

Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Физико-Механический институт

Работа допущена к защите
Директор высшей школы
_____ А.М. Кривцов
«_____» _____ 2022 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

РАБОТА МАГИСТРА

ТЕМА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

по направлению подготовки 01.04.03 Механика и математическое моделирование
Направленность (профиль) 01.04.03_04 Математическое моделирование
процессов нефтегазодобычи

Выполнил

студент гр. 5040103/10401

А.А. Муравцев

Руководитель

должность,

степень, звание

И.О. Фамилия

Консультант

должность, степень

И.О. Фамилия

Консультант

по нормоконтролю

И.О. Фамилия

Санкт-Петербург

2022

СОДЕРЖАНИЕ

Введение	3
Глава 1. Исследование данных до обучения моделей	5
1.1. Тепловая карта корреляций	5
1.2. Распределения выходных параметров	7
1.3. Диаграмма рассеяния	8
1.4. Выводы.....	10
Глава 2. Обучение моделей	11
2.1. Линейная регрессия.....	11
2.2. Регрессия ближайших соседей.....	13
2.3. Регрессия на основе метода опорных векторов.....	15
2.4. Чрезвычайно рандомизированные деревья	17
2.5. Расширенные с помощью базисных функций линейные модели	19
2.6. Выводы.....	21
Заключение	22
Список использованных источников	23
Приложение 1. Код программы для построения метамоделей	24

ВВЕДЕНИЕ

В процессе своей работы исследователи часто используют упрощённые аппроксимации для описания сложных физических процессов. Разработано большое количество способов построения таких аппроксимаций. Большинство способов основаны на данных, полученных в результате конечного числа экспериментов, проведённых над рассматриваемой моделью.

В случае компьютерных моделей аппроксимации могут быть использованы для увеличения скорости расчёта результата новых экспериментов (с ранее не рассматриваемой комбинацией входных параметров).

В работе [2] подробно описано построение таких аппроксимаций (называемых метамоделями) с помощью искусственных нейронных сетей и градиентного бустинга на основе деревьев регрессии в случае фиксированной обучающей выборки. В рамках работы [2] было рассмотрено 2 датасета, которые были сгенерированы при моделировании притока жидкости к вертикальной скважине в гидродинамическом симуляторе ECLIPSE Blackoil. Первый датасет был получен при рассмотрении сценария разработки с поддержанием пластового давления (ППД), начиная с первого месяца. А при генерации данных второго датасета сначала (до существенного падения значений дебитов) рассматривался естественный режим и только потом включалась система ППД. Для обоих датасетов варьировались 4 входных параметра геологии пласта: пористость матрицы, проницаемость матрицы, пористость трещин и проницаемость трещин; на выходе отслеживались дебиты нефти в течение пяти лет.

Целью данной работы является построение других моделей машинного обучения на основе тех же входных данных и сравнение полученных результатов с результатами работы [2]. Все модели будут строиться на основе фиксированной обучающей выборки. Построение с помощью дополнляемой выборки подробно описано в брошюрах [5; 4].

В качестве моделей машинного обучения в данной работе выбраны широко известные модели регрессии: линейной (linear regression), ближайших соседей (the nearest neighbours regression), на основе метода опорных векторов (support vector machine regression) и на основе чрезвычайно рандомизированных деревьев (extra trees regressor). Также рассмотрены расширенные с помощью базисных функций линейные модели.

Для достижения цели были поставлены следующие задачи:

- изучить способы построения известных моделей машинного обучения с помощью языка программирования Python [1; 3];
- оформить визуализацию диаграмм ошибок прогноза и сравнить их с аналогичными диаграммами из работы [2]

ГЛАВА 1. ИССЛЕДОВАНИЕ ДАННЫХ ДО ОБУЧЕНИЯ МОДЕЛЕЙ

В первой главе будет исследован характер данных в обоих рассматриваемых датасетах, для которых в дальнейшем будут строиться аппроксимации.

1.1. Тепловая карта корреляций

Построим тепловую карту корреляций (рис. 1.1) между входными и выходными параметрами первого датасета (сценарий 1).

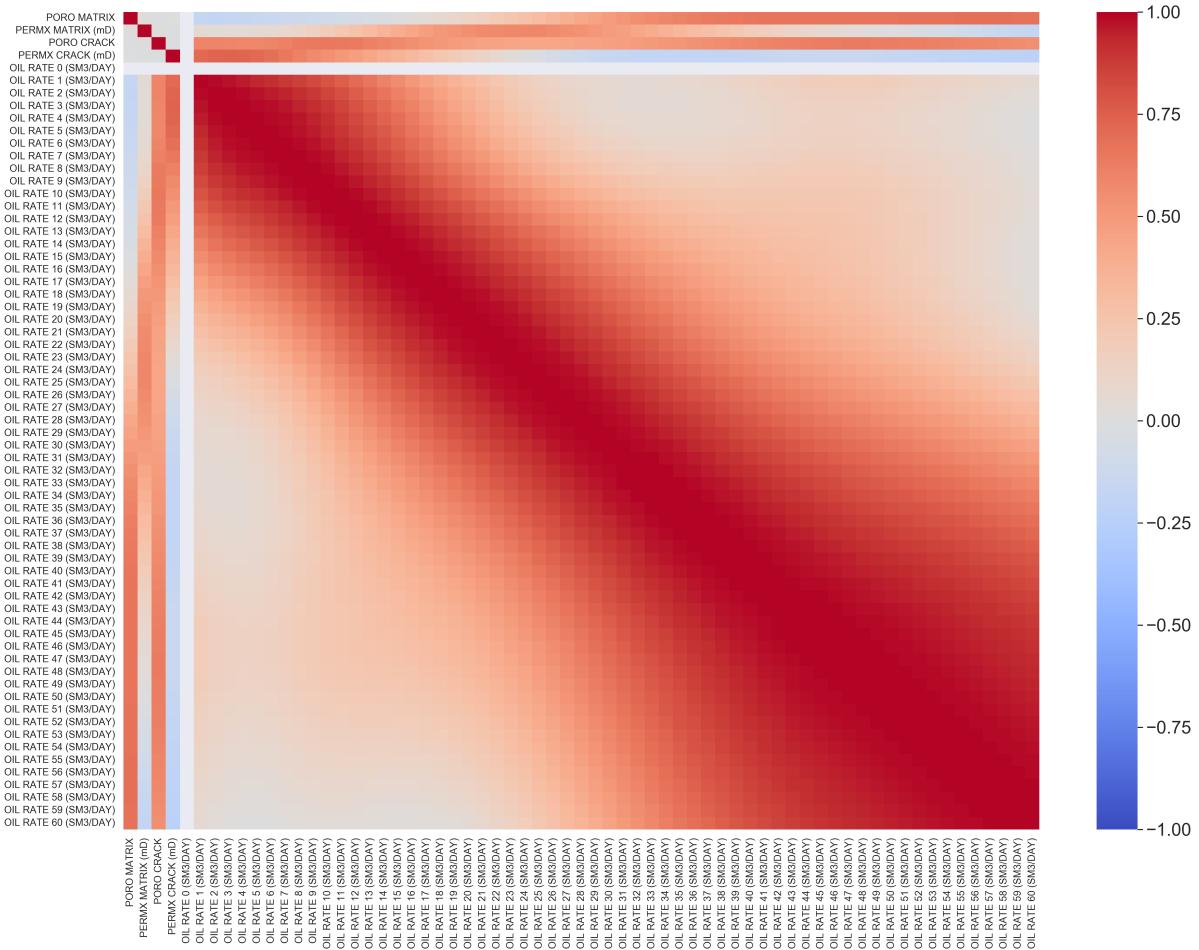


Рис.1.1. Термальная карта корреляций (сценарий 1)

Из построенной карты корреляций видим, что есть сильная линейная корреляция между соседними значениями дебитов, что может быть теоретически использовано для воспроизведения полной последовательности значений дебитов в том случае, когда известны только несколько значений из этой последовательности.

Более значимое для построения аппроксимаций наблюдение состоит в том, что в первые месяцы на значения дебитов нефти наиболее сильно влияют пористость и проницаемость трещин. После первого года влияние проницаемости трещин снижается и возрастает влияние проницаемости матрицы. После второго года снижается влияние проницаемости матрицы и возрастает влияние пористости матрицы и пористости трещин.

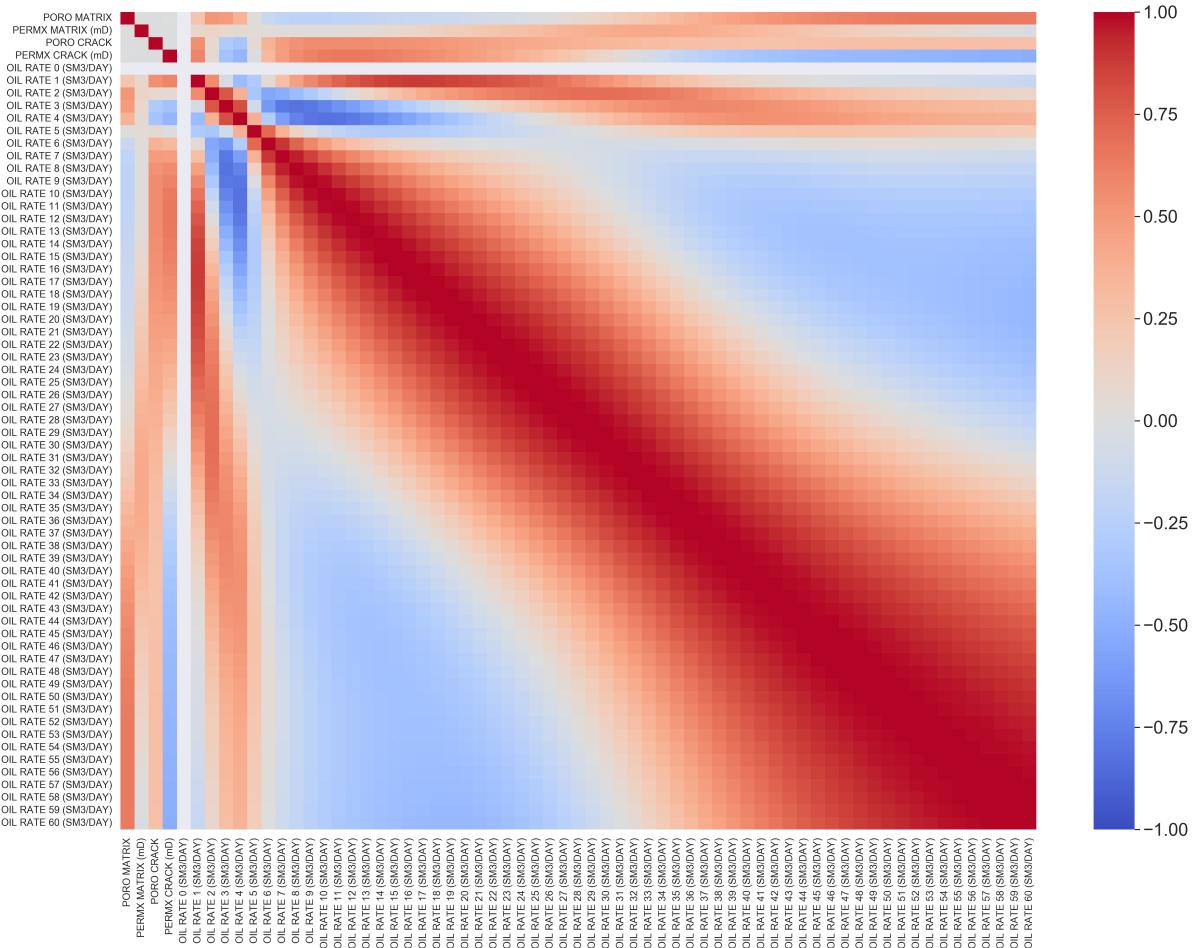


Рис.1.2. Термальная карта корреляций (сценарий 2)

Термальная карта корреляций для второго сценария (рис. 1.2) в первый год сильно отличается от первого сценария, так как во втором сценарии сначала рассматривается другой режим работы (естественный режим) и только затем включается нагнетание. Также во втором сценарии появляется заметная отрицательная корреляция между выходными дебитами, что тоже связано с запаздыванием включения системы ППД по сравнению с первым сценарием.

1.2. Распределения выходных параметров

Построим распределения выходных параметров для того, чтобы проанализировать, каким образом изменялось распределение возможных значений выходных параметров с течением времени. Для первого сценария разработки (рис. 1.3) видим, что большинство значений дебитов в первый месяц сосредоточены в области более низких значений, затем при переходе к концу первого года большинство возможных значений дебитов плавно переходит в область высоких значений. К концу пятого года разработки большинство значений сосредоточены в области средних значений.

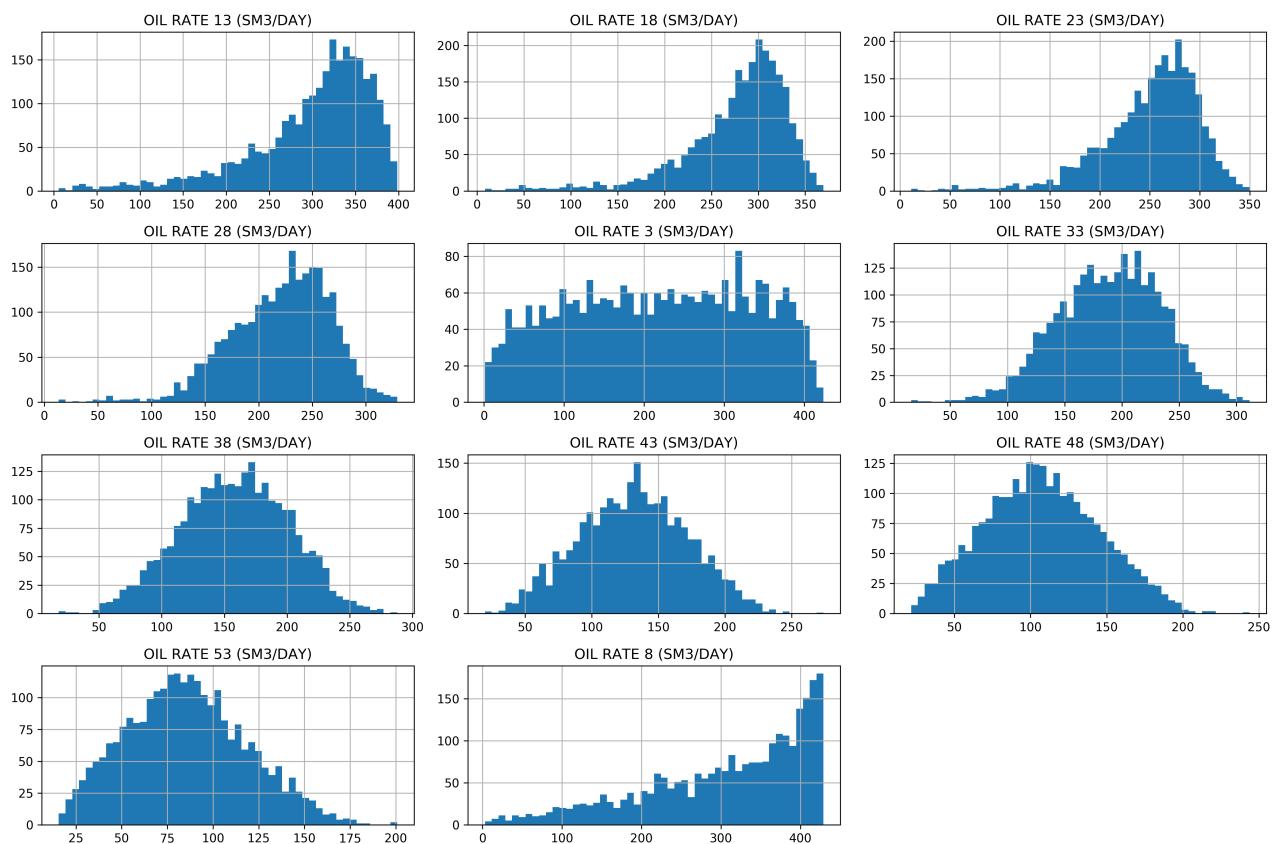


Рис.1.3. Распределения нескольких выходных параметров (сценарий 1)

Во втором сценарии (рис. 1.4) в первый год разработки выражено разделение значений дебитов на кластеры, так как кейсы делятся на группы по критерию начала включения системы ППД. Далее характер распределений аналогичен первому сценарию.

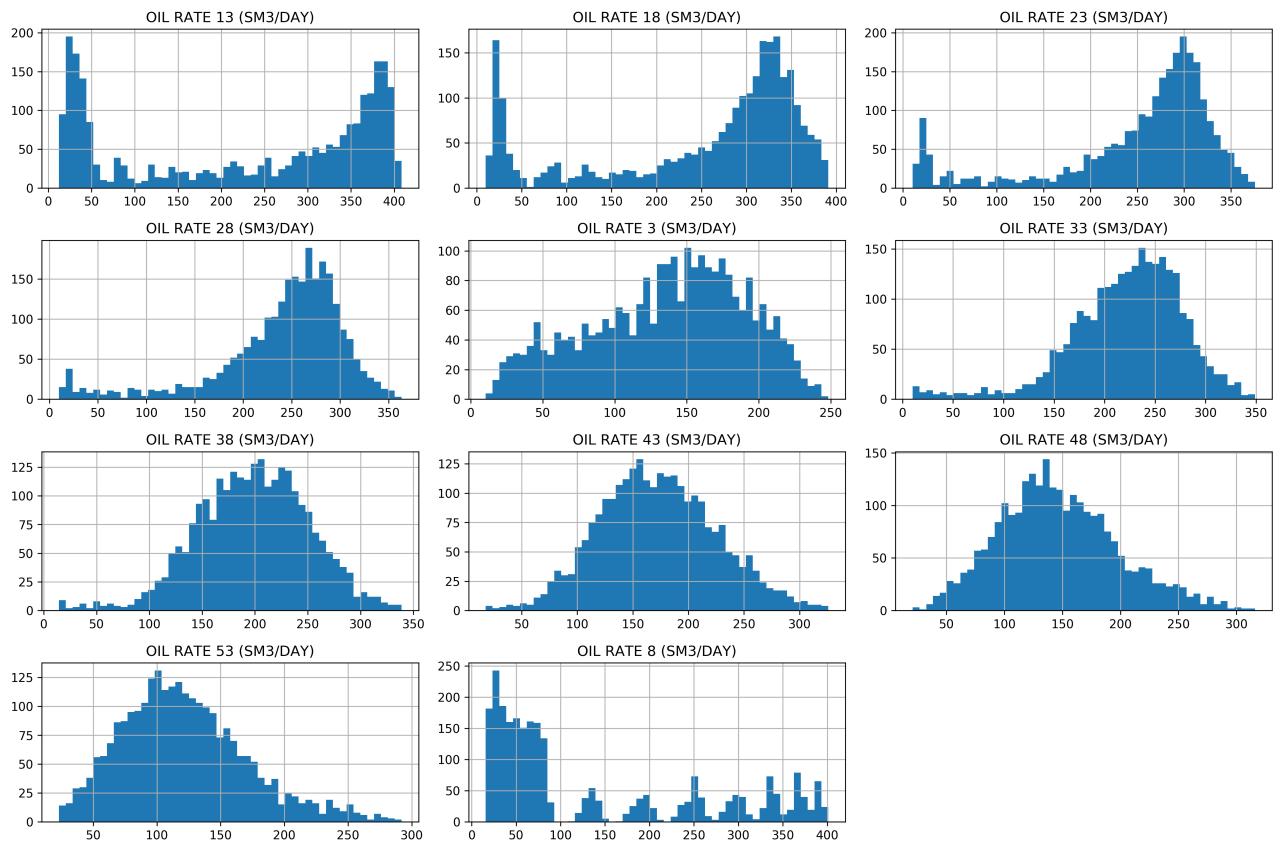


Рис.1.4. Распределения нескольких выходных параметров (сценарий 2)

1.3. Диаграмма рассеяния

Диаграммы рассеяния позволяют наглядно проверить утверждения, полученные при анализе тепловых карт. Например, для первого сценария из тепловой карты видели, что наиболее существенный вклад в значения дебитов в первый месяц оказывают пористость и проницаемость трещин. Построив диаграмму рассеяния (рис. 1.5) для значений дебитов нефти в первый месяц и соответствующих значений проницаемости трещин (по оси абсцисс), пористости трещин (ранжирование значений с помощью цвета), видим, что замеченные ранее из тепловой карты линейные связи явно выделяются: на диаграмме линейный тренд и плавный переход по цветовой гамме.

Другими словами, видим, что возможно предсказать примерное значение дебита нефти в первый месяц из проницаемости и пористости трещин, а пористость и проницаемость матрицы оказывают незначительное влияние на это значение.

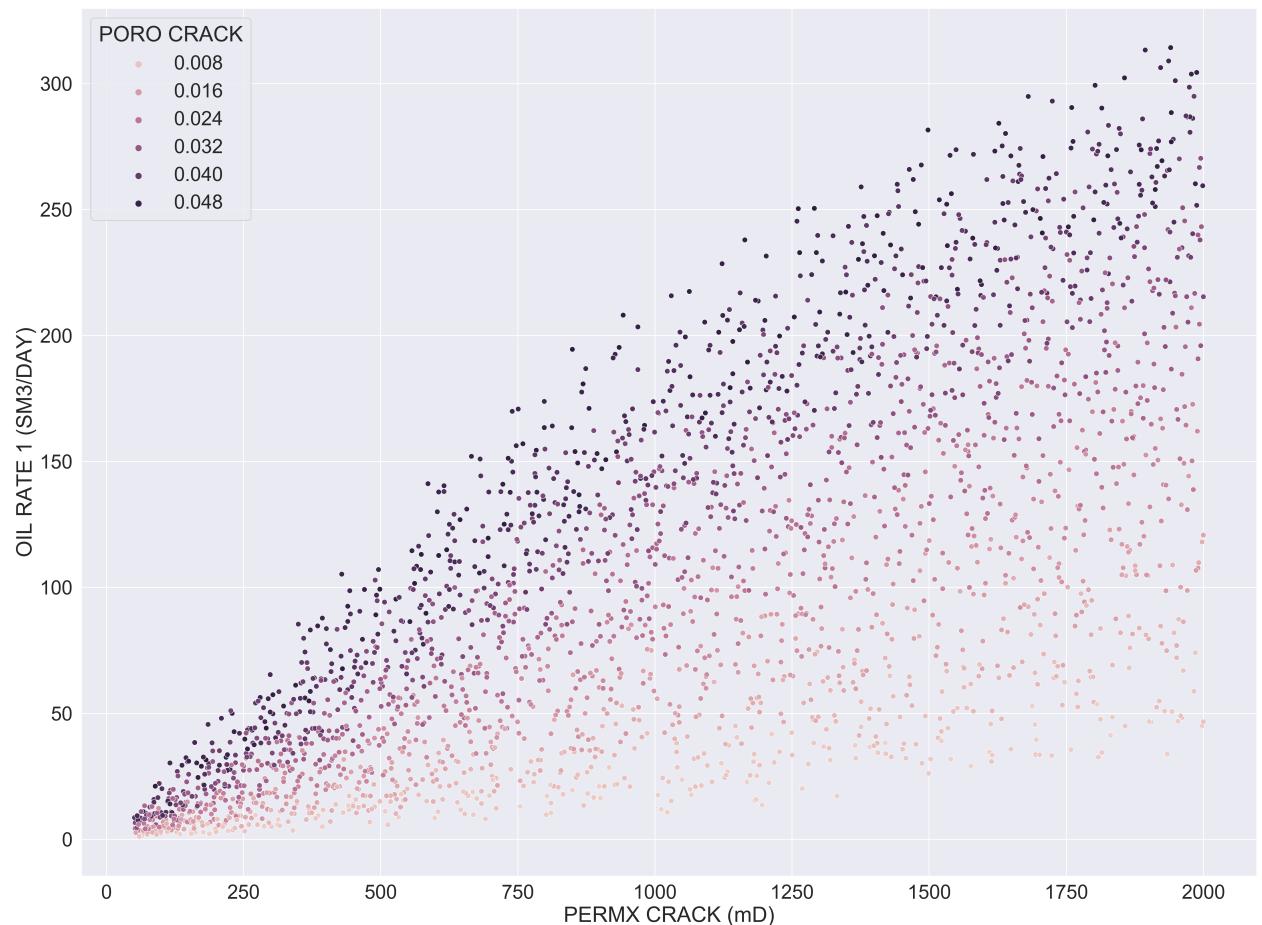


Рис.1.5. Диаграмма рассеяния, первый месяц (сценарий 1)

Похожая диаграмма рассеяния имеет место быть для значений дебитов в последний месяц и соответствующих значений пористости матрицы и пористости трещин (рис. 1.6), так как на тепловой карте (см. рис. 1.1) есть значительная линейная корреляция между значениями дебитов в последние месяцы и соответствующими пористостями матрицы и трещин.

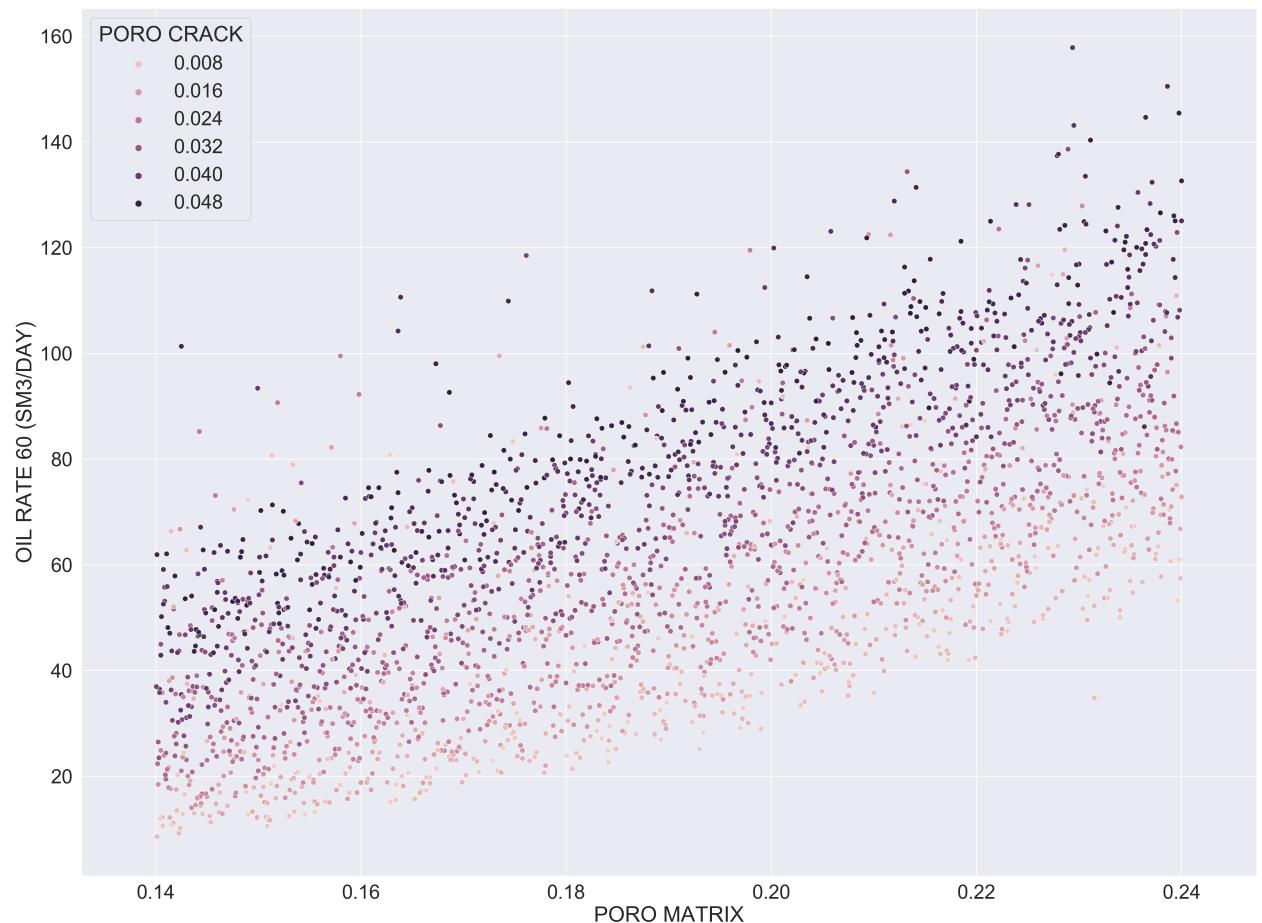


Рис.1.6. Диаграмма рассеяния, последний месяц (сценарий 1)

1.4. Выводы

В результате исследования характера данных в рассматриваемых датасетах (сценарий 1, сценарий 2) для каждого месяца выделены наиболее влияющие на значения дебитов входные параметры. Также проведены визуализации, которые будут полезны для понимания результатов прогноза моделей машинного обучения.

ГЛАВА 2. ОБУЧЕНИЕ МОДЕЛЕЙ

Во второй главе на основе рассматриваемых датасетов обучены модели машинного обучения и проведено сравнение диаграмм размаха ошибок построенных моделей с аналогичными диаграммами из работы [2]. Код для обучения всех обсуждаемых моделей представлен в приложении 1.

2.1. Линейная регрессия

Модель обучается при выполнении функции `linear_regression` класса `ReservoirMetamodel` (см. приложение 1).

Из диаграмм размаха ошибок (рис. 2.1) видим, что в случае линейной регрессии на некоторых кейсах ошибки достигают высоких значений. Это связано с ограниченной гибкостью линейной модели и небольшим количеством кейсов в области входных параметров, которой соответствуют низкие значения дебитов (менее $5 \text{ м}^3/\text{сут}$ за каждый месяц в течение всех рассматриваемых пяти лет).

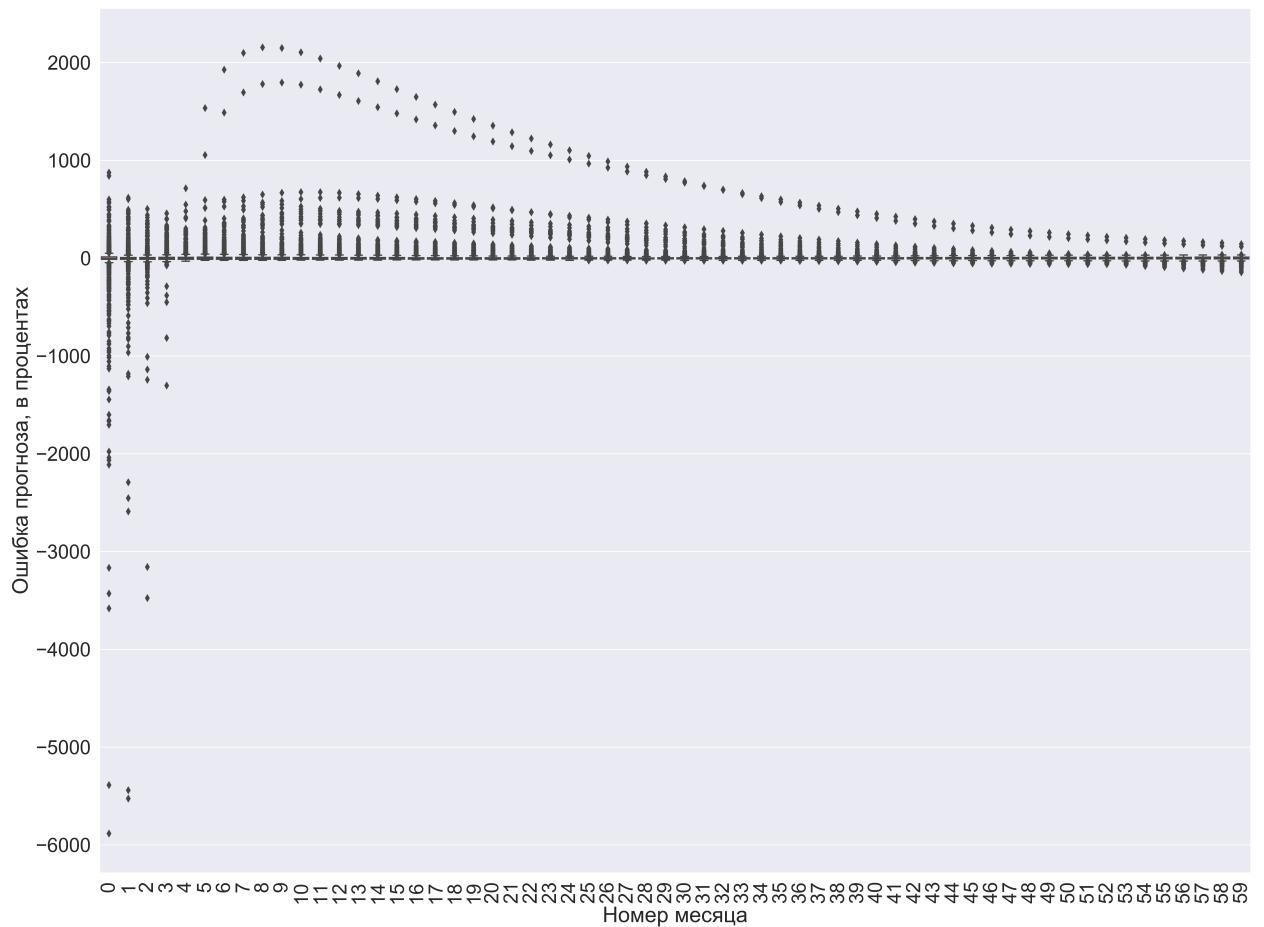


Рис.2.1. Диаграмма ошибок прогноза линейной регрессии (сценарий 1)

Прогнозы линейной регрессии в случае второго сценария разработки (рис. 2.2) значительно хуже первого сценария (что видно из гораздо более длинных усов на диаграммах размаха ошибок в 5-25 месяцы). Во втором сценарии от аппроксимации требуется ещё большая гибкость (так как изменяется режим разработки), что не может быть обеспечено линейной регрессией.

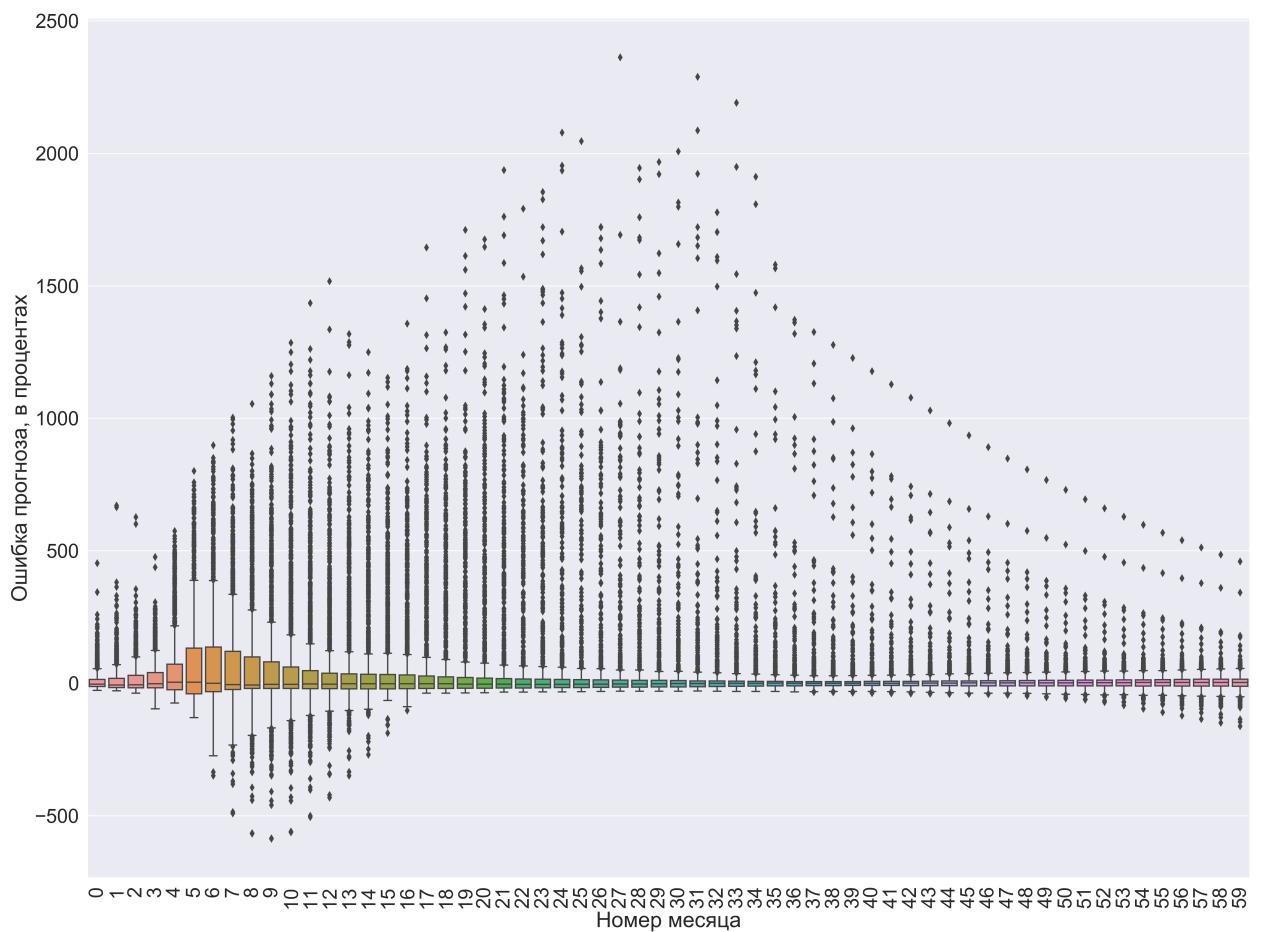


Рис.2.2. Диаграмма ошибок прогноза линейной регрессии (сценарий 2)

2.2. Регрессия ближайших соседей

Модель обучается при выполнении функции `neigh_regression` класса `ReservoirMetamodel` (см. приложение 1).

Видим, что прогнозы регрессии ближайших соседей (рис. 2.3), чем прогнозы линейной регрессии. Но по-прежнему есть высокие значения ошибок вследствие несбалансированности данных (количество примеров с низкими значениями выходных дебитов в обучающей выборке существенно меньше, чем количество примеров со средними или высокими значениями дебитов).

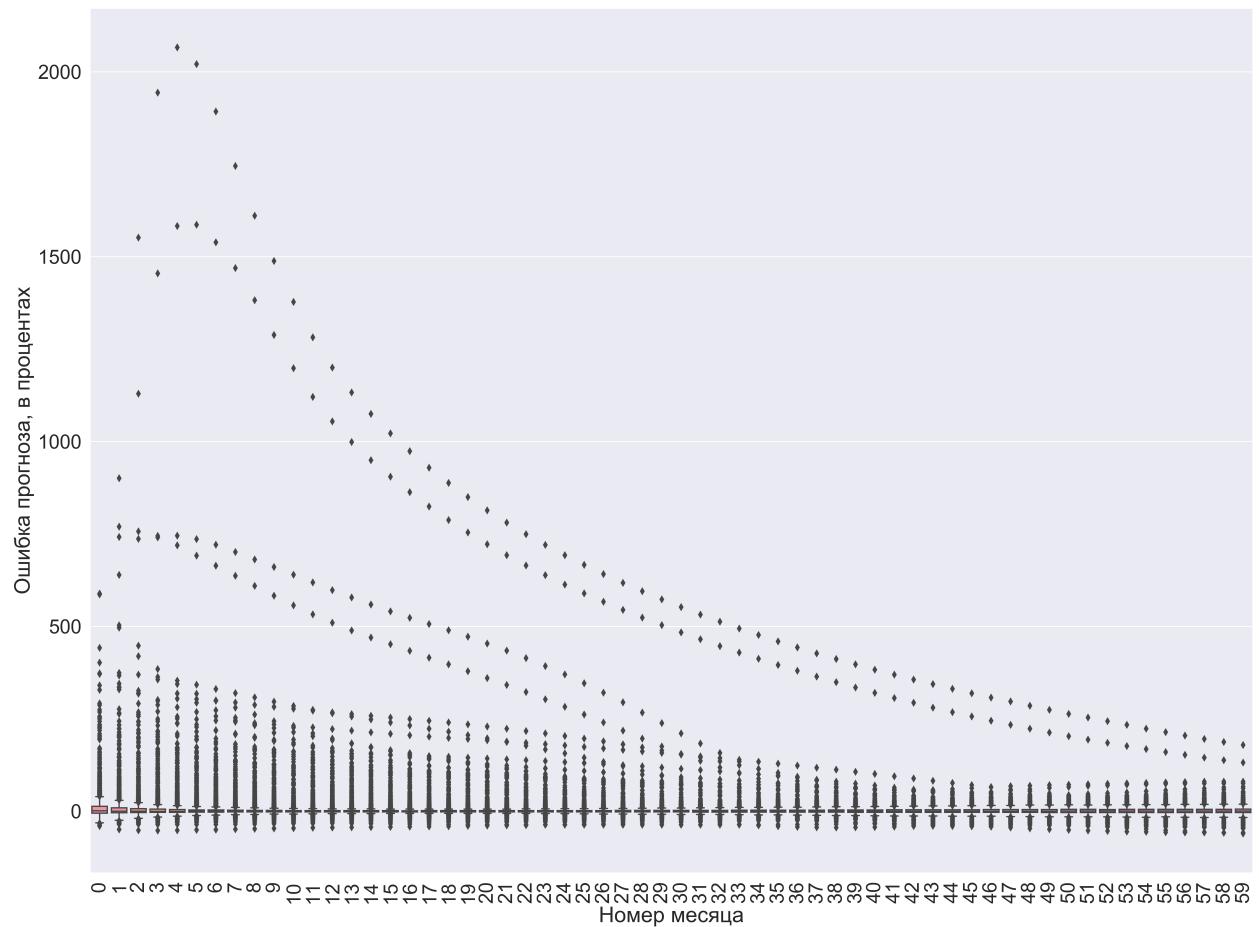


Рис.2.3. Диаграмма ошибок прогноза регрессии ближайших соседей (сценарий 1)

Заметное улучшение точности прогноза (по сравнению с линейной регрессией) наблюдается для многих кейсов второго сценария (рис. 2.4)

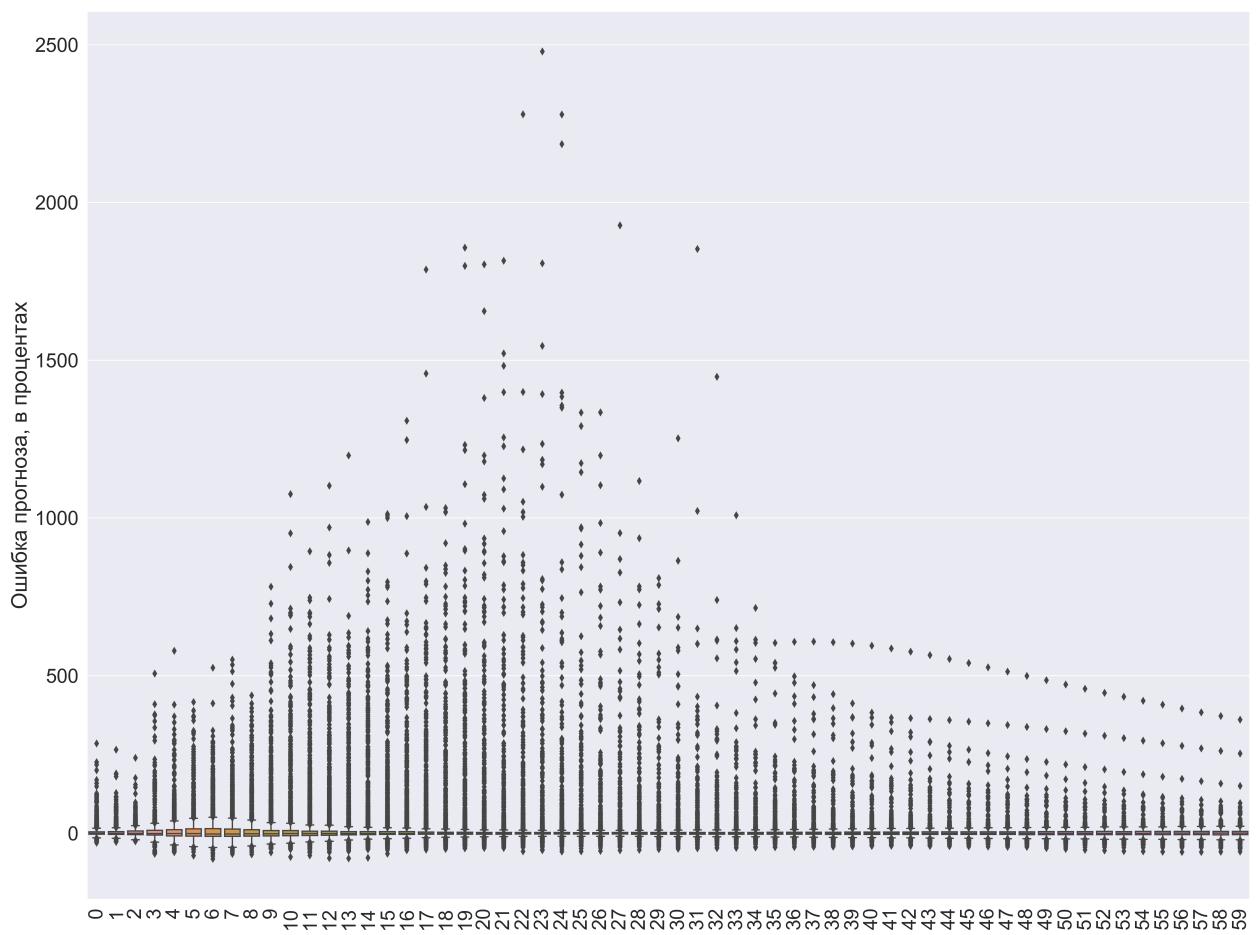


Рис.2.4. Диаграмма ошибок прогноза регрессии ближайших соседей (сценарий 2)

2.3. Регрессия на основе метода опорных векторов

Модель обучается при выполнении функции `svm_regression` класса `ReservoirMetamodel` (см. приложение 1).

Видим, что регрессия опорных векторов в первом сценарии даёт гораздо меньшие ошибки (рис. 2.5), чем линейная регрессия и регрессия ближайших соседей.

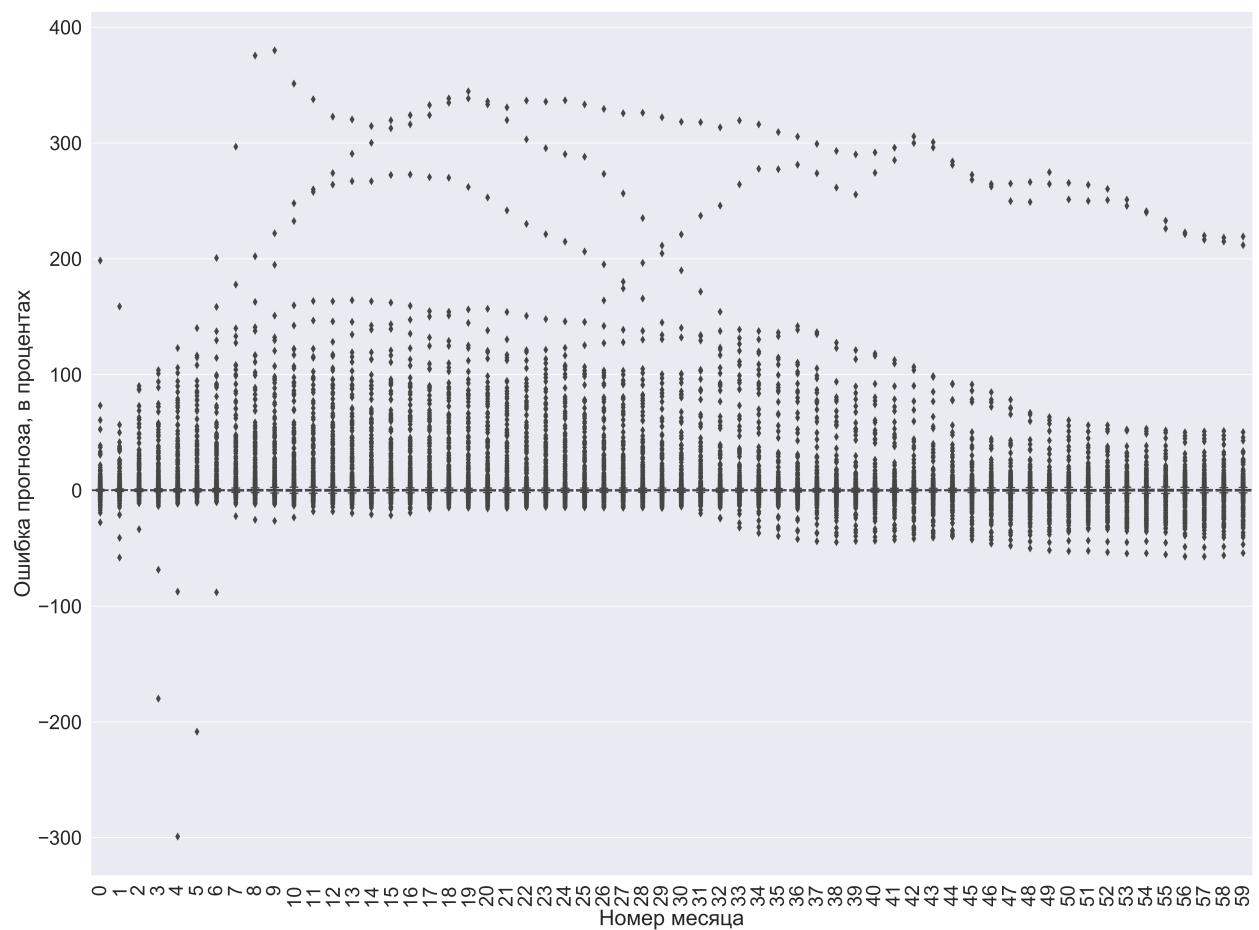


Рис.2.5. Диаграмма ошибок прогноза регрессии опорных векторов (сценарий 1)

Во втором сценарии на некоторых кейсах точность регрессии на основе опорных векторов лучше линейной и ближайших соседей, но по-прежнему есть кейсы с существенными ошибками прогноза (рис. 2.6).

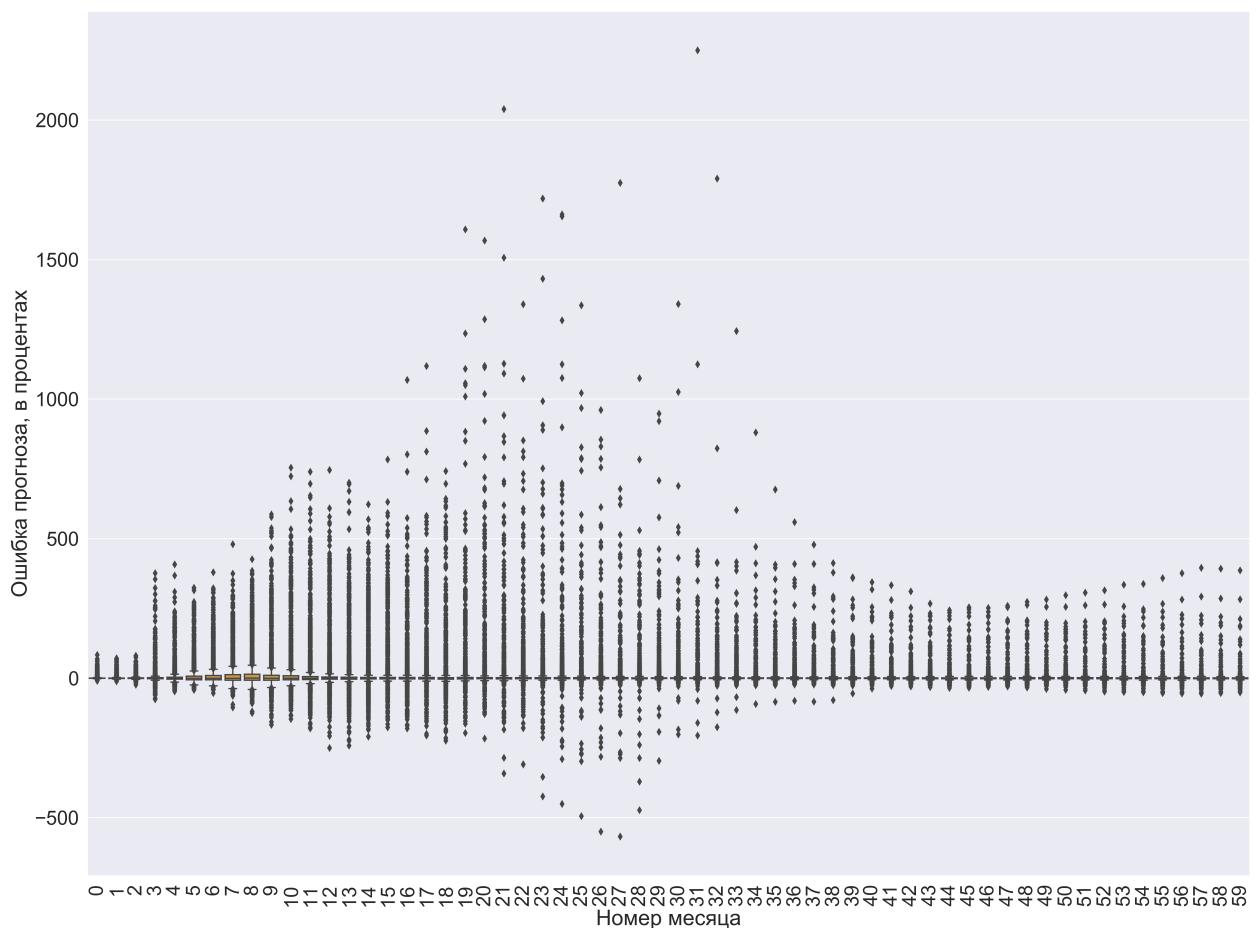


Рис.2.6. Диаграмма ошибок прогноза регрессии опорных векторов (сценарий 2)

2.4. Чрезвычайно рандомизированные деревья

Модель обучается при выполнении функции `extra_trees_regression` класса `ReservoirMetamodel` (см. приложение 1).

Из диаграмм размаха ошибок (рис. 2.7 и рис. 2.8) видим, что регрессия на основе чрезвычайно рандомизированных деревьев может быть полезна при составлении прогноза для данных, подобных данным второго сценария разработки, так как количество кейсов с высокими значениями ошибок заметно сократилось и сами значения этих ошибок снизились.

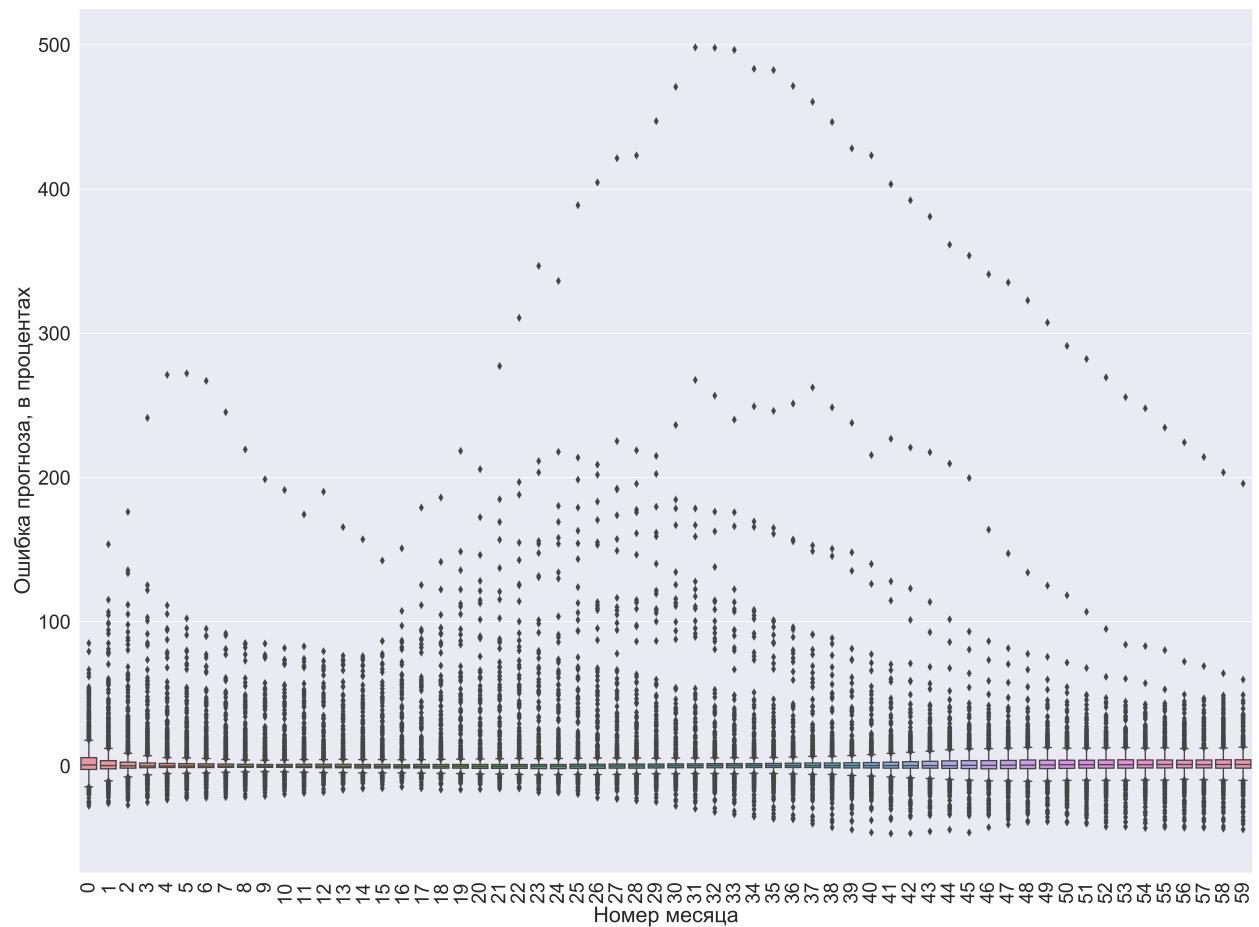


Рис.2.7. Диаграмма ошибок прогноза регрессии на основе чрезвычайно рандомизированных деревьев (сценарий 1)

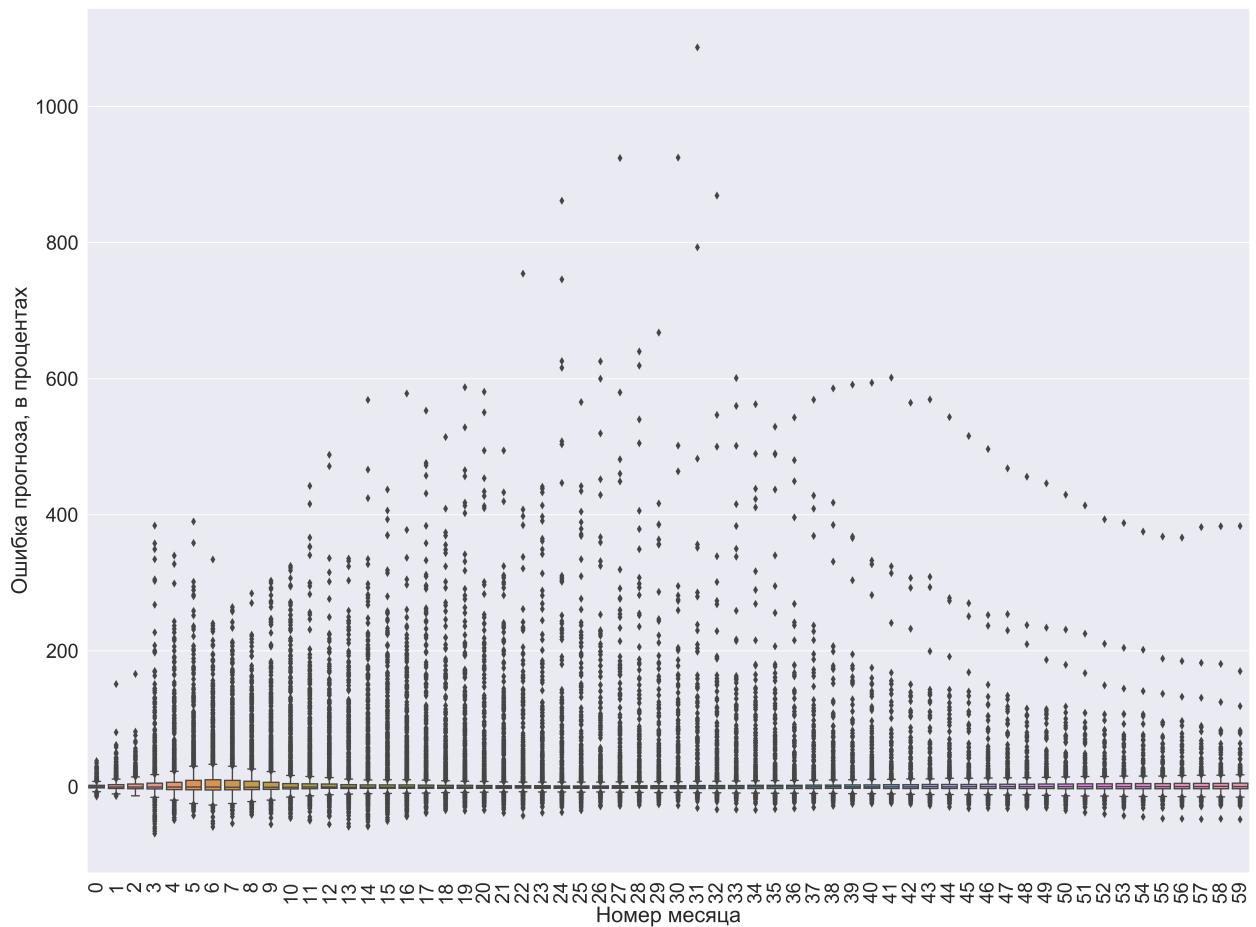


Рис.2.8. Диаграмма ошибок прогноза регрессии на основе чрезвычайно рандомизированных деревьев (сценарий 2)

2.5. Расширенные с помощью базисных функций линейные модели

Модель обучается при выполнении функции `manual_approximation` класса `ReservoirMetamodel` (см. приложение 1).

Из диаграмм размаха ошибок (рис. 2.9 и рис. 2.10) видим, что данной тип аппроксимации лучше линейной регрессии, но хуже остальных рассмотренных ранее аппроксимаций.

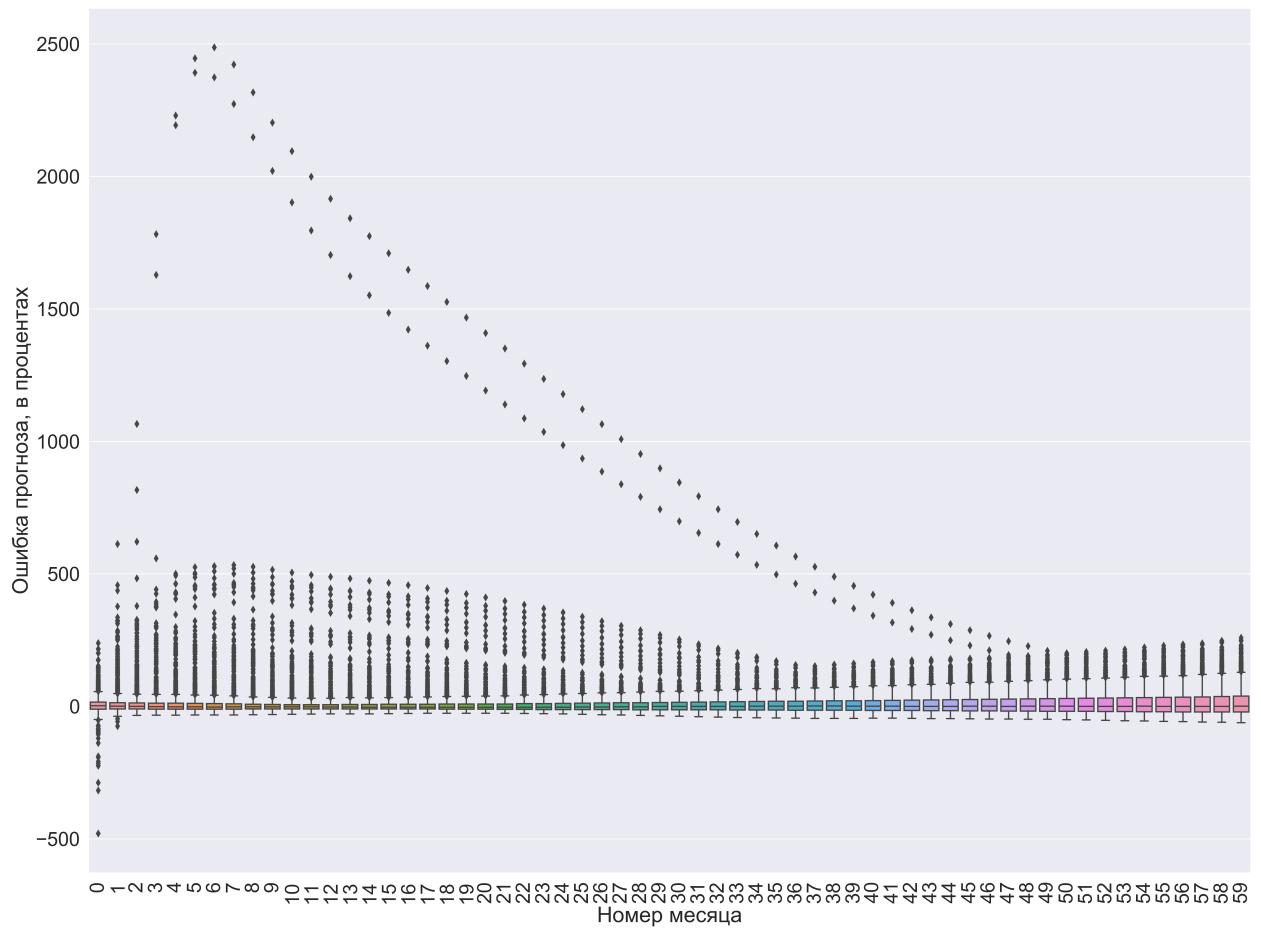


Рис.2.9. Диаграмма ошибок прогноза расширенных линейных моделей (сценарий 1)

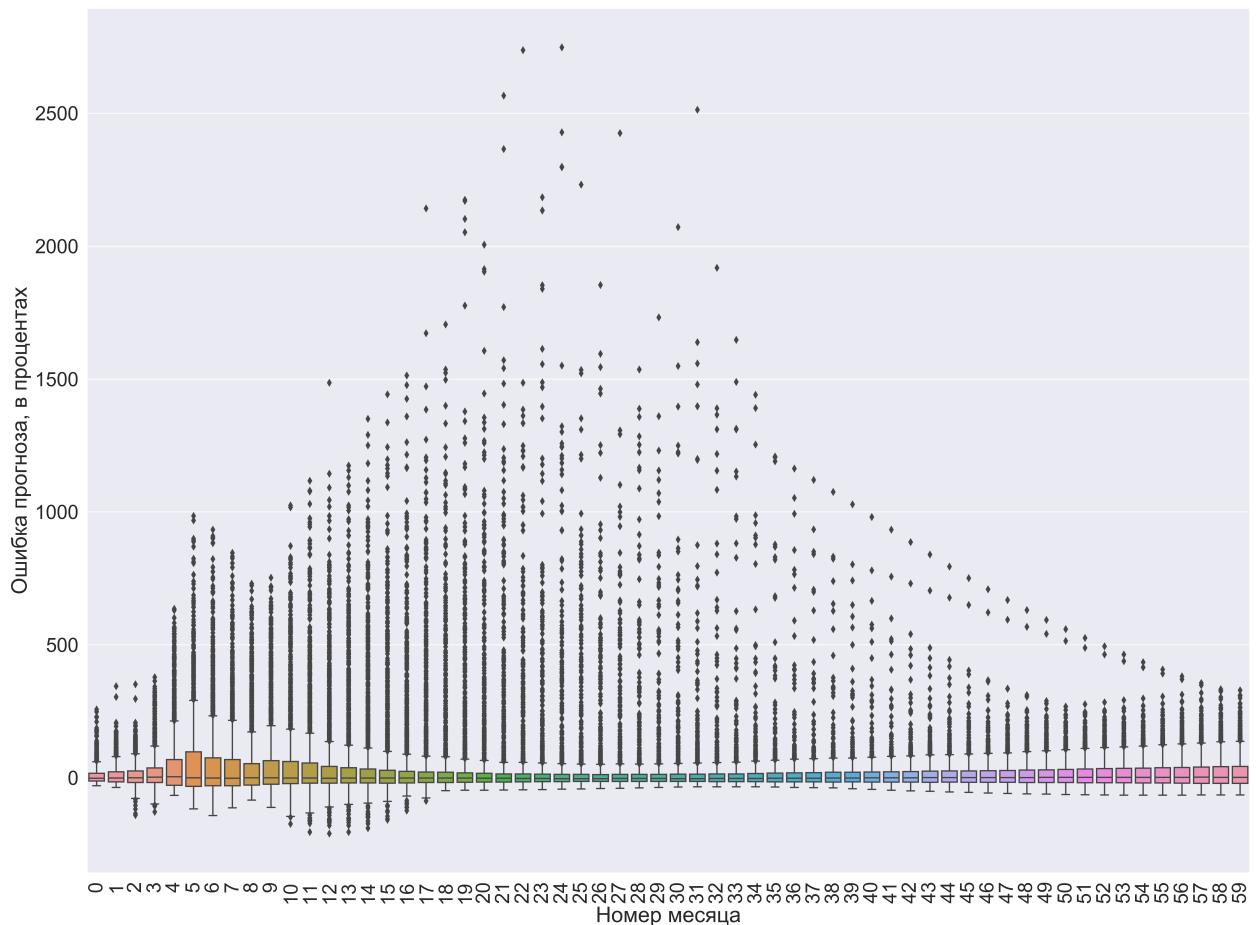


Рис.2.10. Диаграмма ошибок прогноза расширенных линейных моделей (сценарий 2)

2.6. Выводы

В текущей главе были построены аппроксимации разными методами машинного обучения. Наиболее точными оказались регрессия на основе метода опорных векторов и регрессия на основе чрезвычайно рандомизированных деревьев. Однако точность их прогноза не значительно лучше (а для некоторых кейсов хуже) точности прогноза нейронных сетей и градиентного бустинга на основе деревьев регрессии из работы [2].

ЗАКЛЮЧЕНИЕ

В данной работе предприняты попытки улучшить качество аппроксимаций, которые были построены в работе [2].

С помощью регрессии опорных векторов и чрезвычайно рандомизированных деревьев получились более точные результаты прогноза, чем результаты прогноза градиентного бустинга.

Однако построить простую (не ансамблевую) модель с результатами точнее результатов нейронной сети не удалось.

В дальнейшем точность прогноза может быть улучшена с помощью ансамблевых методов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация scikit-learn. — URL: <https://scikit-learn.ru> (дата обращения: 20.05.2022).
2. *Муравицев А. А.* Построение метамодели притока флюида к вертикальной скважине по данным из гидродинамического симулятора. — СПб.: Высшая школа теоретической механики, 2021. — 95 с. — (Сер.: ВКР).
3. *Brownlee J.* Master Machine Learning Algorithms. — 2017. — 162 p.
5. *Cronbecq K.* Surrogate Modelling of Computer Experiments with Sequential Experimental Design. — University of Antwerp, Ghent University, 2011.
4. *Gorrisen D.* Grid-Enabled Adaptive Surrogate Modeling for Computer Aided Engineering. — University of Antwerp, Ghent University, 2010.

Приложение 1

Код программы для построения метамоделей

```

import pandas as pd
import numpy as np
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
import matplotlib.pyplot as plt
import seaborn as sns
import os
from scipy.optimize import curve_fit

def save_fig(fig_id, tight_layout=True, fig_extension='png', resolution=300):
    path = os.path.join('images', fig_id + '.' + fig_extension)
    print('Saving figure', fig_id)
    if tight_layout:
        plt.tight_layout()
    plt.savefig(path, format=fig_extension, dpi=resolution)

class DataFrameSelector(BaseEstimator, TransformerMixin):
    def __init__(self, attribute_names):
        self.attribute_names = attribute_names

    def fit(self, x, y=None):
        return self

    def transform(self, x):
        return x[self.attribute_names].values

class ReservoirMetamodel:
    def __init__(self, data):
        self.data = data
        self.data_train = data[:int(4 * len(data) / 13)]
        self.data_valid = data[int(4 * len(data) / 13):int(5 * len(data) / 13)]
        self.data_test = data[int(5 * len(data) / 13):]

    def print_df(self):
        print(self.data_train)

    def print_info(self):
        print(self.data.info())

```

```

def print_describe(self):
    print(self.data.describe())

def plot_parameters_distributions(self, suffix='_unknown'):
    self.data.iloc[:, 7:59:5].hist(bins=50, figsize=(15, 10))
    save_fig('parameters_histograms' + suffix)

def corr_matrix(self):
    return self.data_train.corr()

def plot_corr_heat_map(self, suffix='_unknown'):
    sns.set(rc={'figure.figsize': (20, 15)}, font_scale=1)
    plt.figure()
    corr_matrix = self.corr_matrix()
    ax = sns.heatmap(corr_matrix, cmap='coolwarm',
                      annot=False, vmin=-1, vmax=1)
    cbar = ax.collections[0].colorbar
    cbar.ax.tick_params(labelsize=20)
    save_fig('corr_matrix' + suffix)

def all_scatter_plot(self):
    sns.set(rc={'figure.figsize': (20, 15)}, font_scale=2)
    plt.figure()
    sns.scatterplot(x=self.data['PERMX CRACK (mD)'],
                    y=self.data['OIL RATE 1 (SM3/DAY)'],
                    hue=self.data['PORO CRACK'])
    save_fig('all_data_scatterplot_1')

def manual_approximation(self):
    sns.set(rc={'figure.figsize': (20, 15)}, font_scale=2)

    def approx_func(X, a1, a2, a3, b):
        x1, x2 = X
        return a1 * x1 + a2 * x2 + a3 * x1 * x2 + b

    ys = []
    for i in range(1, 61):
        popt, _ = curve_fit(approx_func,
                             (self.data_train['PERMX CRACK (mD)'],
                              self.data_train['PORO CRACK']),
                             self.data_train[f'OIL RATE {i} (SM3/DAY)'])
        a1, a2, a3, b = popt
        y = approx_func((self.data_test['PERMX CRACK (mD)'],
                         self.data_test['PORO CRACK']), a1, a2, a3, b)
        if i == 1:
            print(type(y))
        ys.append(y)
    return ys

def data_scaler(self, input):
    num_atrribes = list(input)
    num_pipeline = Pipeline([

```

```

        ('selector', DataFrameSelector(num_attribs)),
        ('std_scaler', StandardScaler())
    ])
train_input_prepared = num_pipeline.fit_transform(input)
return train_input_prepared

def linear_regression(self):
    lin_regs = []
    for i in range(1, 61):
        lin_reg = LinearRegression()
        lin_reg.fit(self.data_scaler(self.data_train.iloc[:, 0:4]),
                    self.data_train[f'OIL RATE {i} (SM3/DAY)'])
        lin_regs.append(lin_reg)
    return lin_regs

def neigh_regression(self):
    neigh_regs = []
    for i in range(1, 61):
        neigh_reg = KNeighborsRegressor(n_neighbors=4)
        neigh_reg.fit(self.data_scaler(self.data_train.iloc[:, 0:4]),
                      self.data_train[f'OIL RATE {i} (SM3/DAY)'])
        neigh_regs.append(neigh_reg)
    return neigh_regs

def svm_regression(self):
    svm_regs = []
    for i in range(1, 61):
        svm_reg = SVR(C=250.0, epsilon=0.01)
        svm_reg.fit(self.data_scaler(self.data_train.iloc[:, 0:4]),
                    self.data_train[f'OIL RATE {i} (SM3/DAY)'])
        svm_regs.append(svm_reg)
    return svm_regs

def extra_trees_regression(self):
    extra_trees_regs = []
    for i in range(1, 61):
        extra_trees_reg = ExtraTreesRegressor(n_estimators=500,
                                              random_state=42)
        extra_trees_reg.fit(self.data_scaler(self.data_train.iloc[:, 0:4]),
                            self.data_train[f'OIL RATE {i} (SM3/DAY)'])
        extra_trees_regs.append(extra_trees_reg)
    return extra_trees_regs

def plot_errors_boxplot(self, approx='linear_regression_test_errors_boxplot',
                       suffix='_unknown'):
    sns.set(rc={'figure.figsize': (20, 15)}, font_scale=2)
    relative_errors = []
    if approx == 'manual_approximation_test_errors_boxplot':
        y = self.manual_approximation()
        for i in range(1, 61):
            relative_error = (y[i - 1] -
                              self.data_test[f'OIL RATE {i} (SM3/DAY)']) \

```

```

                / self.data_test[f'OIL RATE {i} (SM3/DAY)'] * 100
        relative_errors.append(relative_error)
    elif approx == 'extra_trees_regression_test_errors_boxplot':
        extra_trees_regs = self.extra_trees_regression()
        for i in range(1, 61):
            relative_error = (extra_trees_regs[i - 1].predict(
                self.data_scaler(self.data_test.iloc[:, 0:4])) -
                np.array(self.data_test[f'OIL RATE {i}' +
                                      f'(SM3/DAY)'])) / \
                (np.array(self.data_test[f'OIL RATE {i}' +
                                      f'(SM3/DAY)'])) * 100
            relative_errors.append(relative_error)
    elif approx == 'nearest_neighs_approximation_test_errors_boxplot':
        neigh_regs = self.neigh_regression()
        for i in range(1, 61):
            relative_error = (neigh_regs[i - 1].predict(
                self.data_scaler(self.data_test.iloc[:, 0:4])) -
                np.array(self.data_test[f'OIL RATE {i}' +
                                      f'(SM3/DAY)'])) / \
                (np.array(self.data_test[f'OIL RATE {i}' +
                                      f'(SM3/DAY)'])) * 100
            relative_errors.append(relative_error)
    elif approx == 'svm_approximation_test_errors_boxplot':
        svm_regs = self.svm_regression()
        for i in range(1, 61):
            relative_error = (svm_regs[i - 1].predict(
                self.data_scaler(self.data_test.iloc[:, 0:4])) -
                np.array(self.data_test[f'OIL RATE {i}' +
                                      f'(SM3/DAY)'])) / \
                (np.array(self.data_test[f'OIL RATE {i}' +
                                      f'(SM3/DAY)'])) * 100
            relative_errors.append(relative_error)
    else:
        lin_regs = self.linear_regression()
        for i in range(1, 61):
            relative_error = (lin_regs[i - 1].predict(
                self.data_scaler(self.data_test.iloc[:, 0:4])) -
                np.array(self.data_test[f'OIL RATE {i}' +
                                      f'(SM3/DAY)'])) / \
                (np.array(self.data_test[f'OIL RATE {i}' +
                                      f'(SM3/DAY)'])) * 100
            relative_errors.append(relative_error)
    plt.figure()
    plt.xticks(rotation=90)
    sns.boxplot(data=relative_errors)
    plt.xlabel('Номер месяца')
    plt.ylabel('Ошибка прогноза, в процентах')
    save_fig(approx + suffix)

if __name__ == '__main__':
    dataset_file = 'reservoir_train800_scenario1.csv'

```

