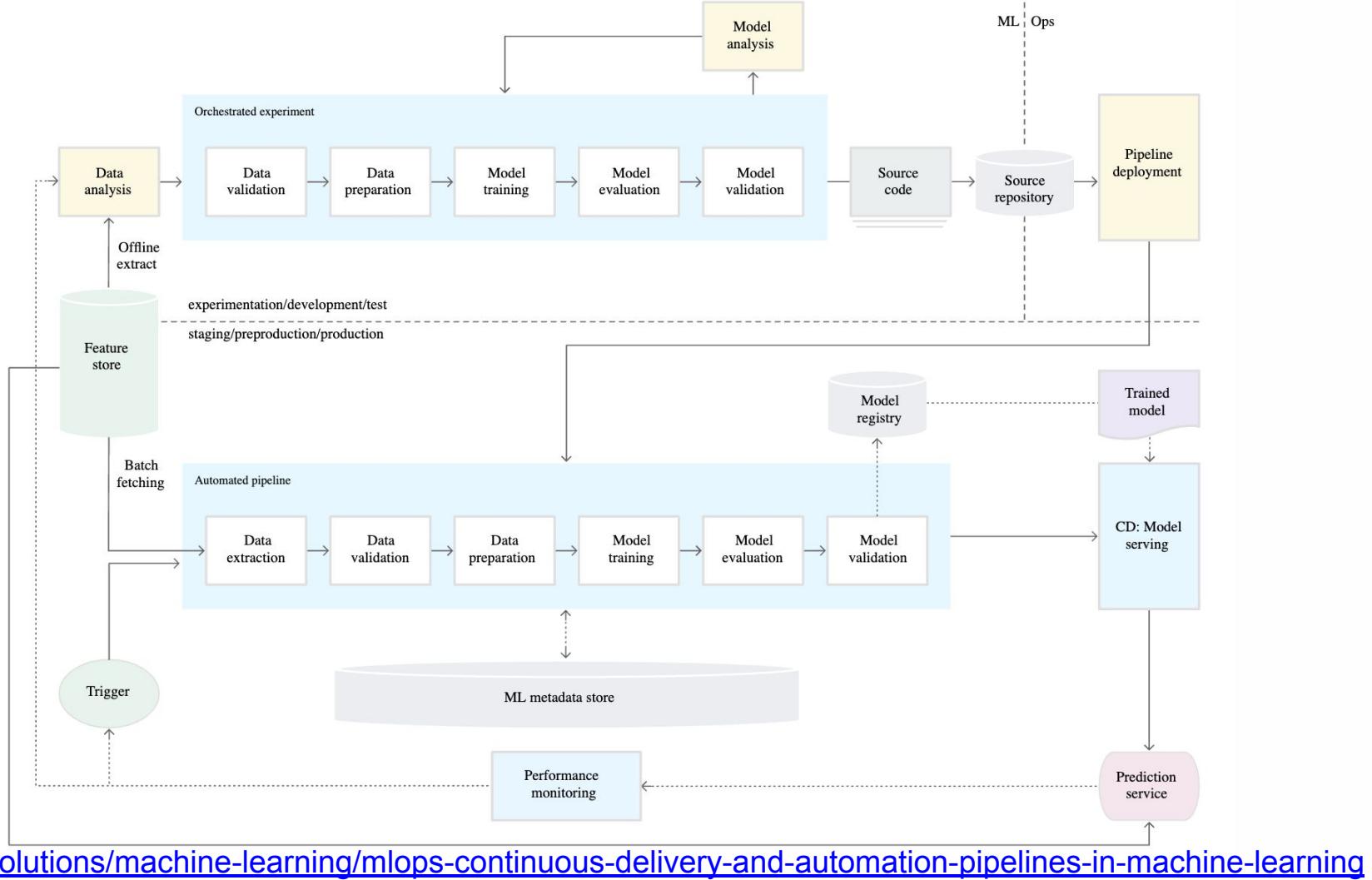
- не поревьюен
- не воспроизводим
- не обобщаемый

Код использования модели

- Написан бэкендером
- Он норм

MLOPS Level 1

Automated pipeline = prod code



Поговорим про код

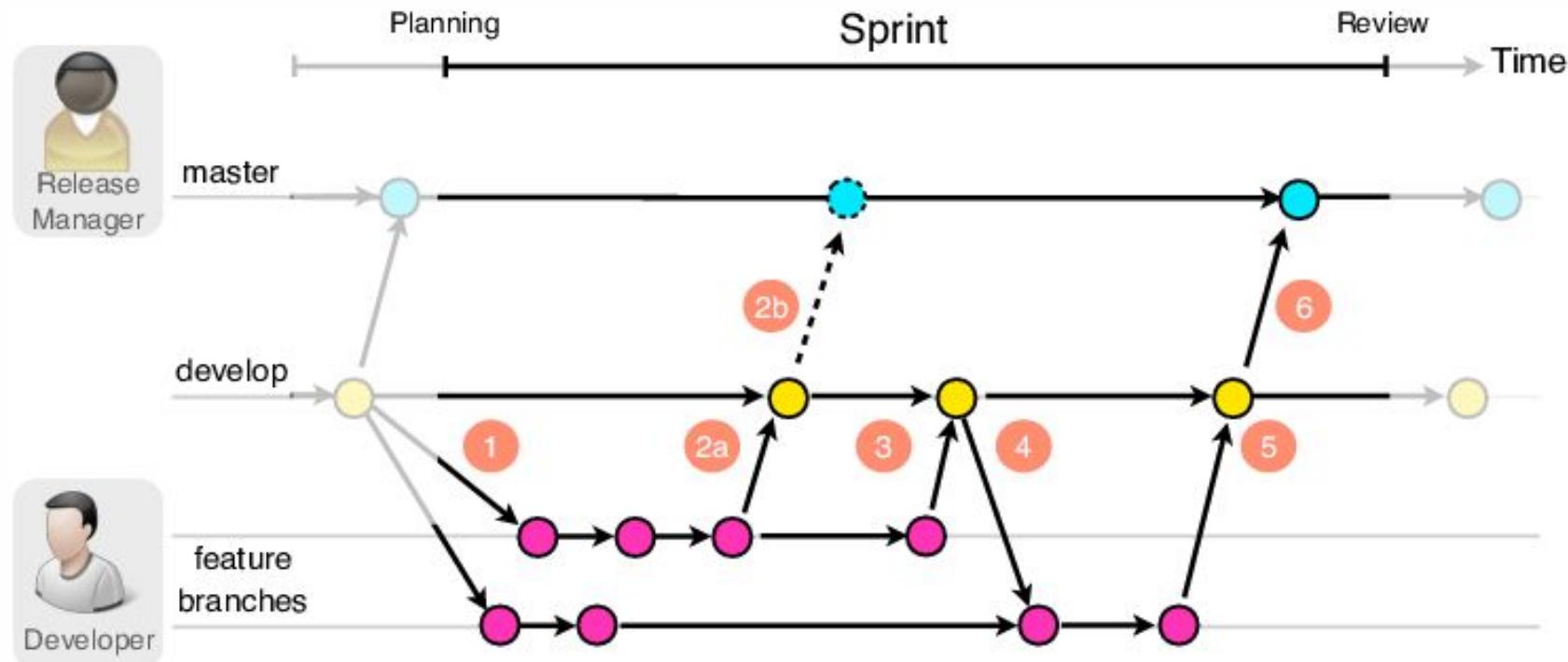
Читаемый код



Ревью

Репозиторий

Создаем репозиторий до начала работы, а не в её конце!



PULL REQUEST



Читаемые названия

```
for i in range(n):
    for j in range(m):
        for k in range(l):
            temp_value = X[i][j][k] * 12.5
            new_array[i][j][k] = temp_value + 150
```



Читаемые названия

```
for i in range(n):
    for j in range(m):
        for k in range(l):
            temp_value = X[i][j][k] * 12.5
            new_array[i][j][k] = temp_value + 150
```

- магические константы
- короткие/непонятные/не привязанные к задаче имена переменных

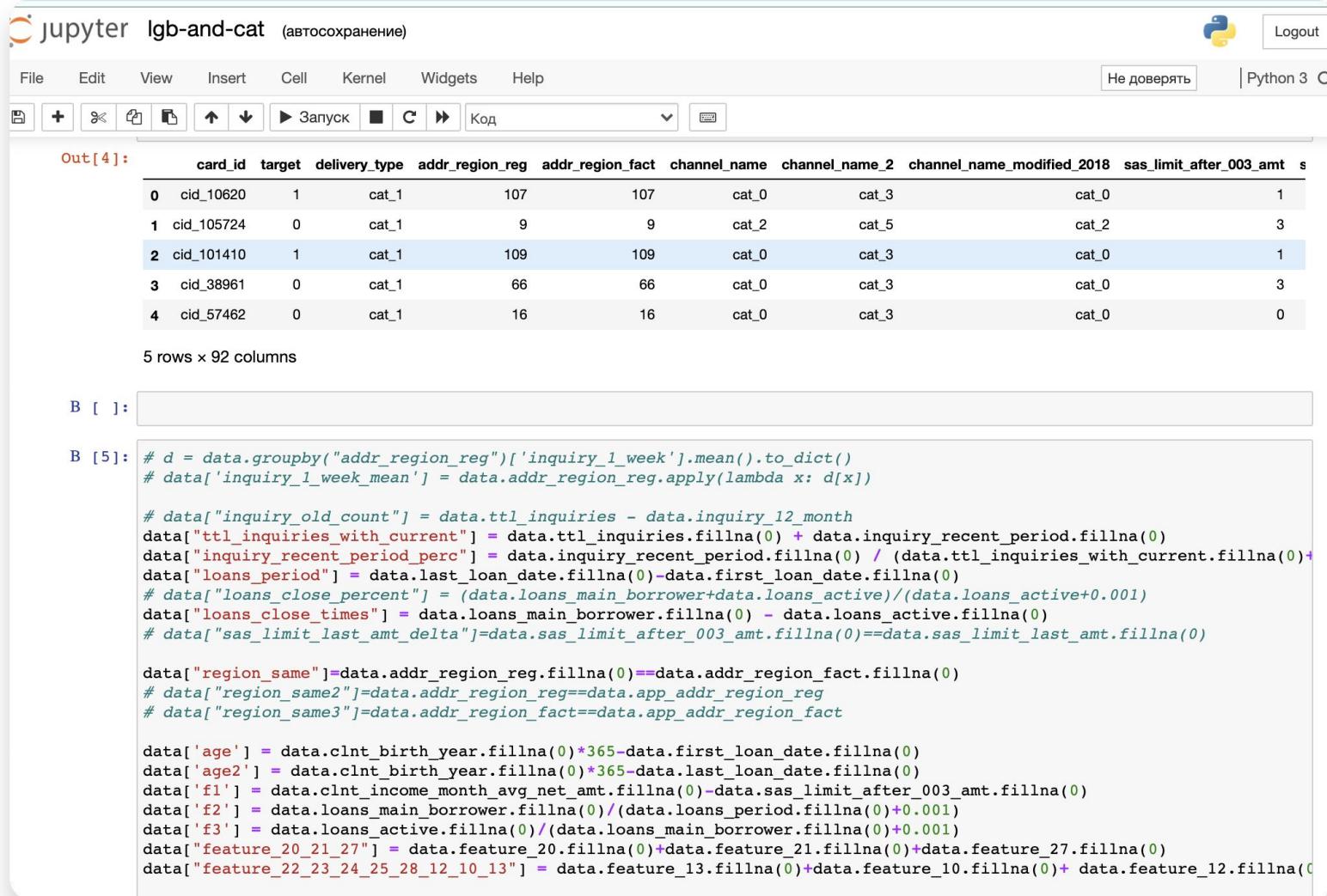
Читаемые названия

```
PIXEL_NORMALIZATION_FACTOR = 12.5
```

```
PIXEL_OFFSET_FACTOR = 150
```

```
for row_index in range(row_count):
    for column_index in range(column_count):
        for color_channel_index in range(color_channel_count):
            normalized_pixel_value = (
                original_pixel_array[row_index][column_index][color_channel_index]
                * PIXEL_NORMALIZATION_FACTOR
            )
            transformed_pixel_array[row_index][column_index][color_channel_index] = (
                normalized_pixel_value + PIXEL_OFFSET_FACTOR
            )
```

Нет портнякам



The screenshot shows a Jupyter Notebook interface with the title "jupyter lgb-and-cat (автосохранение)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. A toolbar below the menu has icons for file operations and cell execution. The status bar indicates "Не доверять" (Do not trust) and "Python 3".

Cell 4 output:

	card_id	target	delivery_type	addr_region_reg	addr_region_fact	channel_name	channel_name_2	channel_name_modified_2018	sas_limit_after_003_amt	s
0	cid_10620	1	cat_1	107	107	cat_0	cat_3	cat_0	1	
1	cid_105724	0	cat_1	9	9	cat_2	cat_5	cat_2	3	
2	cid_101410	1	cat_1	109	109	cat_0	cat_3	cat_0	1	
3	cid_38961	0	cat_1	66	66	cat_0	cat_3	cat_0	3	
4	cid_57462	0	cat_1	16	16	cat_0	cat_3	cat_0	0	

5 rows × 92 columns

Cell 5 output:

```
B [ ]:
```

```
B [5]: # d = data.groupby("addr_region_reg")['inquiry_1_week'].mean().to_dict()
# data['inquiry_1_week_mean'] = data.addr_region_reg.apply(lambda x: d[x])

# data["inquiry_old_count"] = data.ttl_inquiries - data.inquiry_12_month
data["ttl_inquiries_with_current"] = data.ttl_inquiries.fillna(0) + data.inquiry_recent_period.fillna(0)
data["inquiry_recent_period_perc"] = data.inquiry_recent_period.fillna(0) / (data.ttl_inquiries_with_current.fillna(0)+data["loans_period"])
data["loans_close_percent"] = (data.loans_main_borrower+data.loans_active)/(data.loans_active+0.001)
data["loans_close_times"] = data.loans_main_borrower.fillna(0) - data.loans_active.fillna(0)
# data["sas_limit_last_amt_delta"] = data.sas_limit_after_003_amt.fillna(0) == data.sas_limit_last_amt.fillna(0)

data["region_same"] = data.addr_region_reg.fillna(0) == data.addr_region_fact.fillna(0)
# data["region_same2"] = data.addr_region_reg == data.app_addr_region_reg
# data["region_same3"] = data.addr_region_fact == data.app_addr_region_fact

data['age'] = data.clnt_birth_year.fillna(0)*365 - data.first_loan_date.fillna(0)
data['age2'] = data.clnt_birth_year.fillna(0)*365 - data.last_loan_date.fillna(0)
data['f1'] = data.clnt_income_month_avg_net_amt.fillna(0) - data.sas_limit_after_003_amt.fillna(0)
data['f2'] = data.loans_main_borrower.fillna(0)/(data.loans_period.fillna(0)+0.001)
data['f3'] = data.loans_active.fillna(0)/(data.loans_main_borrower.fillna(0)+0.001)
data["feature_20_21_27"] = data.feature_20.fillna(0)+data.feature_21.fillna(0)+data.feature_27.fillna(0)
data["feature_22_23_24_25_28_12_10_13"] = data.feature_13.fillna(0)+data.feature_10.fillna(0)+ data.feature_12.fillna(0)
```

Нет портянкам

- Используйте функции и классы, они помогут сделать код обозримым

```
def run_training() → NoReturn:  
    data = load_dataset(file_name=config.TRAINING_DATA_FILE)  
  
    X_train, X_test, y_train, y_test = train_test_split(  
        data[config.FEATURES], data[config.TARGET], test_size=0.1, random_state=0  
    )  
  
    pipeline.fit(X_train[config.FEATURES], y_train)  
  
    _logger.info(f"saving model version: {_version}")  
    save_pipeline(pipeline_to_persist=pipeline.price_pipe)
```



Маленькие функции

- Если функция занимает более 50 строк, то вероятно, её лучше разбить
- Если вы копируете код - пора создать функцию. (но можно и раньше)



SOLID

- The Single-responsibility principle: a class should only have a single responsibility
- The Open–closed principle: software entities ... should be open for extension, but closed for modification.
- The Liskov substitution principle: objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program
- The Interface segregation principle: many client-specific interfaces are better than one general-purpose interface.
- The Dependency inversion principle: depend upon abstractions, [not] concretions

Работа с ресурсами

```
f = open("hello.txt", "w")
f.write("hello!")
f.close()
```

Работа с ресурсами

```
f = open("hello.txt", "w")
try:
    f.write("hello!")
finally:
    f.close()
```



Работа с ресурсами

```
with open("hello.txt", "w") as f:  
    f.write("hello!")
```



Typing

```
def fit_on_data(model, features, target):  
    model.fit(features.to_numpy(), target.to_numpy())  
    return model
```

Typing

```
def fit_on_data(model, features, target):  
    model.  
    mode  
    retu  
        if  
        ifn  
        ifnn  
        len  
            .  
            .  
            .
```



Typing

```
def fit_on_data_typing(  
    model: ForestClassifier, features: pd.DataFrame, target: pd.DataFrame  
) -> ForestClassifier:  
    model.fit(features.to_numpy(), target.to_numpy())  
    return model
```

Typing

```
def fit_on_data_typing(
    model: ForestClassifier, features: pd.DataFrame, target: pd.DataFrame
) -> ForestClassifier:
    model.
    mode|m| fit(self, X, y, samp...      BaseForest | numpy())
    retu|f| estimators_                 BaseForest
        |f| verbose                   BaseForest
        if                         if expr
        ifn                        if expr is None
    m| apply(self, X)               BaseForest
    f| bootstrap                  BaseForest
    f| class_weight                BaseForest
    f| classes_                   ForestClassifier
```

Typing

```
def fit_on_data_typing(  
    model: ForestClassifier, features: pd.DataFrame, target  
) -> ForestClassifier:  
    features.  
    model.f m to_numpy(self, dtype,... DataFrame npy())  
    return m info DataFrame  
        m join(self, other, on,... DataFrame  
        m append(self, other, i... DataFrame  
        p shape DataFrame  
        .
```

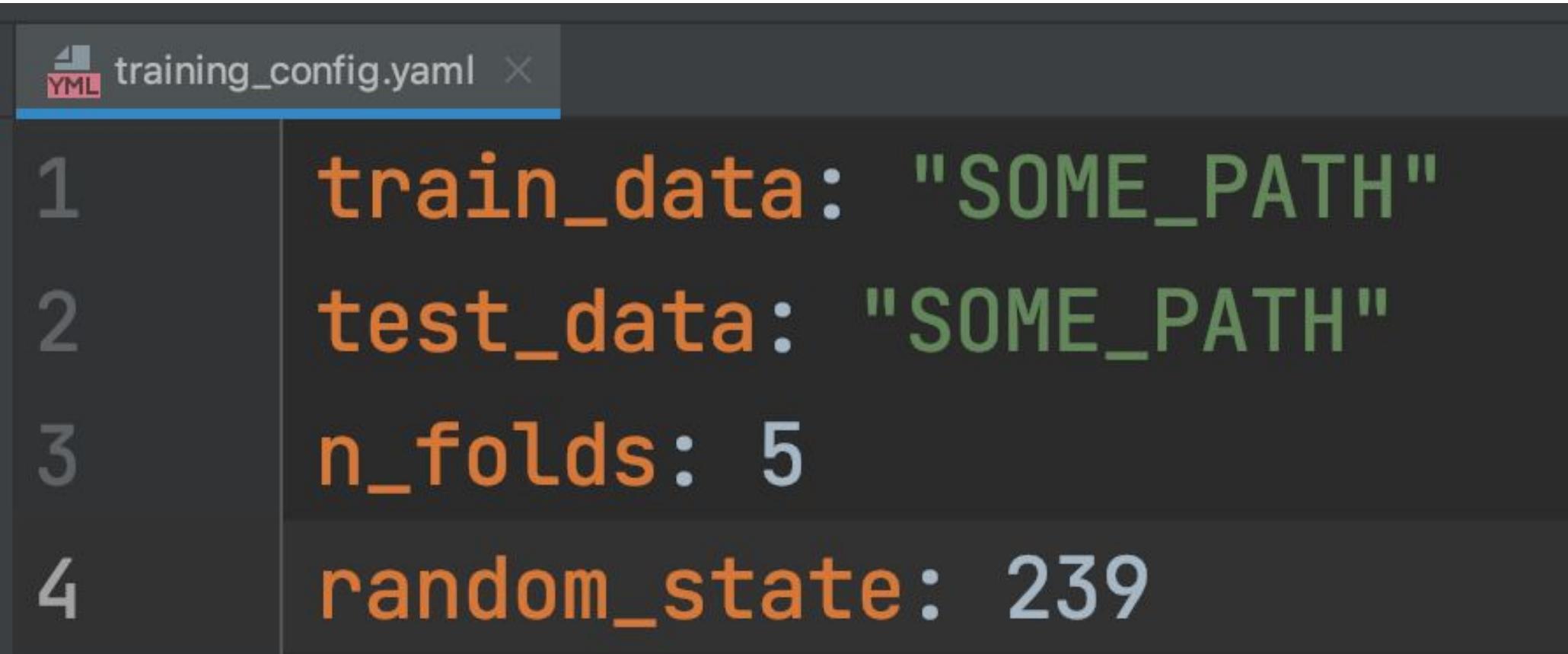
Проверка типов в python



https://mypy.readthedocs.io/en/stable/cheat_sheet_py3.html

Конфигурации

Конфиг в yaml



The image shows a screenshot of a code editor with a dark theme. A file named "training_config.yaml" is open. The file contains four lines of YAML configuration data:

```
1 train_data: "SOME_PATH"
2 test_data: "SOME_PATH"
3 n_folds: 5
4 random_state: 239
```



Считывание конфига из yaml

```
def read_config(config_path: str) -> Dict[str, Union[int, str]]:  
    with open(config_path, "r") as input_stream:  
        config = yaml.safe_load(input_stream)  
    return config
```

Используем конфиги из yaml

```
def test_read_config(config_path: str):
    config = read_config(config_path)
    assert config["n_folds"] == 5
    assert config["test_data"]
    all_fields = ["n_folds", "test_data", "train_data", "random_state"]

    assert set(all_fields) == set(config.keys())
```

DataClasses

```
@dataclass
class TrainingParams:
    train_data: str
    test_data: str
    random_state: int
    n_folds: int = field(default=5)

TrainingParamsSchema = class_schema(TrainingParams)

def read_config_to_dataclass(config_path: str) -> TrainingParams:
    params_dict = read_config(config_path)
    return TrainingParamsSchema().load(params_dict)
```

DataClasses

```
def test_read_config_dataClass(config_path: str):  
    params: TrainingParams = read_config_to_dataclass(config_path)  
    assert params.n_folds == 5  
    assert params.~
```

• n_folds

TrainingParams

• test_data

TrainingParams

par

(expr)

Python Object Serialization libraries



serpyco ↗

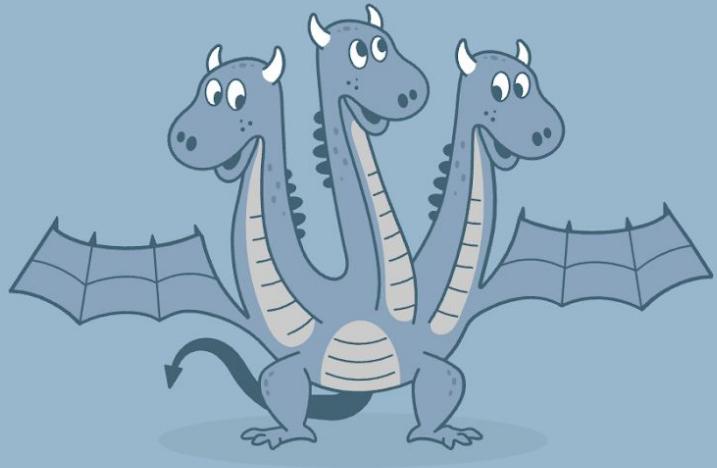
Project ID: 9168810

⌚ 256 Commits ⚡ 3 Branches 🏷 50 Tags 📁 1.5 MB Files 📂 321.5 MB Storage

Fast serialization of dataclasses using Cython.

Посмотрим на конфиг

Hydra



Hydra

A framework for elegantly configuring complex applications

[Get Started](#)

Star 4,001

<https://hydra.cc/docs/intro/>

Форматирование и codestyle

Python convention

Naming Styles

The table below outlines some of the common naming styles in Python code and when you should use them:

Type	Naming Convention	Examples
Function	Use a lowercase word or words. Separate words by underscores to improve readability.	function, my_function
Variable	Use a lowercase single letter, word, or words. Separate words with underscores to improve readability.	x, var, my_variable
Class	Start each word with a capital letter. Do not separate words with underscores. This style is called camel case.	Model, MyClass
Method	Use a lowercase word or words. Separate words with underscores to improve readability.	class_method, method
Constant	Use an uppercase single letter, word, or words. Separate words with underscores to improve readability.	CONSTANT, MY_CONSTANT, MY_LONG_CONSTANT
Module	Use a short, lowercase word or words. Separate words with underscores to improve readability.	module.py, my_module.py
Package	Use a short, lowercase word or words. Do not separate words with underscores.	package, mypackage



Строки в питоне

```
s1 = "QWERTY"  
s2 = 'QWERTY'
```



Строки в питоне

```
s1 = "QWERTY"  
s2 = 'QWERTY'
```

Всегда используйте что-то одно

Форматирование строк в питоне

```
def test_string_format():
    name = "Misha"
    s0 = "Hello, " + name
    s1 = "Hello, %s" % name # classic
    s2 = "Hello, {}".format(name) # py3 format
    s3 = f"Hello, {name}" # fstring

    assert s0 == s1 == s2 == s3
```

PEP8



Real Python

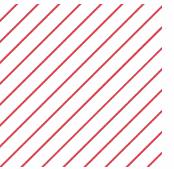
<https://www.python.org/dev/peps/pep-0008/>

BLACK

```
def very_important_function(template: str, *variables, file: os.PathLike,
                           engine: str, header: bool = True, debug: bool = False):
    """Applies `variables` to the `template` and writes to `file`."""
    with open(file, 'w') as f:
        ...

# out:

def very_important_function(
    template: str,
    *variables,
    file: os.PathLike,
    engine: str,
    header: bool = True,
    debug: bool = False,
):
    """Applies `variables` to the `template` and writes to `file`."""
    with open(file, "w") as f:
        ...
```



BLACK

```
# in
j = [1,
     2,
     3

λ
|
# out:

j = [1, 2, 3]
```



BLACK

```
# in  
  
ImportantClass.important_method(exc, limit, lookup_lines, capture_locals, extra_argument)  
  
# out:  
  
ImportantClass.important_method(  
    exc, limit, lookup_lines, capture_locals, extra_argument  
)
```

Логирование

Print vs Logging

```
def some_function_with_print():
    print("start doing")
    try:
        do_stuff()
    except RuntimeError as e:
        print(e)
    print("end doing")
```

Print vs Logging

```
def some_function_with_logging():
    logging.info("start doing")
    try:
        do_stuff()
    except RuntimeError as e:
        logging.error(e)
    logging.info("end doing")
```

Kibana

52,505 hits

New Save Open Share 10 seconds Last 7 days Search... (e.g. status:200 AND extension:PHP) Uses lucene query syntax Search

Add a filter +

Selected Fields

- ? _source
- Available Fields**

 - Popular
 - t request

- @timestamp
- t @version
- t _id
- t _index
- # _score
- t _type
- t agent

May 27th 2018, 12:29:23.986 - June 3rd 2018, 12:29:23.986 — Auto

Count

@timestamp per 3 hours

Time	_source
June 3rd 2018, 12:26:56.000	source: /var/log/nginx/pawel-blog-access.log host: pawel-blog name: Other beat.hostname: pawel-blog beat.version: 6.2.4 beat.name: pawel-blog verb: HEAD request: / httpversion: 1.1 os: Other os_name: Other @timestamp: June 3rd 2018, 12:26:56.000 device: Other clientip: 108.162.229.239 @version: 1 tags: beats_input_codec_plain_applied many�-many http:// 0 many 1 string: 48 863 many timestamp: Fri

PLEASE, DON'T

```
print = logging.info
|
def some_function_with_print():
    print("start doing")
    try:
        do_stuff()
    except RuntimeError as e:
        print(e)
    print("end doing")
```

Документирование

SELF-DOCUMENTED



Самодокументируемый код

Документация

```
class ExampleError(Exception):
```

"""Exceptions are documented in the same way as classes.

The `__init__` method may be documented in either the class level docstring, or as a docstring on the `__init__` method itself.

Either form is acceptable, but the two should not be mixed. Choose one convention to document the `__init__` method and be consistent with it.

Note:

Do not include the `self` parameter in the ``Args`` section.

Args:

msg (str): Human readable string describing the exception.

code (:obj:`int`, optional): Error code.

Attributes:

msg (str): Human readable string describing the exception.

code (int): Exception error code.

"""

```
def __init__(self, msg, code):
```

self.msg = msg

self.code = code

Организация и дистрибуция Python кода



Модуль

Модуль это просто файл .py

Там могут быть

- импорты других модулей
- объявления функций, классов, переменных
- исполняемый код

Модуль

```
import os

def example_fun() -> str:
    return os.curdir

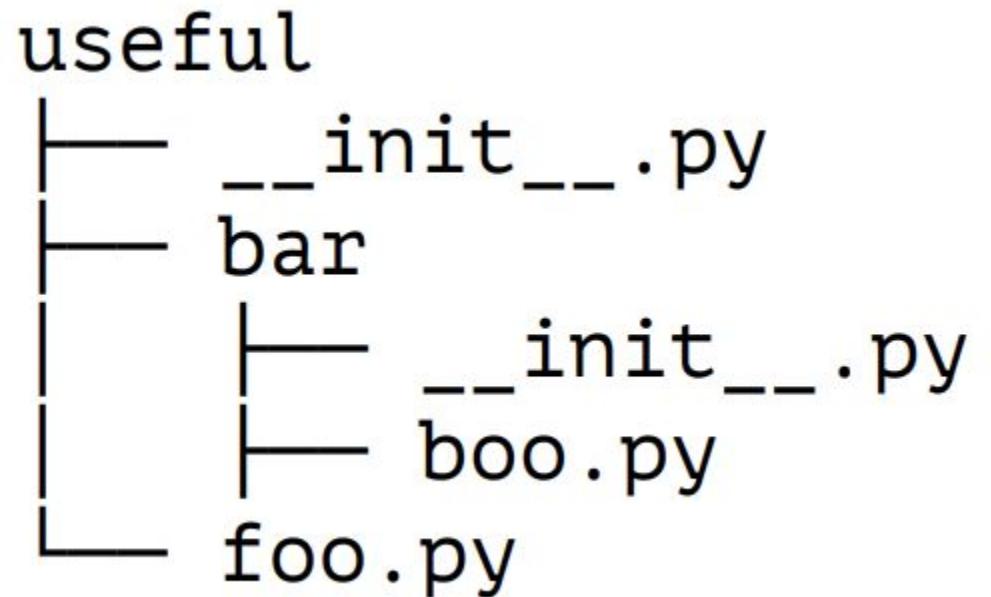
if __name__ == "__main__":
    print(example_fun())
```

Пакет

Пакет это просто папка с модулями

Там могут быть

- Модули
- Вложенные пакеты
- `__init__.py`



__init__.py

- При импорте пакета (**import package, from my_cool_lib import package**) импортируется только __init__.py (выполняется все то, что написано внутри)
- Служит фасадом

```
from ._dict_vectorizer import DictVectorizer
from ._hash import FeatureHasher
from .image import img_to_graph, grid_to_graph
from . import text

__all__ = ['DictVectorizer', 'image', 'img_to_graph', 'grid_to_graph', 'text',
          'FeatureHasher']
```

Структура проекта

```
helloworld-project
├── helloworld
│   ├── __init__.py
│   └── core.py
├── setup.py
└── README.txt
```

setup.py

```
#!/usr/bin/env python

from distutils.core import setup

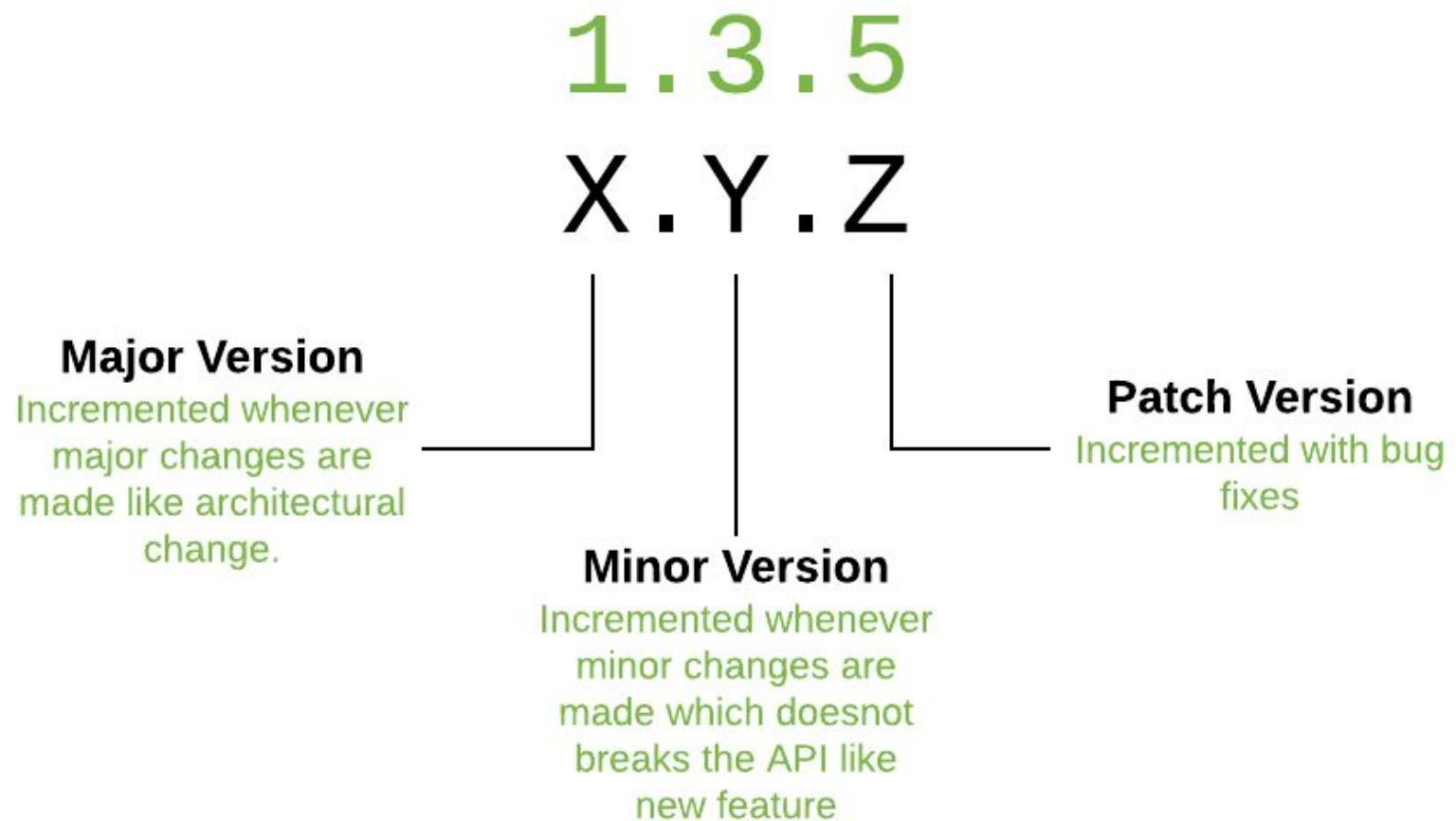
setup(name='Distutils',
      version='1.0',
      description='Python Distribution Utilities',
      author='Greg Ward',
      author_email='gward@python.net',
      url='https://www.python.org/sigs/distutils-sig/',
      packages=[ 'distutils', 'distutils.command' ],
)
```



setup.py

```
setup(  
    name="dvc",  
    version=version,  
    description="Git for data scientists – manage your code and data together",  
    long_description=open("README.rst", "r", encoding="UTF-8").read(),  
    author="Dmitry Petrov",  
    author_email="dmitry@dvc.org",  
    download_url="https://github.com/iterative/dvc",  
    license="Apache License 2.0",  
    install_requires=install_requires,  
    extras_require={  
        "all": all_remotes,  
        "gs": gs,  
        "gdrive": gdrive,  
        "s3": s3,  
        "azure": azure,  
        "oss": oss,  
        "ssh": ssh,  
        "ssh_gssapi": ssh_gssapi,  
        "hdfs": hdfs,  
        "webhdfs": webhdfs,  
        "webdav": webdav,  
        "tests": tests_requirements,  
    },  
    keywords="data-science data-version-control machine-learning git"  
    " developer-tools reproducibility collaboration ai",  
    python_requires=">=3.6",  
    classifiers=[  
        "Development Status :: 4 - Beta",  
        "Programming Language :: Python :: 3",  
        "Programming Language :: Python :: 3.6",  
        "Programming Language :: Python :: 3.7",  
        "Programming Language :: Python :: 3.8",  
    ],  
    packages=find_packages(exclude=["tests"]),  
    include_package_data=True,  
    url="http://dvc.org",
```

Semantic Versioning





Дистрибуция

```
python setup.py sdist
```

```
python setup.py sdist
```

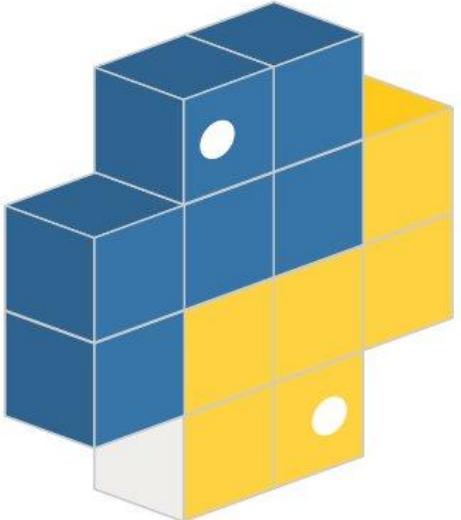
Проект, содержащий setup.py можно собрать

> **python setup.py sdist -> my_project.tar.gz**

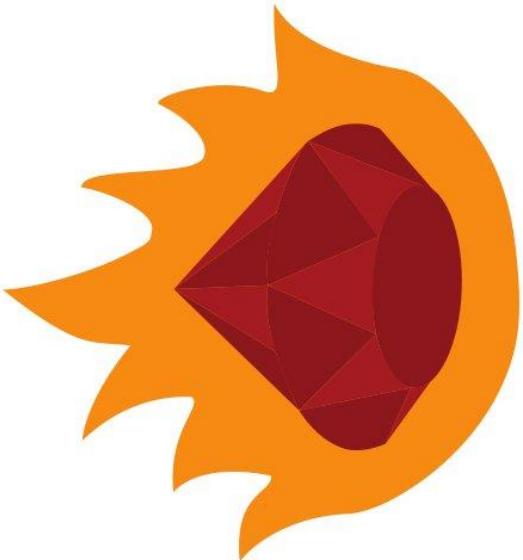
Этот пакет можно устанавливать и распространять

> **pip install my_project.tar.gz**

Репозитории пакетов



PyPI



Gemfury



Nexus

Зависимости



Зависимости

pip install -r requirements.txt

sklearn==1.6.0

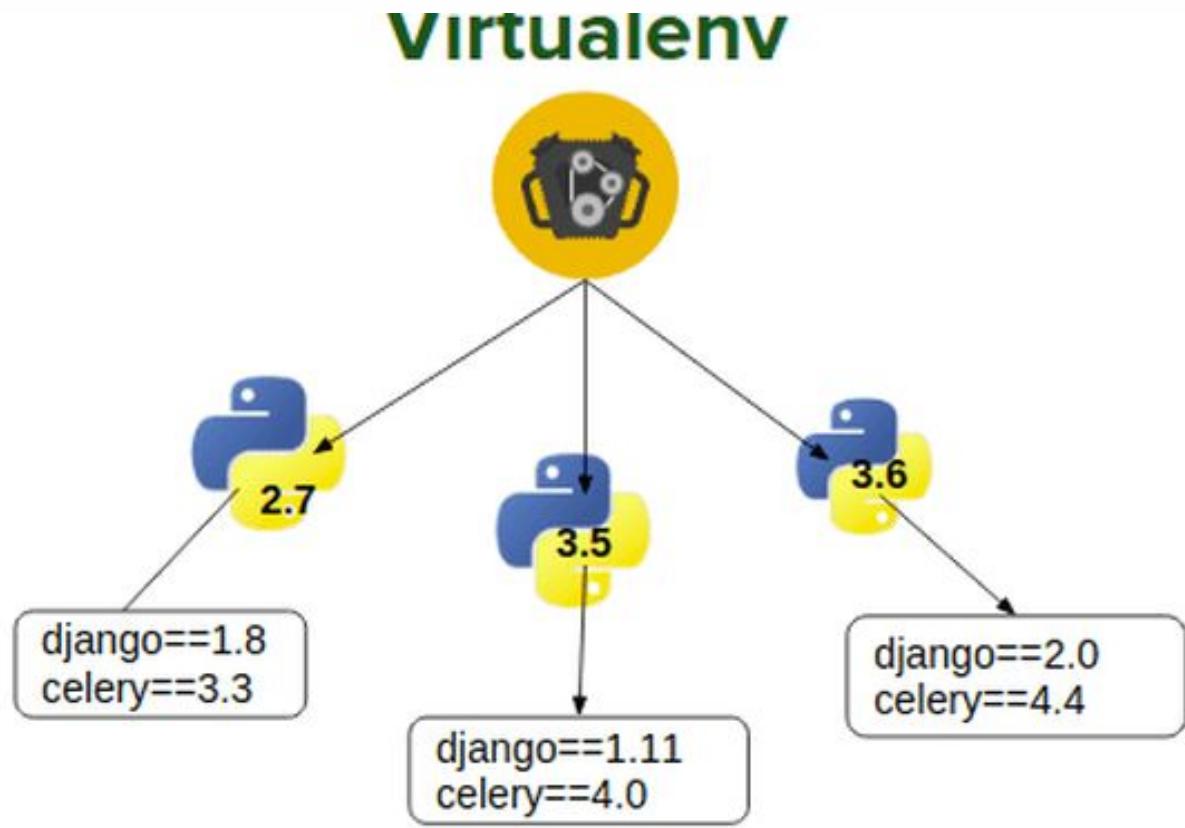
dataclasses==0.8

pyyaml==3.11

marshmallow-dataclass==8.3.0

torch>=1.5

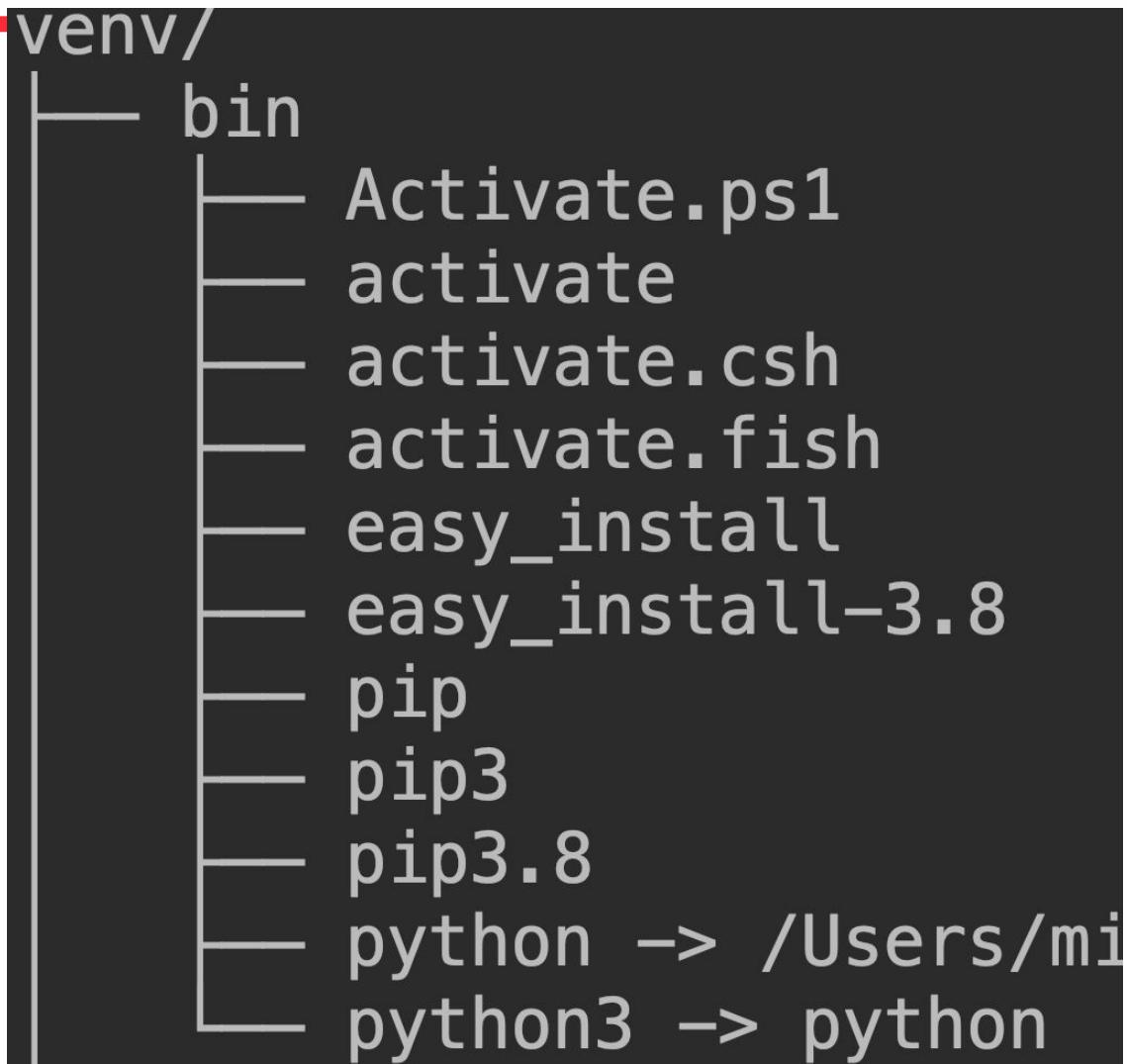
Виртуальные окружения



VENV структура

```
venv/
  └── bin
  └── include
  └── lib
      └── pyvenv.cfg
```

VENV структура - bin



```
venv/
└── bin
    ├── Activate.ps1
    ├── activate
    ├── activate.csh
    ├── activate.fish
    ├── easy_install
    ├── easy_install-3.8
    ├── pip
    ├── pip3
    ├── pip3.8
    └── python → /Users/mi
        └── python3 → python
```

The image shows a terminal window displaying the contents of the 'bin' directory within a virtual environment ('venv'). The 'bin' directory contains several scripts: 'Activate.ps1', 'activate', 'activate.csh', 'activate.fish', 'easy_install', 'easy_install-3.8', 'pip', 'pip3', 'pip3.8', and 'python'. The 'python' script is a symbolic link pointing to '/Users/mi/python', and the 'python3' script is a symbolic link pointing to 'python'. A red horizontal bar highlights the 'venv/' prefix at the top of the terminal window.

VENV структура - site-packages

```
lib
└── python3.8
    └── site-packages
        ├── __pycache__
        ├── easy_install.py
        ├── pip
        ├── pip-20.1.1.dist-info
        ├── pkg_resources
        ├── setuptools
        └── setuptools-47.1.0.dist-info
```

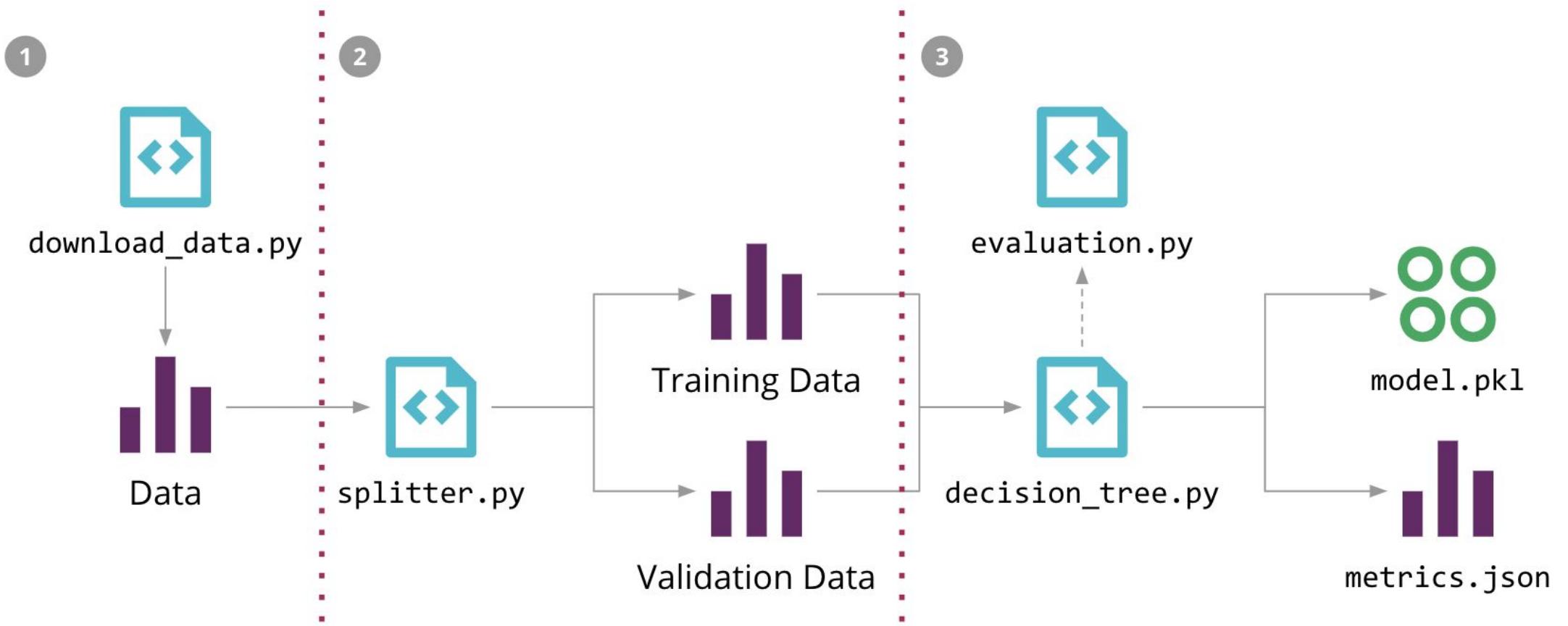


Инструменты для контроля виртуальных окружений

- **virtualenv**
- **virtualenvwrapper**
- **conda**
- **pyenv**
- **poetry**

Структура DS проекта

ML проект



Cookiecutter DS template

```
├── LICENSE
├── Makefile           <- Makefile with commands like `make data` or `make train`
├── README.md          <- The top-level README for developers using this project.
└── data
    ├── external        <- Data from third party sources.
    ├── interim         <- Intermediate data that has been transformed.
    ├── processed       <- The final, canonical data sets for modeling.
    └── raw              <- The original, immutable data dump.

    └── docs             <- A default Sphinx project; see sphinx-doc.org for details

    └── models           <- Trained and serialized models, model predictions, or model summaries

    └── notebooks        <- Jupyter notebooks. Naming convention is a number (for ordering),
                                the creator's initials, and a short '-' delimited description, e.g.
                                `1.0-jqp-initial-data-exploration`.

    └── references        <- Data dictionaries, manuals, and all other explanatory materials.

    └── reports           <- Generated analysis as HTML, PDF, LaTeX, etc.
        └── figures         <- Generated graphics and figures to be used in reporting

    └── requirements.txt   <- The requirements file for reproducing the analysis environment, e.g.
                                generated with `pip freeze > requirements.txt`
```

Cookiecutter DS template

```
|   ├── setup.py          <- Make this project pip installable with `pip install -e`  
|   └── src               <- Source code for use in this project.  
|       ├── __init__.py    <- Makes src a Python module  
|       └── data           <- Scripts to download or generate data  
|           └── make_dataset.py  
|       └── features        <- Scripts to turn raw data into features for modeling  
|           └── build_features.py  
|       └── models          <- Scripts to train models and then use trained models to make  
|                           predictions  
|           ├── predict_model.py  
|           └── train_model.py  
|       └── visualization   <- Scripts to create exploratory and results oriented visualizations  
|           └── visualize.py  
└── tox.ini            <- tox file with settings for running tox; see tox.readthedocs.io
```

Тестирование

Зачем тесты?

```
# test_capitalize.py

def capital_case(x):
    return x.capitalize()

def test_capital_case():
    assert capital_case('semaphore') == 'Semaphore'
```

1. Код без тестов -- это сломанный код
2. Даже если всё работает сейчас, вы сломаете его завтра. Без тестов вы не узнаете об этом.
3. Заставляет вас писать код лучше
4. Легче владеть кодом коллективно



Статья про тестирование в ML

- 1) Pre-train tests to ensure correct implementation
- 2) Post-train tests to ensure expected learned behaviour

Доклады про тестирование



КАК ТЕСТИРОВАТЬ DS-КОД

АЛЕКСЕЙ
МОГИЛЬНИКОВ

Lean
DataScience

A presentation slide featuring a man in a dark suit and glasses speaking at a podium. He is holding a small device and has a purple lanyard around his neck. The background is white, and there are blue and white geometric shapes on the left side of the slide.



ML REPA

Data Fest 2020 Online

Testing for Data Science Hands-on Guide

Julia Antokhina

What you'll learn

Practical issues on clean code, patterns, code review, refactoring, static code analysis, etc.

Cool stories about the engineering part of DS projects and simple howto's to improve your engineering skills.

A promotional image for the Data Fest 2020 Online event. It features a yellow header with the ML REPA logo and the event name. Below is a yellow section with the title "Testing for Data Science Hands-on Guide" and the speaker's name, Julia Antokhina. A teal hexagon contains the text "What you'll learn" followed by a list of topics. To the right is a photo of a smiling woman with dark hair and a headband. The background is yellow with some geometric shapes.