Datacon

Team 3

У нас было...

Основной датасет:
883 строки экспериментальных данных.
Таргет – ZOI_drug_NP (размер зоны ингибирования роста бактерий комплексом антибиотик-наночастица Ag)

Бактериальные дескрипторы:
записи о систематике и свойствах 59 бактерий

Дескрипторы лекарств:
42 записи о названии лекарств и их хиральных SMILES

```
Int64Index: 883 entries, 0 to 882
Data columns (total 16 columns):
 #   Column
---  ------
 0   Bacteria
 1   NP_Synthesis
 2   Drug
 3   Drug_class_drug_bank
 4   Drug_dose
 5   NP_concentration
 6   NP size_min
 7   NP size_max
 8   NP size_avg
 9   shape
 10  method
 11  ZOI_drug
 12  ZOI_NP
 13  ZOI_drug_NP
 14  fold_increase_in_antibacterial_activity (%)
 15  MDR_check
```

Data cleaning

Предобработка: численные признаки проверяем на не-

```
df['ZOI_drug'].unique()
```
- удобно для поиска проблем

ZOI_drug	ZOI_NP	ZOI_drug_NP
32+	6.3	32+
32+	6.3	32+
32+	25	32+
32+	25	32+
0	14	17+2

NP_concentration
170/85/42.29

dtype = object

Результат - все числовые признаки dtype = float. Содержат NaN

Исправления:
1)32+ = 32 (и т. п.)
2)17+2 = 17
3)170/85/42.xx были удалены

Data cleaning

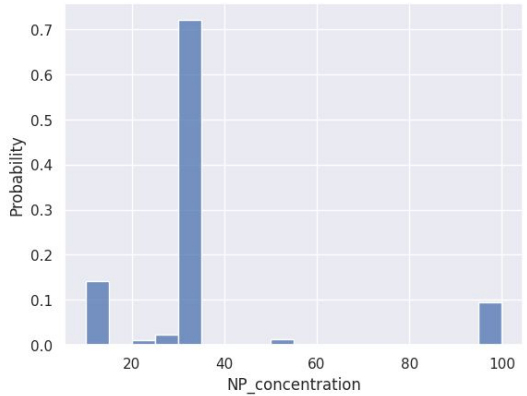
Предобработка: строковые признаки чистим и подготавливаем к кодированию

1) Исправляем опечатки
2) str.strip()
3) Приводим колонки к одному виду (abc -> Abc), значения в lowercase
4) В Drug_class_drug_bank есть лекарства, которые отнесены сразу к 2 классам - оставляем один
5) Тип синтеза НЧ: либо зеленый, либо химический синтез, без подробностей

```
Bacteria - nans: 0
NP_Synthesis - nans: 0
Drug - nans: 241
Drug_class_drug_bank - nans: 241
Drug_dose - nans: 306
NP_concentration - nans: 250
avg_NP_size - nans: 0
shape - nans: 0
method - nans: 0
ZOI_drug - nans: 256
ZOI_NP - nans: 412
ZOI_drug_NP - nans: 299
fold_increase_in_antibacterial_activity (%) - nans: 432
MDR_check - nans: 0
min_max_NP_size - nans: 0
```

Data cleaning

Удаление NaN



1)**ZOI_drug_NP - target, вынуждены выкинуть NaNs**

2) ZOI_NP выкинем, так как слишком много nans (412/547)

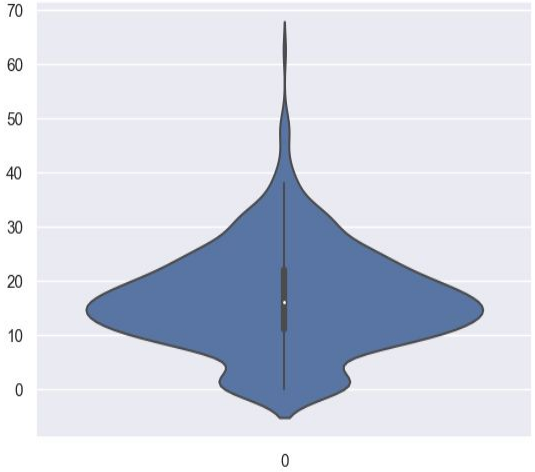
3) В остальных фичах заменяем на моду или по KNN

тут NaN заменяется на моду

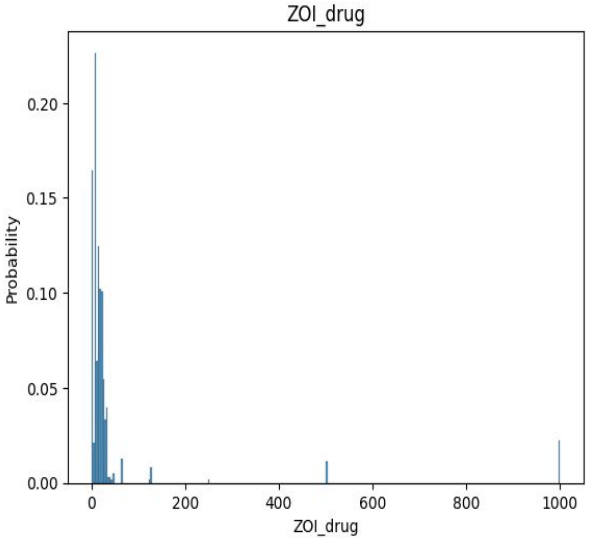
Data cleaning

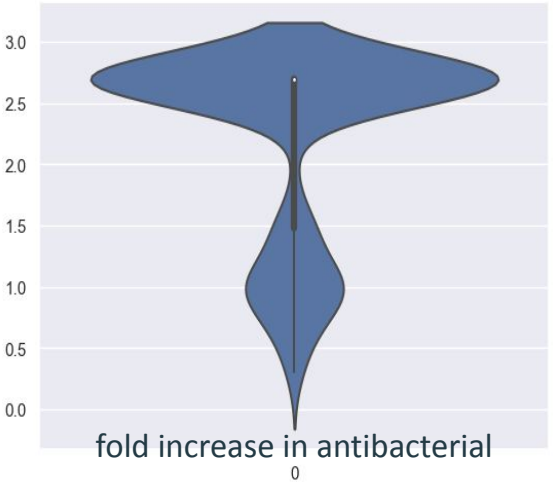
Удаление NaN

Пропуски в ZOI_drug, Drug_dose и fold_increase_in_antibacterial_activity(%) потом заменим **с помощью KNN**, обучив на почищенных данных



Drug_dose

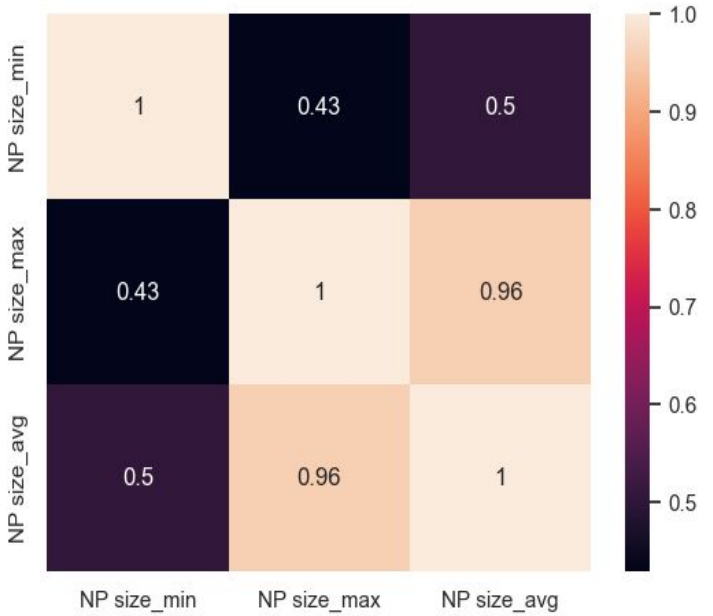




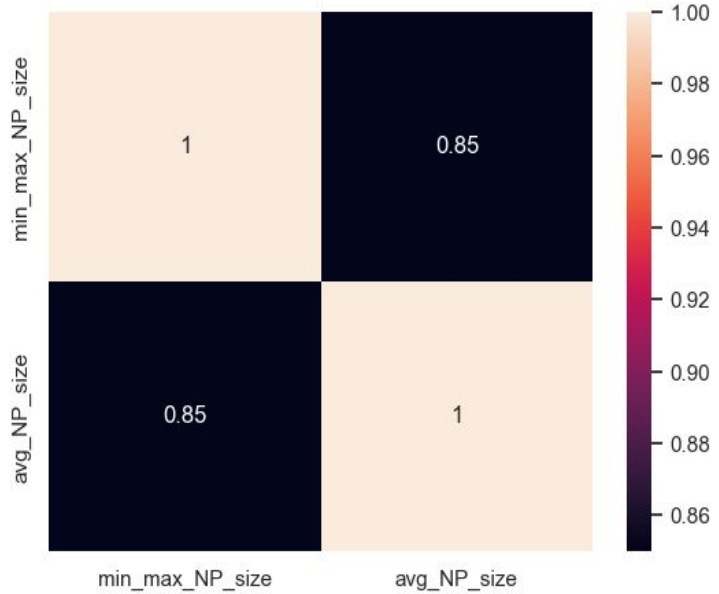
fold increase in antibacterial activity

Data analysis

Корреляции NP_size_min, NP_size_max, NP_size_avg



max - min
= размах



Размах все равно коррелирует, оставили только средний размер НЧ

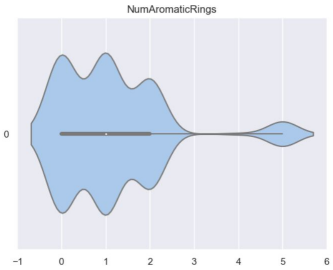
Data analysis

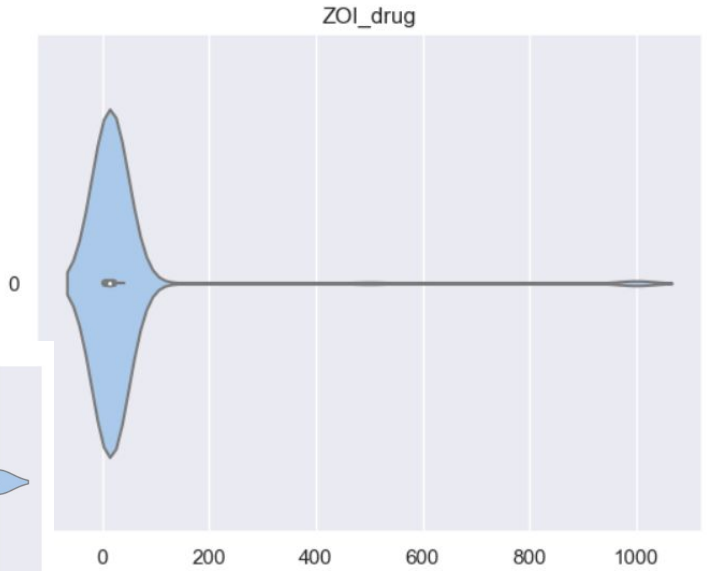
1) Drug_dose заменим на lg(Drug_dose), есть явные порядки

```
df['Drug_dose'].unique()
✓ 0.0s
array([  nan,  30. ,  10. , 500. ,   2. ,   5. , 100. , 300. ,  25.
        20. ,  50. ,   2.5,  15. ])
```

2) Обработка выбросов

3) Любуемся на violin-plot’ы





Drug_descriptors cleaning and analysis

1) Удаляем prefered_name и chemID, есть SMILES
2) Удаляем дубликаты, добавляем neomycin
3) Загружаем дополнительные признаки: **pKa**, дескрипторы из RDKit

```
descs_to_add = [
    'LabuteASA', 'NumHDonors', 'NumHAcceptors',
    'MolLogP', 'MolWt', 'Chi0', 'RingCount', 'TPSA',
    'NumAliphaticRings', 'NumAromaticRings', 'NumAromaticHeterocycles',
    'MinPartialCharge', 'MaxPartialCharge', 'BertzCT'
]
```

(и затем удаляем сильно коррелирующие дескрипторы)

```
last_drop = ['LabuteASA', 'MolWt', 'Chi0', 'NumHDonors', 'TPSA', 'min_max_NP_size']
```

pKa estimation [J. Chem. Inf. Mod. 61(1) https://doi.org/10.1021/acs.jcim.1c00075]

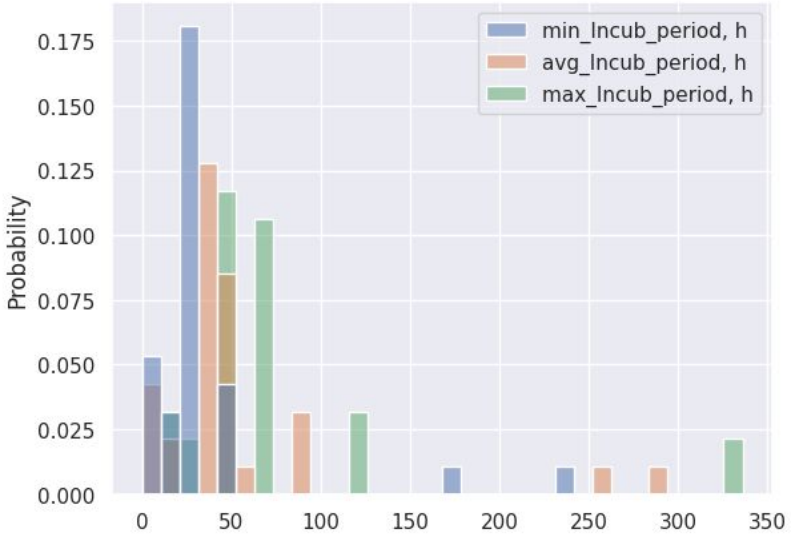
Bacterial_descriptors cleaning and analysis

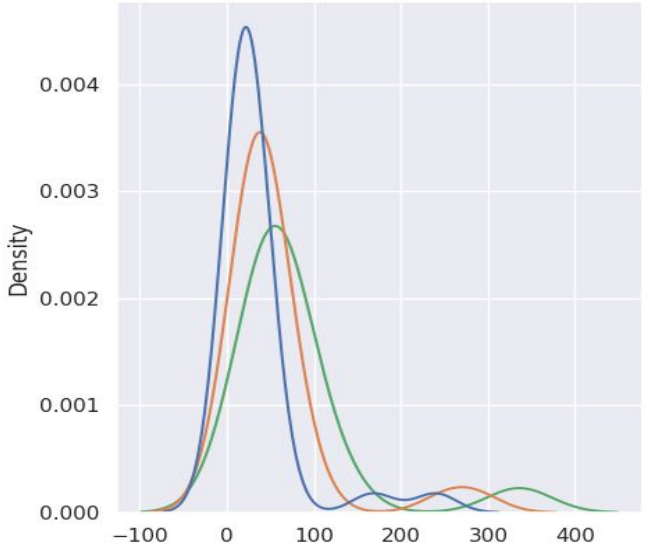
1) Удаляем бактерии, отсутствующие в data
2) Исправляем опечатки
3) Столбцы **subkingdom** и **clade** дропаем, много NaNs
4) Пропуски заполняем **данными из NCBI** или модой

```
df['Bacteria'] = df['Bacteria'].replace({
    'Acinetobacter baumanii': 'Acinetobacter baumannii',
    'Bacillus spp.': 'Bacillus sp.',
    'Salmonella  typhi': 'Salmonella typhi'
})
```

Bacterial_descriptors cleaning and analysis

avg и min/max_Incub_period: те же маневры с размахом.





Заполняем пропуски и некорректные значения модой, затем заменили мин/макс на размах. Он сильно коррелировал со средним, пришлось выкинуть.

Слияние БД

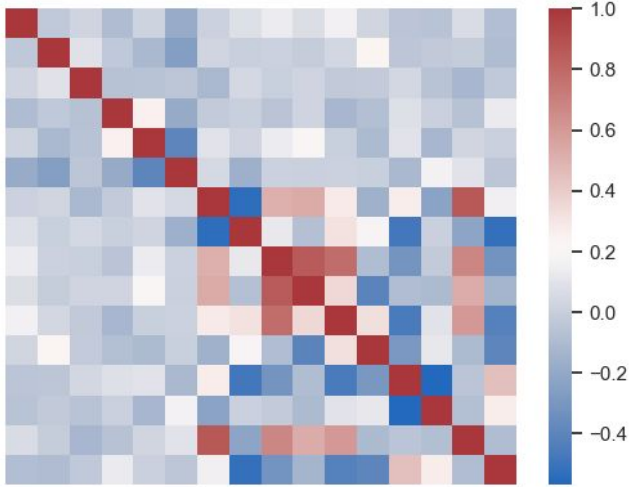
```
df.drop_duplicates(inplace=True)
print(df.shape)

df = df.merge(df_bac, on='Bacteria', how='left')

df = df.merge(df_drug, on='Drug', how='left')

df.drop_duplicates(inplace=True)
print(df.shape)
```

✓ 0.0s

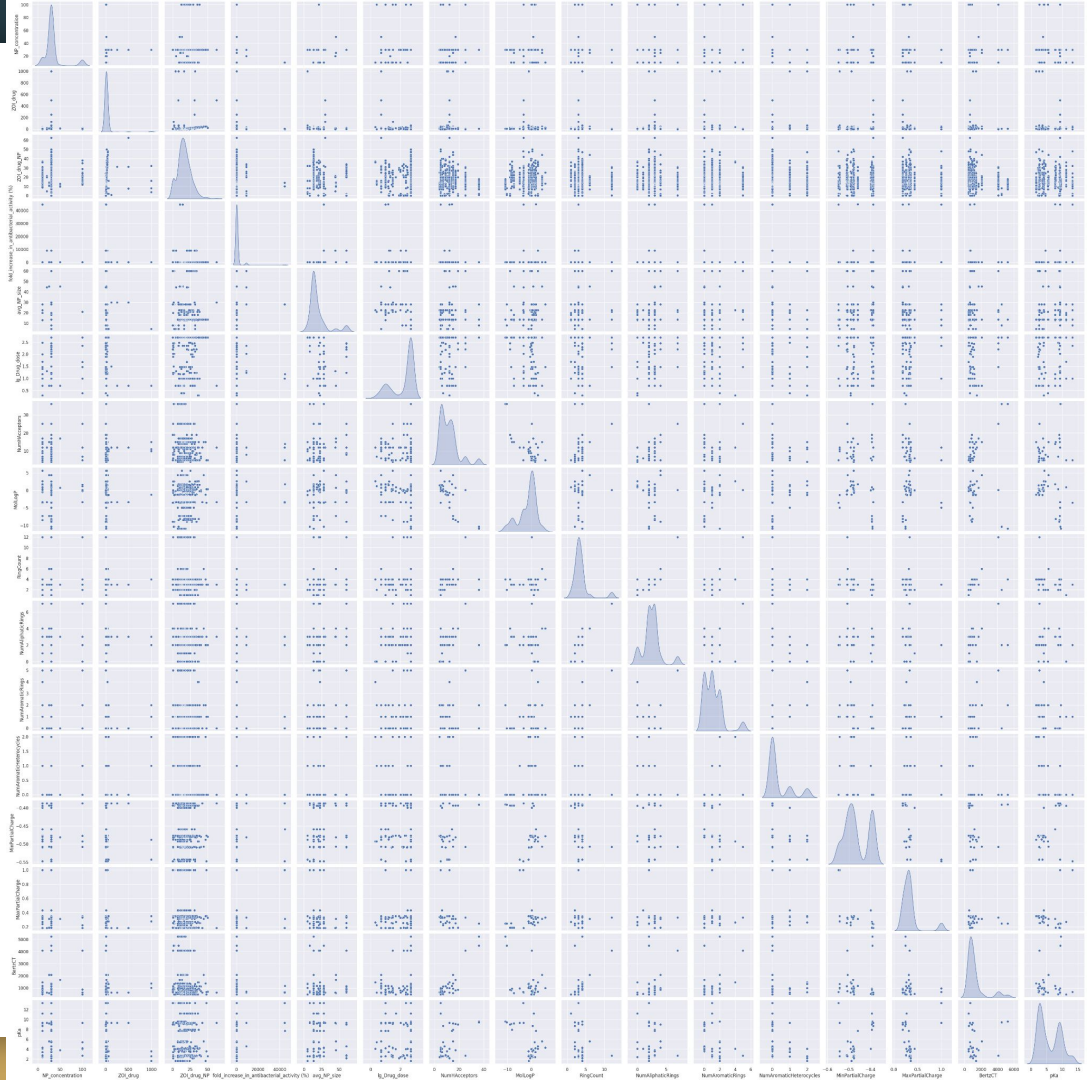
(547, 14) - размер до
(547, 42) - размер после

LEFT INNER JOIN - в общую БД попадают только те бактерии/лекарства, которые есть в df

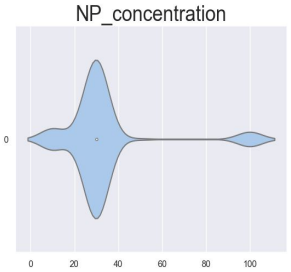
Pairplot and corrmap

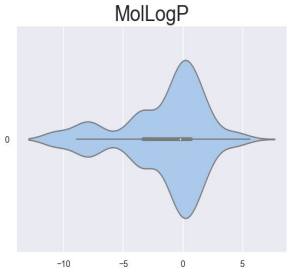


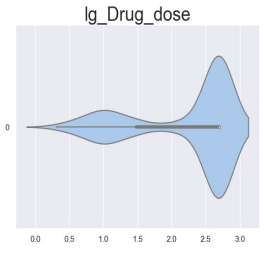
Корреляций нет, на pairplot проблем не видно

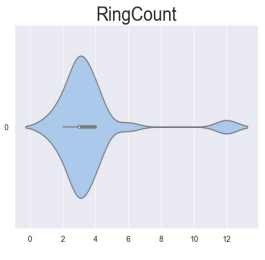


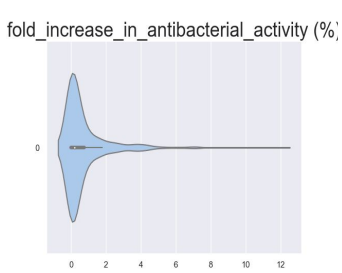
Распределение непрерывных признаков

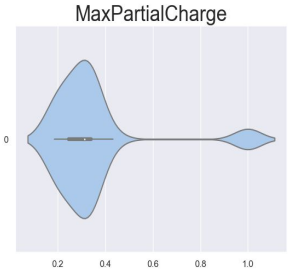


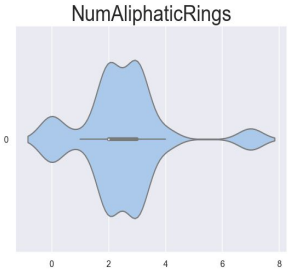


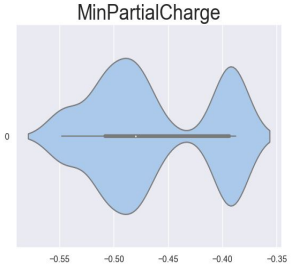


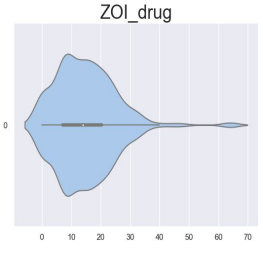


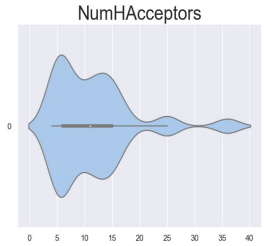


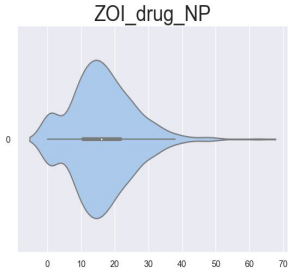


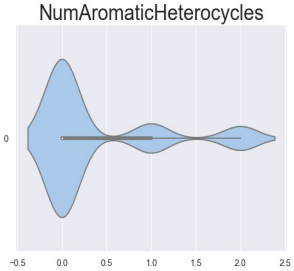


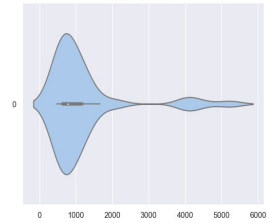


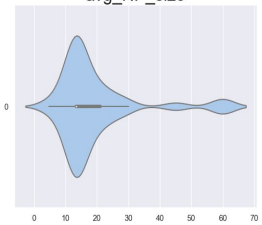


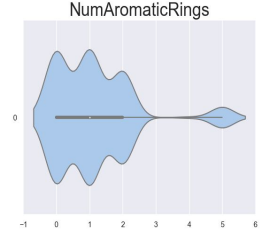


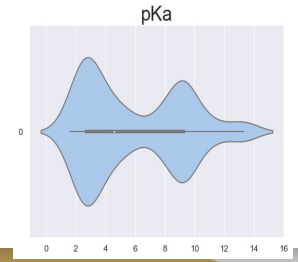




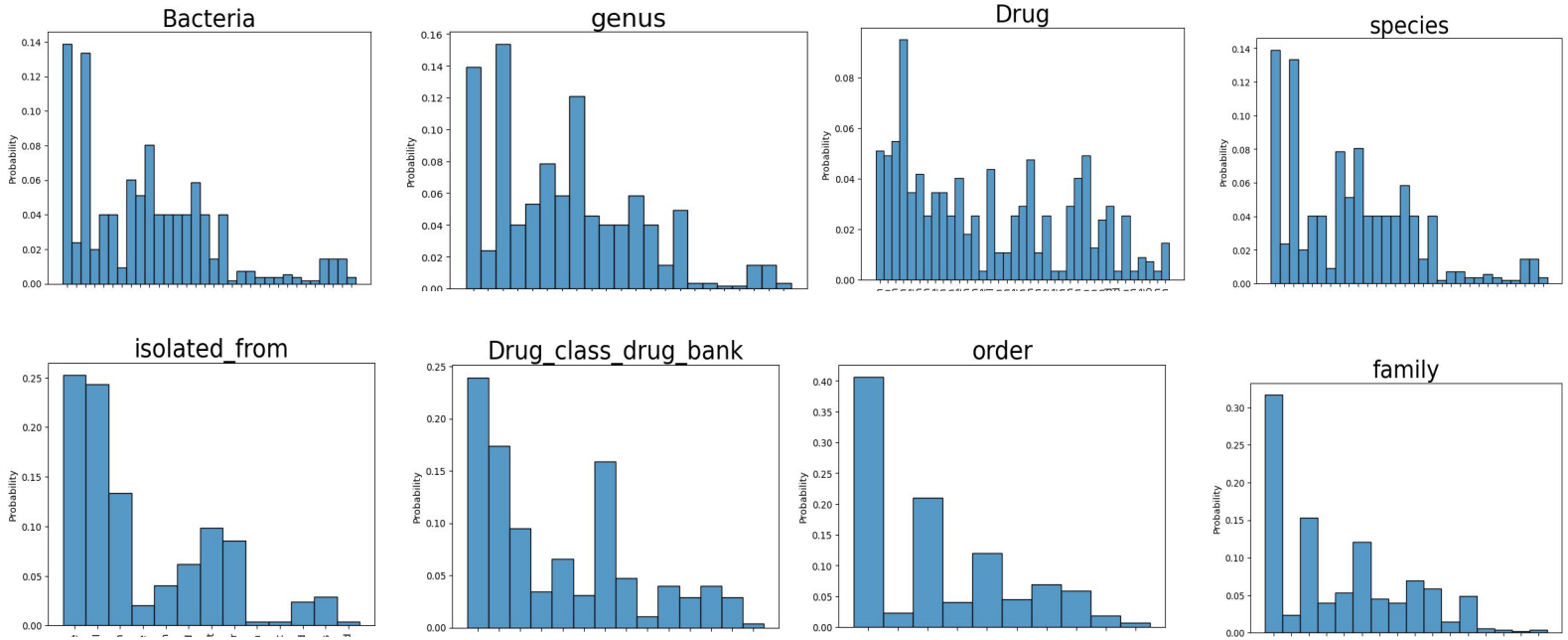




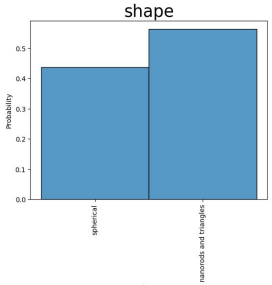


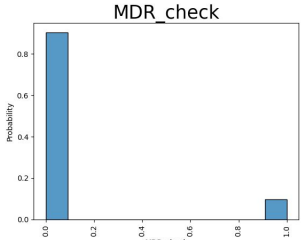


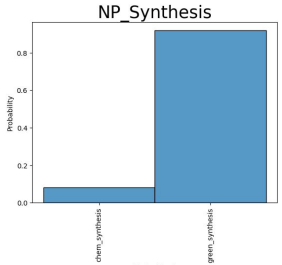
Распределение категориальных признаков

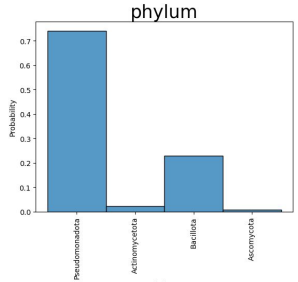


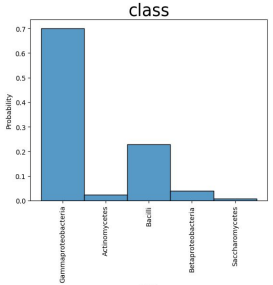
Распределение категориальных признаков

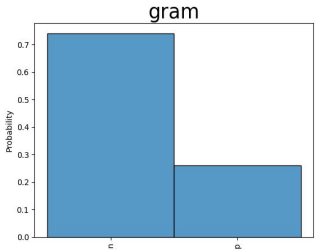


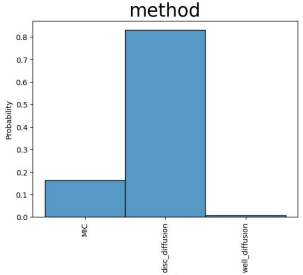


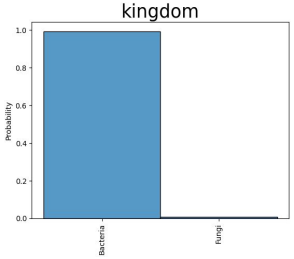












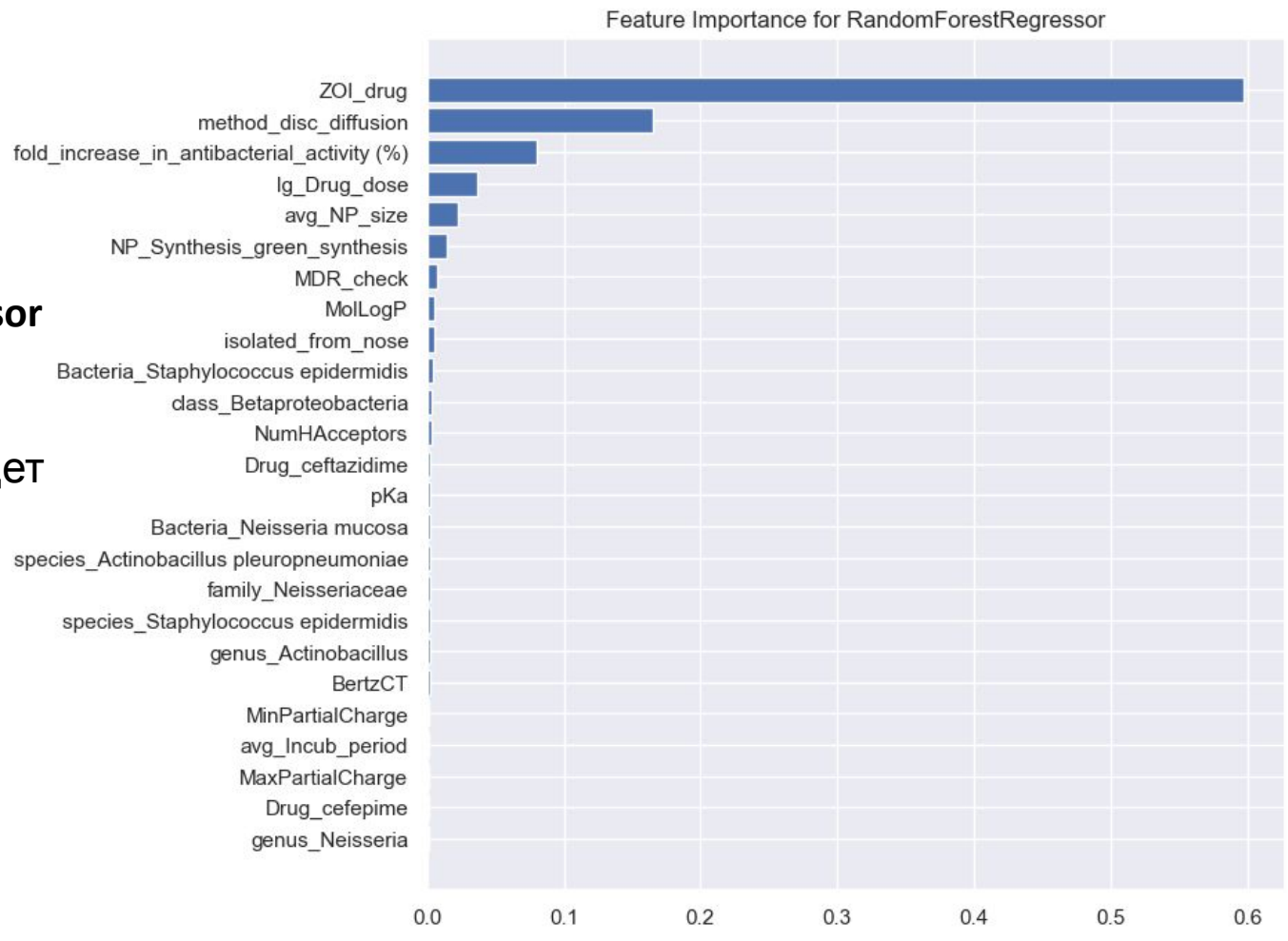
Сильный дизбаланс классов. Помешает универсальности применения модели

ML - первые наблюдения

RandomForestRegressor
GridSearchCV
просто чтобы посмотреть, что будет

R^2 = 0.77

Улучшаем



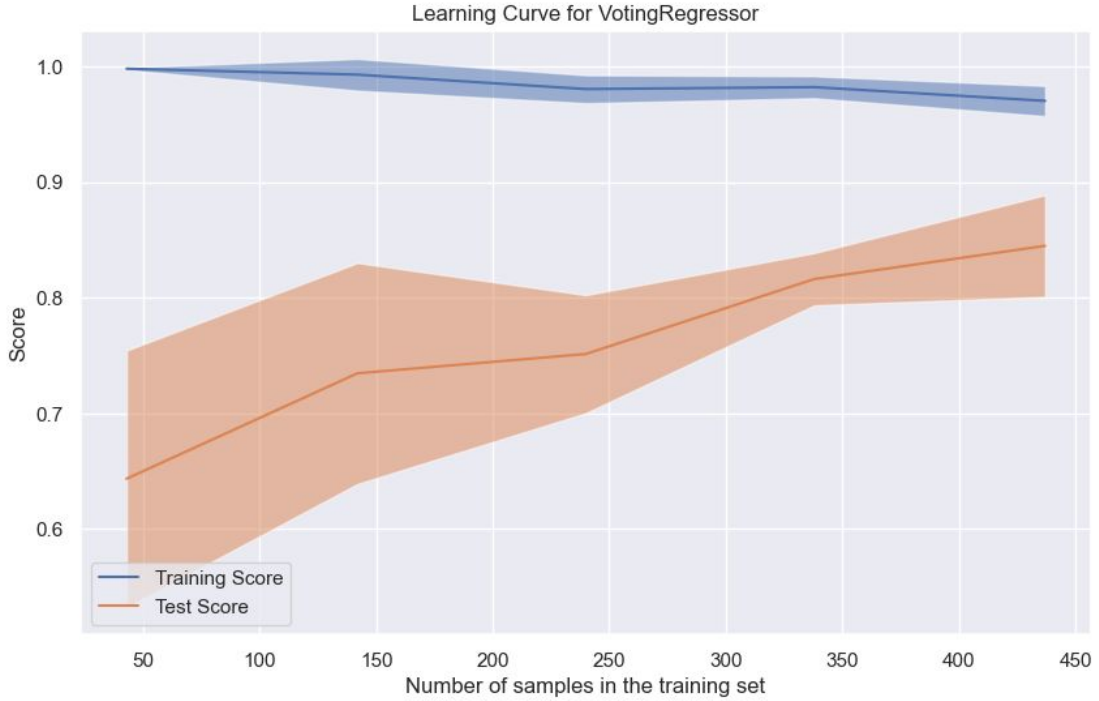
ML-модели

1) Кодирование кат. признаков: нумерация классов и one-hot (разница мала)
2) LazyRegressor: отчет о метриках классических ML-регрессоров из коробки

Лучшие: **ExtraTreesRegressor** (ансамбль реш. деревьев),
CatBoost (в LazyPredict его нет) *R^2 = 0.83*

3) Оставим 30 наиболее важных признаков - метрика вырастает
4) GridSearchCV
5) **VotingRegressor** для CatBoost и ExtraTreesRegressor
6) Упаковка обученных моделей для предсказаний

Графики обучения и R2 на кросс-валидации



One-Hot

```
-- onehot_final --
--- Model: CatBoostRegressor ---

Best params: {'model__depth': 3, 'model__
Best CV R2 score: 0.819
R2 score on unseen data: 0.829

--- Model: ExtraTreesRegressor ---

Best params: {'model__criterion': 'absolu
Best CV R2 score: 0.818
R2 score on unseen data: 0.852
```

Codes

```
-- codes_final --
--- Model: CatBoostRegressor ---

Best params: {'model__depth': 2,
Best CV R2 score: 0.824
R2 score on unseen data: 0.851

 -- Model: ExtraTreesRegressor --

est params: {'model__criterion':
est CV R2 score: 0.832
2 score on unseen data: 0.852
```

VotingRegressor

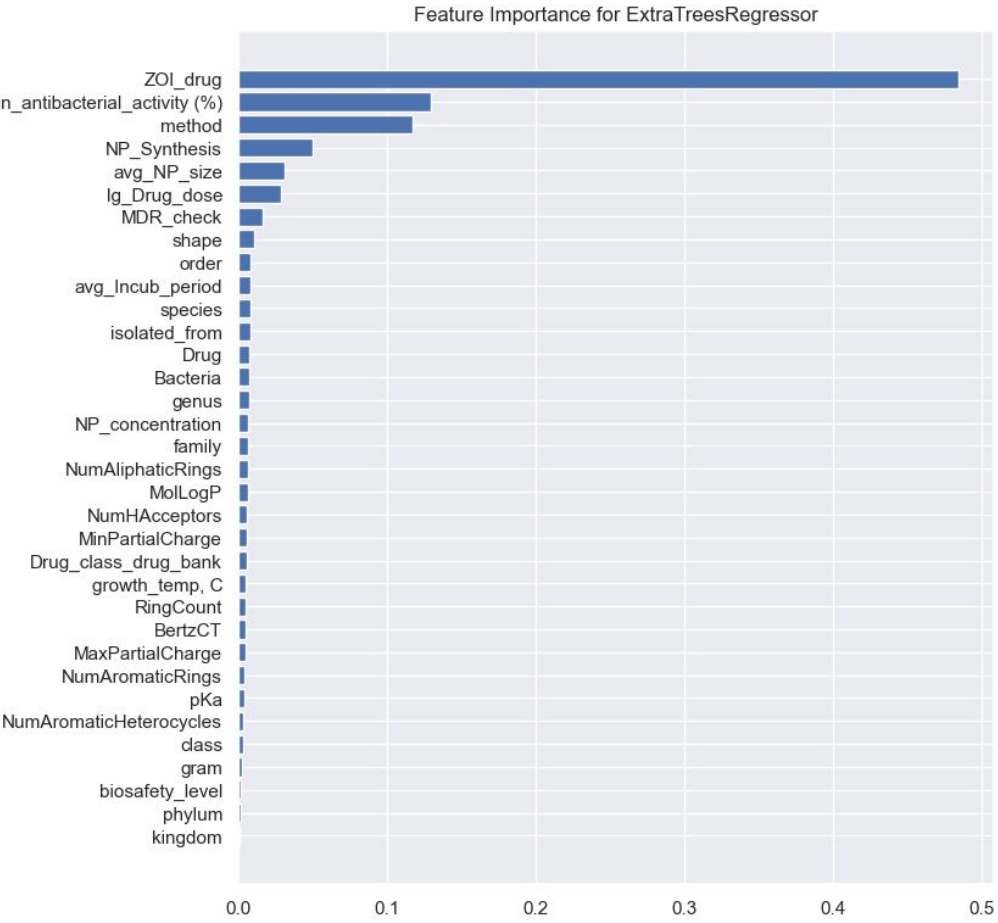
One-Hot

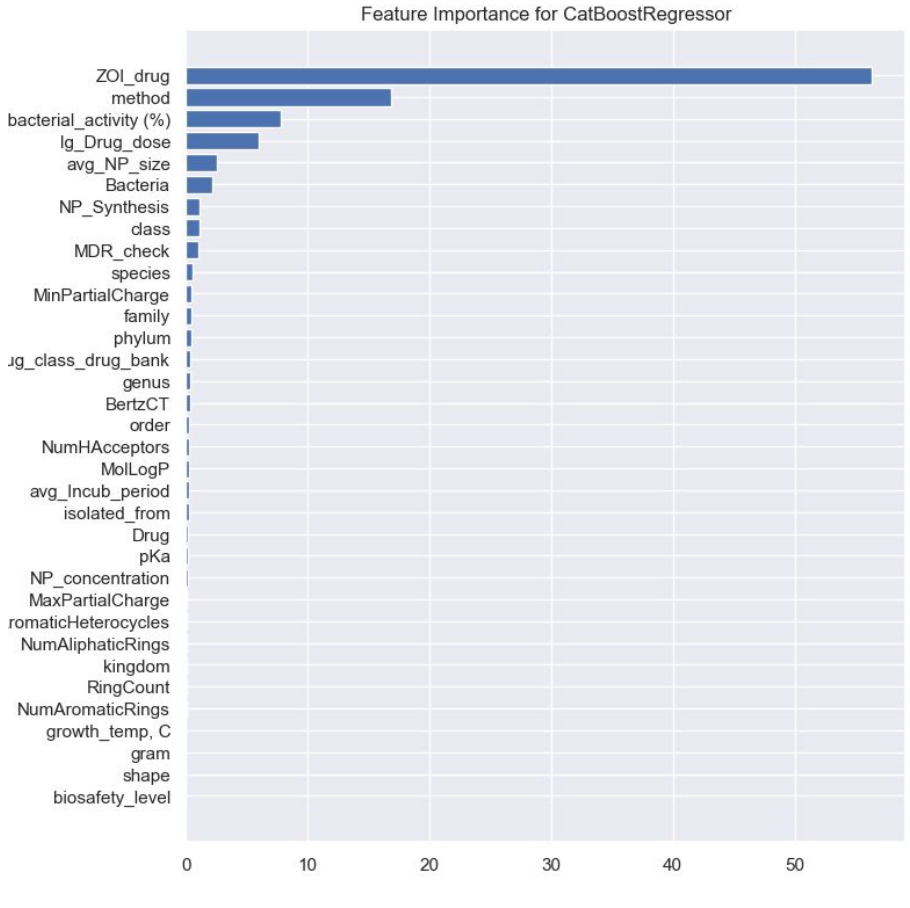
```
---
R2 score: 0.831 with a standard deviation 0.022
R2 score max: 0.851
```

Codes

```
---
R2 score: 0.839 with a standard deviation 0.025
R2 score max: 0.861
```

Feature importance



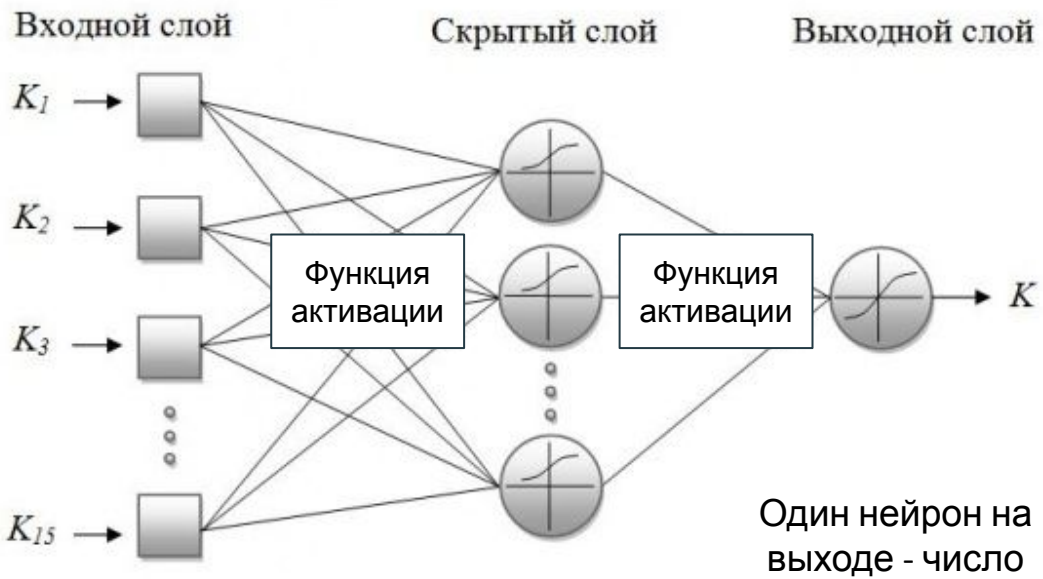


Нейросеть (полносвязная, PyTorch)

Сделали перебор параметров:

a. тип оптимизатора (Adam, SGD)
b. функция активации (ReLU, ELU, Leaky ReLU, sigmoid)
c. Дропаут (делать/нет)

Лучший $R^2 = 0.914$

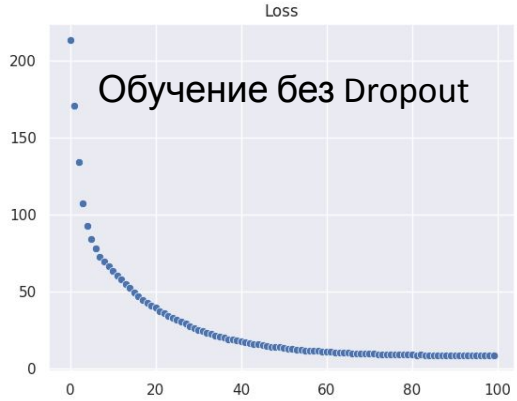


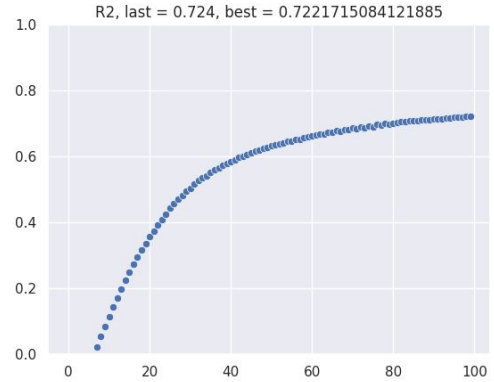
В конечном варианте - 198 признаков (большинство - OneHot)

Один скрытый слой
204 // 3 нейрона

Один нейрон на выходе - число “ZOI_drug_NP”

Реализация и типичные кривые обучения





Очень сложно подобрать оптимальный learning rate: данных мало

Оптимизировать архитектуру подбором?

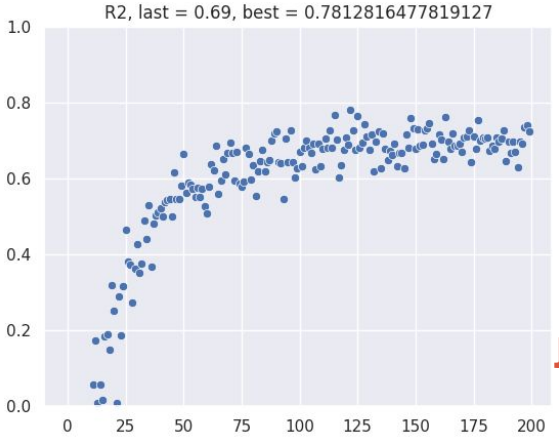
Где-то упустили random_state

Лучший $R^2 = 0.914$

```python
class DenseNN(nn.Module):
    def __init__(self, in_features: int, activation=F.relu, do_dropout=False):
        super(DenseNN, self).__init__()
        self.activation = activation
        self.fc1 = nn.Linear(in_features, in_features)
        self.fc2 = nn.Linear(in_features, in_features // 3)
        self.fc3 = nn.Linear(in_features // 3, 1)
        self.do_dropout = do_dropout
        self.dropout = nn.Dropout(0.1)

    def forward(self, x):
        x = self.activation(self.fc1(x))

        if self.do_dropout:
            x = self.dropout(x)

        x = self.activation(self.fc2(x))

        if self.do_dropout:
            x = self.dropout(x)

        x = self.fc3(x)
        return x
```

```python
model = DenseNN(X.shape[1], activation=activation_name, do_dropout=do_dropout)
loss_fn = loss_fn_name()
optimizer = optimizer_lr[0](model.parameters(), lr=optimizer_lr[1])

for n_epoch in range(n_epochs):
    for i in range(0, len(X), batch_size):
        Xbatch = X[i:i+batch_size]

        y_pred = model(Xbatch)

        ybatch = y[i:i+batch_size]

        loss = loss_fn(y_pred, ybatch)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

Реализация и типичные кривые обучения



Очень сложно подобрать оптимальный learning rate: данных мало

Оптимизировать архитектуру подбором?

Где-то упустили random_state

Лучший $R^2 = 0.914$

```python
class DenseNN(nn.Module):
    def __init__(self, in_features: int, activation=F.relu, do_dropout=False):
        super(DenseNN, self).__init__()
        self.activation = activation
        self.fc1 = nn.Linear(in_features, in_features)
        self.fc2 = nn.Linear(in_features, in_features // 3)
        self.fc3 = nn.Linear(in_features // 3, 1)
        self.do_dropout = do_dropout
        self.dropout = nn.Dropout(0.1)

    def forward(self, x):
        x = self.activation(self.fc1(x))

        if self.do_dropout:
            x = self.dropout(x)

        x = self.activation(self.fc2(x))

        if self.do_dropout:
            x = self.dropout(x)

        x = self.fc3(x)
        return x
```

```python
model = DenseNN(X.shape[1], activation=activation_name, do_dropout=do_dropout)
loss_fn = loss_fn_name()
optimizer = optimizer_lr[0](model.parameters(), lr=optimizer_lr[1])

for n_epoch in range(n_epochs):
    for i in range(0, len(X), batch_size):
        Xbatch = X[i:i+batch_size]

        y_pred = model(Xbatch)

        ybatch = y[i:i+batch_size]

        loss = loss_fn(y_pred, ybatch)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

Выводы

1. Данные грязненькие, но рабочие
2. Данных мало, но достаточно для хороших предсказаний
3. Улучшения - сбалансировать классы, заполнить NaN реальными данными, оптимизировать нейронку

Подходящие модели:

- ExtraTreesRegressor, ансамбль реш. деревьев …….. R^2 на CV 0.832
- CatBoost, бустинг на деревьях ……..……..……..……..……..……... R^2 на CV 0.824
- VotingRegressor на предыдущих двух ……. R^2 на CV 0.839
- Полносвязная нейросеть, 1 скрытый слой ……..……..……..……. R^2 на тесте 0.913

Главный продукт: ipynb с решением, ipynb для предсказаний с сохраненными моделями

Спасибо за внимание!

Над решением работали:

- Артем
- Артем
- Антон
- Андрей
- Ангелина

Задавайте вопросы!