Лабораторная работа

Методы регрессионного анализа

Работу подготовили:

Панов Олег, Михаил Бабушкин, Денис Чашин, Анатолий Мезенов, Никита Боровик

Постановка задачи

В качестве задания требуется провести регрессионный анализ данных, для того чтобы оценить значение **целевой** переменной на основе **факторных**.

Полученные результаты всех методов **сравним** между собой.

> Грубо говоря, натренировать модели регрессионного анализа данных, выбрать **лучшую** из ~~худших~~ полученных моделей и показать полученные результаты.

Входные данные

> Тренировочных выборок: **5**
> Тестовых выборок: **10**

Данные выборки содержат информацию о **комментариях** в социальной сети **Facebook** ~~запрещена на территории РФ~~.

- Размер всего датасета: **0.5GB**
- Размер тренировочного датасета **602808** строк
- Уникальных строк в тренировочном датасете **602400**
- Размер тестового датасета **1089** строк

Атрибуты (*переменные*)

Датасет содержит **54** переменные.

Факторными переменными являются:

- Количество просмотров
- Длина поста
- Категория поста
- День недели
- Время дня
- *и т.д.*

Целевые значения

Одна из переменных является целевой - **Target Variable**

> Это количество комментариев под постом в следующие **H часов** *[1-24]*

H часов - это факторная переменная!!!

Селектор

- Это инструмент для выбора определенного подмножества данных из более крупного набора, часто используемый для фильтрации или извлечения конкретных столбцов, строк или признаков в зависимости от заданных критериев или условий.
- Был реализован класс **RegressionSelectorHolder** представляющий собой "разветвитель" обучения модели по **selected variable**.

> В нашем случае селекция идет по "H Local"

Логирование

- Логирование при расчетах машинного обучения играет важную роль в обеспечении **прозрачности и воспроизводимости результатов**. Оно позволяет записывать и сохранять информацию о параметрах модели, метриках, процессе обучения и промежуточных результатах, что полезно для **отладки, анализа ошибок и оптимизации моделей**.

lab3.log

```
TIME: 2023-12-08 23:52:44.781798

NeuralNetworkRegressionModel

key: 1
params: {'activation': 'relu', 'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (250, 50), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 2500, 'random_state': 42, 'solver': 'adam'}
TRAIN_R2: -0.09459937563342802

key: 2
params: {'activation': 'relu', 'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (250, 50), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 2500, 'random_state': 42, 'solver': 'sgd'}
TRAIN_R2: 0.9464763119704609

key: 3
params: {'activation': 'relu', 'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (250, 50), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 2500, 'random_state': 42, 'solver': 'adam'}
TRAIN_R2: 0.8199445433435928

key: 4
params: {'activation': 'relu', 'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (250, 50), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 2500, 'random_state': 42, 'solver': 'adam'}
TRAIN_R2: 0.126093067337968

key: 5
params: {'activation': 'relu', 'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (250, 50), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 2500, 'random_state': 42, 'solver': 'sgd'}
TRAIN_R2: 0.8995099299081557

key: 6
params: {'activation': 'relu', 'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (250, 50), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 2500, 'random_state': 42, 'solver': 'adam'}
TRAIN_R2: 0.525711359249276

key: 7
params: {'activation': 'relu', 'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (250, 50), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 2500, 'random_state': 42, 'solver': 'adam'}
TRAIN_R2: 0.5259054689897142

key: 8
params: {'activation': 'relu', 'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (250, 50), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 2500, 'random_state': 42, 'solver': 'sgd'}
TRAIN_R2: 0.11782352422945153

key: 9
params: {'activation': 'relu', 'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (250, 50), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 2500, 'random_state': 42, 'solver': 'adam'}
TRAIN_R2: -0.0920336428495585

key: 10
params: {'activation': 'relu', 'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (250, 50), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 2500, 'random_state': 42, 'solver': 'sgd'}
TRAIN_R2: 0.481081860411381
```

Логирование

- В нашем случае, поскольку расчеты производились на удаленном сервере с соединением по **SSH** в консоли **Linux**, логирование позволило на ранних этапах **отследить и исправить ошибку**.

```
  Downloading argon2_cffi_bindings-21.2.0-cp36-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (86 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 86.2/86.2 KB 1.1 MB/s eta 0:00:00
Collecting executing>=1.2.0
  Downloading executing-2.0.1-py2.py3-none-any.whl (24 kB)
Collecting asttokens>=2.1.0
  Downloading asttokens-2.4.1-py2.py3-none-any.whl (27 kB)
Collecting pure-eval
  Downloading pure_eval-0.2.2-py3-none-any.whl (11 kB)
Collecting jsonpointer>1.13
  Downloading jsonpointer-2.4-py2.py3-none-any.whl (7.8 kB)
Collecting isoduration
  Downloading isoduration-20.11.0-py3-none-any.whl (11 kB)
Collecting webcolors>=1.11
  Downloading webcolors-1.13-py3-none-any.whl (14 kB)
Collecting fqdn
  Downloading fqdn-1.5.1-py3-none-any.whl (9.1 kB)
Collecting uri-template
  Downloading uri_template-1.3.0-py3-none-any.whl (11 kB)
Collecting cffi>=1.0.1
  Downloading cffi-1.16.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (443 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 443.9/443.9 KB 10.7 MB/s eta 0:00:00
Collecting pycparser
  Downloading pycparser-2.21-py2.py3-none-any.whl (118 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 118.7/118.7 KB 10.7 MB/s eta 0:00:00
Collecting arrow>=0.15.0
  Downloading arrow-1.3.0-py3-none-any.whl (66 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 66.4/66.4 KB 9.5 MB/s eta 0:00:00
Collecting types-python-dateutil>=2.8.10
  Downloading types_python_dateutil-2.8.19.14-py3-none-any.whl (9.4 kB)
Using legacy 'setup.py install' for maxminddb-geolite2, since package 'wheel' is not installed.
Installing collected packages: webencodings, wcwidth, types-python-dateutil, pytz, pure-eval, ptyprocess, json5, fastjsonschema, widgetsnbextension, websocket-client, webcolors, urllib3, uri-template, tzdata, typing-extensions, traitlets, tornado, tomli, tinycss2, threadpoolctl, tenacity, soupsieve, sniffio, six, setuptools, send2trash, rpds-py, rfc3986-validator, pyzmq, pyyaml, python-json-logger, pyparsing, pygments, pycparser, psutil, prompt-toolkit, prometheus-client, platformdirs, pillow, pexpect, parso, pandocfilters, packaging, overrides, numpy, networkx, nest-asyncio, mistune, markupsafe, kiwisolver, jupyterlab-widgets, jupyterlab-pygments, jsonpointer, joblib, idna, graphviz, fqdn, fonttools, executing, exceptiongroup, defusedxml, decorator, debugpy, cycler, charset-normalizer, certifi, babel, attrs, terminado, scipy, rfc3339-validator, requests, referencing, qtpy, python-dateutil, plotly, maxminddb, matplotlib-inline, jupyter-core, jinja2, jedi, contourpy, comm, cffi, bleach, beautifulsoup4, async-lru, asttokens, anyio, stack-data, scikit-learn, pandas, maxminddb-geolite2, matplotlib, jupyter-server-terminals, jupyter-client, jsonschema-specifications, arrow, argon2-cffi-bindings, seaborn, jsonschema, isoduration, ipython, imbalanced-learn, catboost, argon2-cffi, nbformat, ipywidgets, ipykernel, imblearn, qtconsole, nbclient, jupyter-events, jupyter-console, nbconvert, jupyter-server, notebook-shim, jupyterlab-server, jupyter-lsp, jupyterlab, notebook, jupyter
  Attempting uninstall: setuptools
    Found existing installation: setuptools 59.6.0
    Uninstalling setuptools-59.6.0:
      Successfully uninstalled setuptools-59.6.0
  Running setup.py install for maxminddb-geolite2 ... done
Successfully installed anyio-4.1.0 argon2-cffi-23.1.0 argon2-cffi-bindings-21.2.0 arrow-1.3.0 asttokens-2.4.1 async-lru-2.0.4 attrs-23.1.0 babel-2.13.1 beautifulsoup4-4.12.2 bleach-6.1.0 catboost-1.2.2 certifi-2023.11.17 cffi-1.16.0 charset-normalizer-3.3.2 comm-0.2.0 contourpy-1.2.0 cycler-0.12.1 debugpy-1.8.0 decorator-5.1.1 defusedxml-0.7.1 exceptiongroup-1.2.0 executing-2.0.1 fastjsonschema-2.19.0 fonttools-4.46.0 fqdn-1.5.1 graphviz-0.20.1 idna-3.6 imbalanced-learn-0.11.0 imblearn-0.0 ipykernel-6.27.1 ipython-8.18.1 ipywidgets-8.1.1 isoduration-20.11.0 jedi-0.19.1 jinja2-3.1.2 joblib-1.3.2 json5-0.9.14 jsonpointer-2.4 jsonschema-4.20.0 jsonschema-specifications-2023.11.2 jupyter-1.0.0 jupyter-client-8.6.0 jupyter-console-6.6.3 jupyter-core-5.5.0 jupyter-events-0.9.0 jupyter-lsp-2.2.1 jupyter-server-2.12.1 jupyter-server-terminals-0.4.4 jupyterlab-4.0.9 jupyterlab-pygments-0.3.0 jupyterlab-server-2.25.2 jupyterlab-widgets-3.0.9 kiwisolver-1.4.5 markupsafe-2.1.3 matplotlib-3.8.2 matplotlib-inline-0.1.6 maxminddb-2.5.1 maxminddb-geolite2-2018.703 mistune-3.0.2 nbclient-0.9.0 nbconvert-7.12.0 nbformat-5.9.2 nest-asyncio-1.5.8 networkx-3.2.1 notebook-7.0.6 notebook-shim-0.2.3 numpy-1.26.2 overrides-7.4.0 packaging-23.2 pandas-2.1.4 pandocfilters-1.5.0 parso-0.8.3 pexpect-4.9.0 pillow-10.1.0 platformdirs-4.1.0 plotly-5.18.0 prometheus-client-0.19.0 prompt-toolkit-3.0.41 psutil-5.9.6 ptyprocess-0.7.0 pure-eval-0.2.2 pycparser-2.21 pygments-2.17.2 pyparsing-3.1.1 python-dateutil-2.8.2 python-json-logger-2.0.7 pytz-2023.3.post1 pyyaml-6.0.1 pyzmq-25.1.2 qtconsole-5.5.1 qtpy-2.4.1 referencing-0.32.0 requests-2.31.0 rfc3339-validator-0.1.4 rfc3986-validator-0.1.1 rpds-py-0.13.2 scikit-learn-1.3.2 scipy-1.11.4 seaborn-0.13.0 send2trash-1.8.2 setuptools-69.0.2 six-1.16.0 sniffio-1.3.0 soupsieve-2.5 stack-data-0.6.3 tenacity-8.2.3 terminado-0.18.0 threadpoolctl-3.2.0 tinycss2-1.2.1 tomli-2.0.1 tornado-6.4 traitlets-5.14.0 types-python-dateutil-2.8.19.14 typing-extensions-4.8.0 tzdata-2023.3 uri-template-1.3.0 urllib3-2.1.0 wcwidth-0.2.12 webcolors-1.13 webencodings-0.5.1 websocket-client-1.7.0 widgetsnbextension-4.0.9
(runner) mick@RECHNER:~/RemoteProjects/DataAnalysisLabs$ python3 lab3_regression.py
Execution time for 'load_from_csv' took 0.20080065727233887 seconds
Execution time for 'load_from_csv' took 0.38518524169921875 seconds
Execution time for 'load_from_csv' took 0.5407347679138184 seconds
Execution time for 'load_from_csv' took 0.6924324035644531 seconds
Execution time for 'load_from_csv' took 0.8981702327728271 seconds
Execution time for 'load_from_csv' took 0.0030646324157714844 seconds
Execution time for 'load_from_csv' took 0.002722501754760742 seconds
Execution time for 'load_from_csv' took 0.002686738967895508 seconds
Execution time for 'load_from_csv' took 0.0026960372924804688 seconds
```

Разведочный анализ

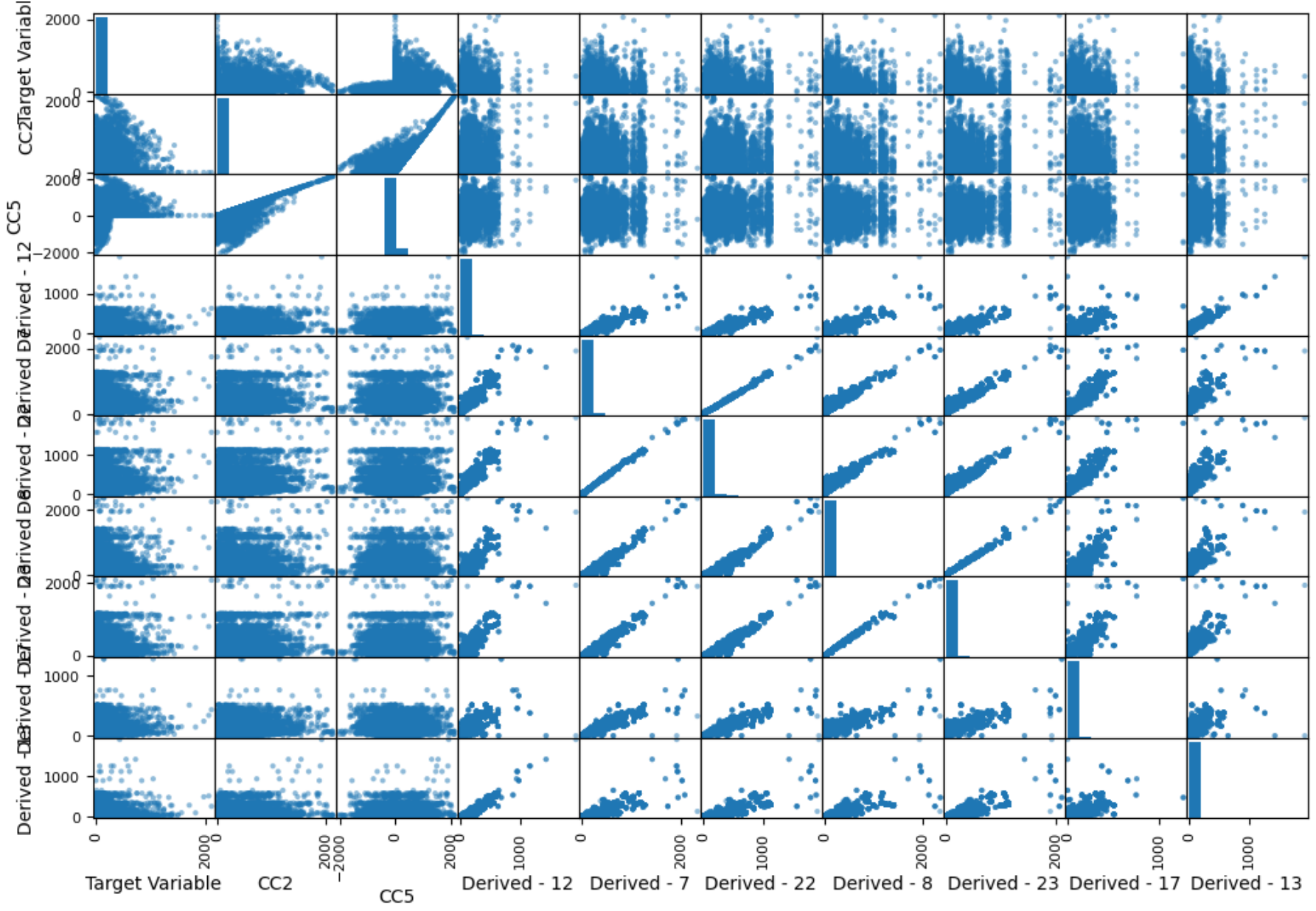
- Это процесс анализа данных в целях выявления основных характеристик, закономерностей, аномалий и тенденций, с использованием различных графических и статистических методов.
- Основная цель разведочного анализа - получить более глубокое понимание структуры данных, выделить ключевые аспекты и гипотезы, которые могут быть дальше исследованы или применены в анализе данных.

Корреляция величин

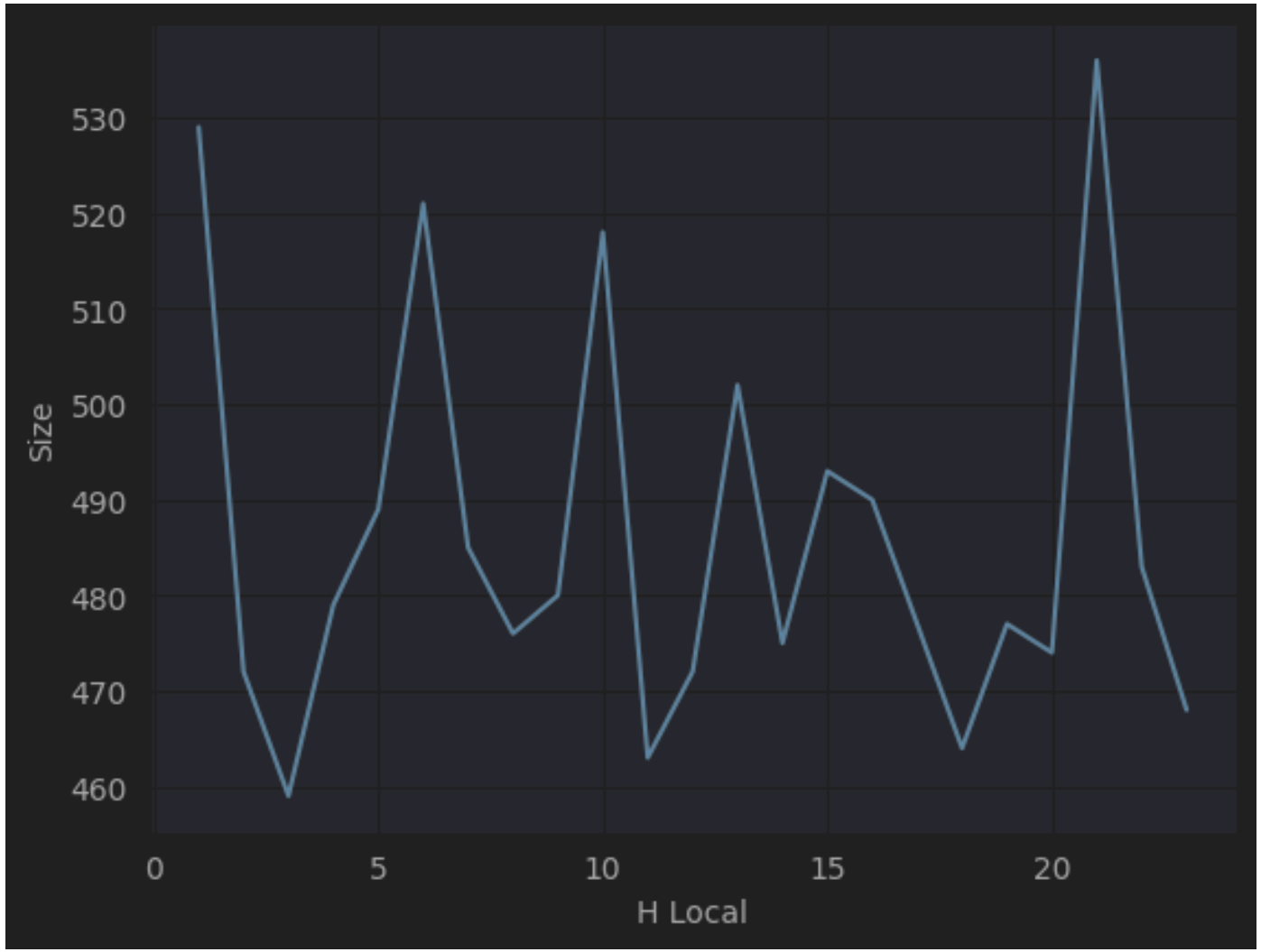
Variable	Correlation
Target Variable	1.000000
CC2	0.537412
CC5	0.372872
Derived - 12	0.366997
Derived - 7	0.357139
Derived - 22	0.354598
Derived - 8	0.350421

Матрица рассеяния

- Это **графическое представление**, в котором каждая пара переменных в наборе данных представлена в виде диаграммы рассеяния **(scatter plot)**.
- Матрица рассеяния используется для **визуального исследования связей** и корреляций между парами переменных, позволяя аналитику быстро выявить потенциальные зависимости и тенденции в данных.
- Каждая ячейка матрицы содержит график, показывающий, как взаимодействуют две конкретные переменные, что помогает выявить структуры и паттерны в данных.



Матрица рассеяния



Распределение размера данных в зависимости от "H Local"

> В промежутке [1:23] значения практически одинаковы!

Однако!!! 😡😡😡😡😡

```
In 15   1  len(training[training['H Local'] == 24])
           Executed in 92ms, 9 Dec at 00:30:25

Out 15        591626
```

Выбор моделей

Для решения проблемы регрессии мы решили выбрать следующие модели, и распределили их между собой.

- **GB, CB** - Михаил Бабушкин
- **NN** - Анатолий Мезенов
- **DT, RF** - Никита Спектровик
- **KNN** - Олег Панов
- **LS, Ridge** - Денис Чашин

> И мы расскажем о них подробнее, но сначала

Напомним вам, насколько важен GridSearch

GridSearchCV – это очень мощный инструмент для автоматического подбирания параметров для моделей машинного обучения. Метод **поиска по сетке** находит наилучшую комбинацию параметров, которые дают **наименьшую ошибку**, путем обычного перебора: **он создает модель для каждой возможной комбинации параметров**.

> Ну а теперь расскажем о важном...

CV *или же кросс-валидация*

> И какую же мы используем 🔥

Кросс-валидация работает путем разделения набора данных на несколько поднаборов, называемых **фолдами**. Затем модель обучается и тестируется несколько раз, каждый раз используя **разные фолды** для тестирования и обучения.

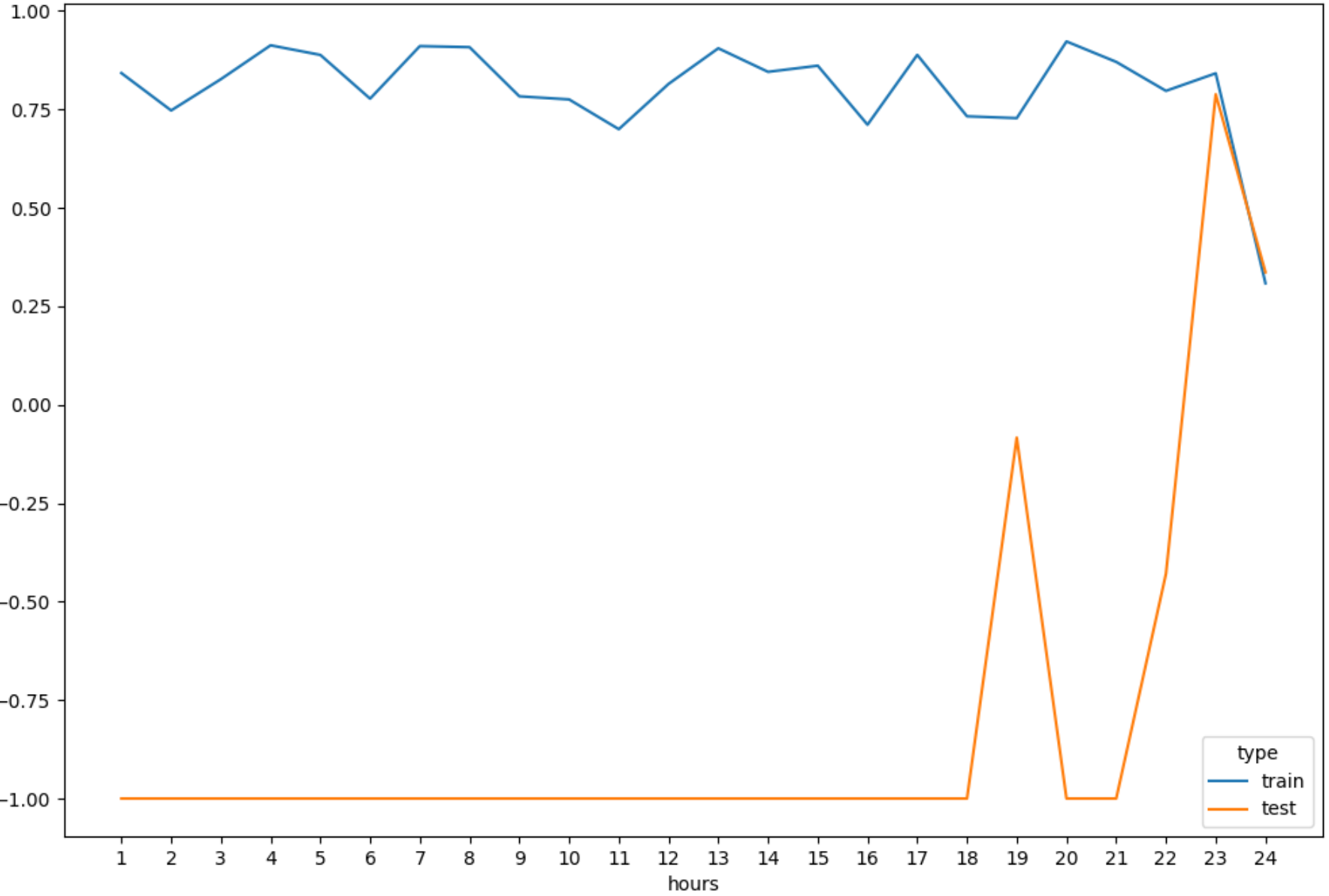
RepeatedStratifiedKFold - это вид кросс-валидации, который помогает учесть разнообразие данных и уменьшить вероятность переобучения модели. Его особенностью является стремление **сохранить баланс классов** в каждом фолде.

Модель LS

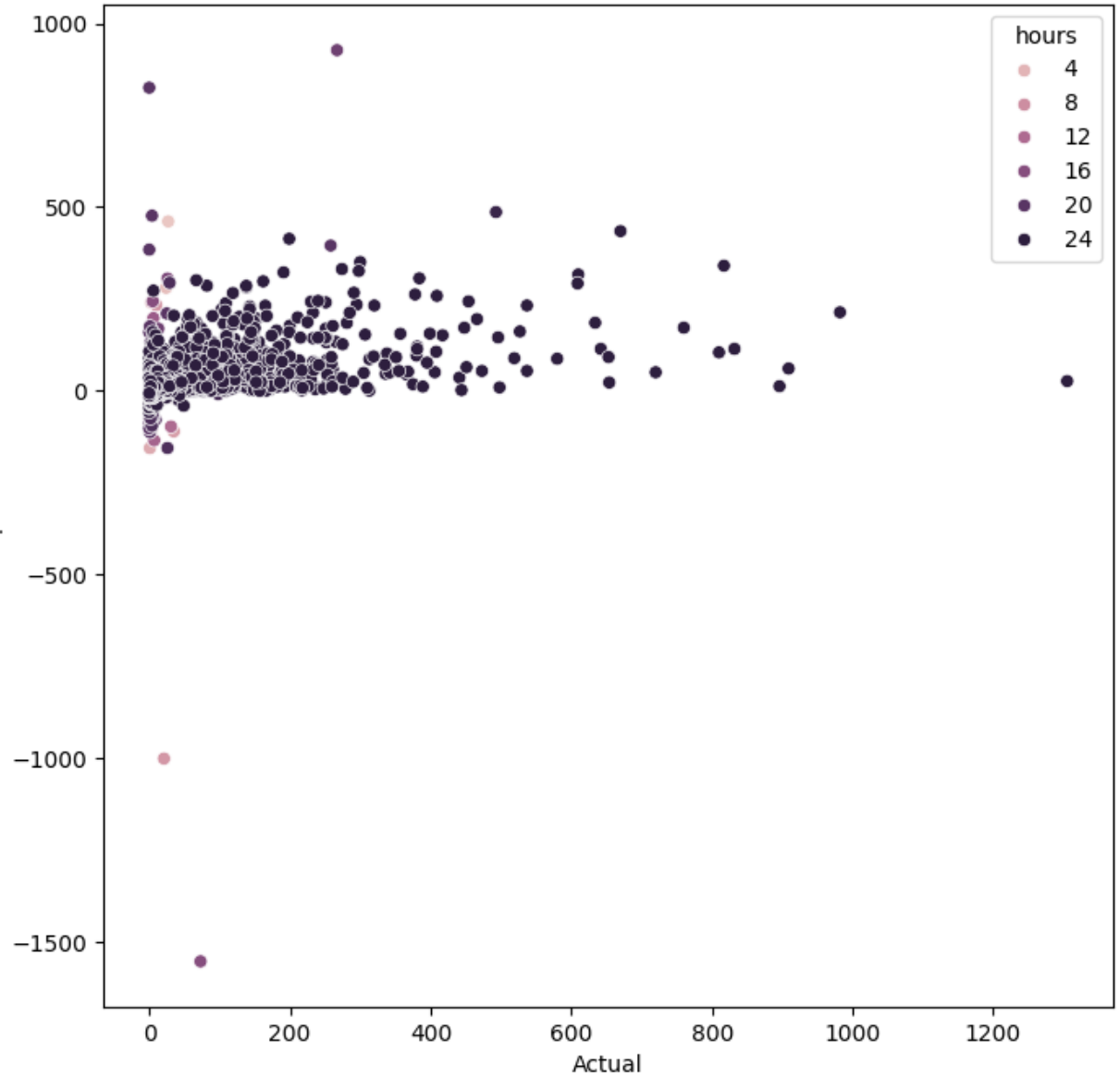
> Метод наименьших квадратов (Least squares)

LS заключается в поиске линейной функции, которая наилучшим образом соответствует данным путем минимизации суммы квадратов разницы между фактическими и предсказанными значениями. Метод **оптимизирует сумму квадратов остатков** и находит оптимальные значения коэффициентов линейной модели.

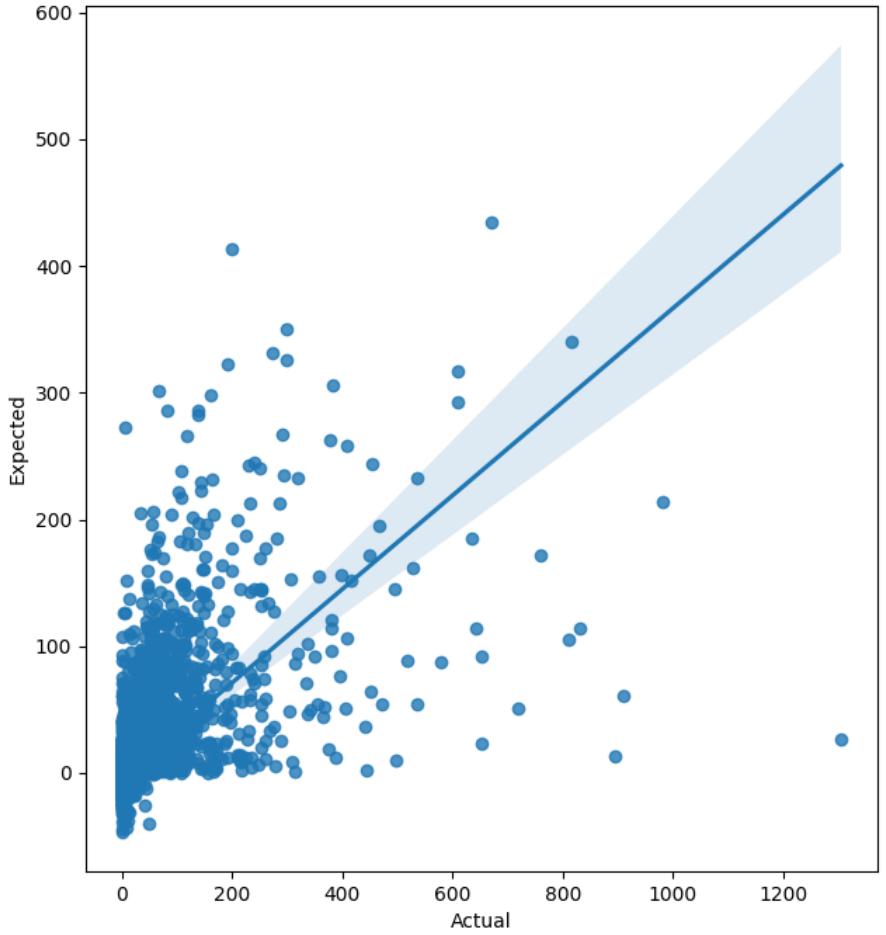
- **Дополнительный плюс** метода состоит в том, что он обеспечивает аналитические *(закрытые)* решения для оценки коэффициентов линейной модели
- **Но** довольно чуствителен к выбросам. Даже небольшие выбросы могут сильно исказить оценки коэффициентов регрессии и делать предсказания менее точными

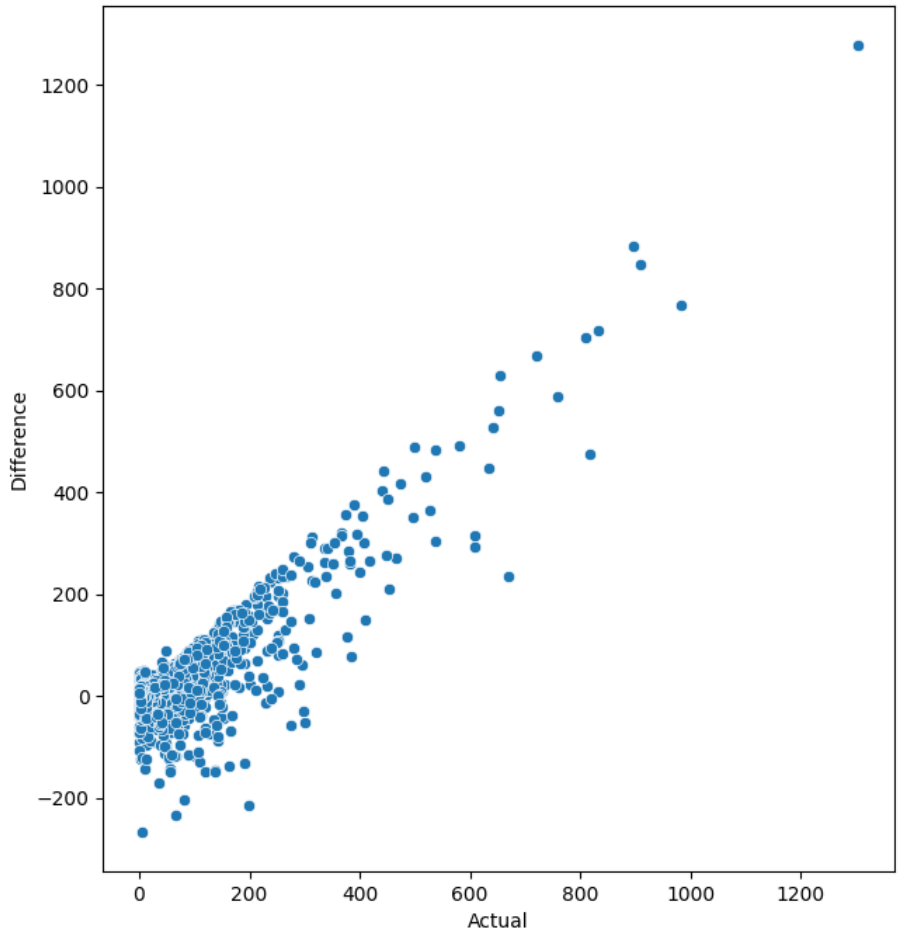


R2 score for LS model

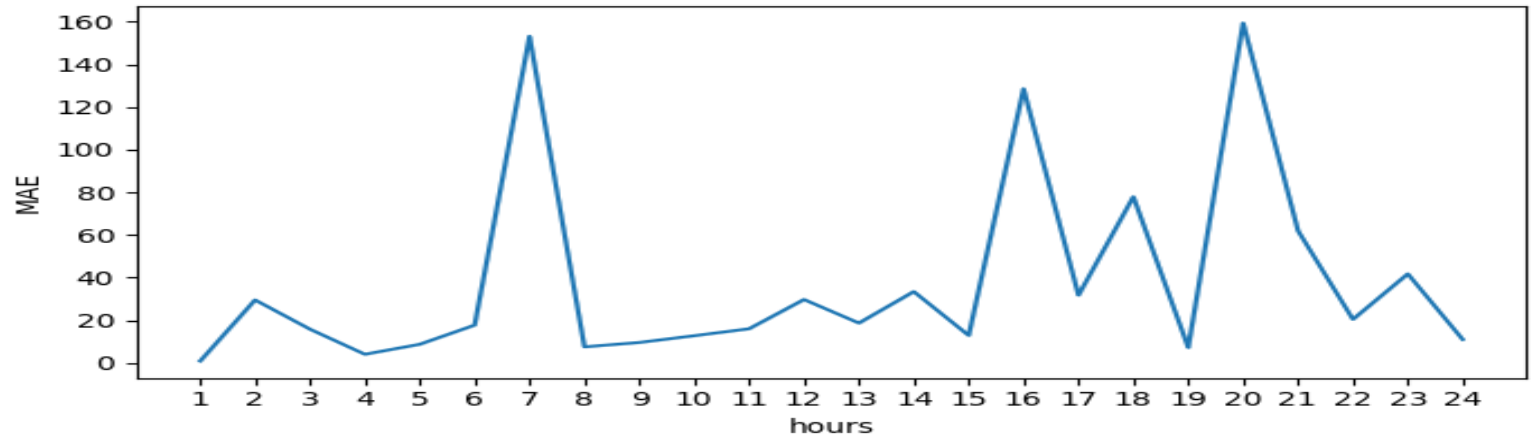


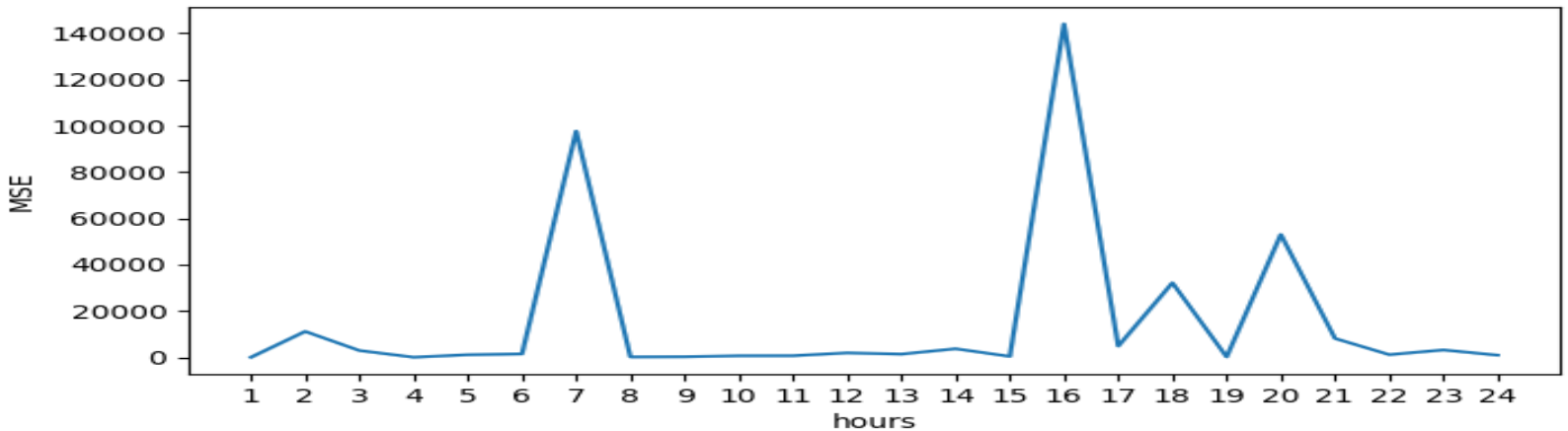
Error plot (overall) for LS model

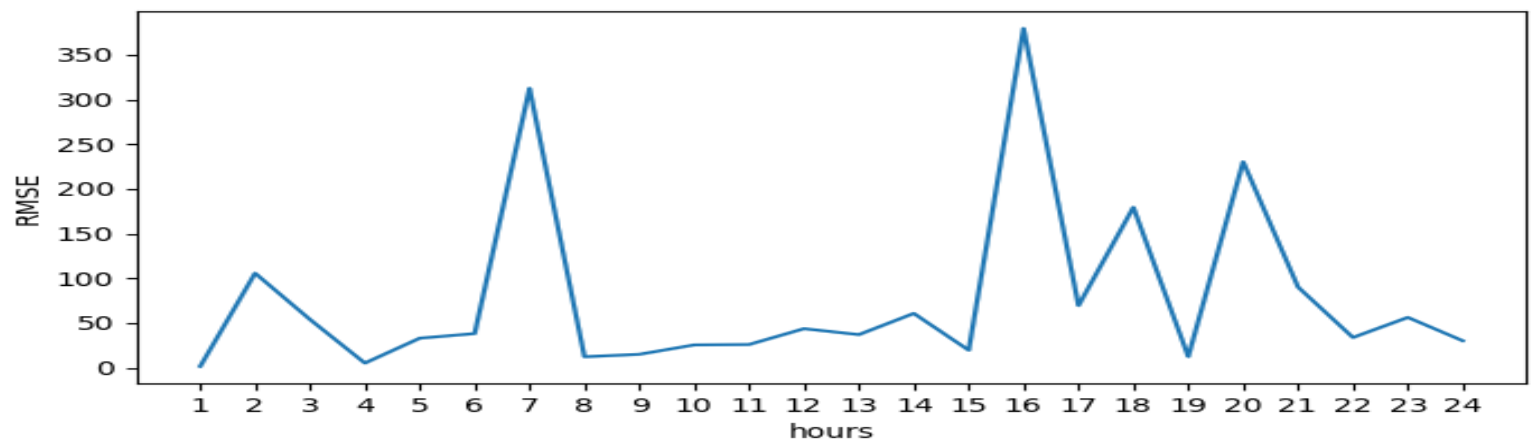




Error plot (24h) for LS model


MAE
0
20
40
60
80
100
120
140
160
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
hours


MSE
0
20000
40000
60000
80000
100000
120000
140000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
hours

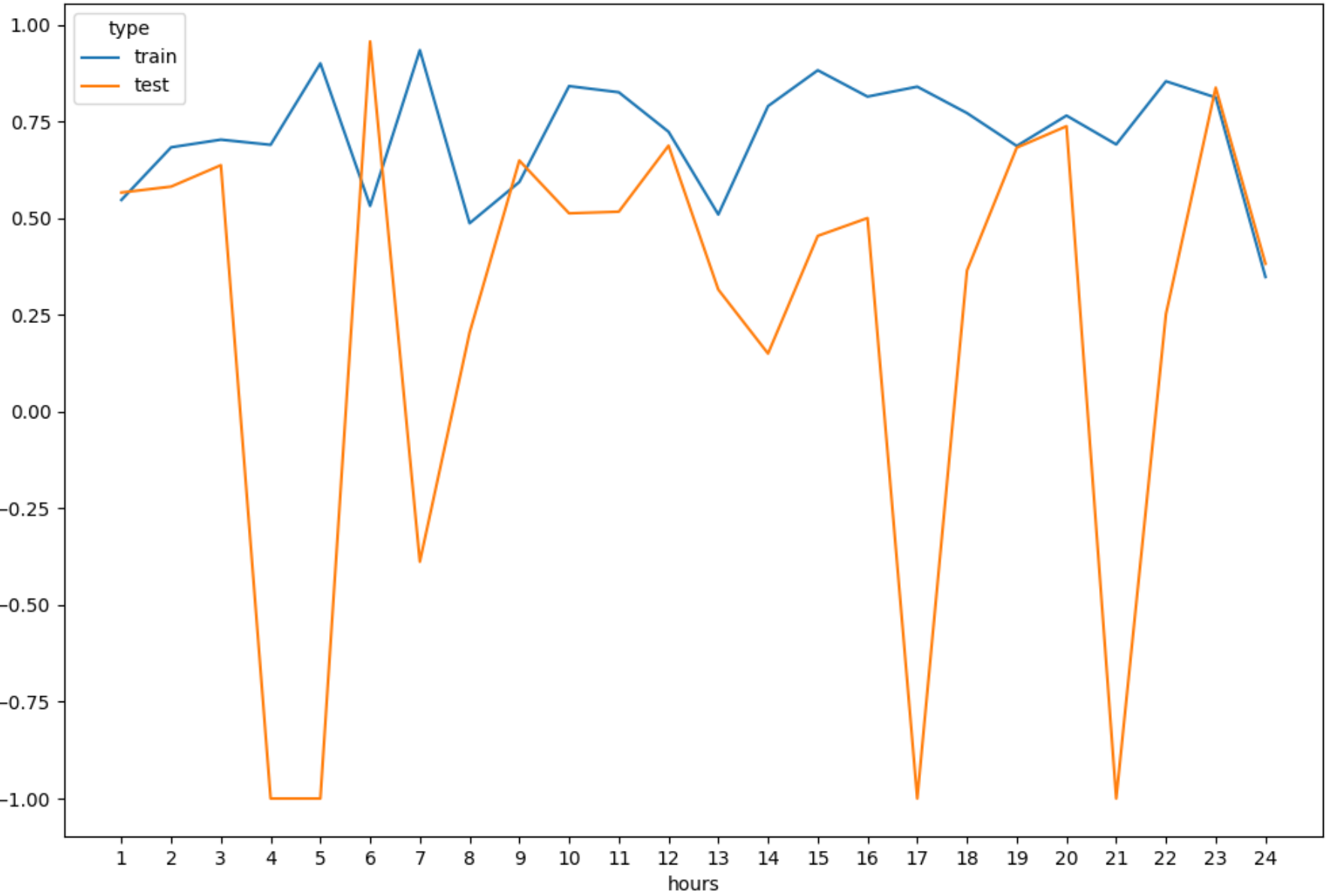

RMSE
0
50
100
150
200
250
300
350
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
hours

Модель Ridge

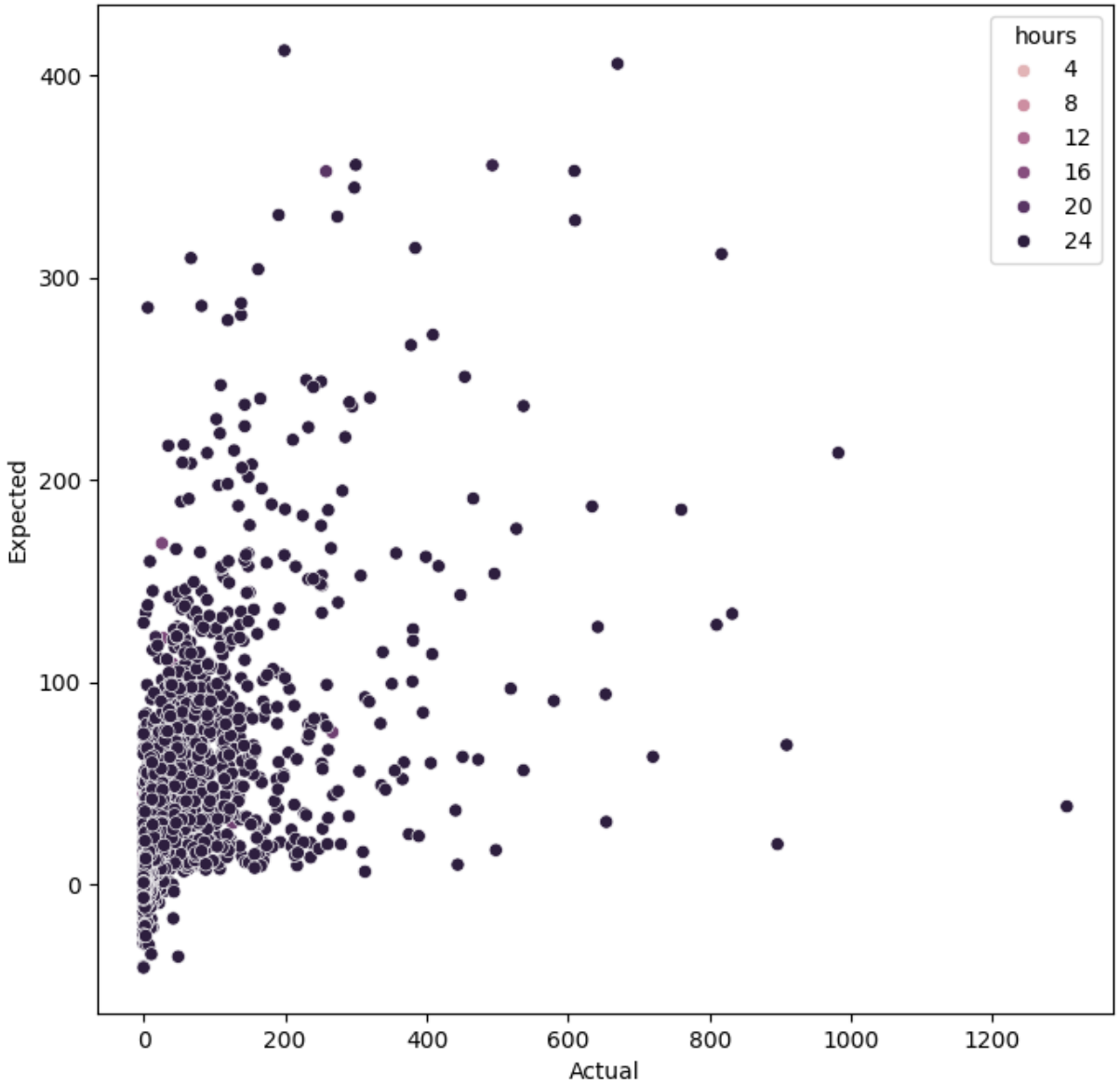
> Ridge ♪

Ridge очень похож на **LS**, ведь он также минимизирует сумму квадратов, но к этой сумме добавляется **штрафование больших значений коэффициентов модели**, что способствует снижению их величины и предотвращает переобучение.

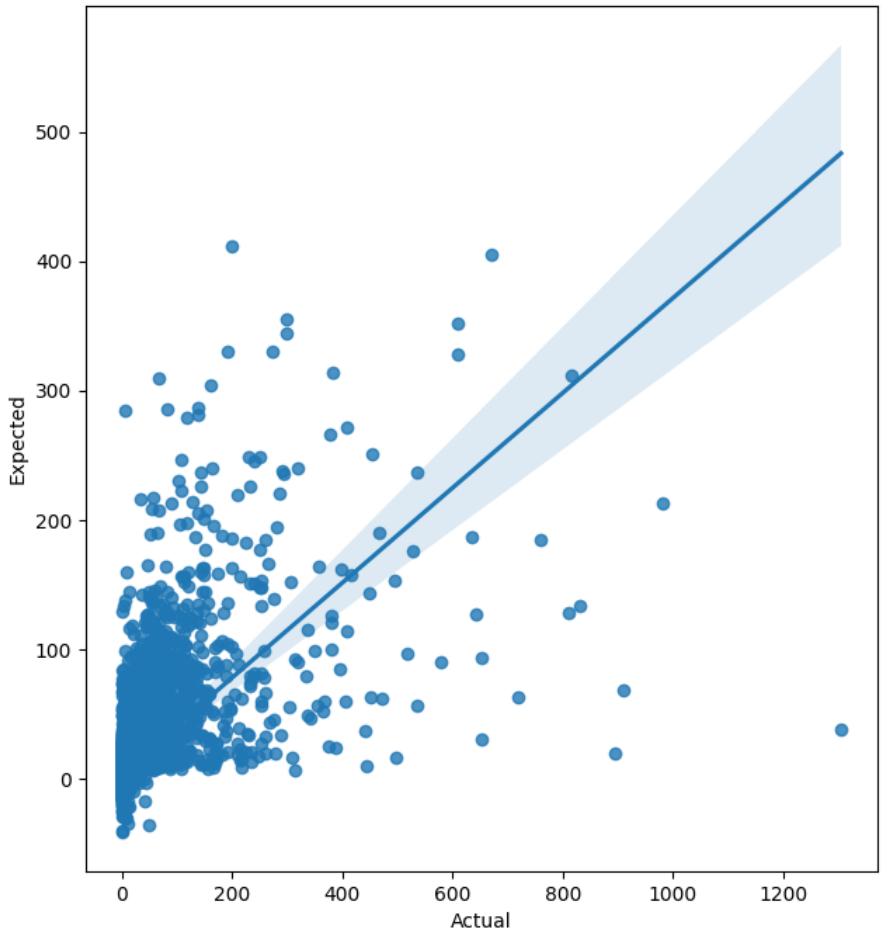
- **Явным плюсом** является стабильность метода, ведь он менее чувствителен к выбросам
- **Но** Из-за регуляризации коэффициенты в Ridge регрессии могут быть менее интерпретируемыми, чем в обычной линейной регрессии, потому что они могут быть уменьшены или даже **занулены**

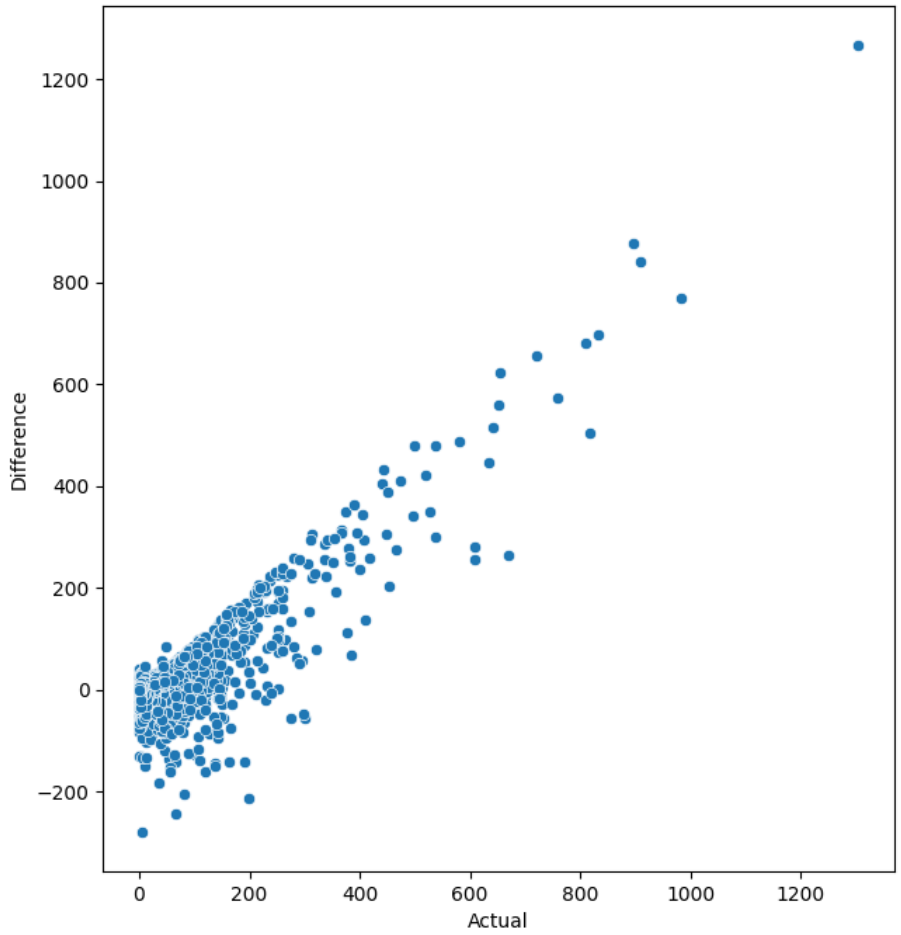


R2 score for Ridge model

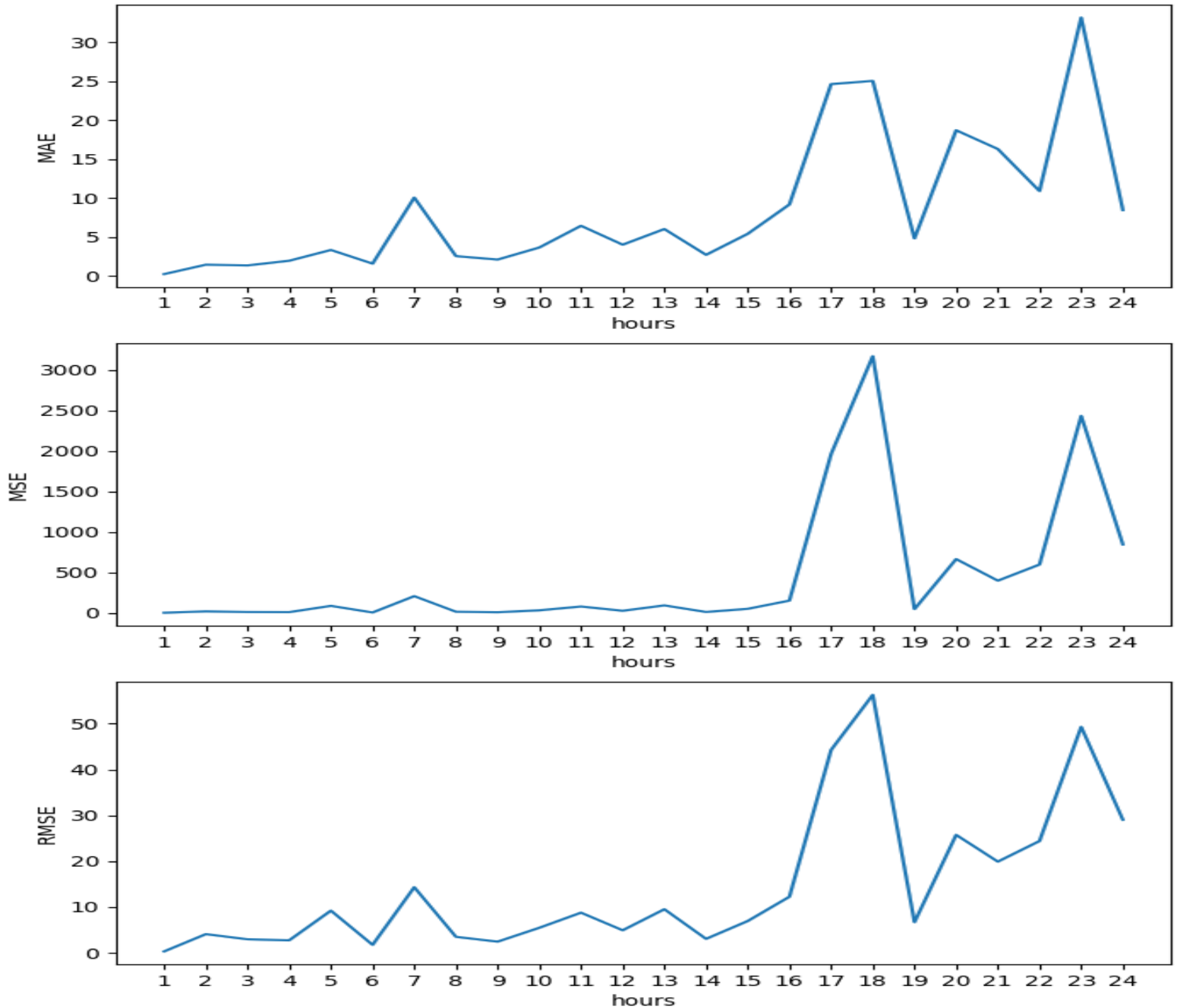


Error plot (overall) for Ridge model





Error plot (24h) for Ridge model

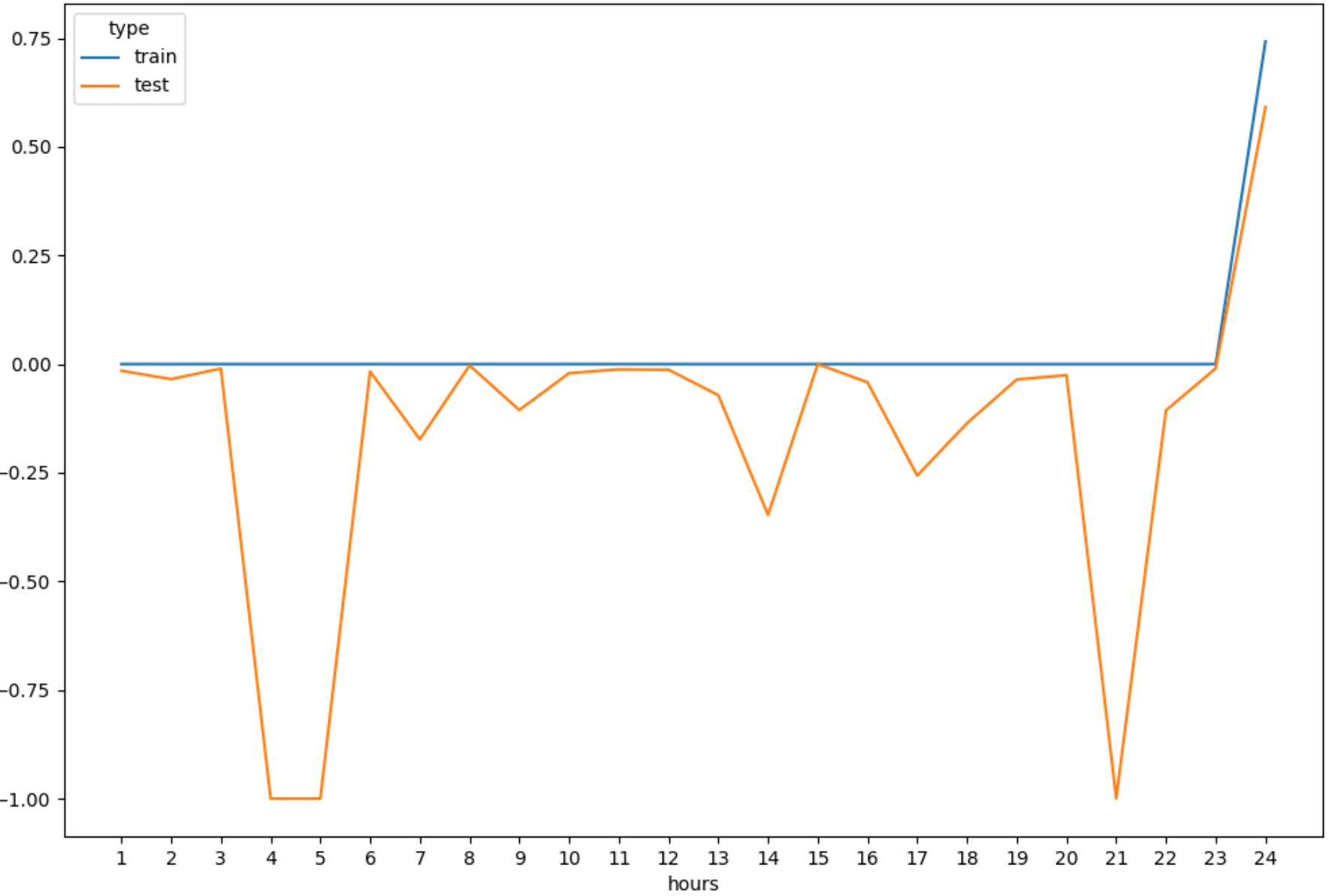

MAE
0
5
10
15
20
25
30
hours
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
MSE
0
500
1000
1500
2000
2500
3000
hours
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
RMSE
0
10
20
30
40
50
hours
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24

Модель DT

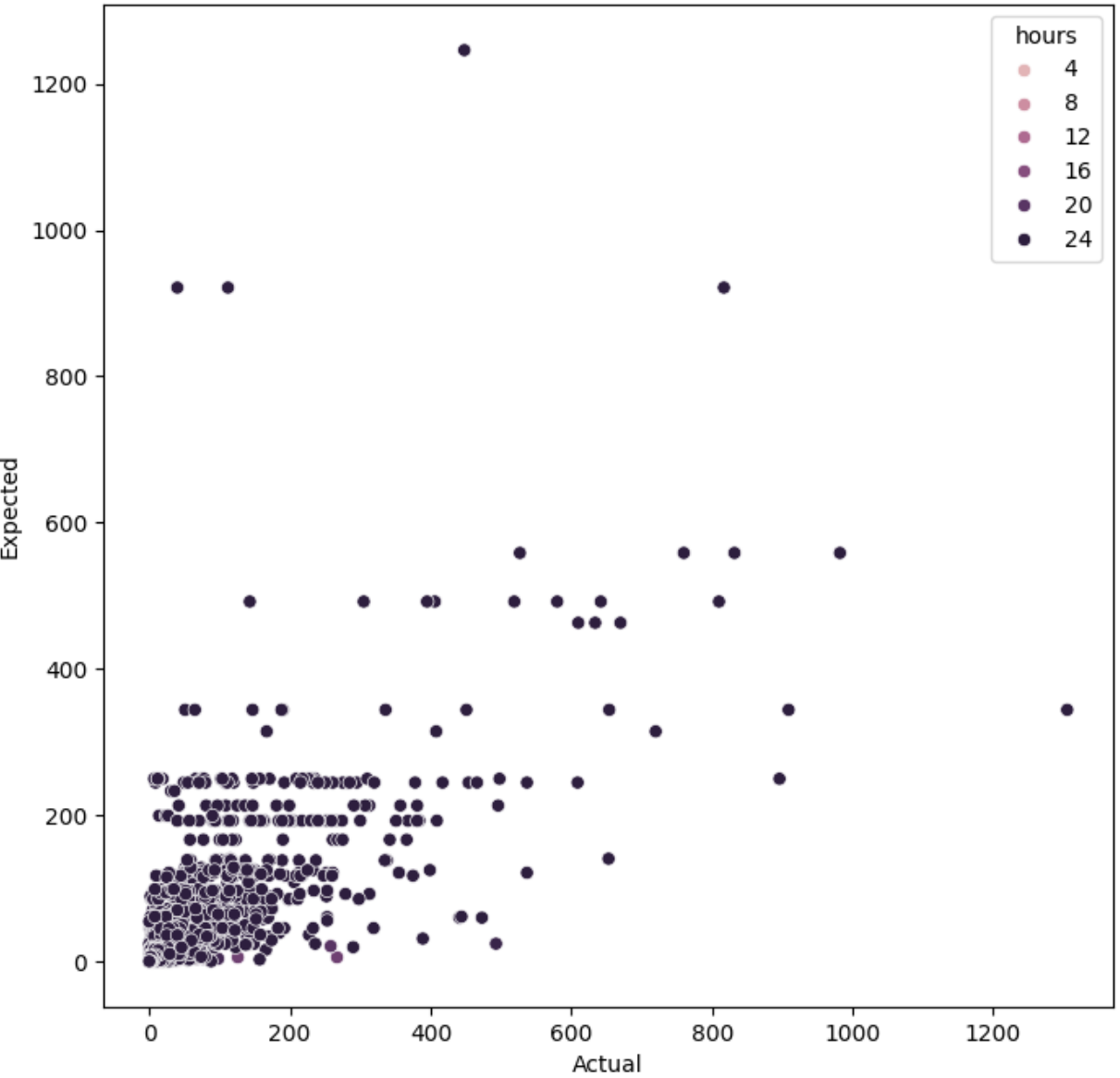
> Decision Tree (Регрессионное дерево)

DT - это метод регрессии, основанный на построении **дерева решений**, где каждый узел представляет собой тест по одному из признаков. Он разбивает данные на подгруппы на основе значений признаков и прогнозирует целевую переменную для каждой подгруппы

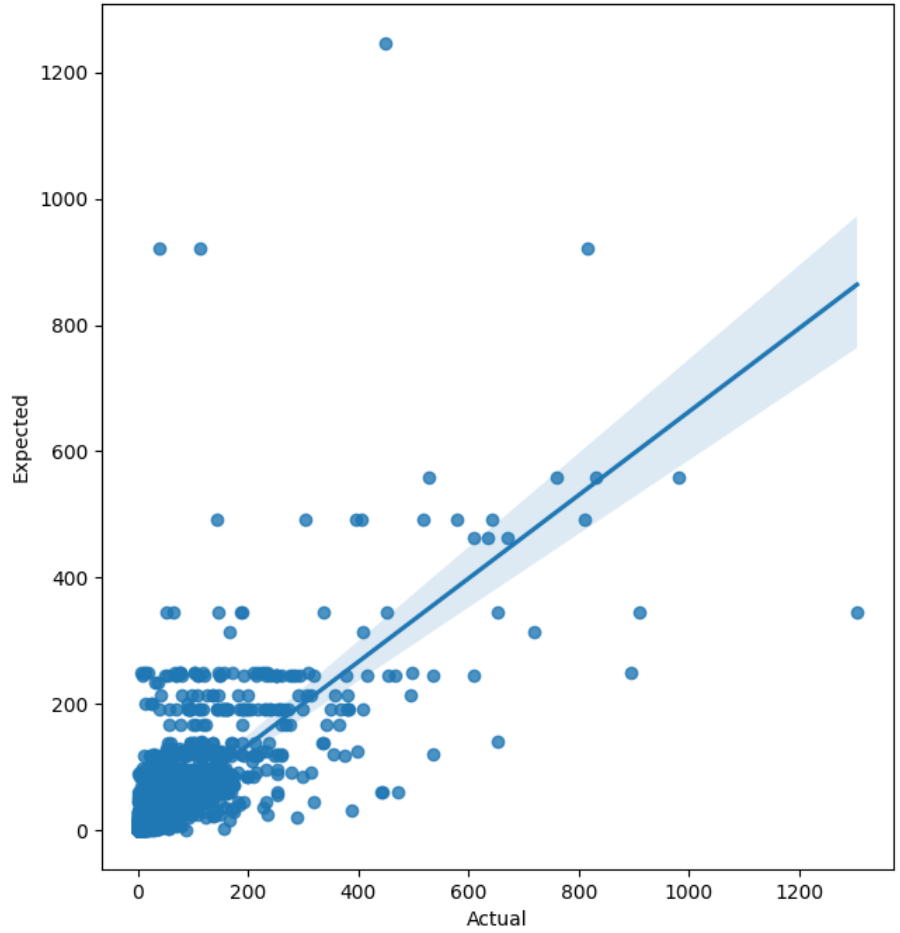
- Чем глубже дерево, тем сложнее правила принятия решений и тем лучше модель.
- Она **хорошо** справляется с обработкой **нелинейных отношений**
- **Но** слишком склонна к переобучению *(возможен случай: хорошо на training, плохо на test)*

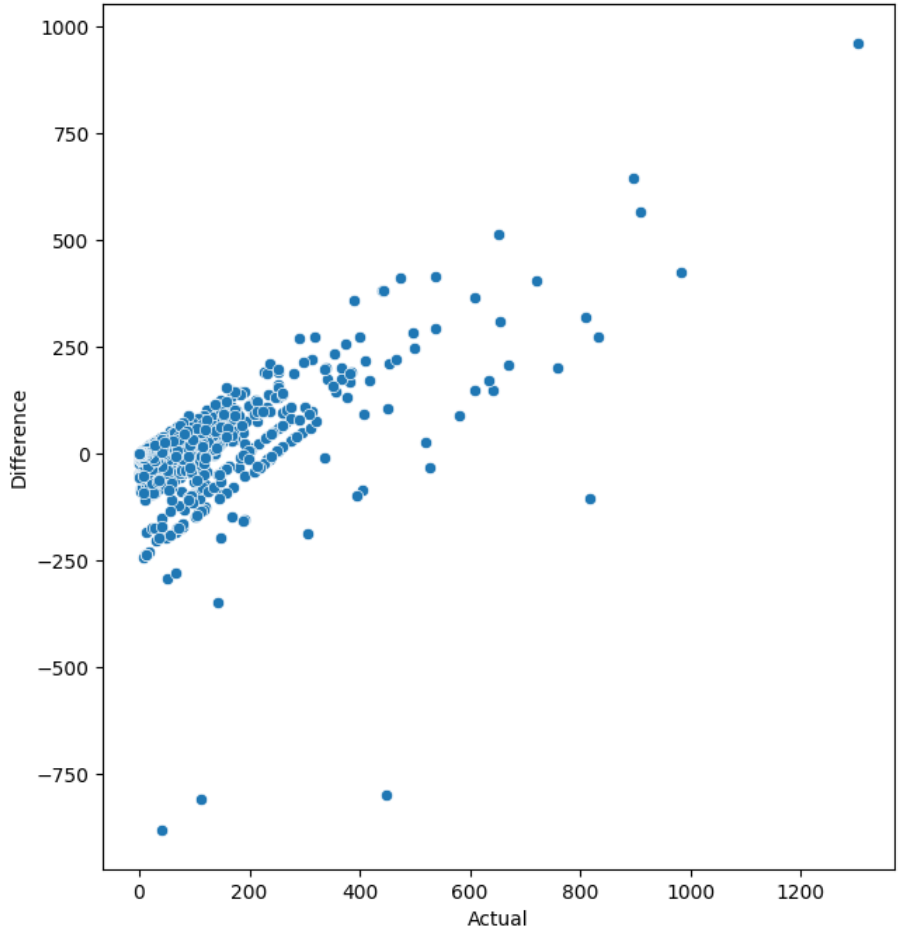


R2 score for Decision Tree model

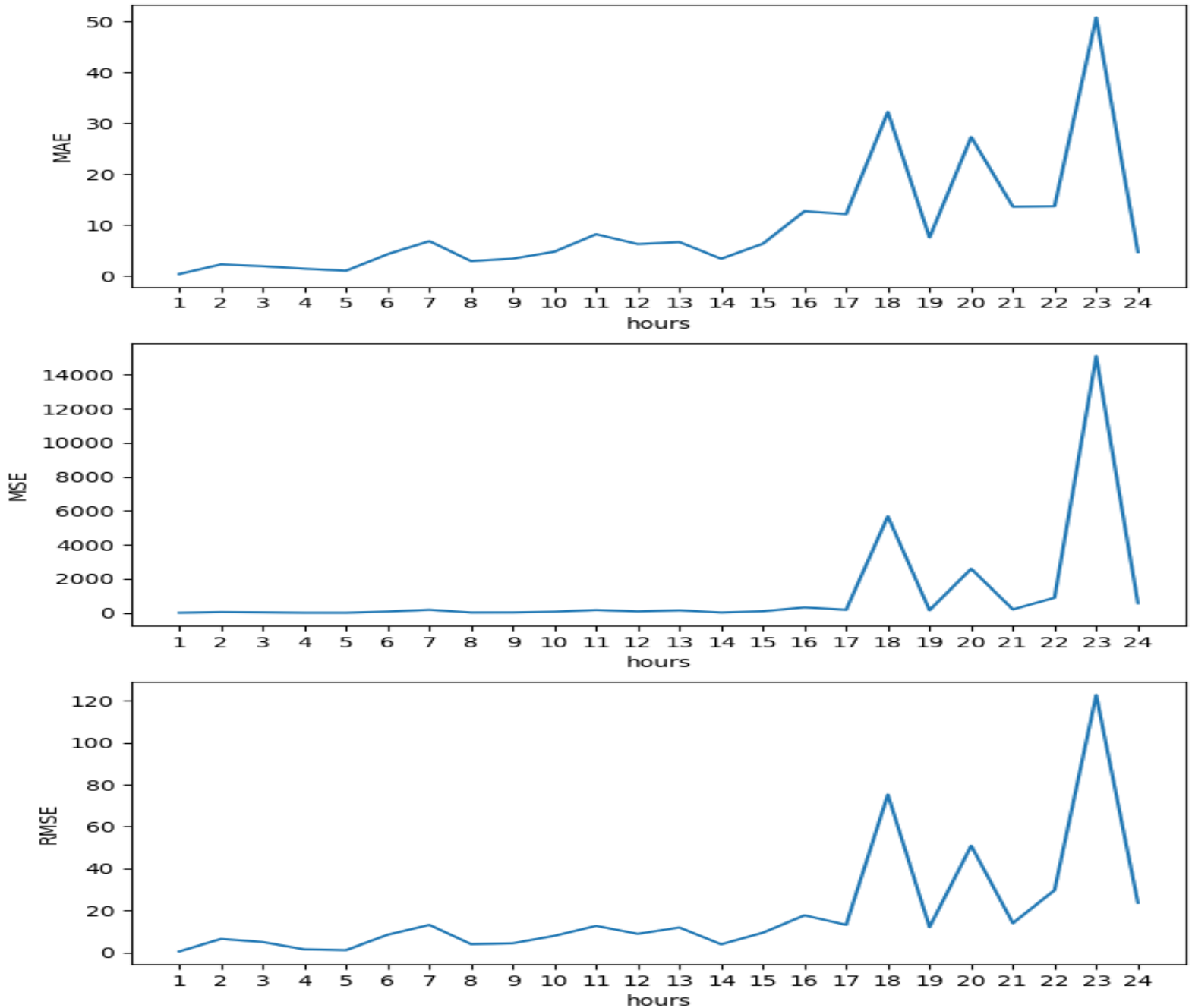


Error plot (overall) for Decision Tree model





Error plot (24h) for Decision Tree model

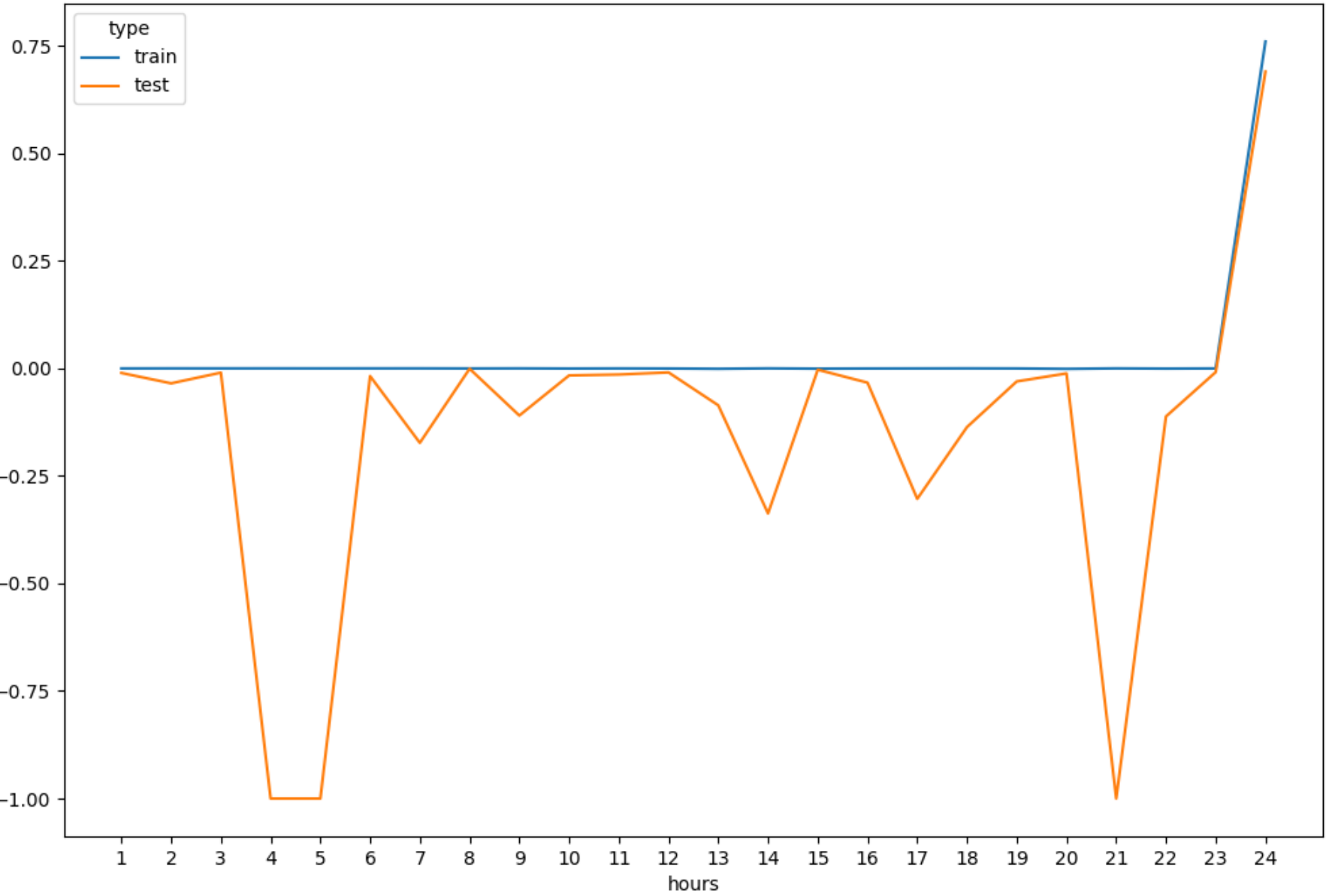

MAE
0
10
20
30
40
50
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
hours
MSE
0
2000
4000
6000
8000
10000
12000
14000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
hours
RMSE
0
20
40
60
80
100
120
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
hours

Модель RF

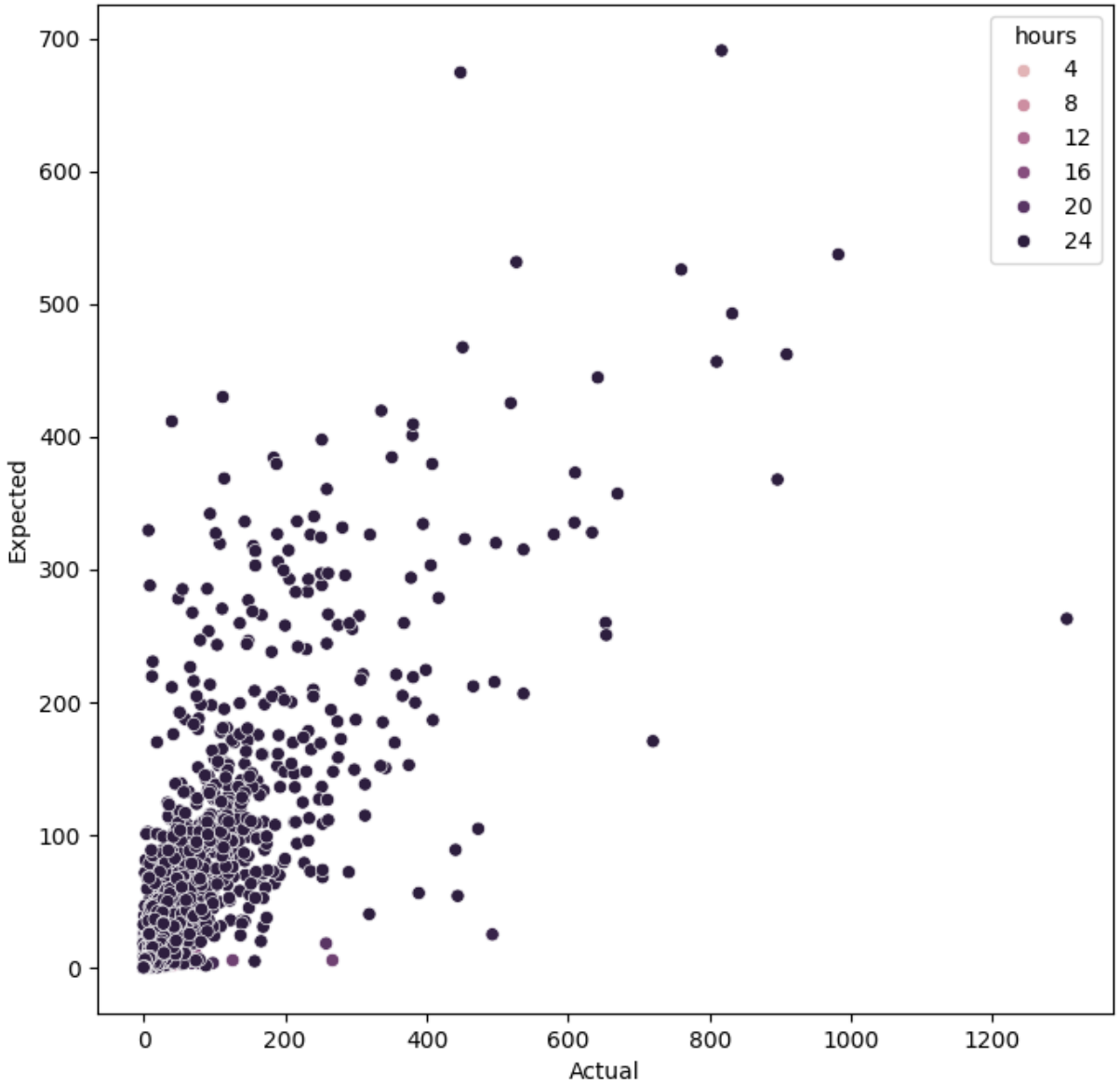
> Random Forest

RF продолжает идею **DT**, ведь он создает лес из этих деревьев. При предсказании модель **усредняет (или взвешивает) предсказания** всех деревьев, что позволяет уменьшить дисперсию и повысить точность предсказаний.

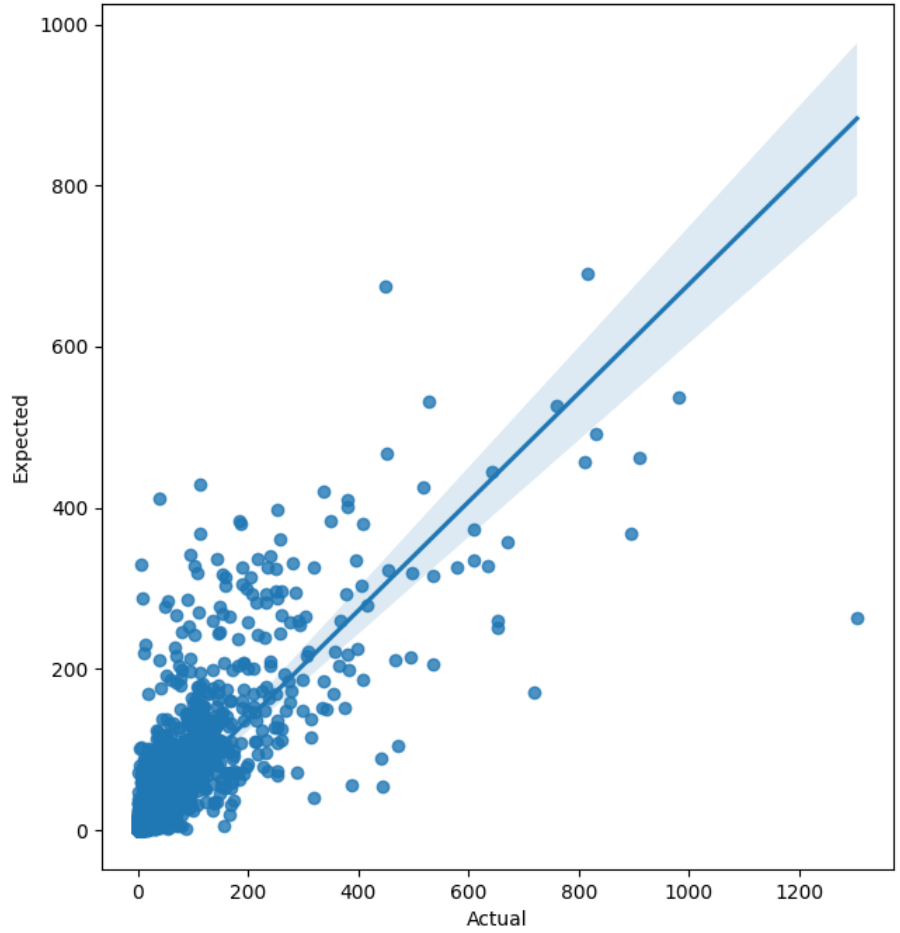
- **Огромный плюс** - высокая точность модели. ~~Верьте на слово~~
- **Но также весомый минус -** из-за большого количества деревьев, модель может быть очень вычислительно затратна

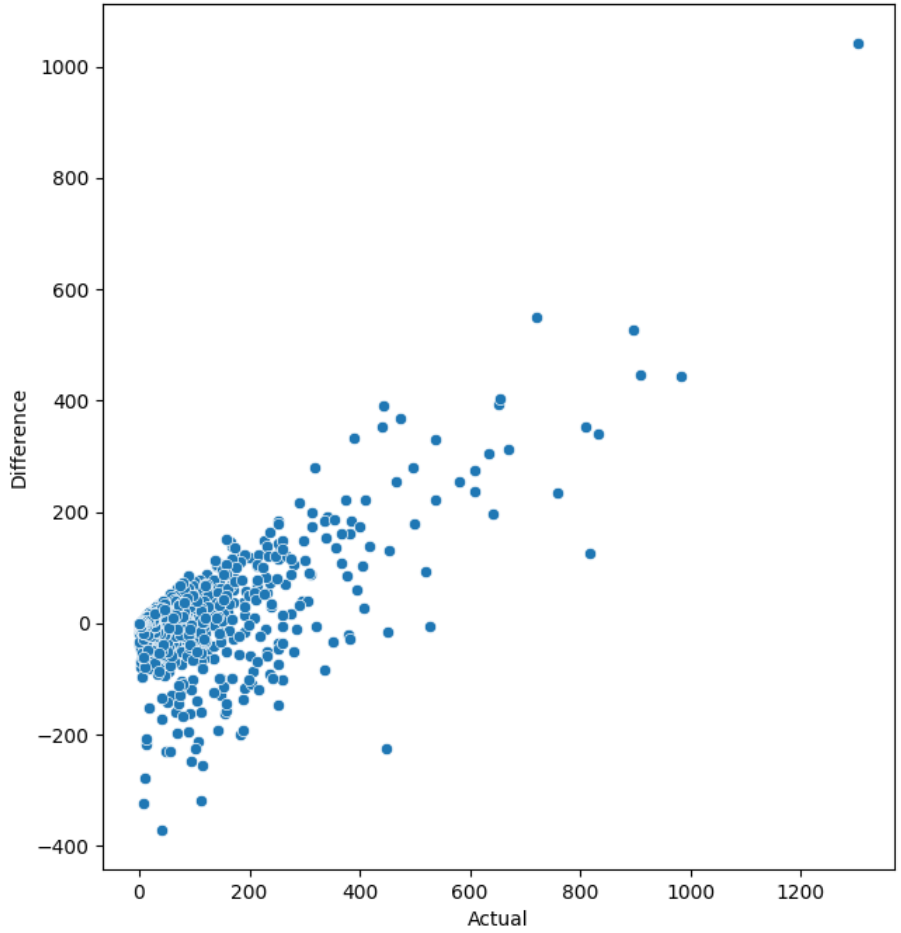


R2 score for Random Forest model

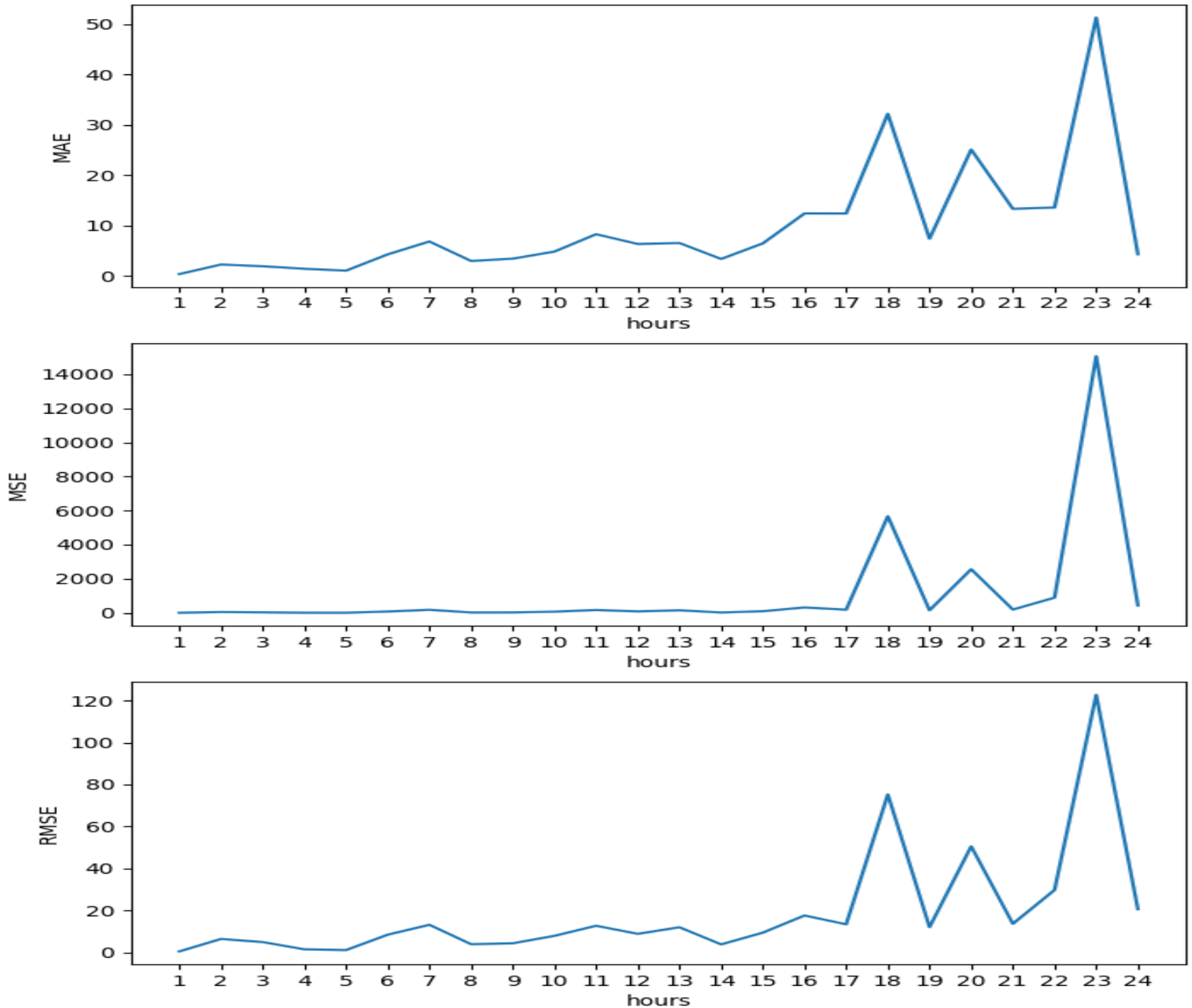


Error plot (overall) for Random Forest model





Error plot (24h) for Random Forest model

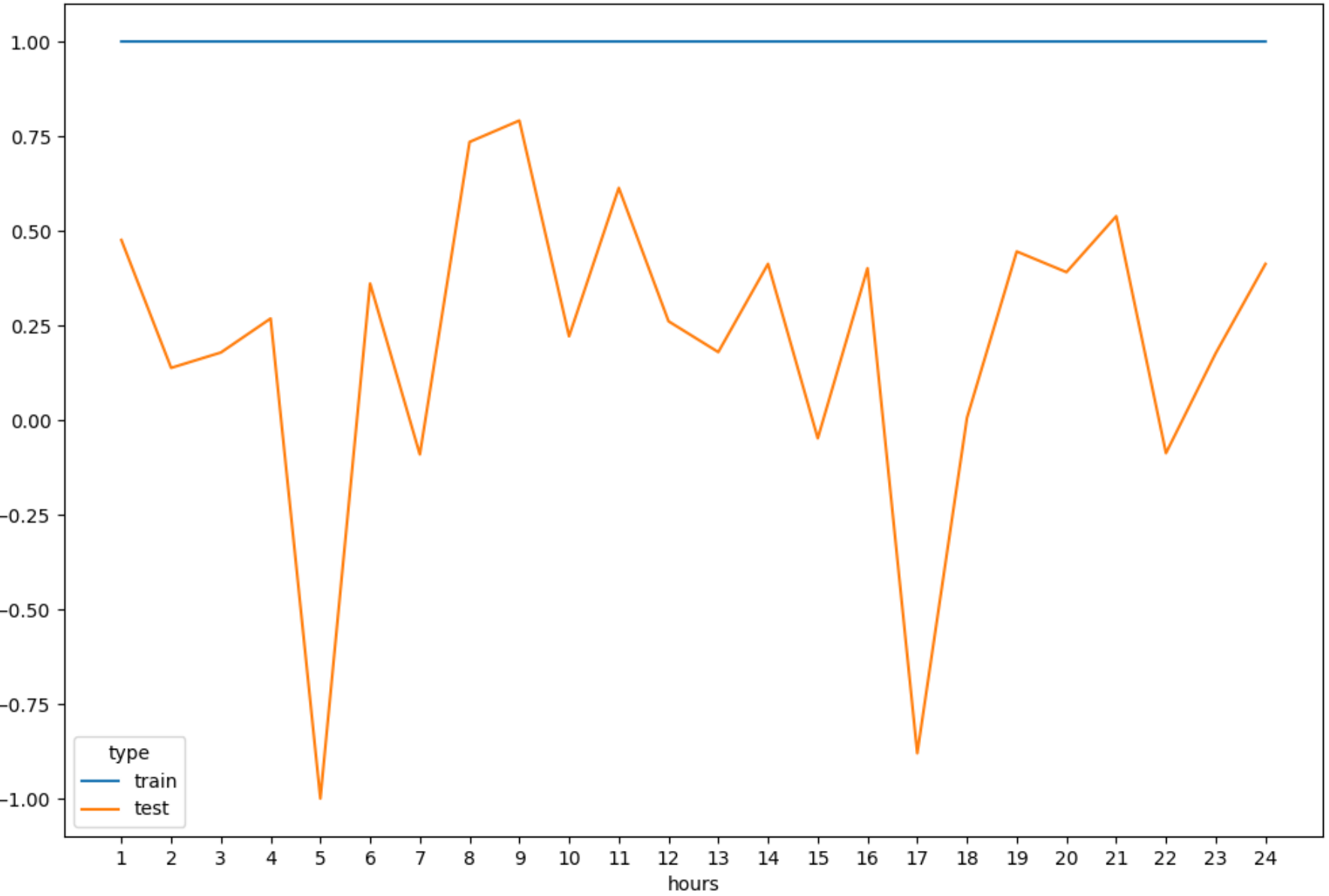

MAE
0
10
20
30
40
50
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
hours
MSE
0
2000
4000
6000
8000
10000
12000
14000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
hours
RMSE
0
20
40
60
80
100
120
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
hours

Модель KNN

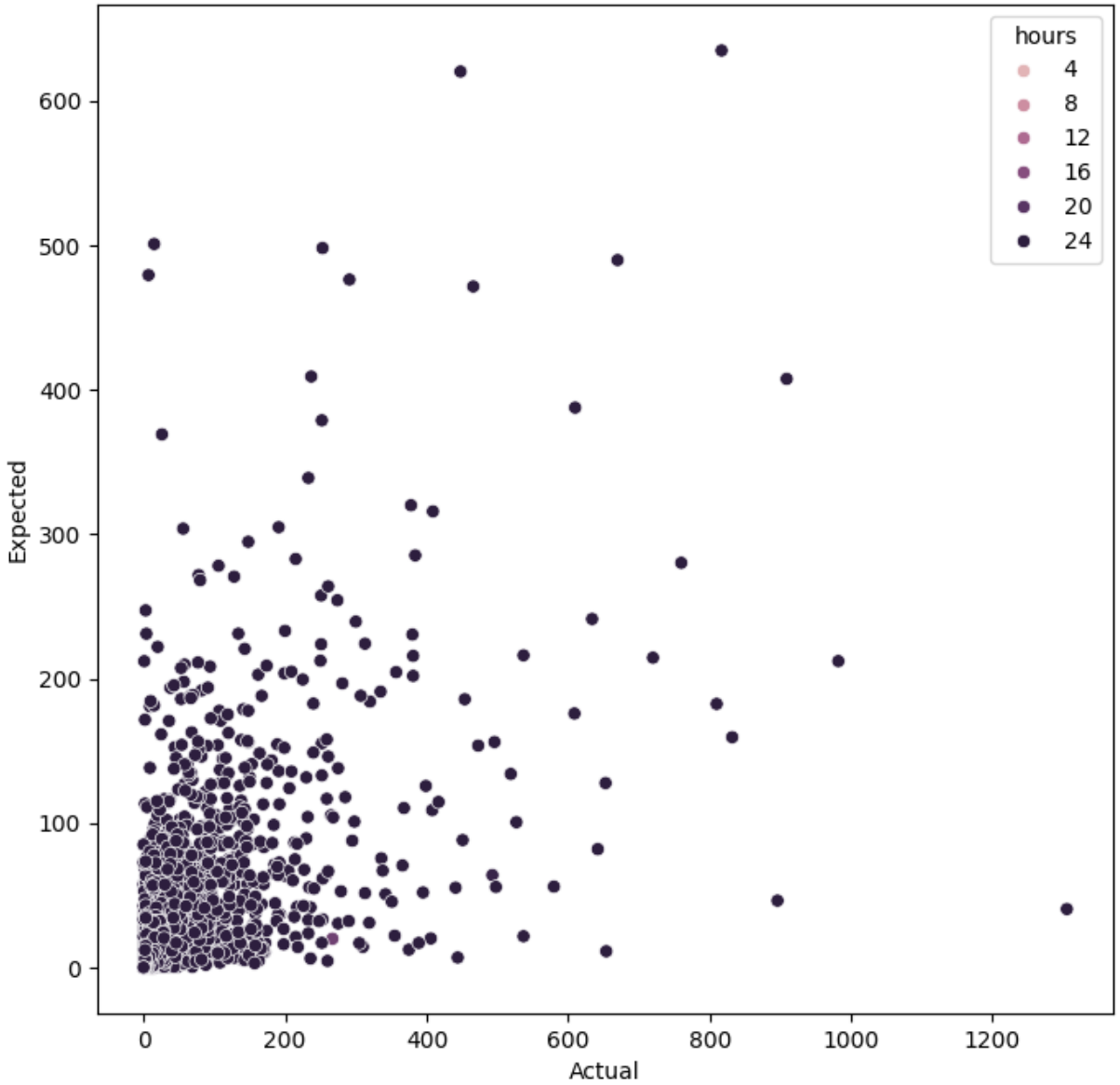
> K Nearest Neighbors (Регрессор "k-ближайших соседей")

Суть **KNN** заключается в том, что он определяет прогноз для нового объекта на основе **среднего** *(или медианного)* значения целевой переменной у её **ближайших соседей** в обучающей выборке.

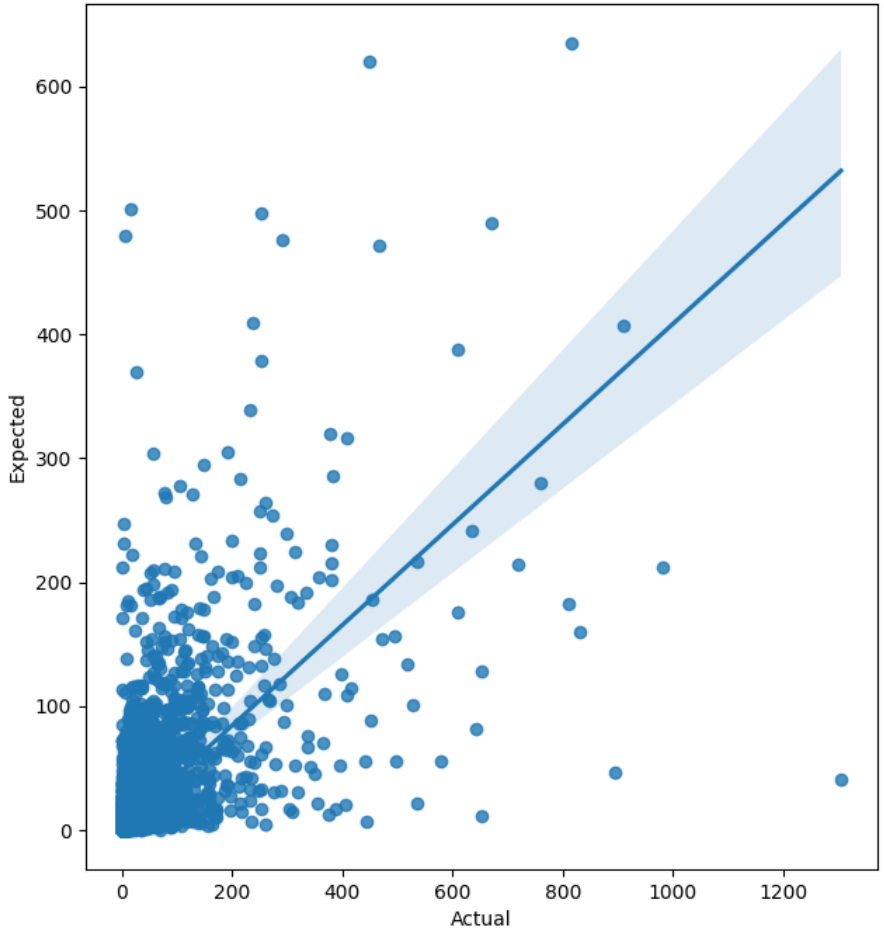
- Модель **хороша**, поскольку ее метод не использует сложную математику, а реализация проста и очевидна.
- **Но не без проблем**: модель очень вычислительно затратна, особенно на больших наборах данных.

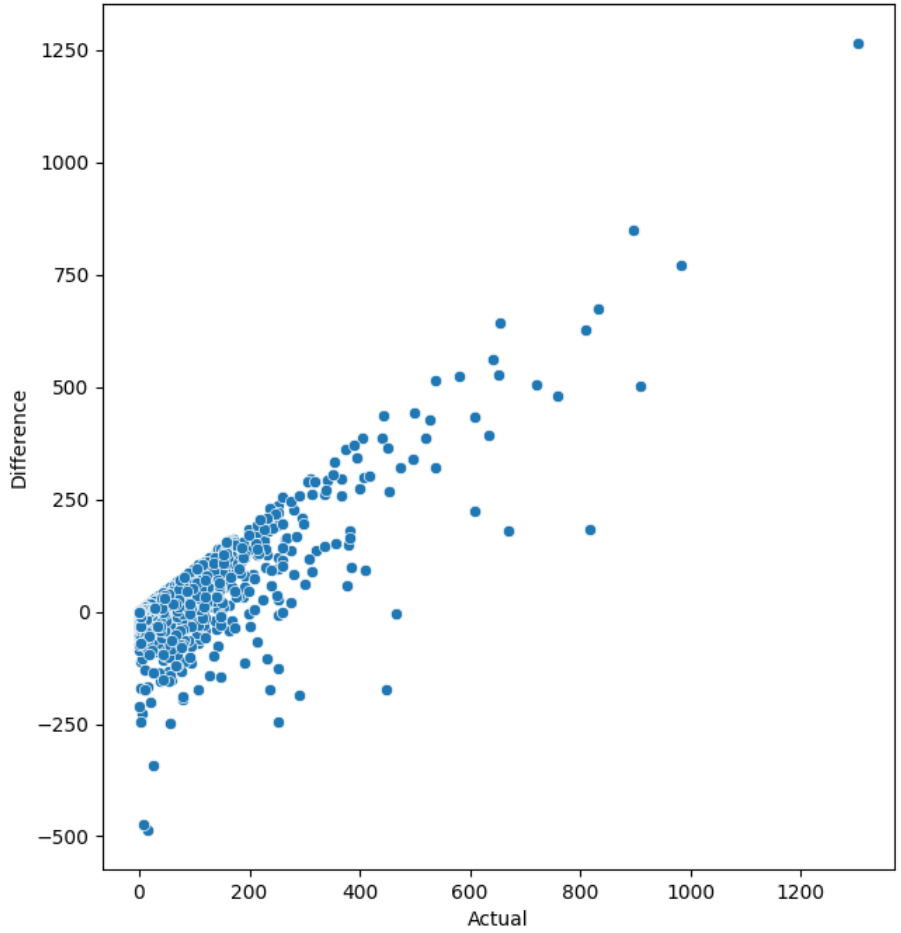


R2 score for KNN model

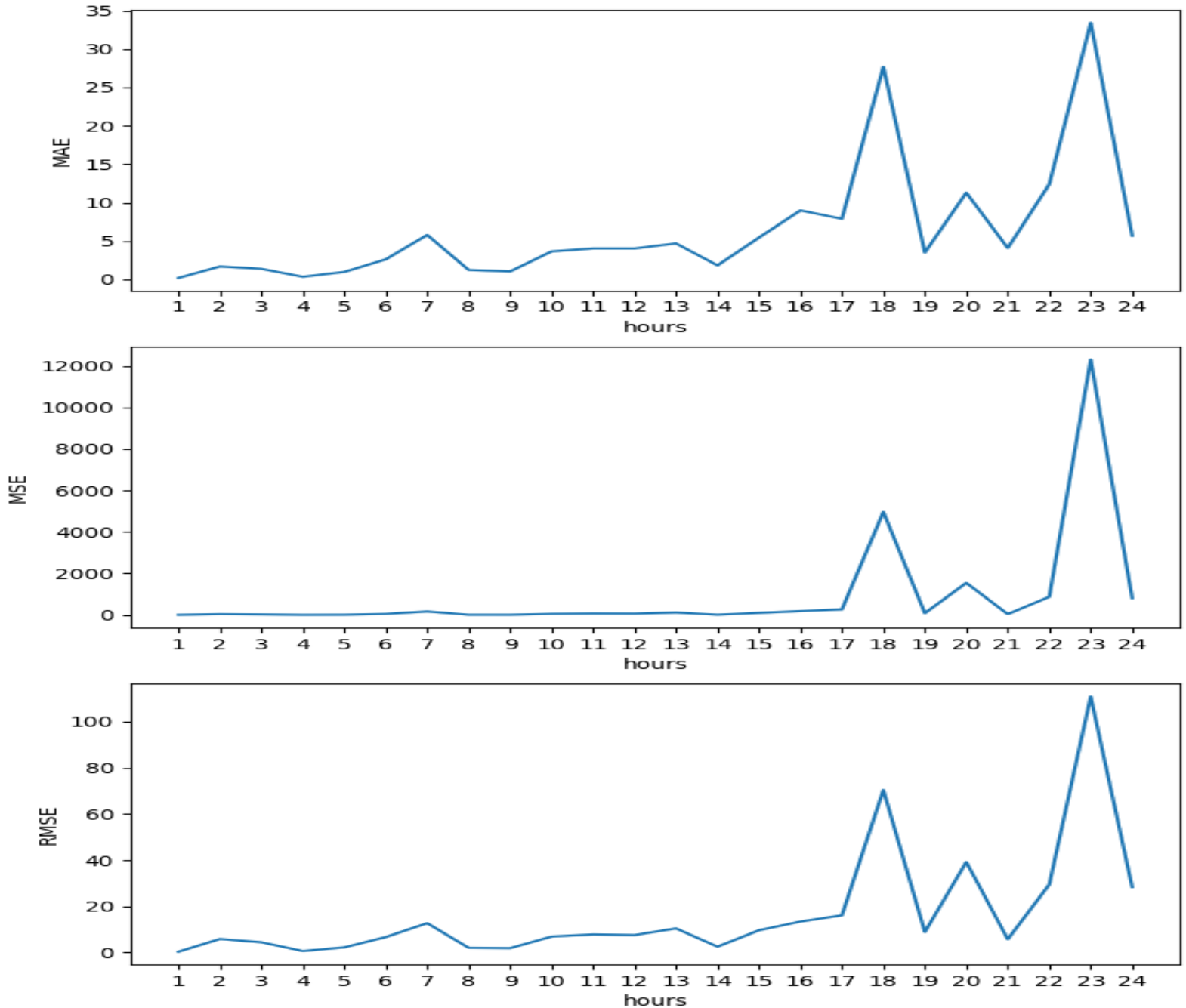


Error plot (overall) for KNN model





Error plot (24h) for KNN model


MAE
0
5
10
15
20
25
30
35
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
hours
MSE
0
2000
4000
6000
8000
10000
12000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
hours
RMSE
0
20
40
60
80
100
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
hours

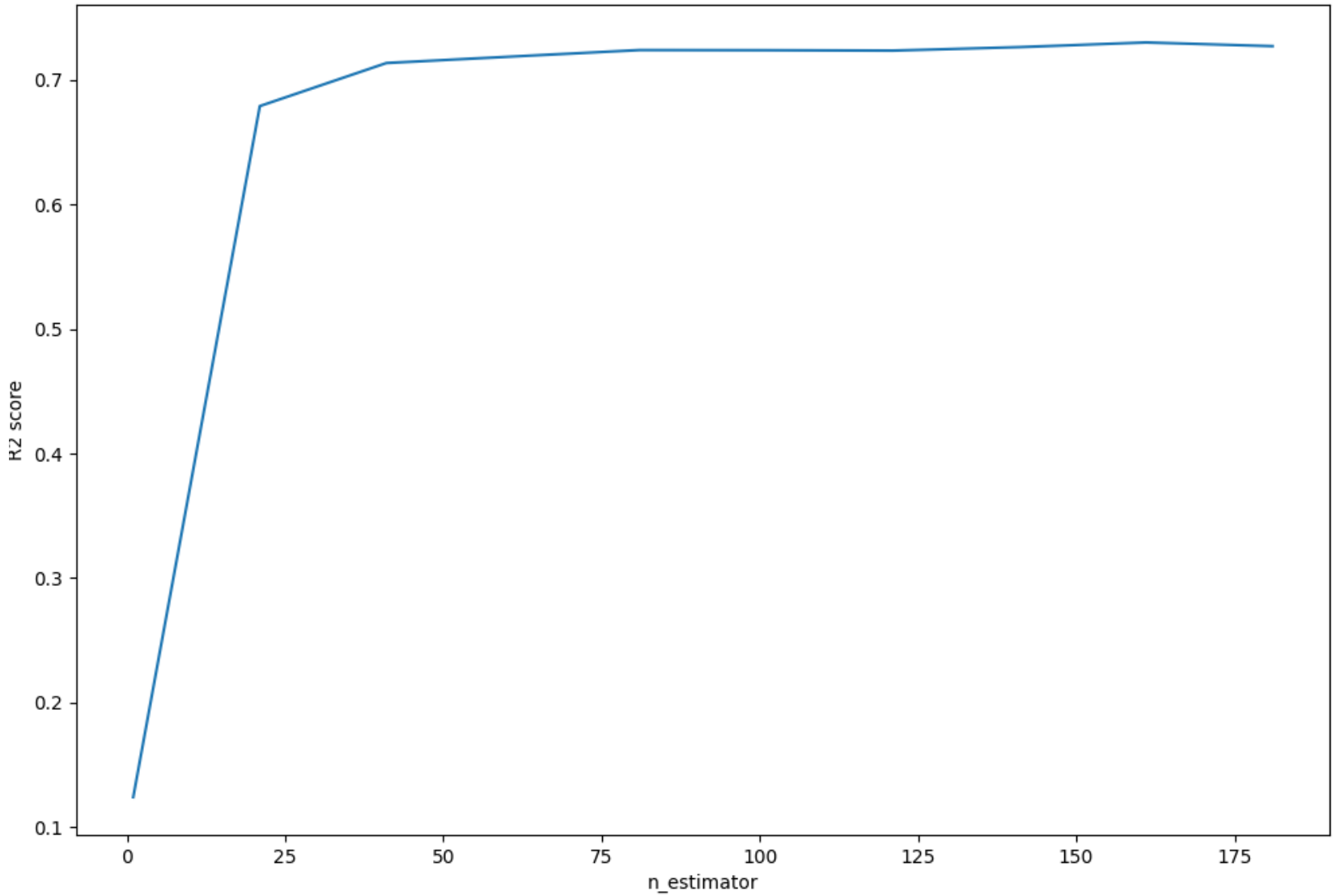
Модель GB

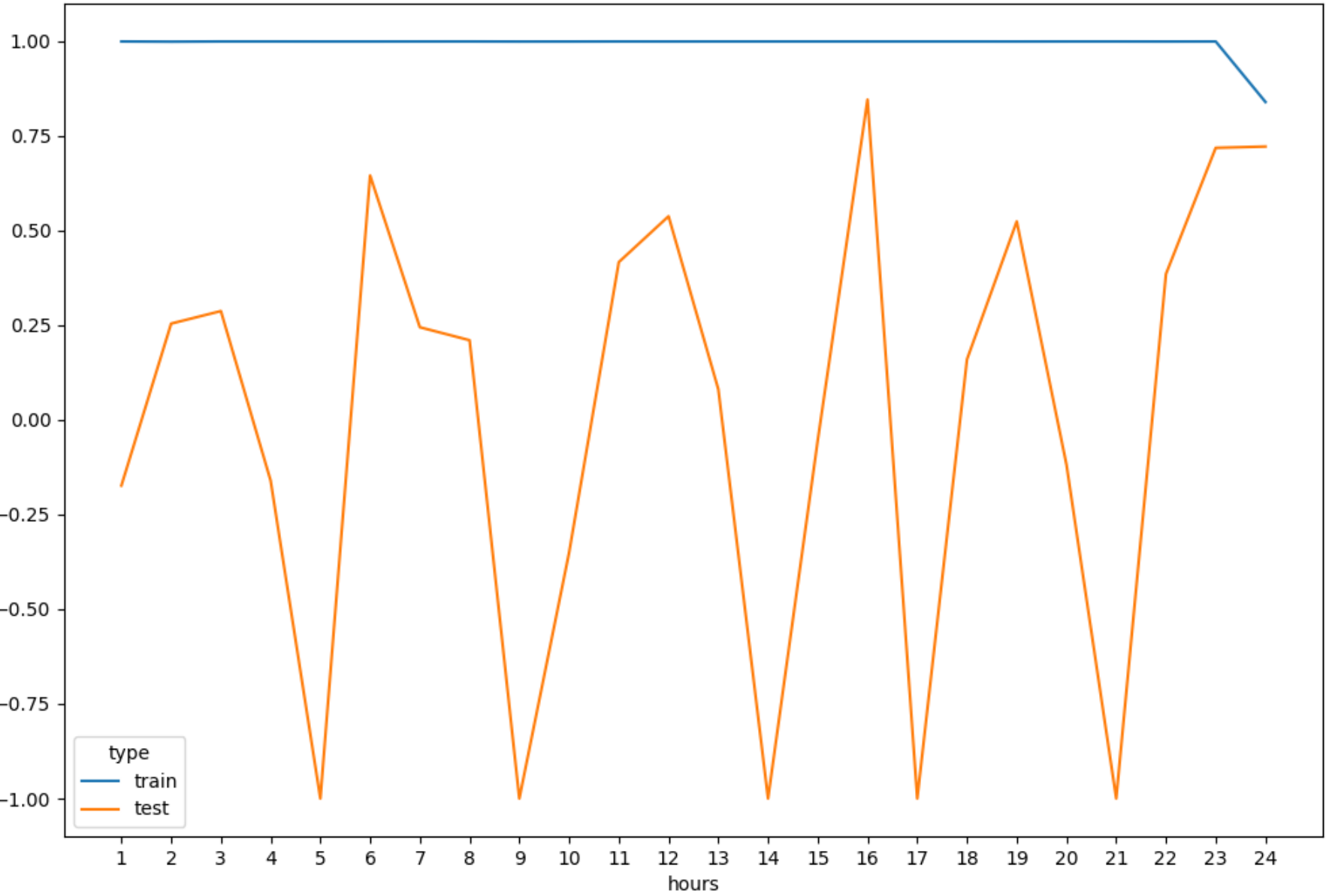
> Gradient Boosting

GB работает путем последовательного добавления новых **моделей (деревьев)** к существующему ансамблю. Каждое новое дерево обучается на данных, учитывая ошибки, сделанные предыдущими деревьями.

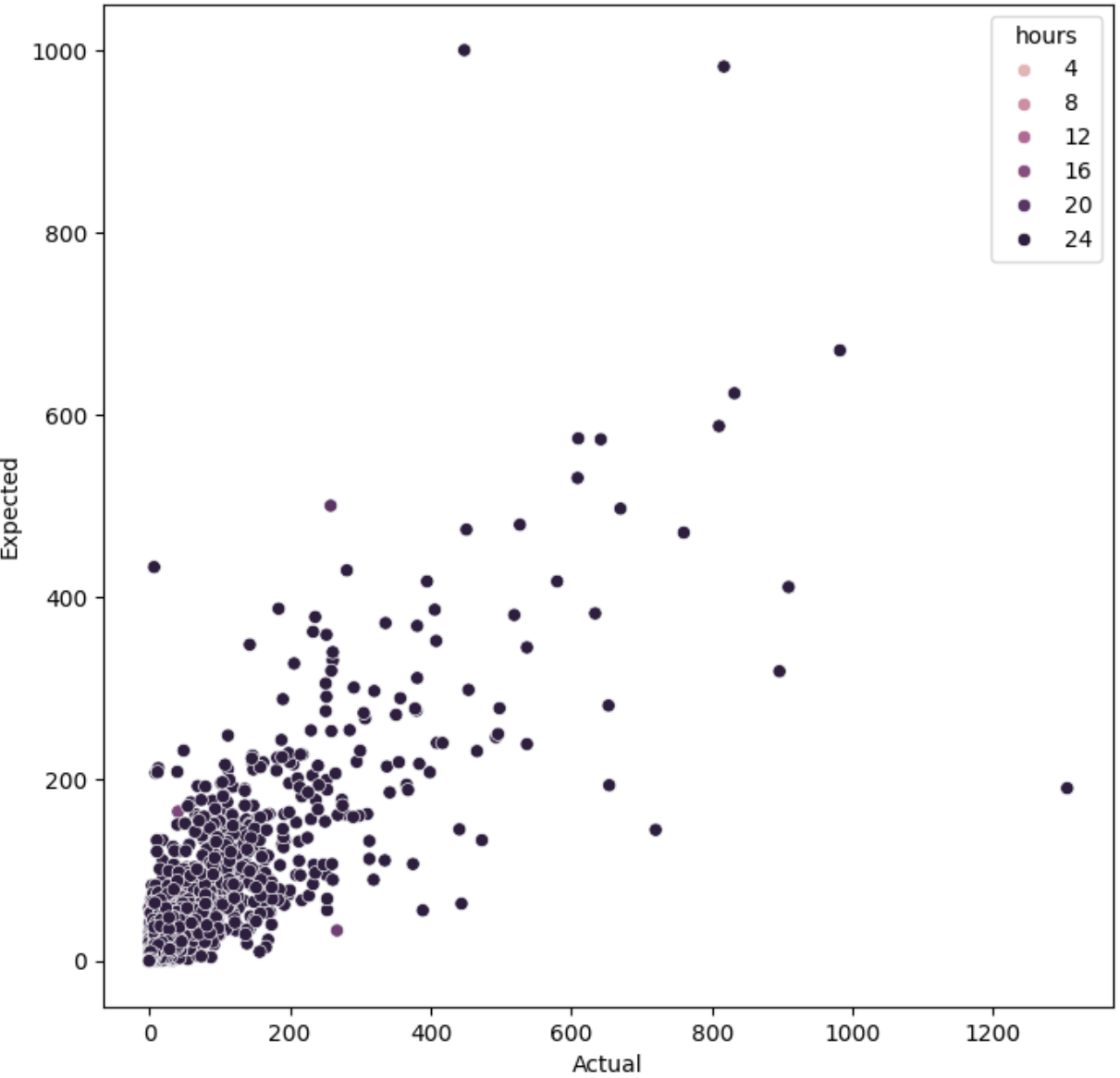
- **Плюсом метода** является его способность автоматически учитывать важность признаков.
- **Но к сожалению,** она вычислительно затратна и если недостаточно ограничить **глубину деревьев или количество итераций**, модель может переобучиться на обучающих данных, что приведет к плохой обобщающей способности модели.

Зависимость R2 от n_estimators

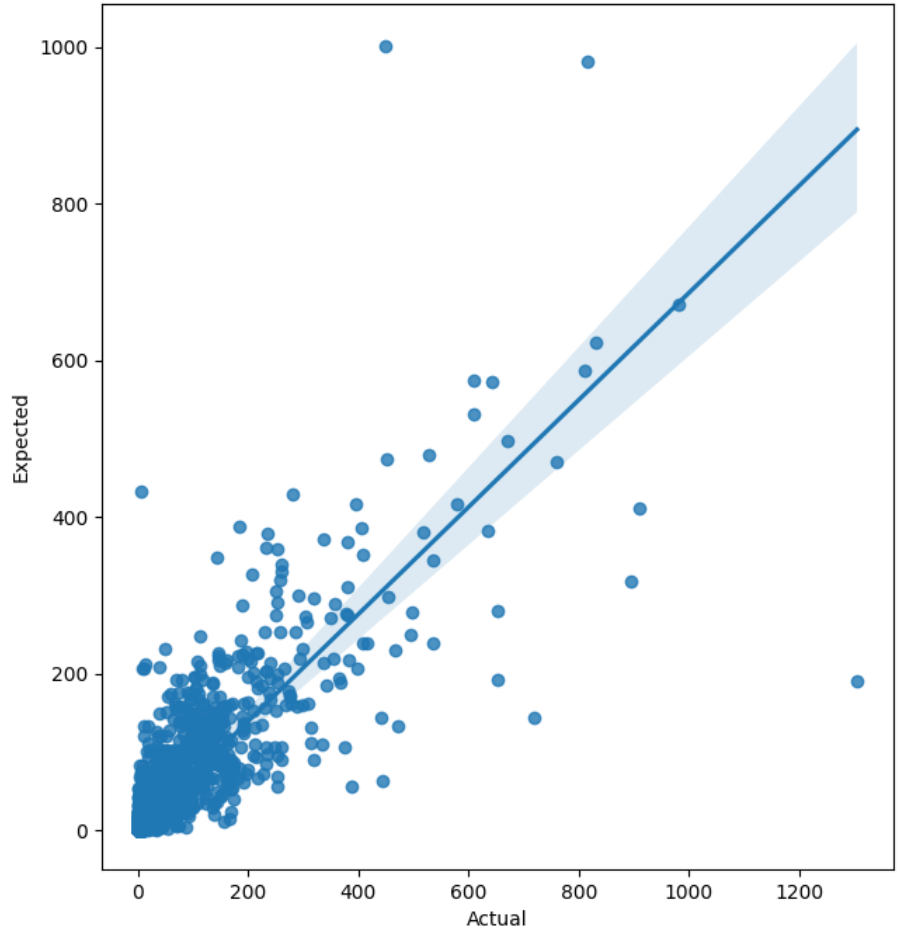


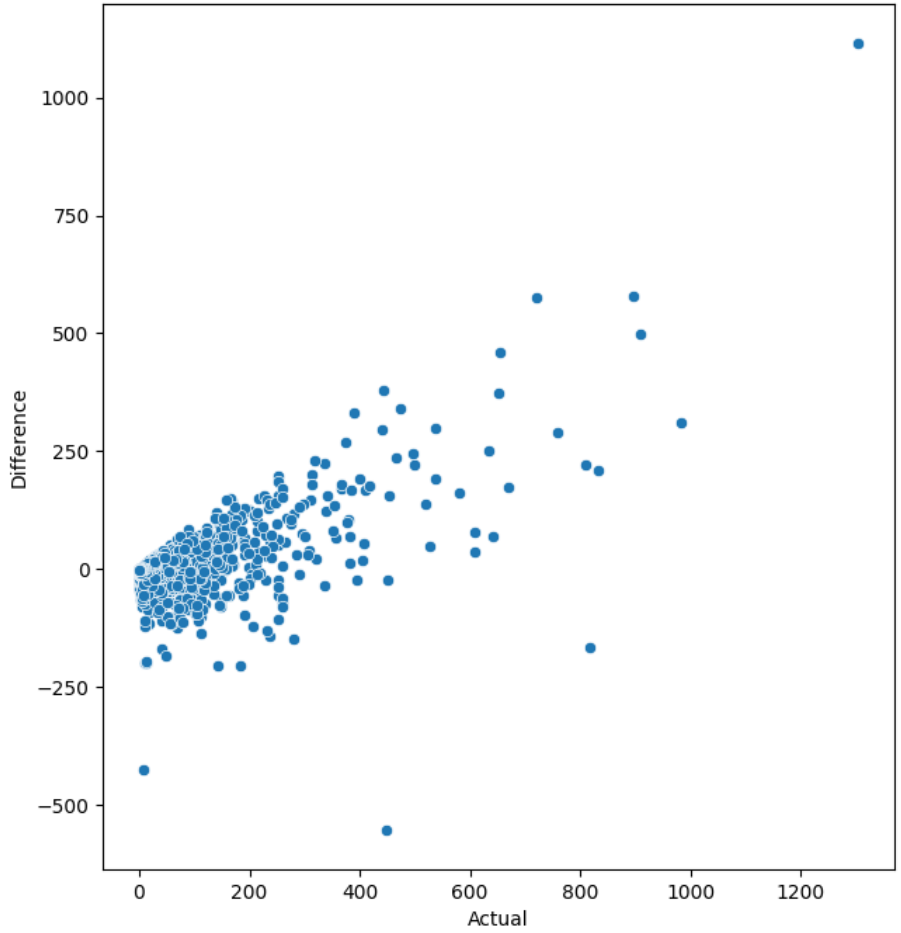


R2 score for Gradient Boosting model

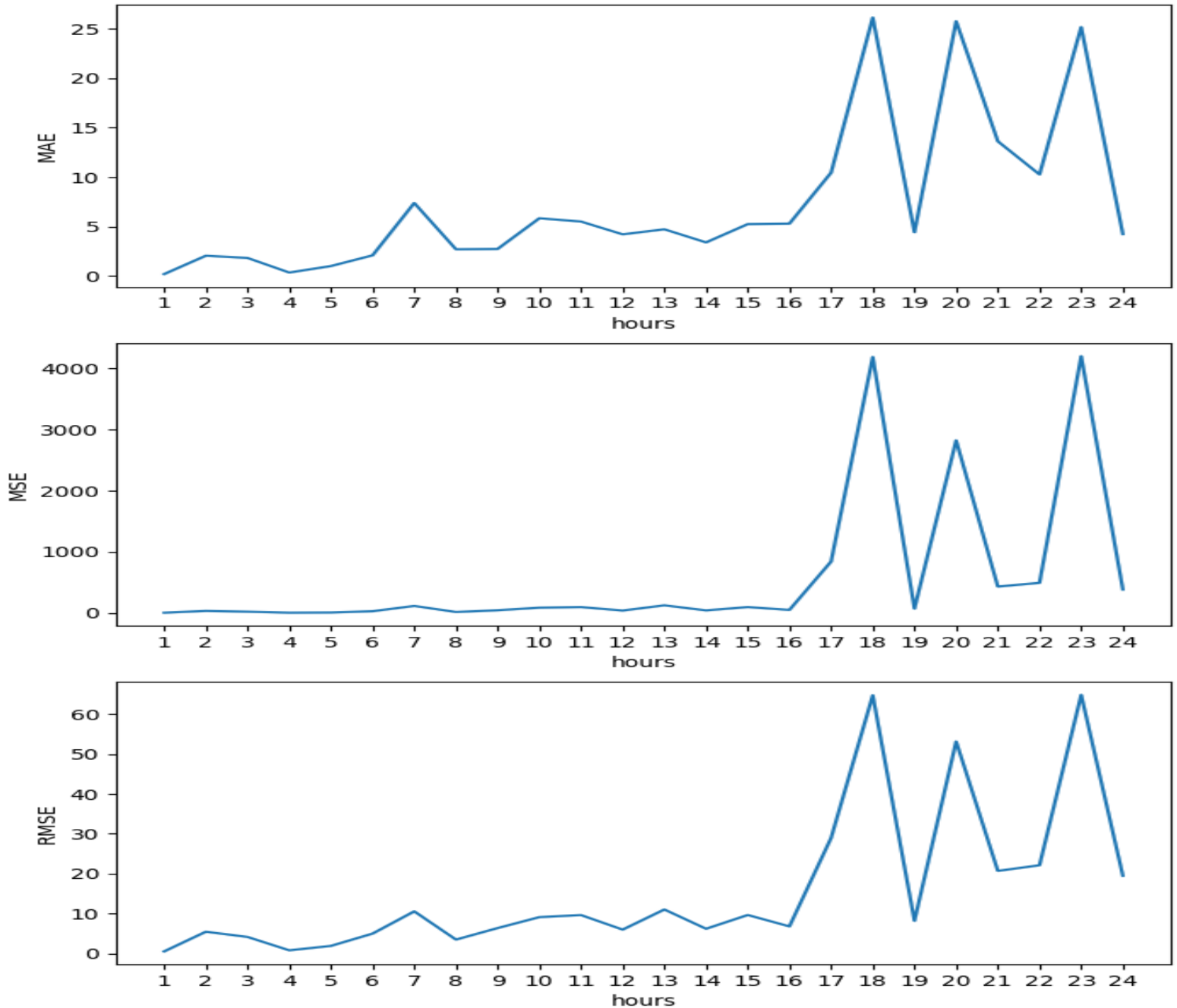


Error plot (overall) for Gradient Boosting model





Error plot (24h) for Gradient Boosting model

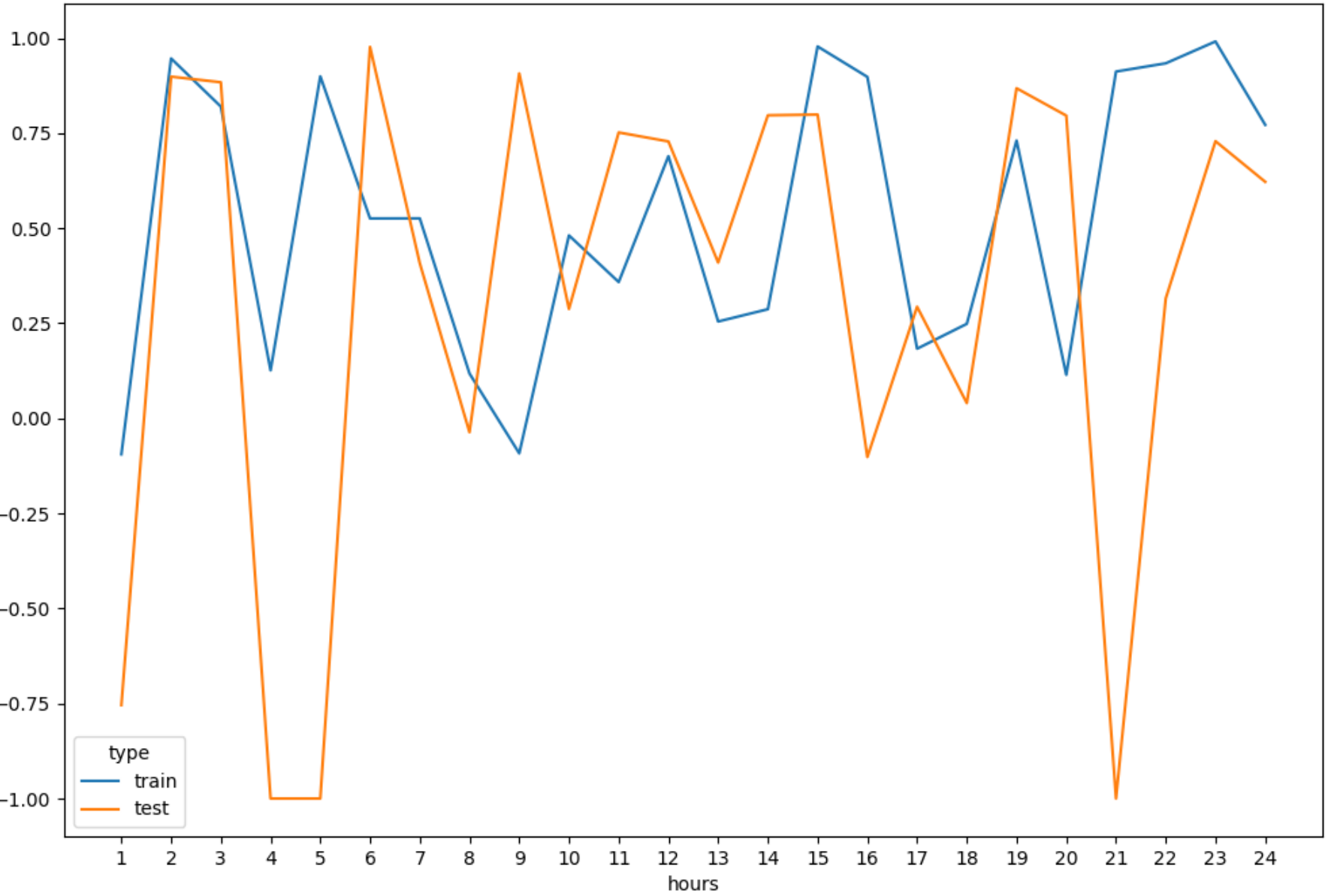

MAE
0
5
10
15
20
25
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
hours
MSE
0
1000
2000
3000
4000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
hours
RMSE
0
10
20
30
40
50
60
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
hours

Модель NN

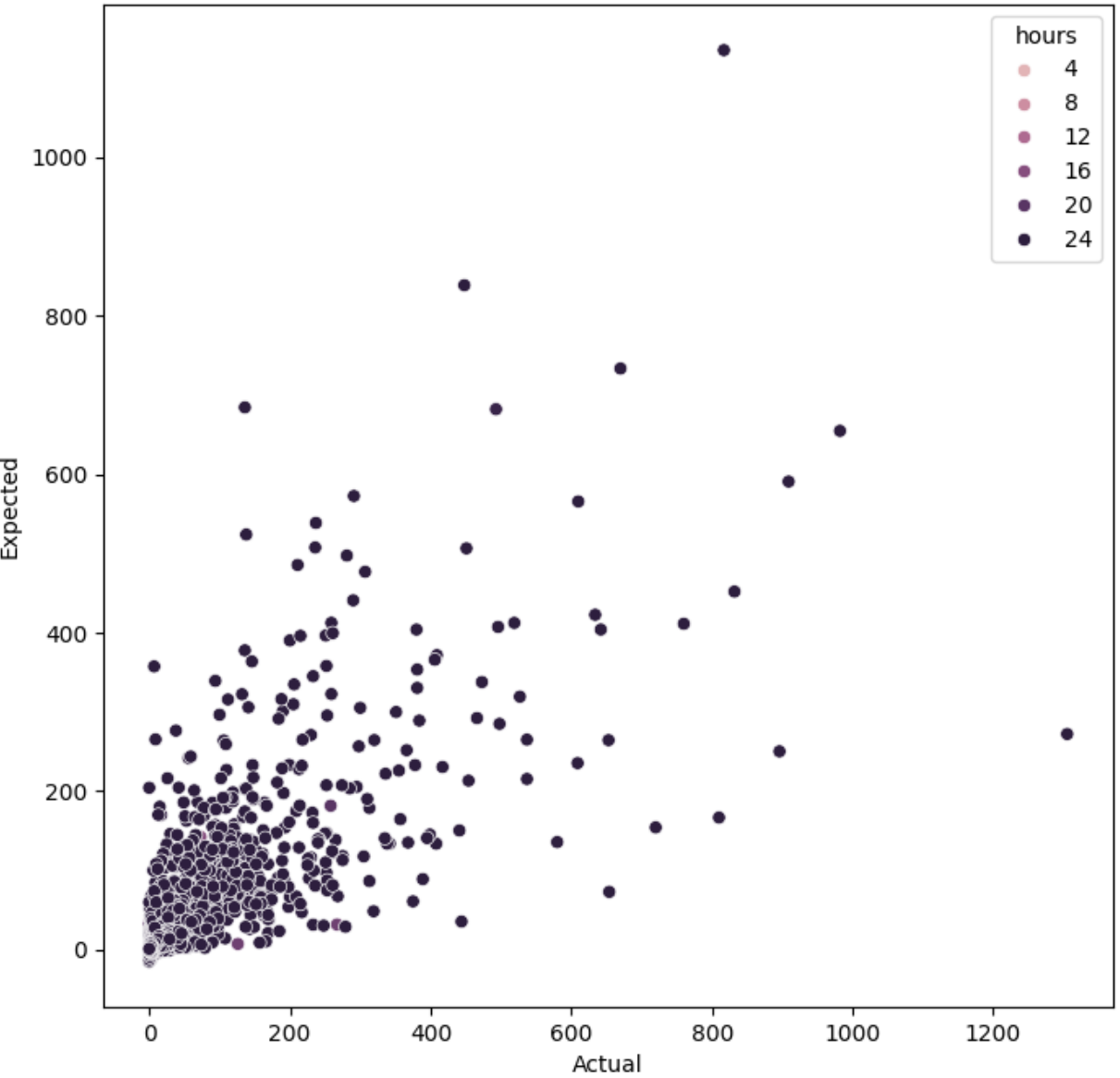
> Neural Network *(ура нейронка!!!)*

NN это тип искусственной **нейронной сети**, используемой для прогнозирования непрерывного выходного значения на основе одного или нескольких входных значений. **В отличие от классификационных моделей**, которые предсказывают категориальные метки, регрессионные модели нейросети **предсказывают количественные выходные данные**

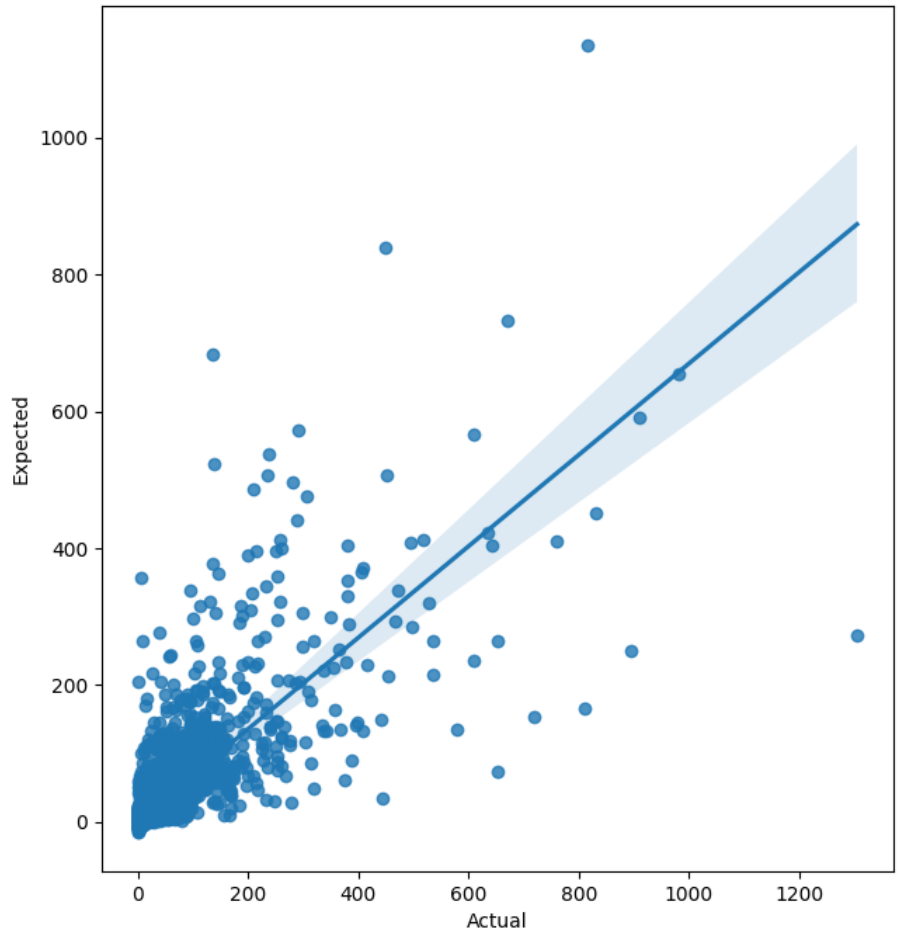
- **Плюсом метода** является его гибкость и адаптивность.
- **Но к сожалению,** нейросети часто требуют больших объемов данных для обучения, значительных вычислительных ресурсов и тщательной настройки гиперпараметров

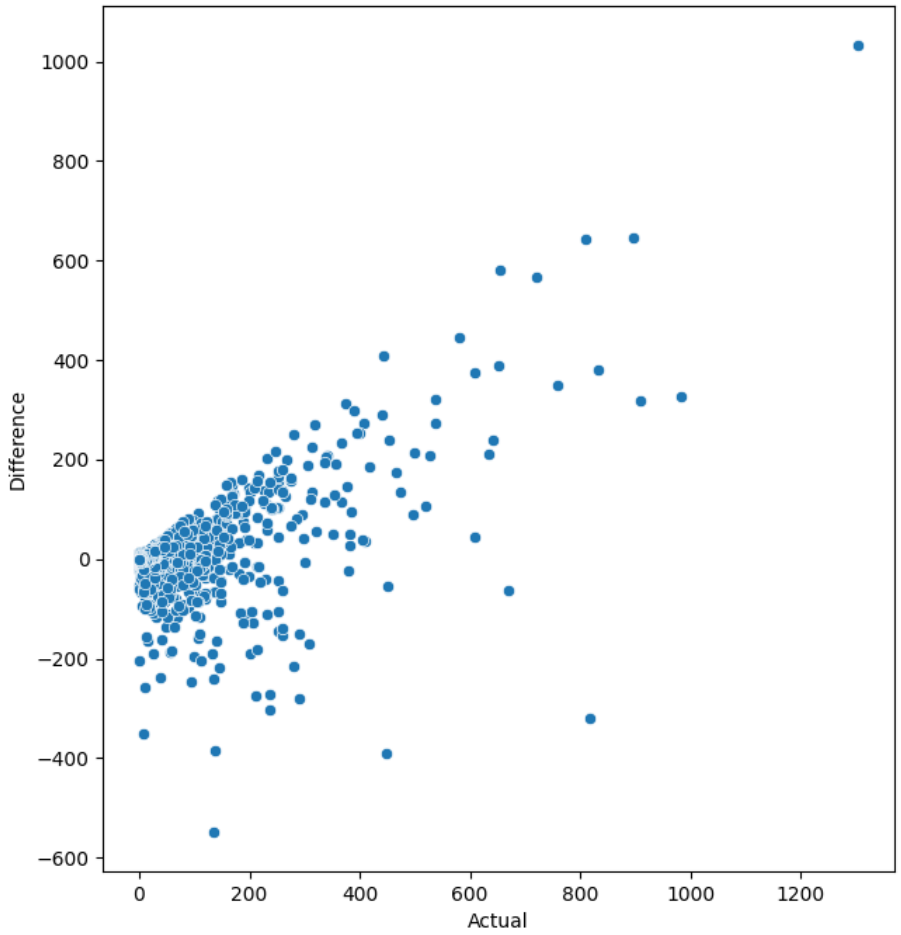


R2 score for Neural Network model

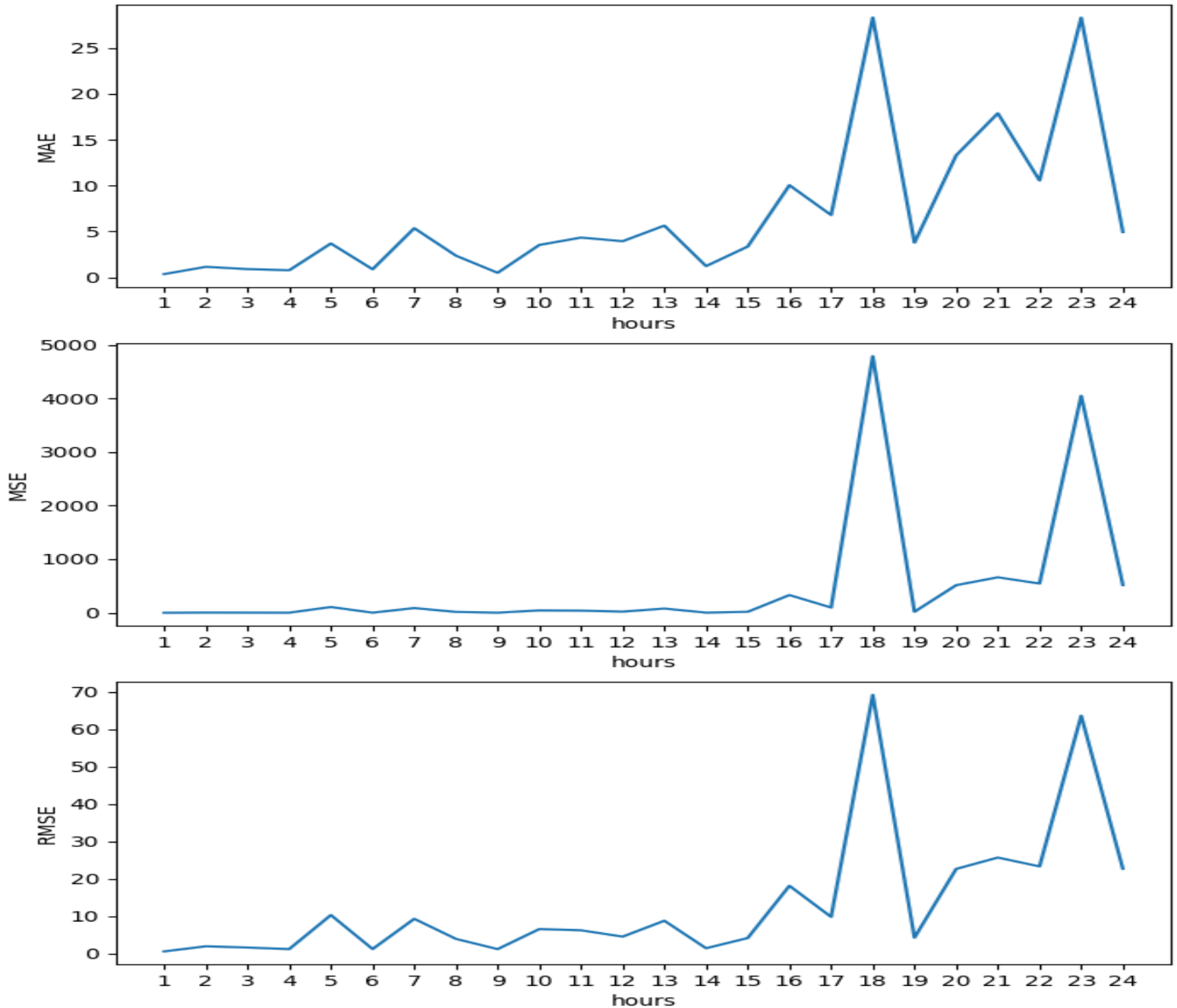


Error plot (overall) for Neural Network model





Error plot (24h) for Neural Network model

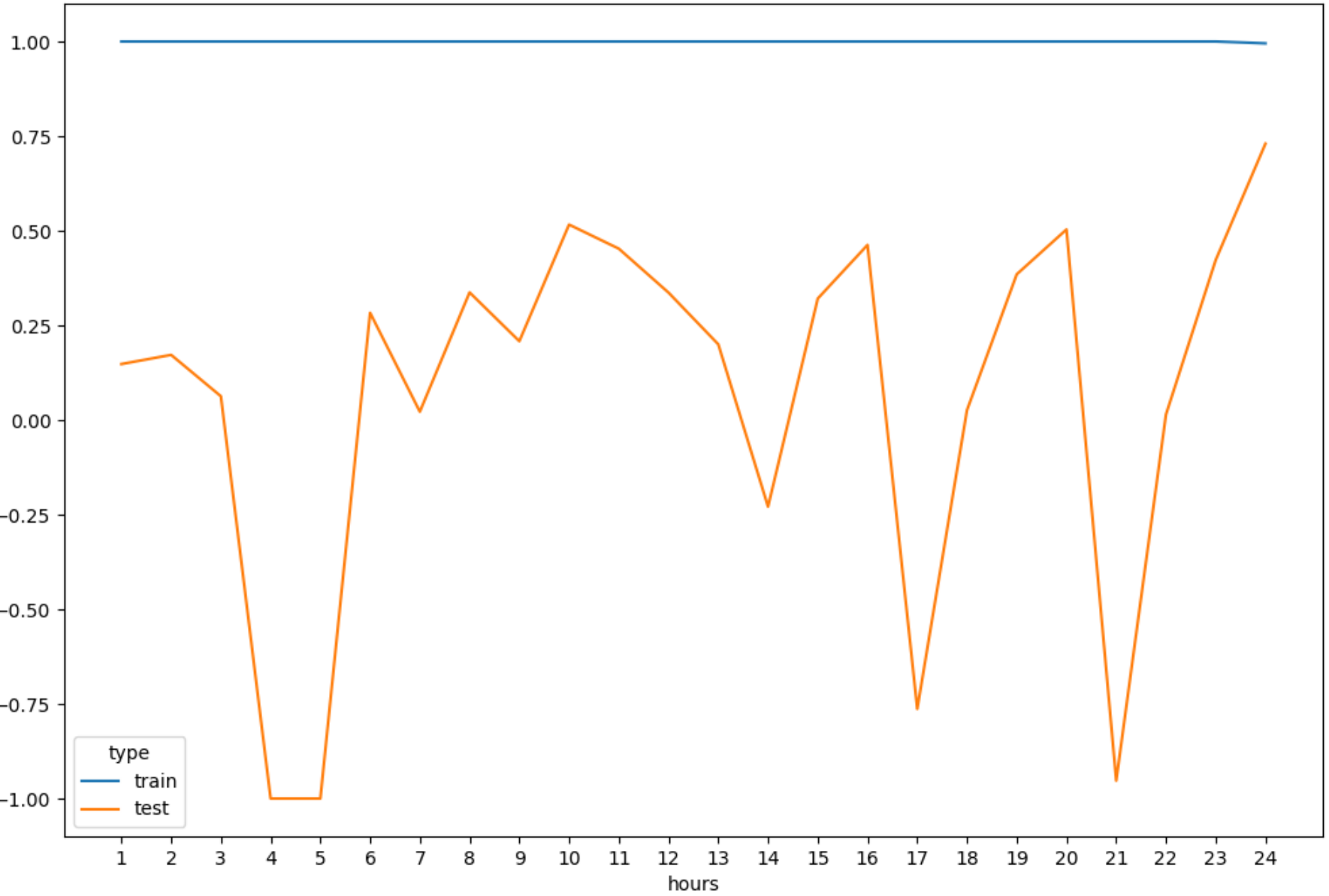

MAE
0
5
10
15
20
25
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
hours
MSE
0
1000
2000
3000
4000
5000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
hours
RMSE
0
10
20
30
40
50
60
70
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
hours

Модель CB

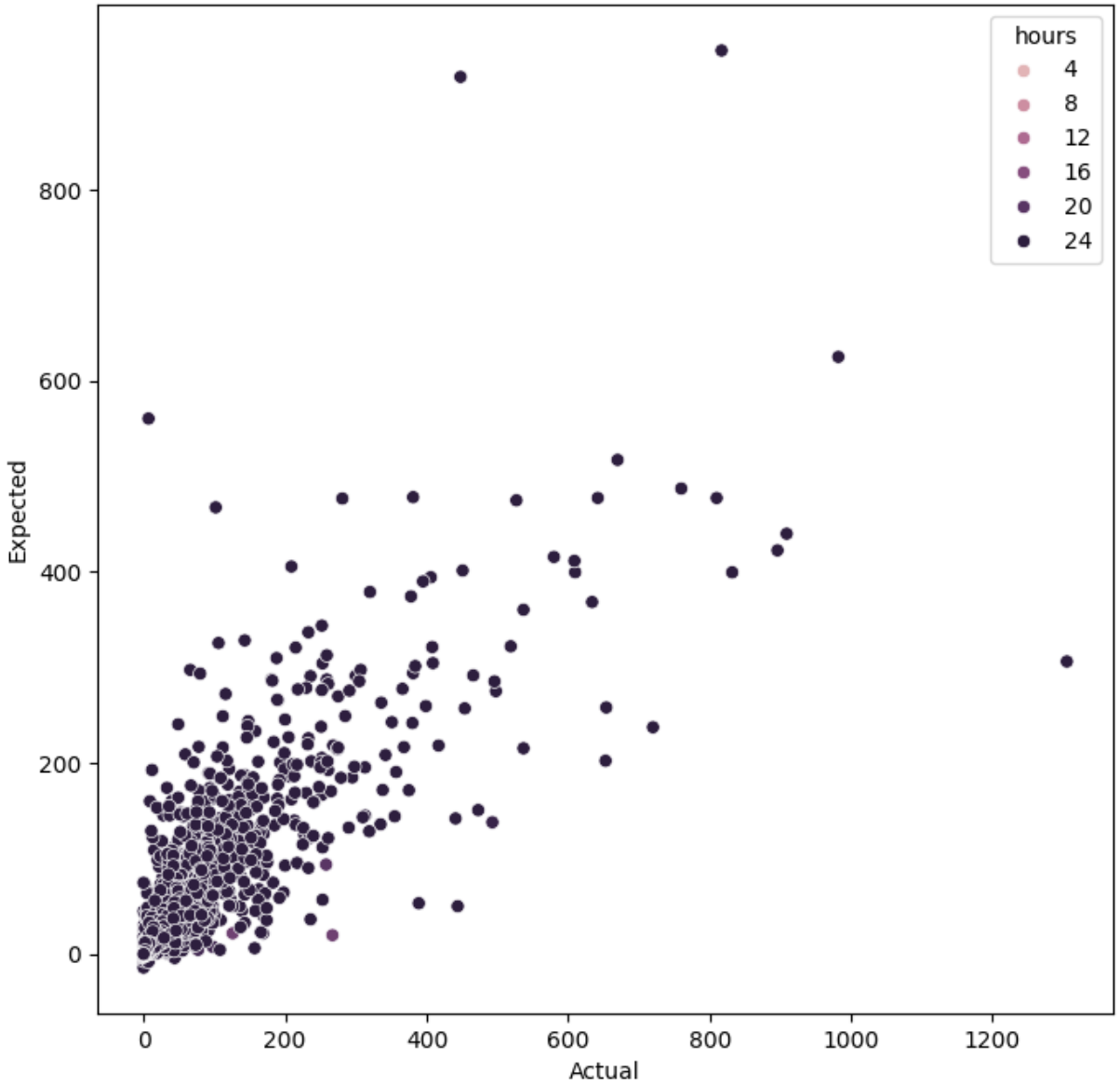
> Cat Boost Regressor 😺😺😺

CB это регрессионная модель, входящая в состав библиотеки **CatBoost**, разработанной **Yandex**. CatBoost (Categorical Boosting) представляет собой алгоритм градиентного бустинга, оптимизированный для работы с **категориальными данными**

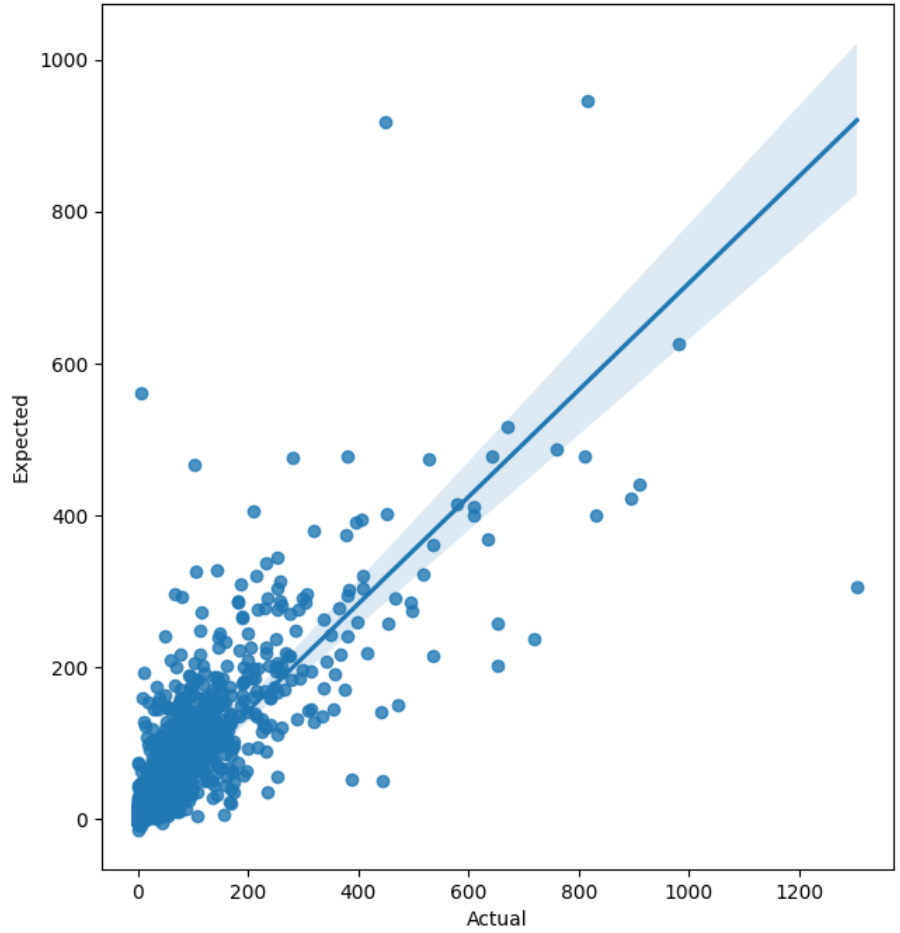
- **Плюс:** он автоматически преобразует категориальные переменные и интегрирует их в модель, что делает его очень удобным и мощным для наборов данных с многочисленными категориальными признаками.
- **Но к сожалению,** CatBoost может требовать больше времени для обучения по сравнению с некоторыми другими алгоритмами машинного обучения, особенно на больших наборах данных

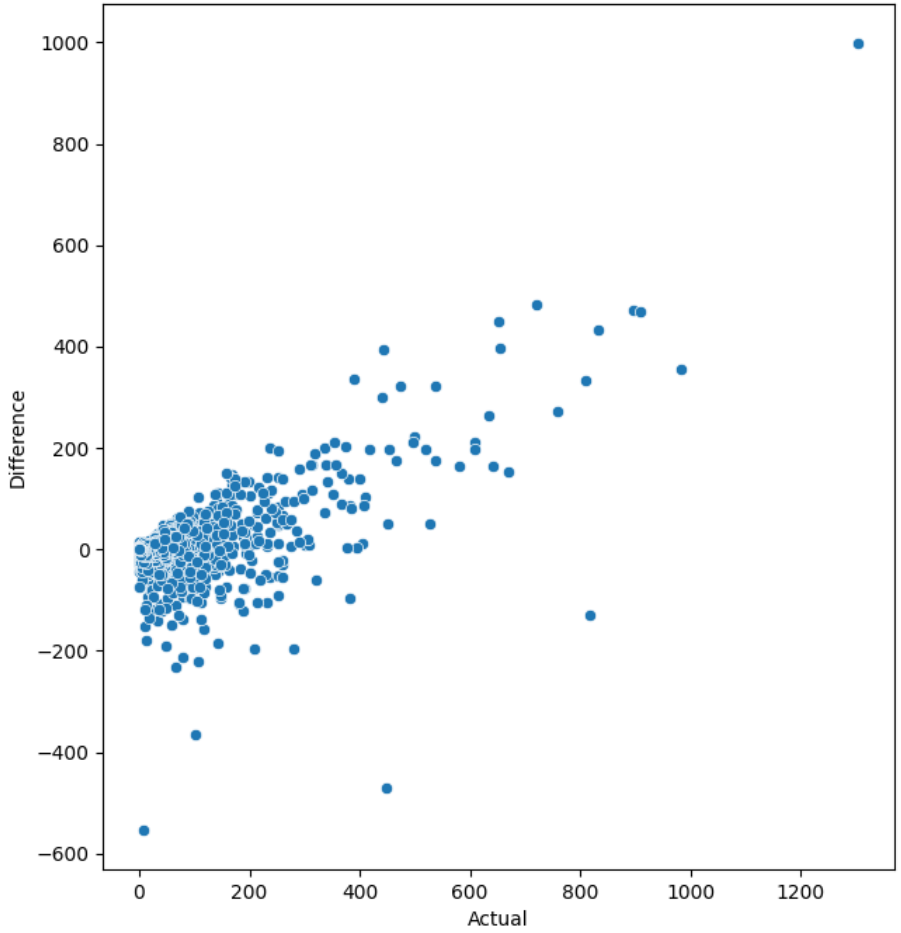


R2 score for CatBoost model

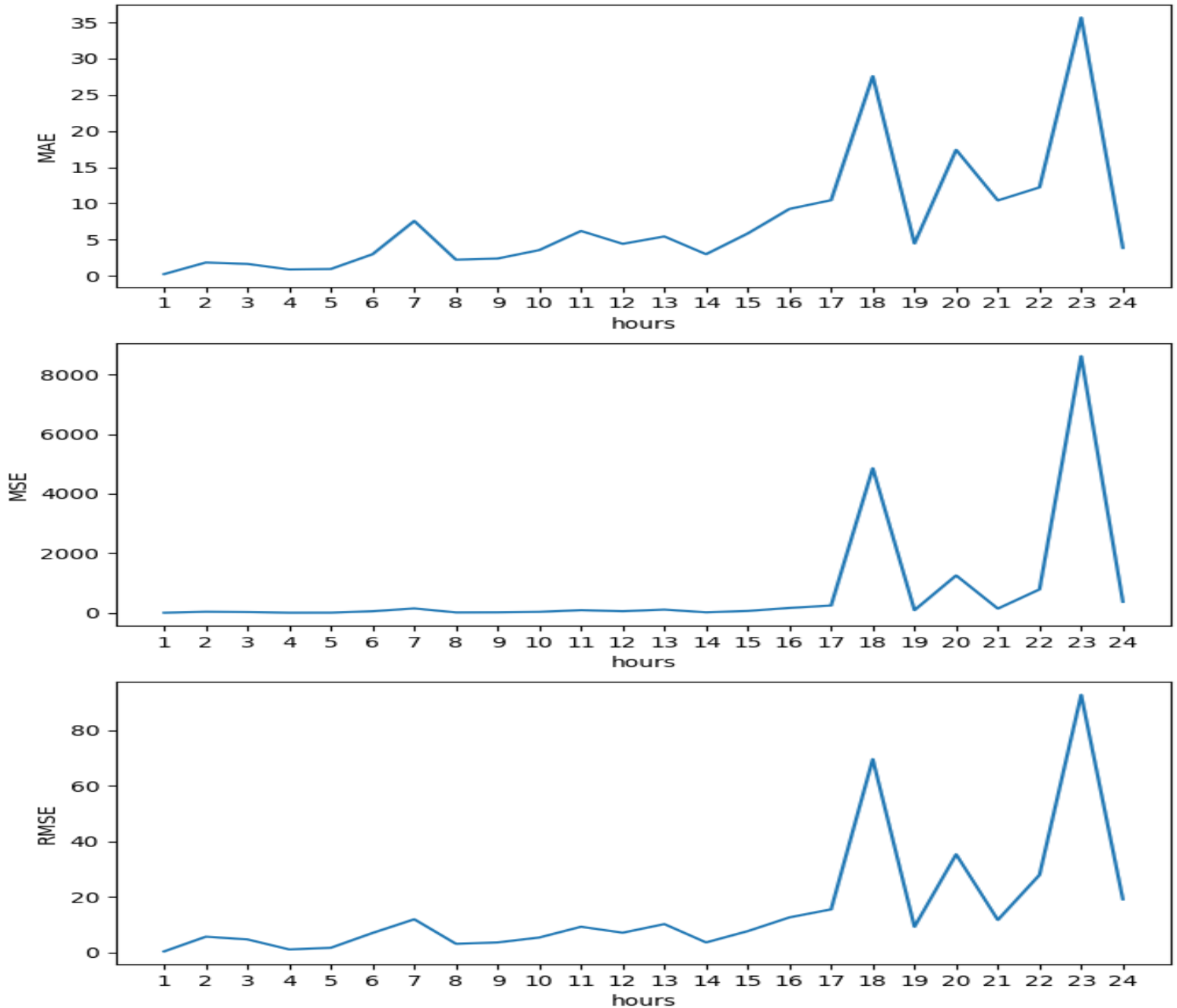


Error plot (overall) for CatBoost model





Error plot (24h) for CatBoost model


MAE
0
5
10
15
20
25
30
35
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
hours
MSE
0
2000
4000
6000
8000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
hours
RMSE
0
20
40
60
80
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
hours

Итоги 🥳🥳🥳

По итогам редакции, лучшими моделями оказались:

1. **Cat Boost Regressor**
2. **Gradient Boosting**
3. **Neural Network**

Почему так?

- **Cat Boost Library** это специализированный инструмент, разработанный в **Yandex** для решения комплексных задач. Алгоритм нахождения регрессии способен на большее, чем предоставляет стандартная библиотека **sklearn**.
- **Gradient Boosting** это современный эффективный способ для моделирования регрессии, как и **Neural Network**.

Работали,

Data Balbesing 🔥

> Что видит консоль линукса, когда Миша ждет окончания расчетов моделей ->