2025

Лабораторная работа № 1

ПРАКТИК: СУХИНИ Е.С
ЕГОР ЯЧМЕННИКОВ, ФЁДОР ДЕМЬЯНОВ, КУЗНЕЦОВА ТАИСИЯ

```python
import numpy as np
import pandas as pd

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.metrics import mean_absolute_error

import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
```

Импортируем нужные библиотеки

```python
data = pd.read_csv('/content/drive/MyDrive/housing.csv')
X = data.drop('MEDV', axis=1)
y = data['MEDV']
```

Считываем данные, делим их на признаки (X) и целевые (y)

```python
if robot:
    # Рзбиение данных на тренировочные и тестовые
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=test_size, random_state=random_state)

    if use_scaler:
        X_train = scale.fit_transform(X_train)
        X_test = scale.transform(X_test)

    # Вычисление коэффициентов и применение настроек на данных
    X_train = poly.fit_transform(X_train)
    X_test = poly.transform(X_test)

    # Создаем объект класса и вычисляем коэффициенты на Train
    model = LinearRegression(fit_intercept=True)
    model.fit(X_train, Y_train)

    # Предиктим
    train_pred = model.predict(X_train)
    test_pred = model.predict(X_test)
```

Код для обучения модели линейной регрессии из sklearn.

Сначала разбиваем данные на тренировочные и тестовые, затем применяем scaler, если нужно, полиномиализируем их, создаём объект модели и обучаем его. Вычисляем Абсолютные Средние Ошибки (MAE) для тренировочных и для тестовых данных.

```python
else:
    # Создаем полиномиальные признаки
    X_poly = poly.fit_transform(X)

    # Разделение данных на тестовые и обучающие
    X_train, X_test, Y_train, Y_test = train_test_split(X_poly, Y, test_size=test_size, random_state=random_state)

    # Инициализация скейлеров
    x_scaler, y_scaler = None, None

    # Масштабирование данных
    if use_scaler:
        X_train = scale.fit_transform(X_train)
        X_test = scale.transform(X_test)

        Y_train = scale.fit_transform(Y_train.values.reshape(-1, 1)).flatten()

    # Находим число обусловленности и находим коэффициеты, решая систему
    XTX = X_train.T @ X_train
    cond_number = np.linalg.cond(XTX)
    try:
        theta = np.linalg.inv(XTX) @ X_train.T @ Y_train
    except np.linalg.LinAlgError:
        theta = np.linalg.pinv(XTX) @ X_train.T @ Y_train

    # Получение предсказаний
    train_pred = X_train @ theta
    test_pred = X_test @ theta

    # Обратное преобразование предсказаний (если скейлер True)
    if use_scaler:
        train_pred = scale.inverse_transform(train_pred.reshape(-1, 1)).flatten()
        test_pred = scale.inverse_transform(test_pred.reshape(-1, 1)).flatten()
        Y_train = scale.inverse_transform(Y_train.reshape(-1, 1)).flatten()

# Расчет ошибок
mae_train = mean_absolute_error(Y_train, train_pred)
mae_test = mean_absolute_error(Y_test, test_pred)
```

Код для реализации линейной регрессии методом Наименьших Квадратов (вручную).

Вычисляем $X^T \cdot X$. Зачем? Мы должны минимизировать разность квадратов, то есть эту функцию: $||X\theta - y||^2$ (θ – искомые коэффициенты). Чтобы найти минимум мы берём производную: $\frac{d}{d\theta}||X\theta - y||^2 = \frac{d}{d\theta}(X\theta - y)^T \cdot (X\theta - y)$ (так получается потому что мы умножаем вектор на самого себя) $= \frac{d}{d\theta}(\theta^T \cdot X^T \cdot X\theta - 2y^T \cdot X\theta + y^T y) = 2X^T \cdot X\theta - 2X^T y = = 0 \rightarrow X^T \cdot X\theta = X^T y$.

Затем решаем $X^T \cdot X\theta = X^T y \rightarrow \theta = (X^T X)^{-1} \cdot X^T y$

В случае, если $X^T X$ вырождена и, следовательно, обратную матрицу найти нельзя, мы ловим исключение и считаем через псевдообратную матрицу.

Также считаем число обусловленности матрицы, показывающее меру устойчивости матрицы. Если число слишком большое, то малые изменения в данных приводят к большим изменениям в θ.

Получив коэффициенты, предсказываем целевую переменную y. Преобразуем данные обратно, если использовали scaler, и считаем MAE.

	Degree	Scaler	MAE Train	MAE Test	Condition Number
0	1	False	3.283	3.481	7.248816e+07
1	1	True	3.237	3.566	9.587028e+01
2	2	False	1.818	2.599	4.512723e+15
3	2	True	1.809	2.572	1.722967e+08
4	3	False	85.499	2899087.931	1.446579e+30
5	3	True	1.844	253.193	2.395357e+17

Таблица для аналитического метода

При Degree = 1 ошибки небольшие, число обусловленности наоборот высокое, но с использованием Scaler'а значительно снижается.

При Degree = 2 ошибки меньше, чем раньше, что говорит о том, что модель стала лучше. Числа обусловленности получаются большими, что свидетельствует о вычислительной неустойчивости.

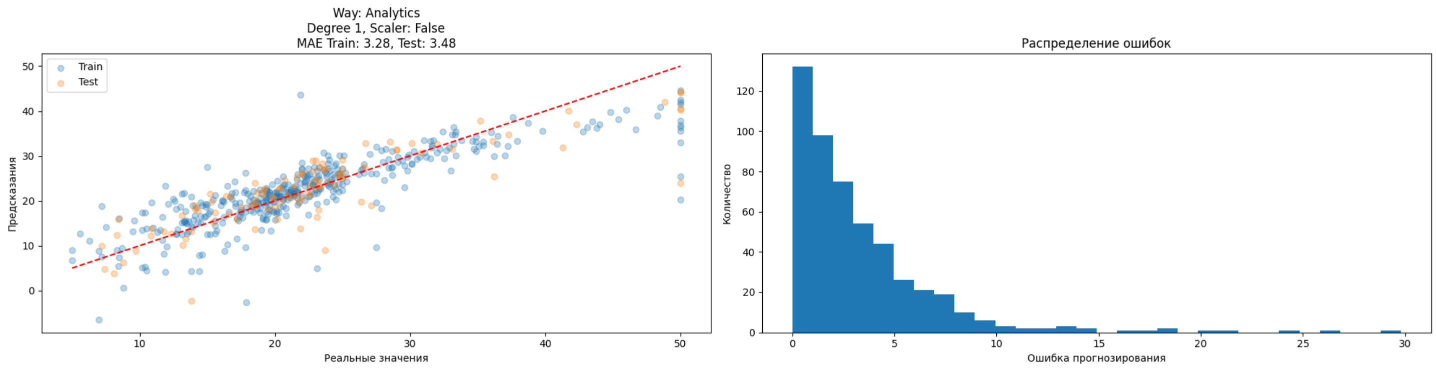
При Degree = 3 ошибки сильно увеличиваются, особенно на тестовых данных, что свидетельствует о переобучении модели. Числа обусловленности так же сильно велики.

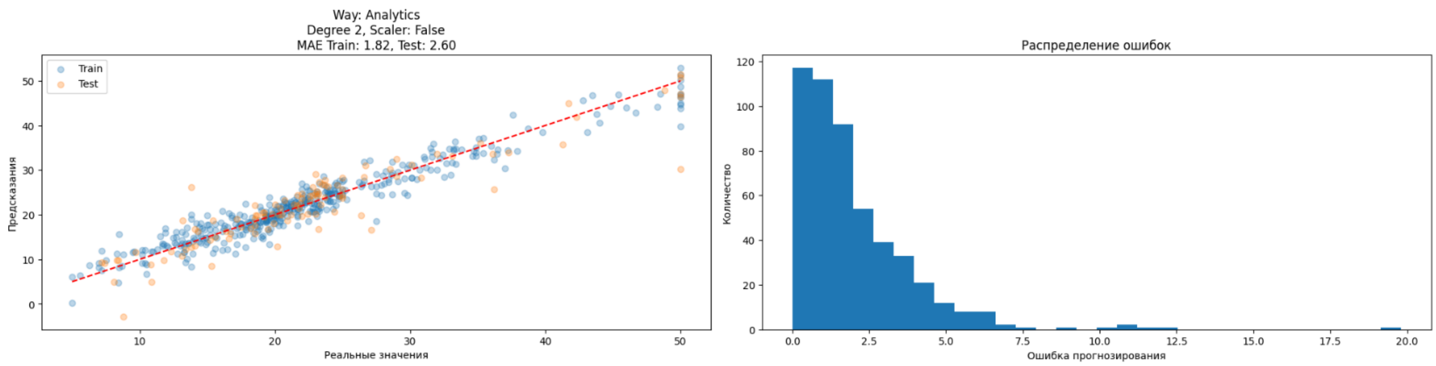
Можно сделать вывод, что лучшим выбором станет полином степени два с предварительным масштабированием.

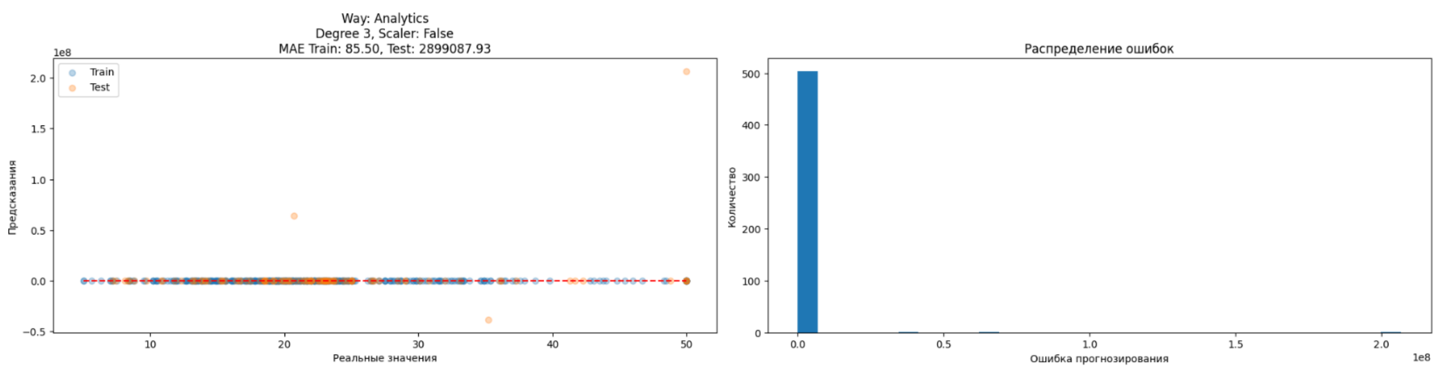
	Degree	Scaler	MAE Train	MAE Test
0	1	False	3.237	3.566
1	1	True	3.237	3.566
2	2	False	1.809	2.572
3	2	True	1.809	2.572
4	3	False	0.645	10.403
5	3	True	0.645	32.570

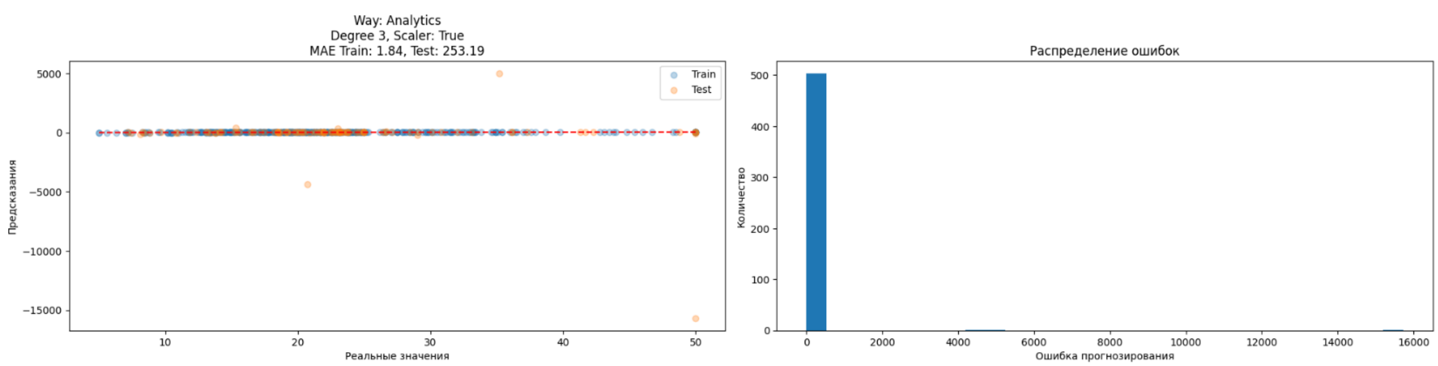
Таблица для sklearn LinearRegression

Выводы такие же.

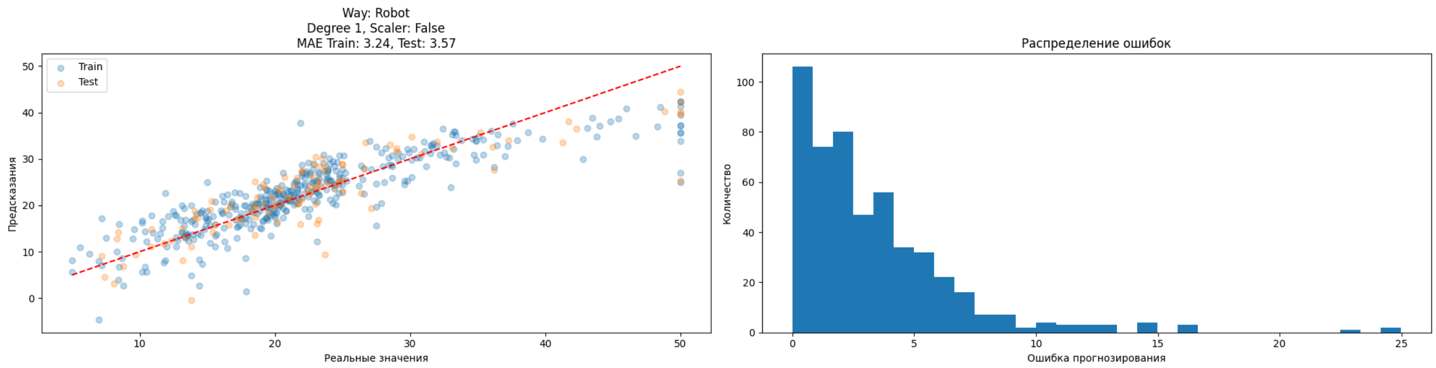


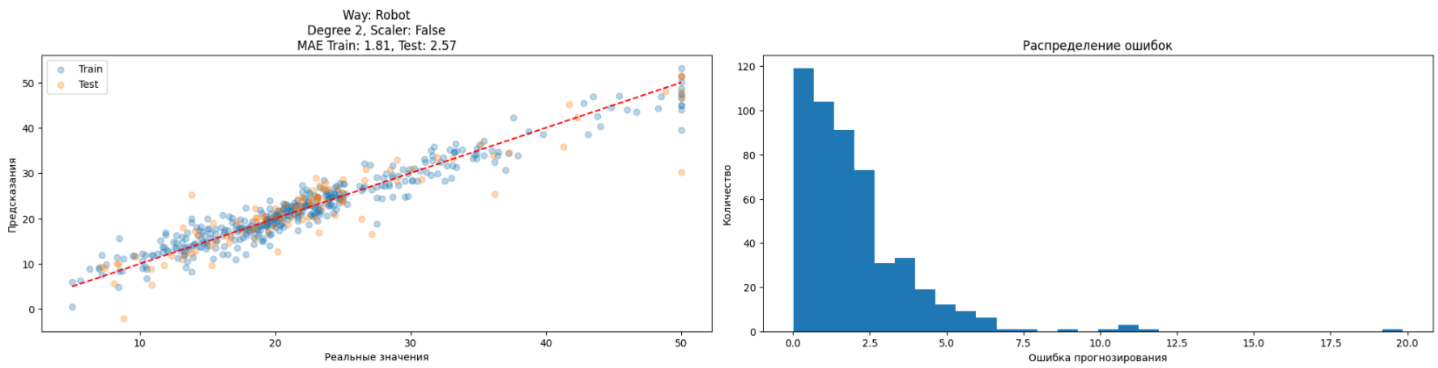


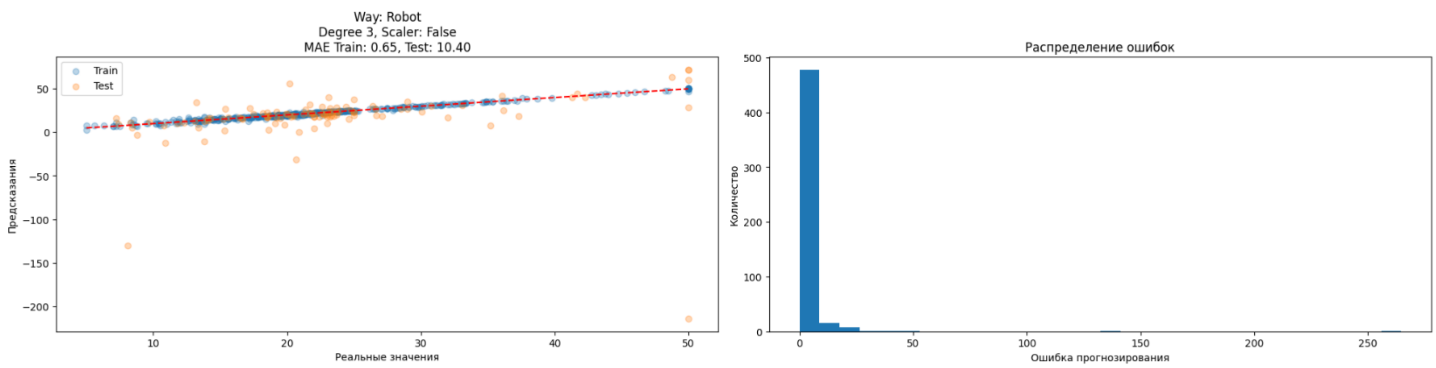


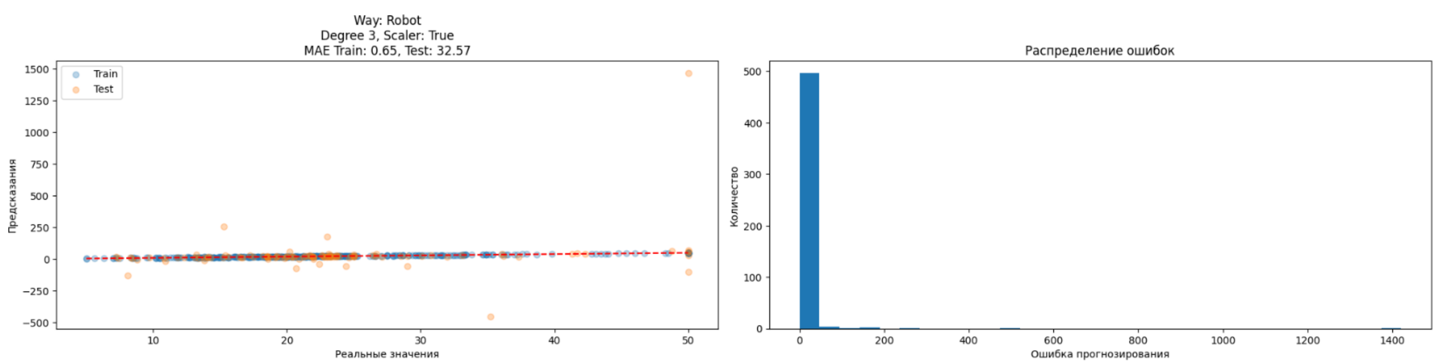


Графики для аналитического метода









Графики sklearn Linear Regression