МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «искуственные нейронные сети»

Тема: Регрессионная модель изменения цен на дома в Бостоне

Студент гр. 8382 _____ Щемель Д.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель

Реализовать предсказание медианной цены на дома в пригороде Бостона в середине 1970-х по таким данным, как уровень преступности, ставка местного имущественного налога и т. д.

Данный набор содержит относительно немного образцов данных: всего 506, разбитых на 404 обучающих и 102 контрольных образца. И каждый признак во входных данных (например, уровень преступности) имеет свой масштаб. Например, некоторые признаки являются пропорциями и имеют значения между 0 и 1, другие — между 1 и 12 и т. д.

Задачи

- Ознакомиться с задачей регрессии
- Изучить отличие задачи регрессии от задачи классификации
- Создать модель
- Настроить параметры обучения
- Обучить и оценить модели
- Ознакомиться с перекрестной проверкой

Требования

1. Объяснить различия задач классификации и регрессии
2. Изучить влияние кол-ва эпох на результат обучения модели
3. Выявить точку переобучения
4. Применить перекрестную проверку по K блокам при различных K
5. Построить графики ошибки и точности во время обучения для моделей, а также усредненные графики по всем моделям

Ход работы

1. Различия задач классификации и регрессии

Задача классификации - задача определения принадлежности входных данных к конкретному классу (дискретному значению). Задача регрессии - задача предсказания непрерывного значения (не дискретного).

2. Изучение влияния количества эпох на результат обучения модели

Параметры модели:

Количество эпох: 100

Количество блоков: 4

Результаты обучения для моделей представлены на рис.1-8. Усредненные результаты для всех моделей приведены на рис.9-10.

Средняя оценка: 2.656462695002556

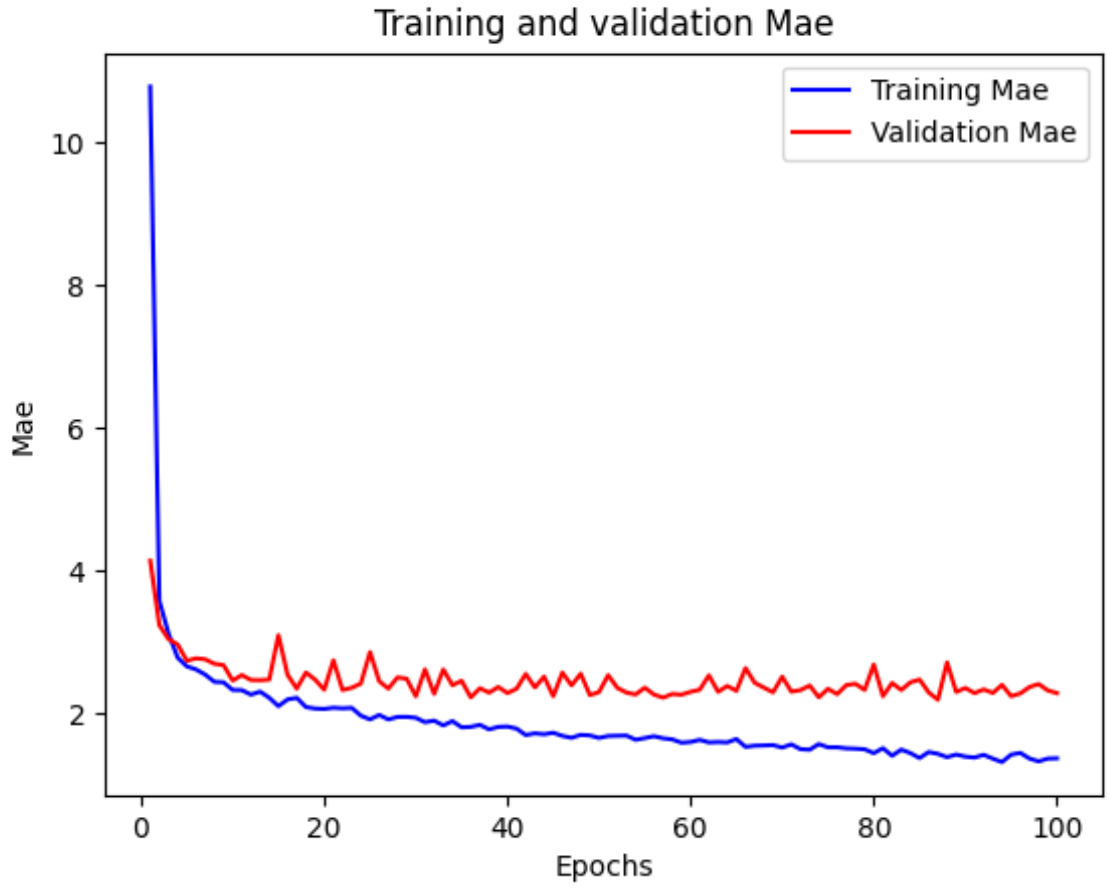

Рис. 1: Mae для модели 1

По последнему графику видно, что значение Mae перестаёт сильно улучшаться после 30ой эпохи.

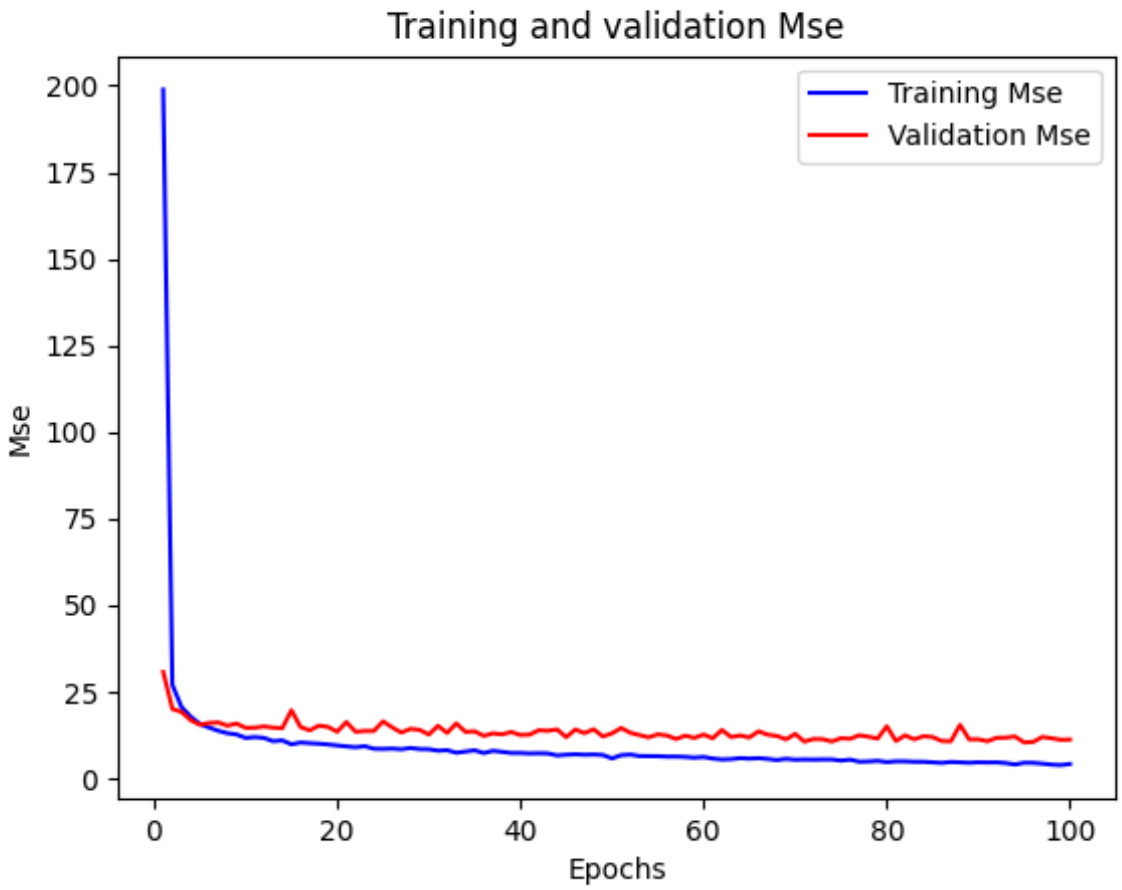

Рис. 2: Mse для модели 1

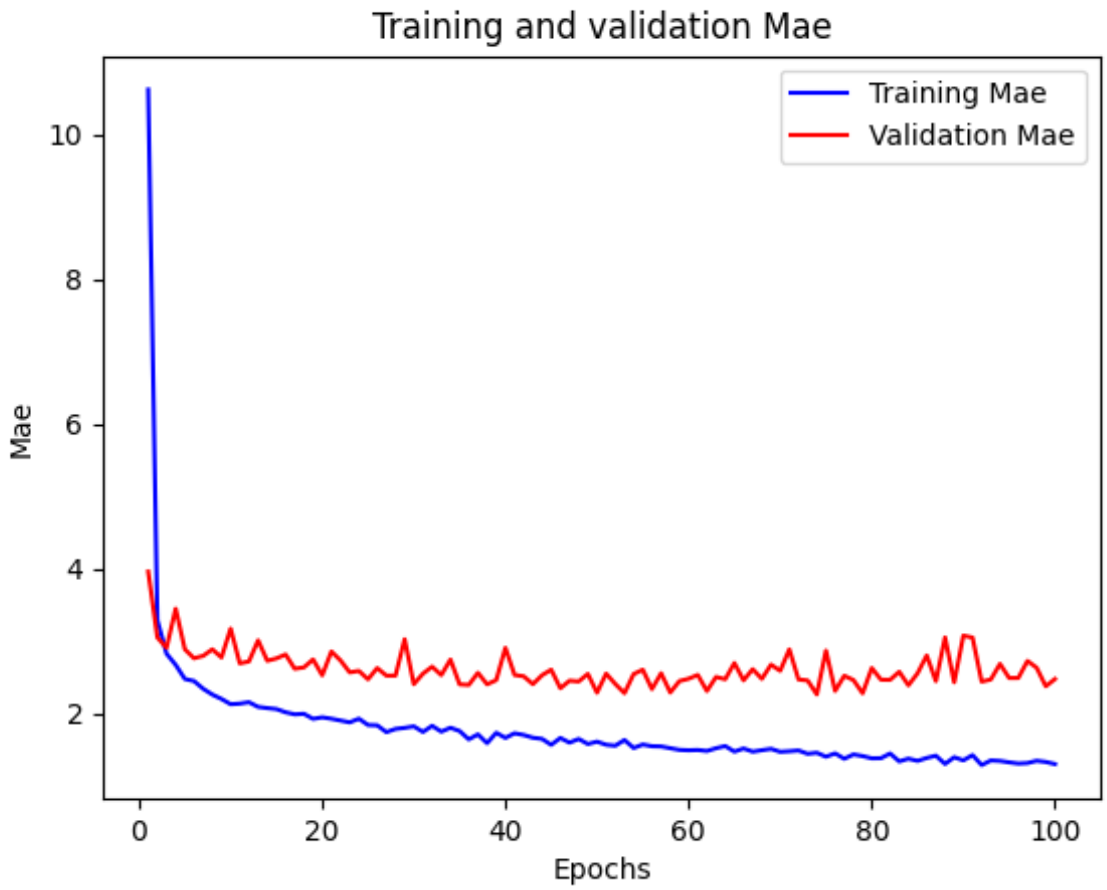

Рис. 3: Mae для модели 2

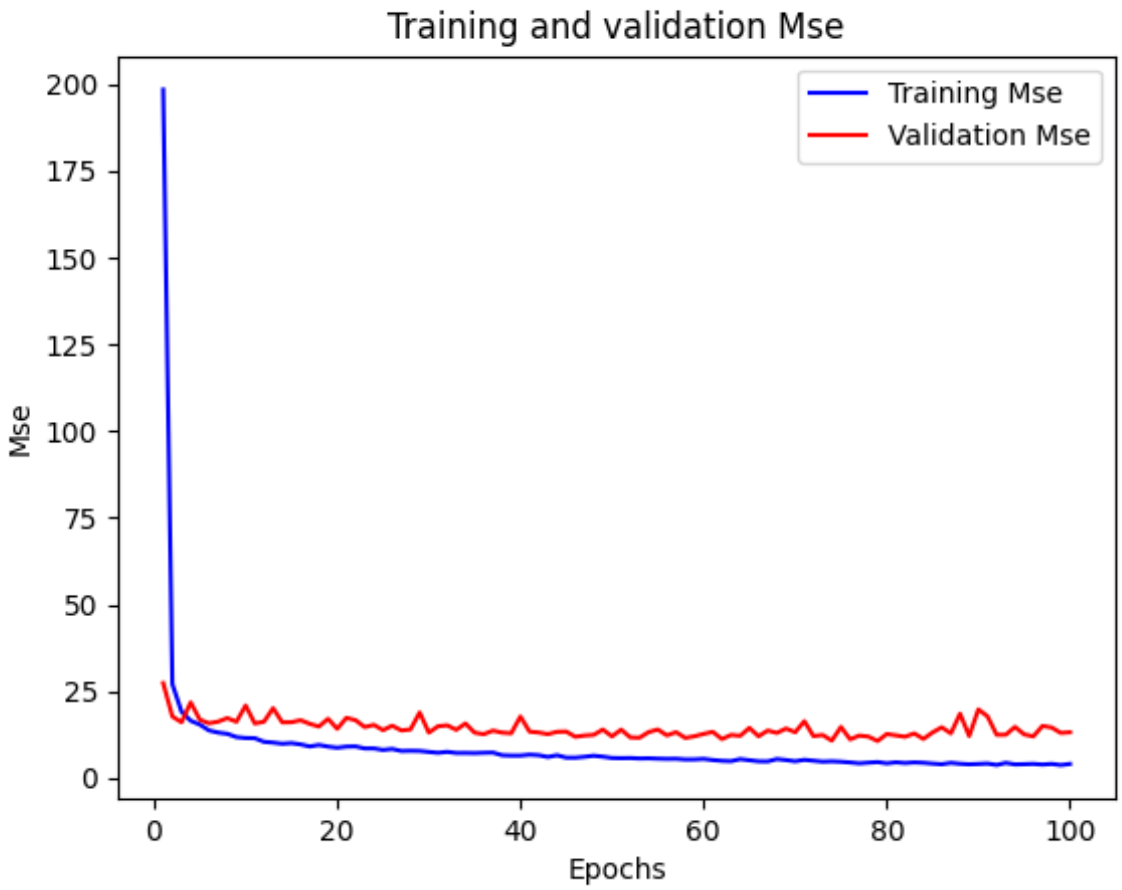

Рис. 4: Mae для модели 2

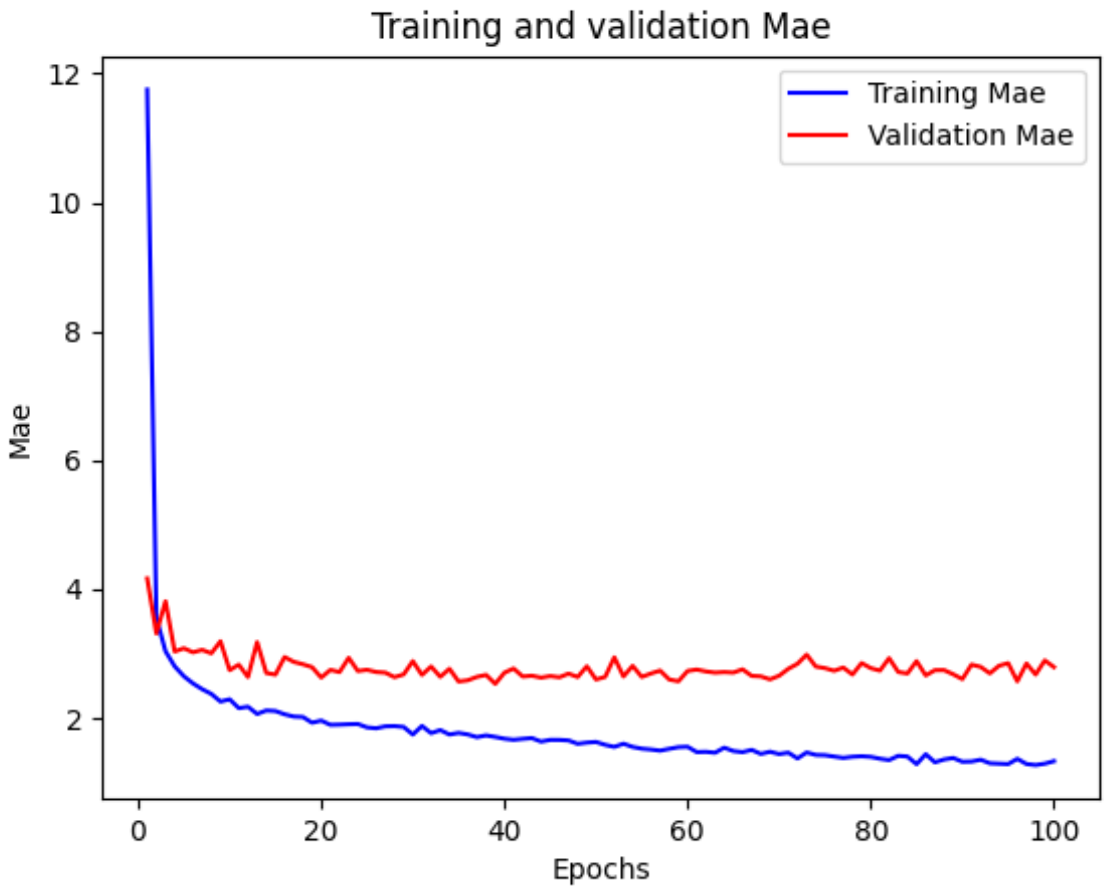

Рис. 5: Mae для модели 3

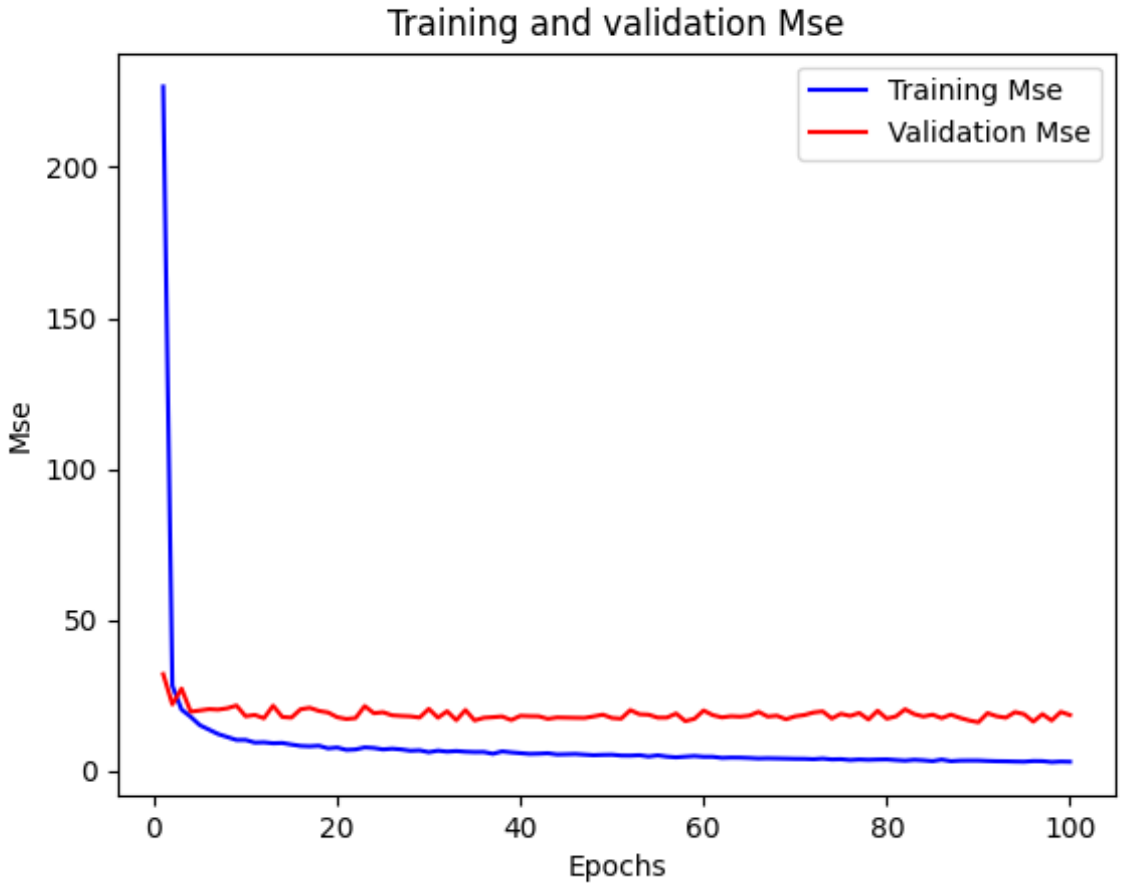

Рис. 6: Mae для модели 3

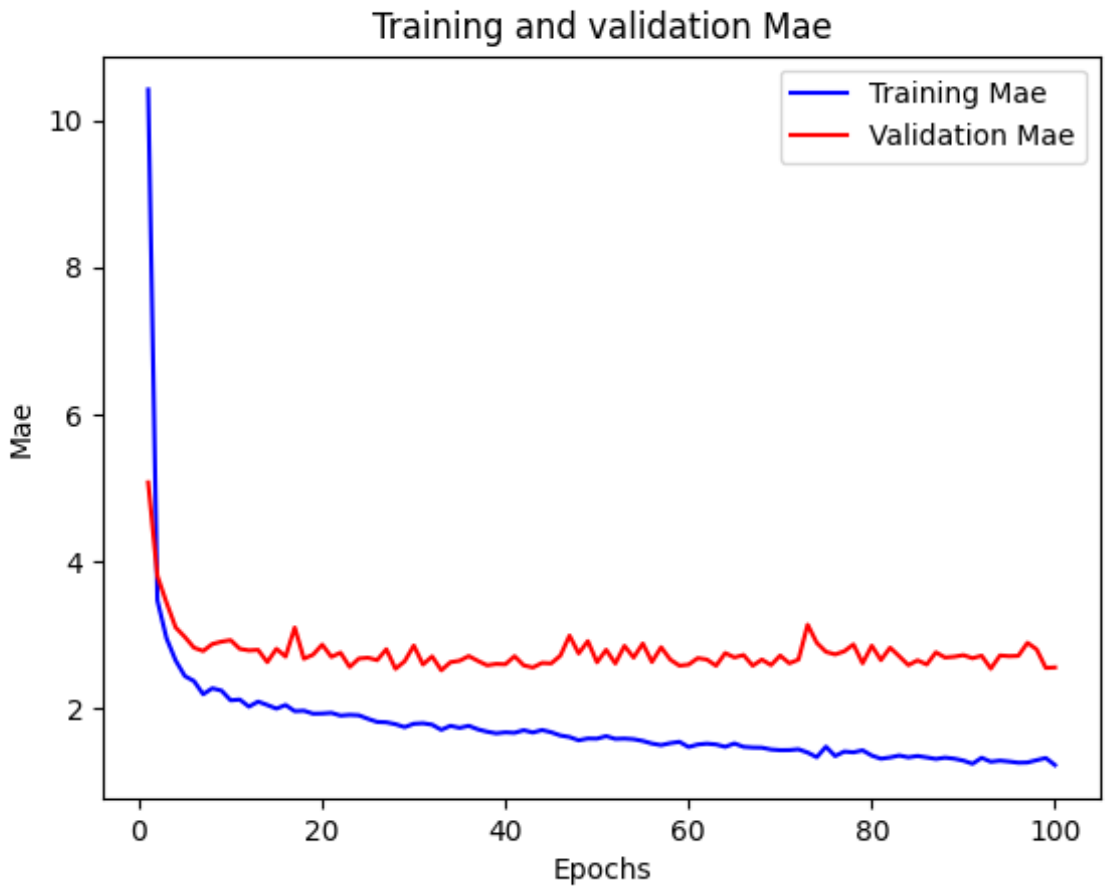

Рис. 7: Mae для модели 4

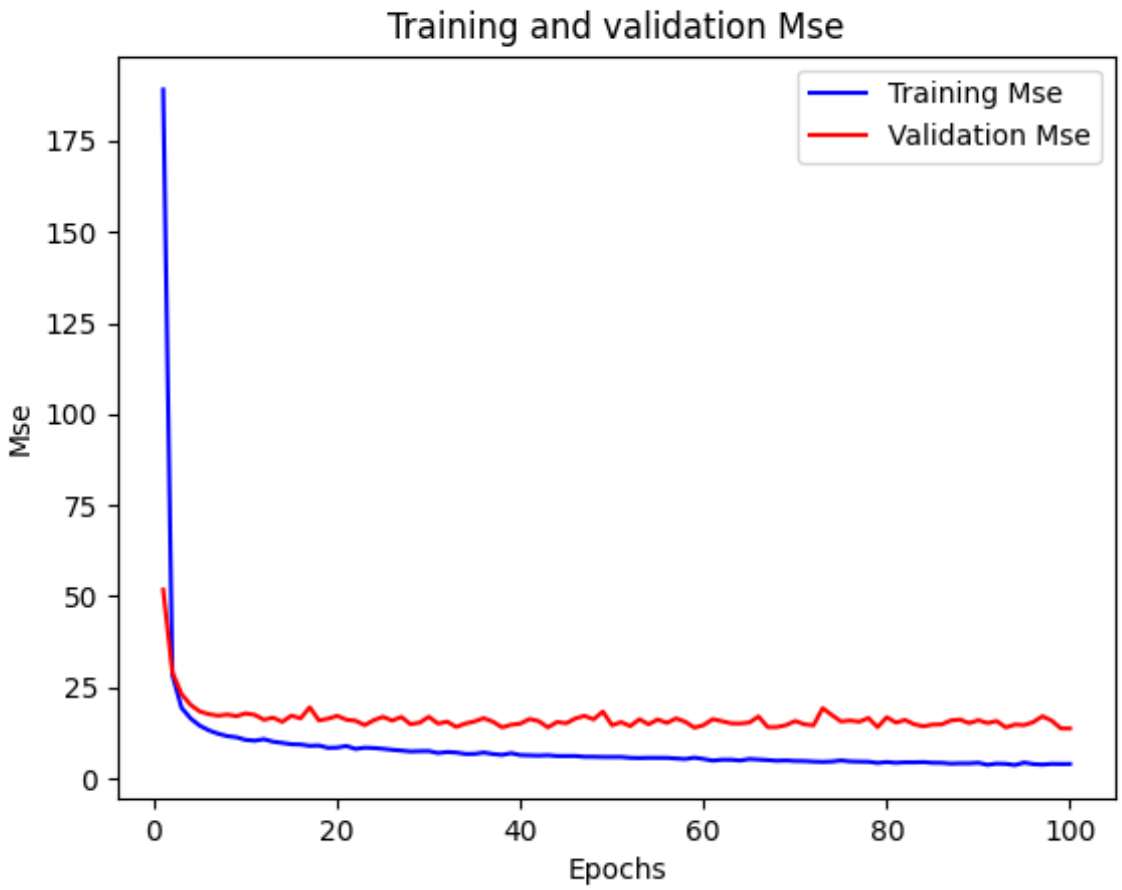

Рис. 8: Mse для модели 4

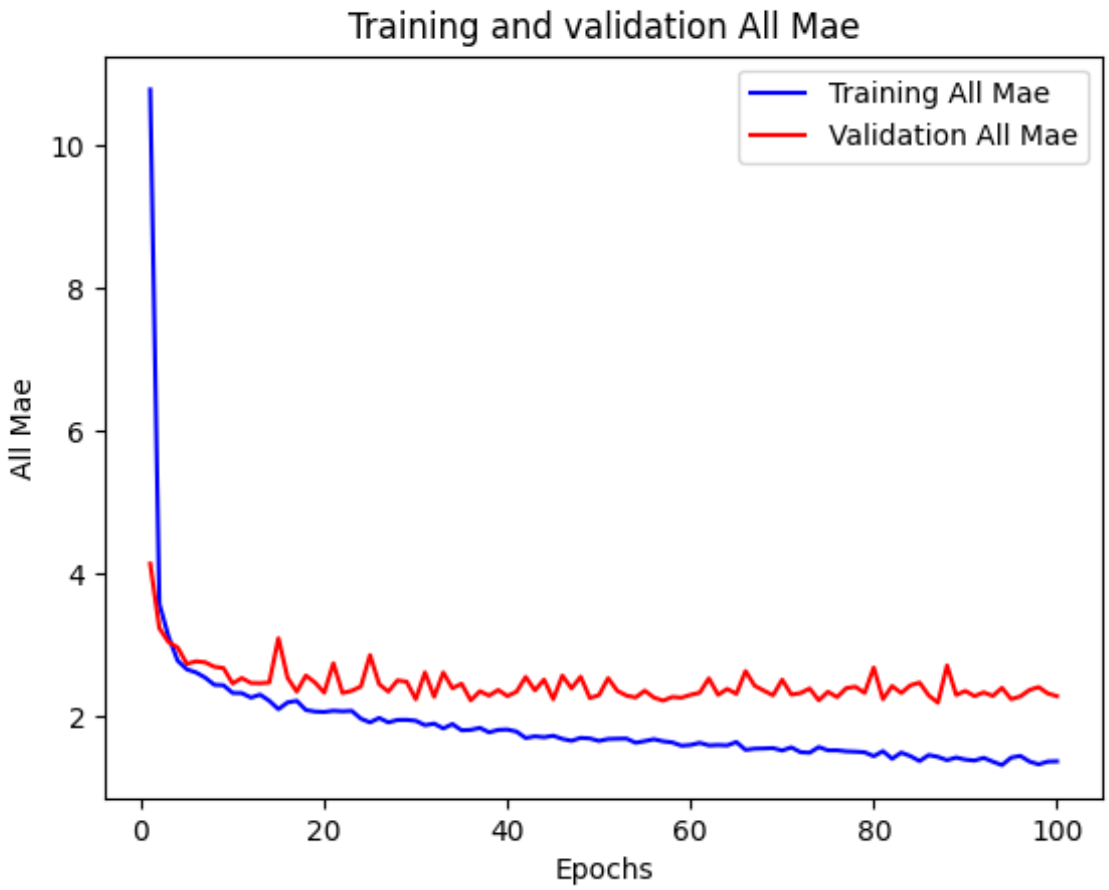

Рис. 9: Mae для модели в среднем

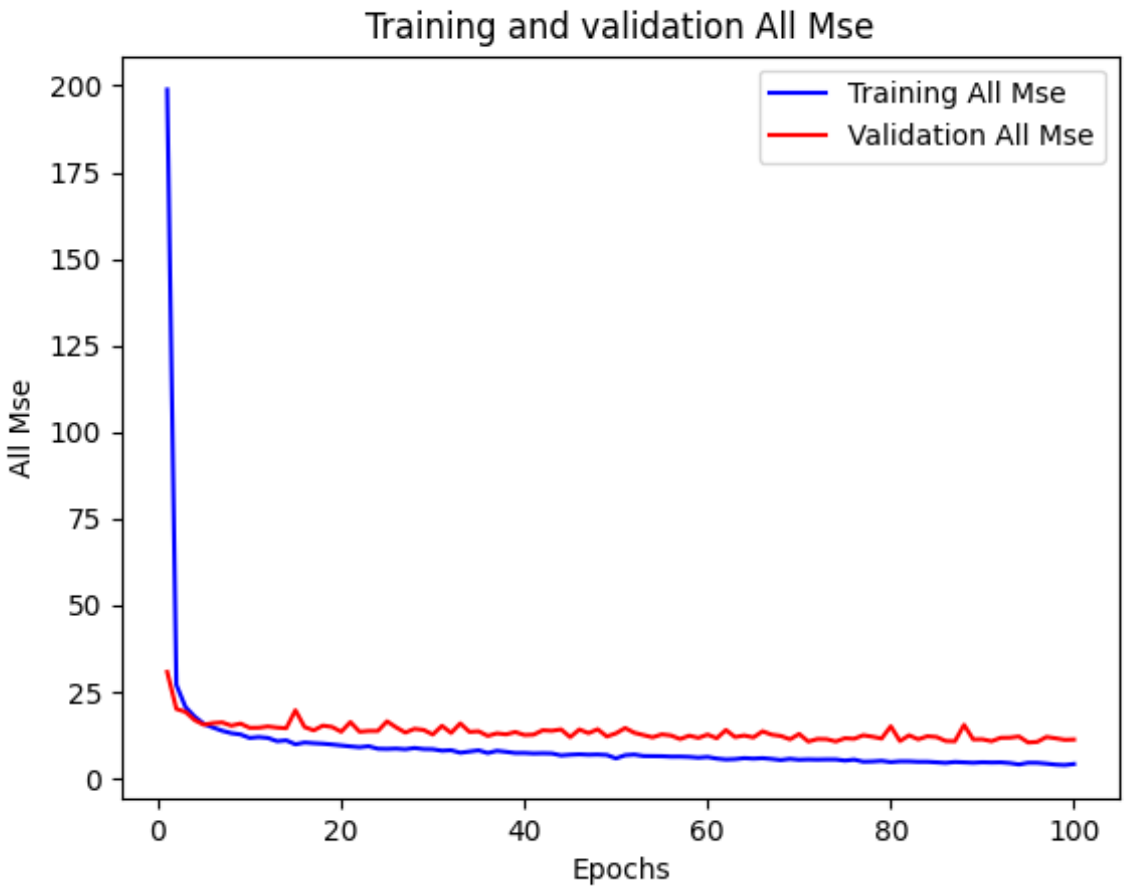

Рис. 10: Mse для модели в среднем

Параметры модели:

Количество эпох: 30

Количество блоков: 4

Результаты обучения для моделей представлены на рис.11-18. Усредненные результаты для всех моделей приведены на рис.19-20.

Средняя оценка: 2.84057746330897

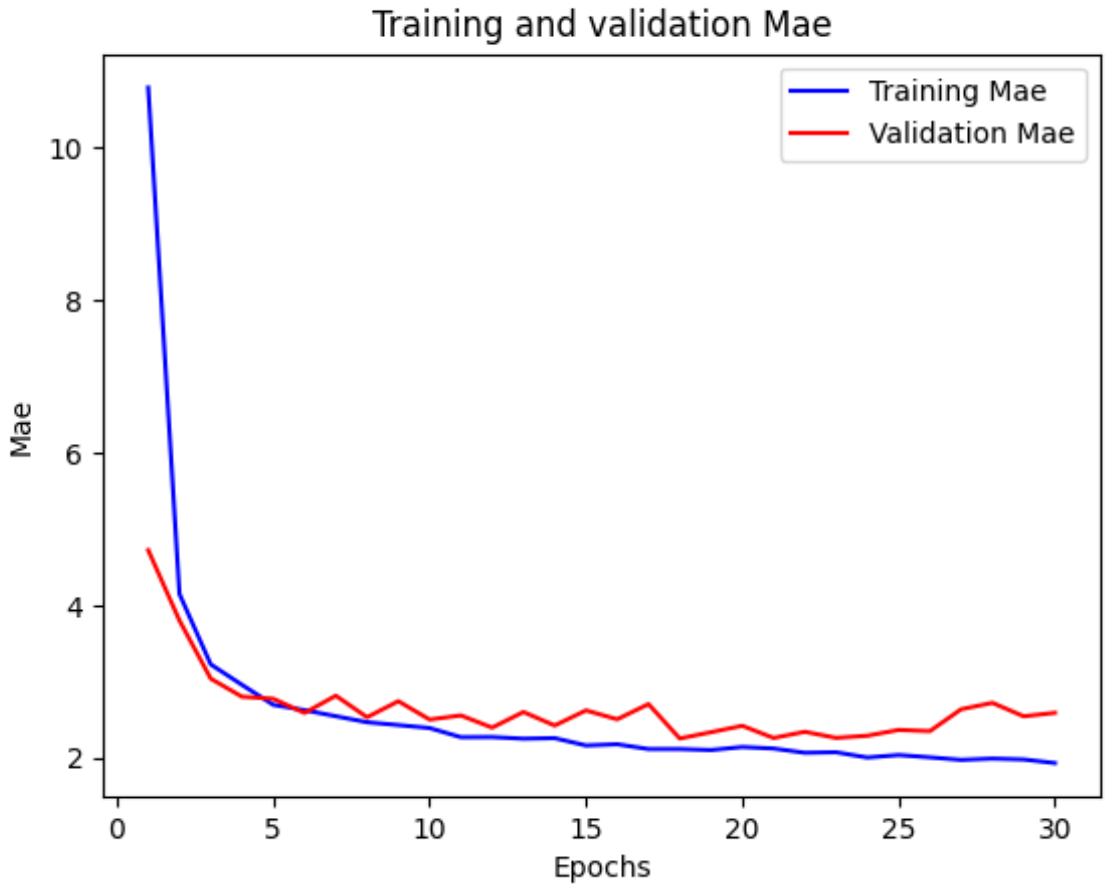

Рис. 11: Mae для модели 1

Значение mse ухудшилось не сильно, но количество эпох удалось сократить на 70.

По последнему графику видно, что, возможно, более удачное количество эпох - 26.

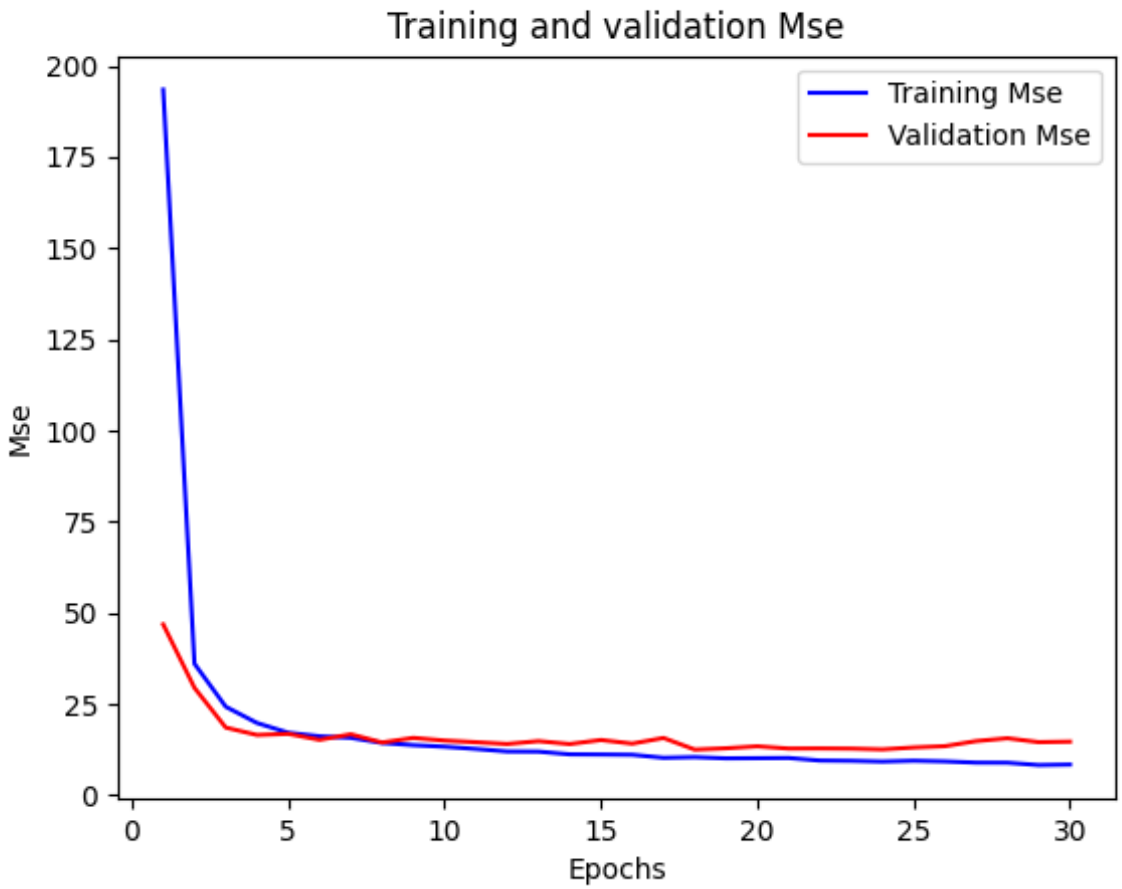

Рис. 12: Mse для модели 1

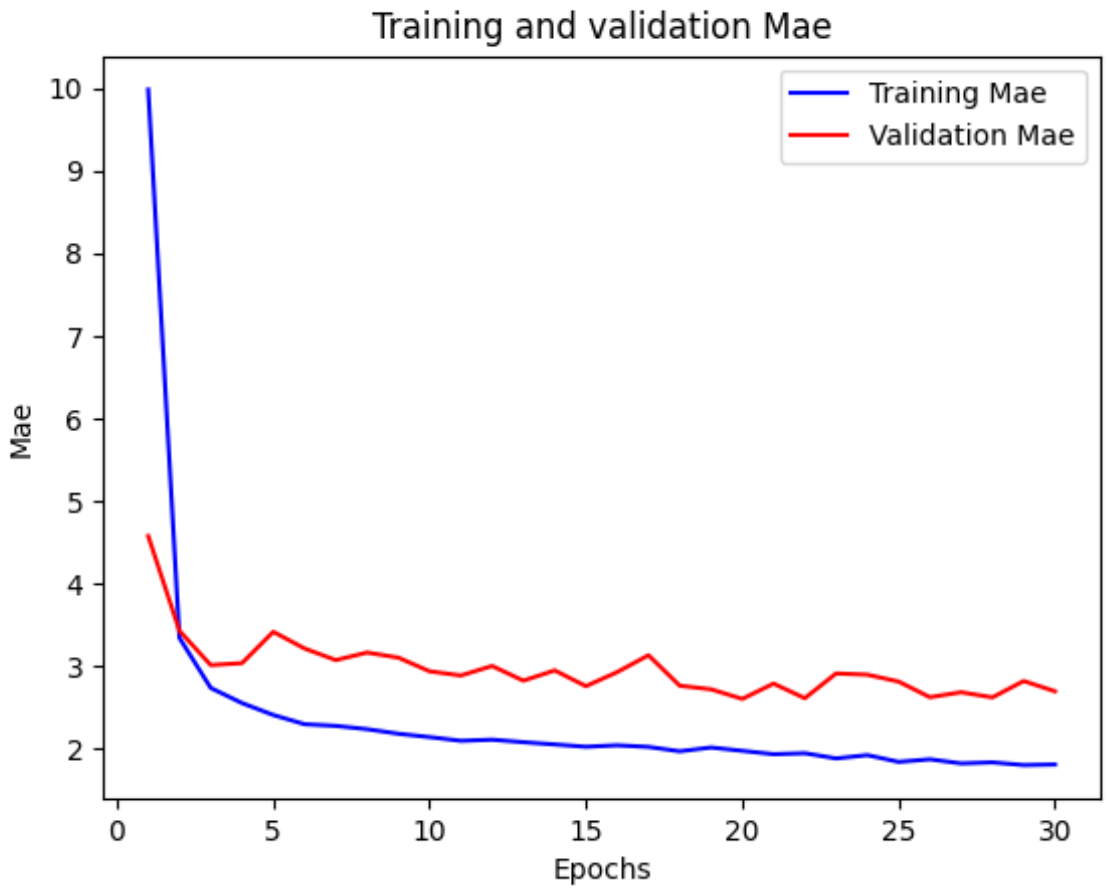

Рис. 13: Mae для модели 2

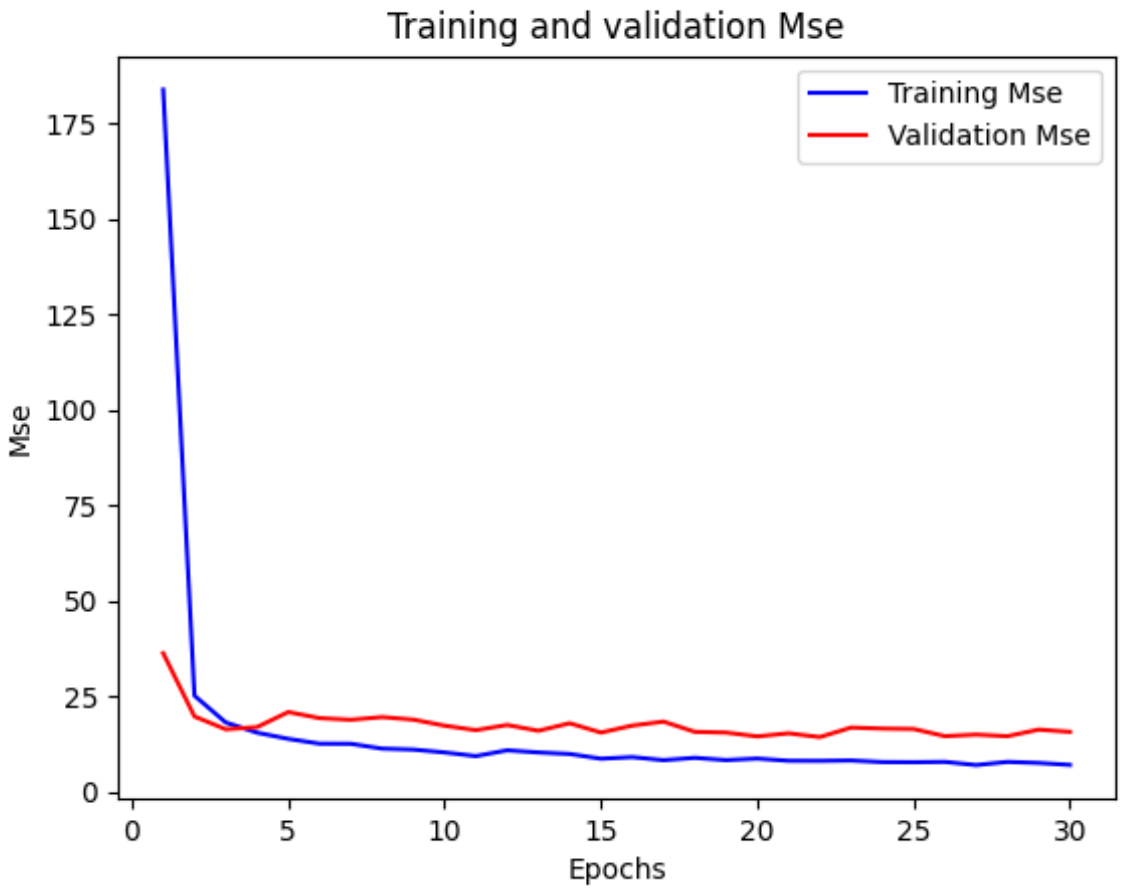

Рис. 14: Mse для модели 2

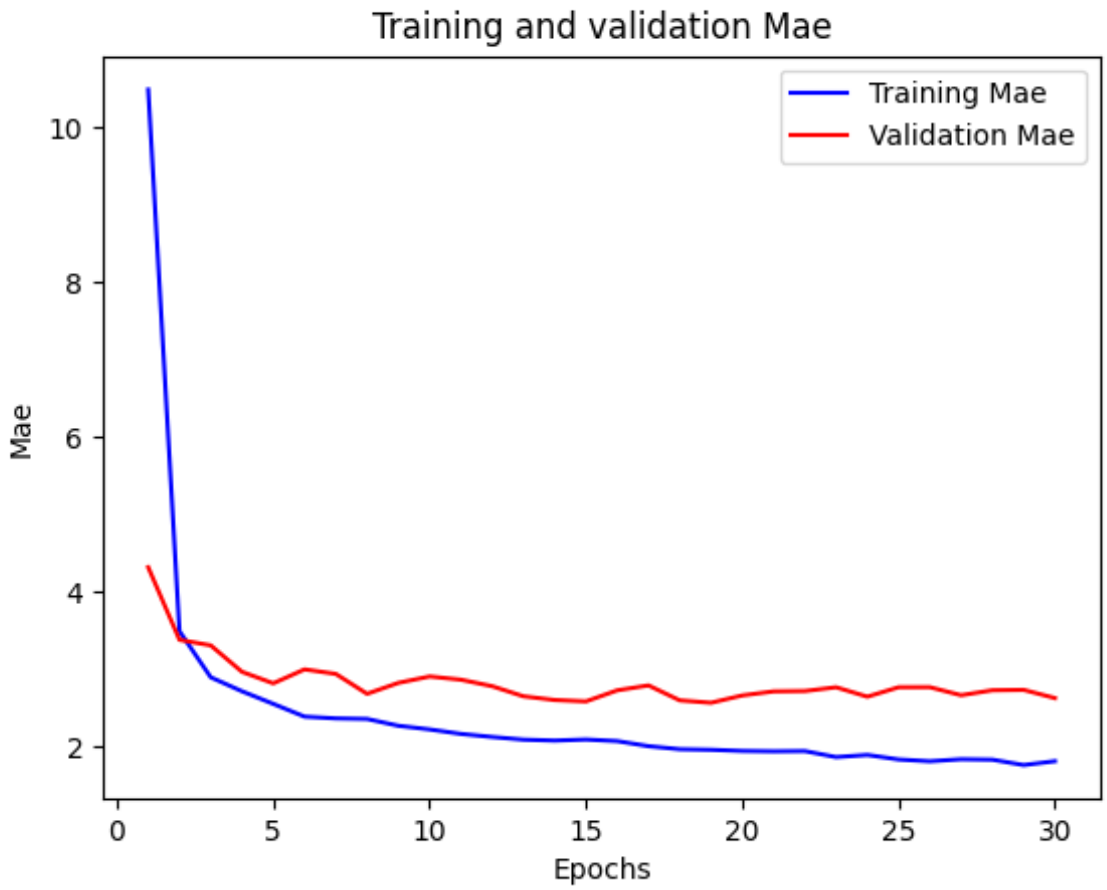

Рис. 15: Mae для модели 3

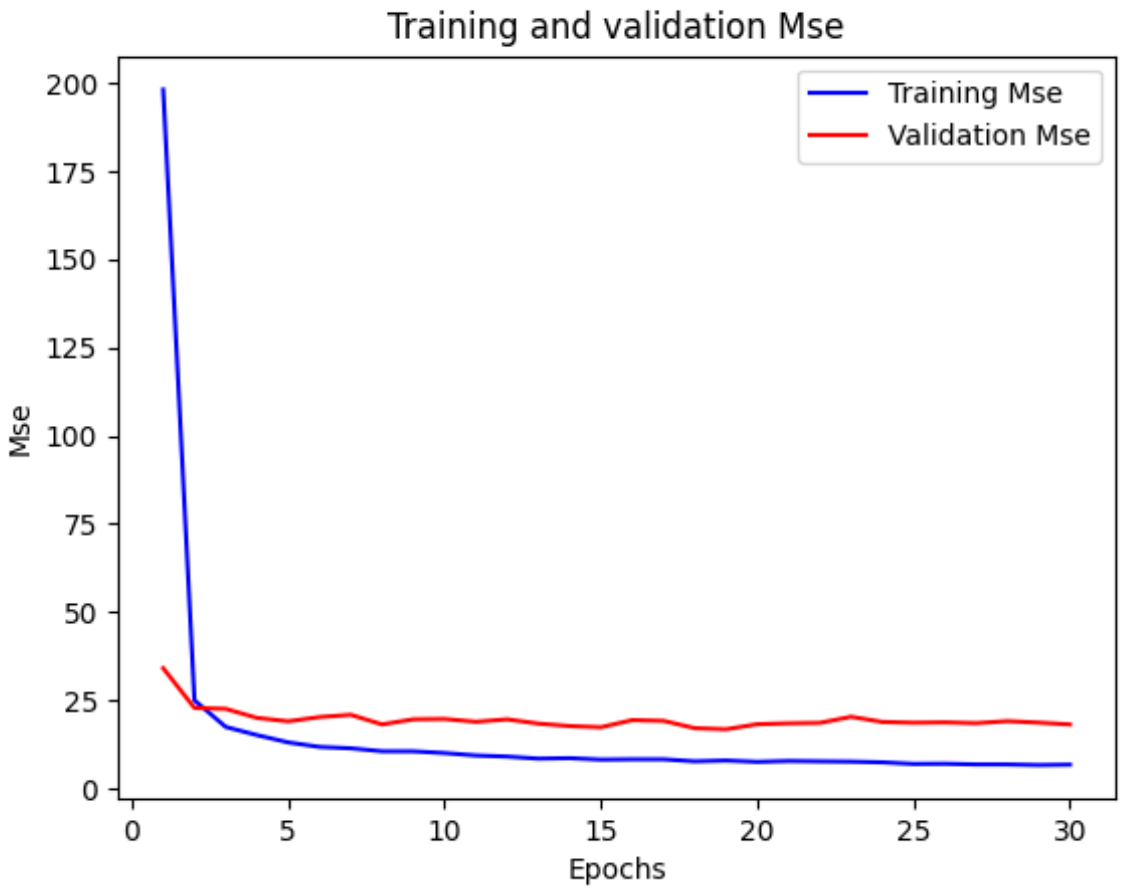

Рис. 16: Mse для модели 3

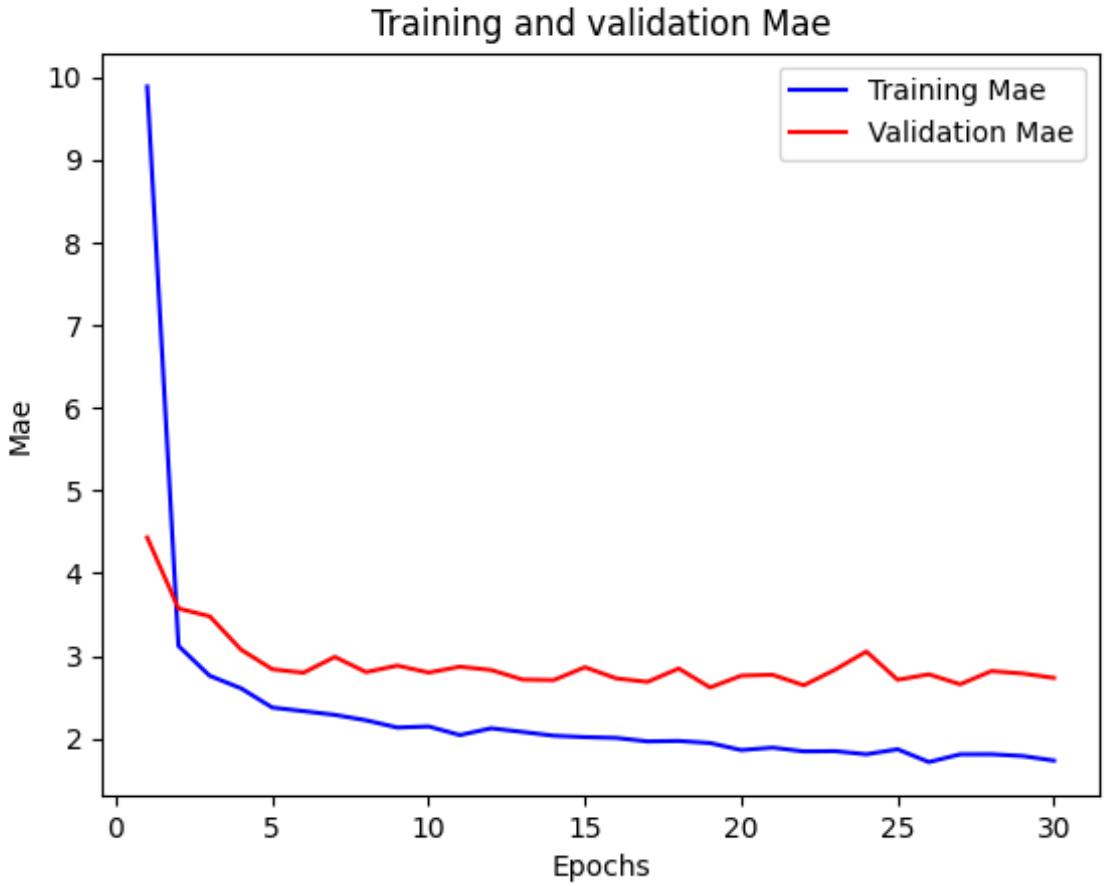

Рис. 17: Mae для модели 4

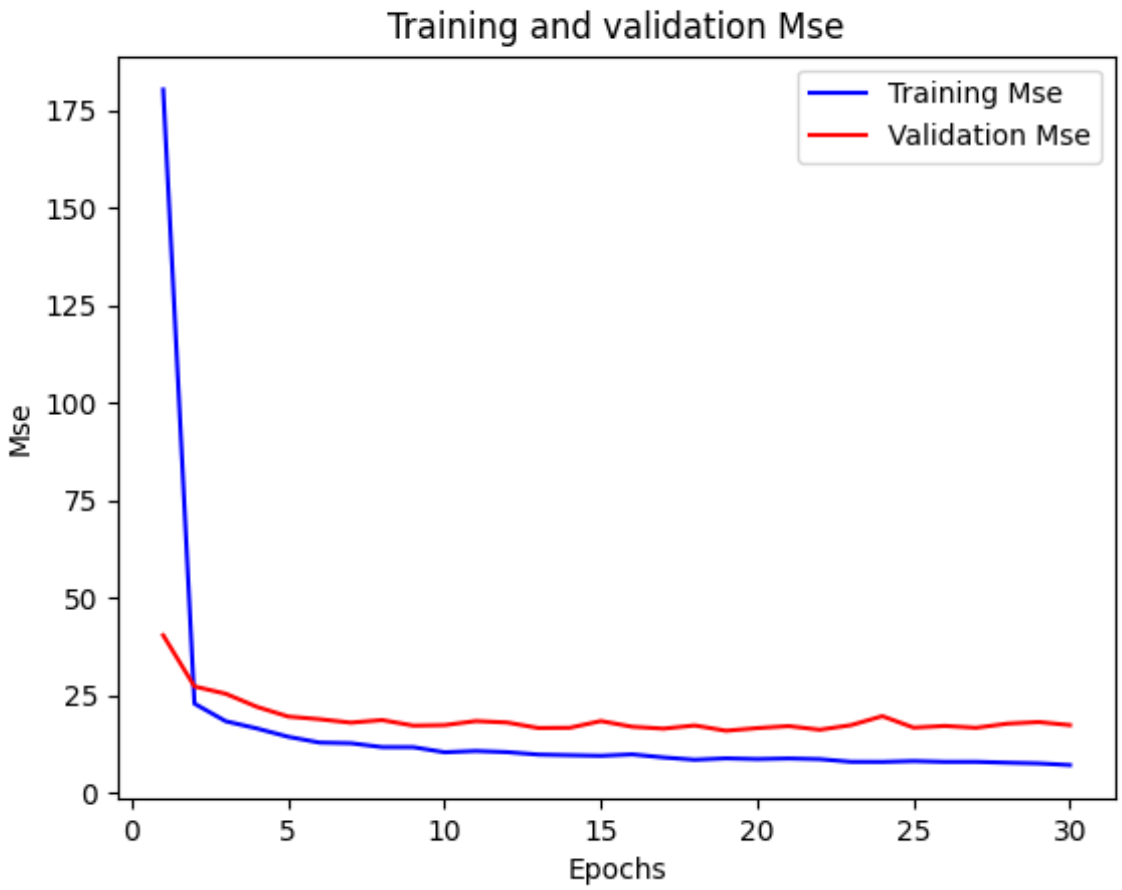

Рис. 18: Mse для модели 4

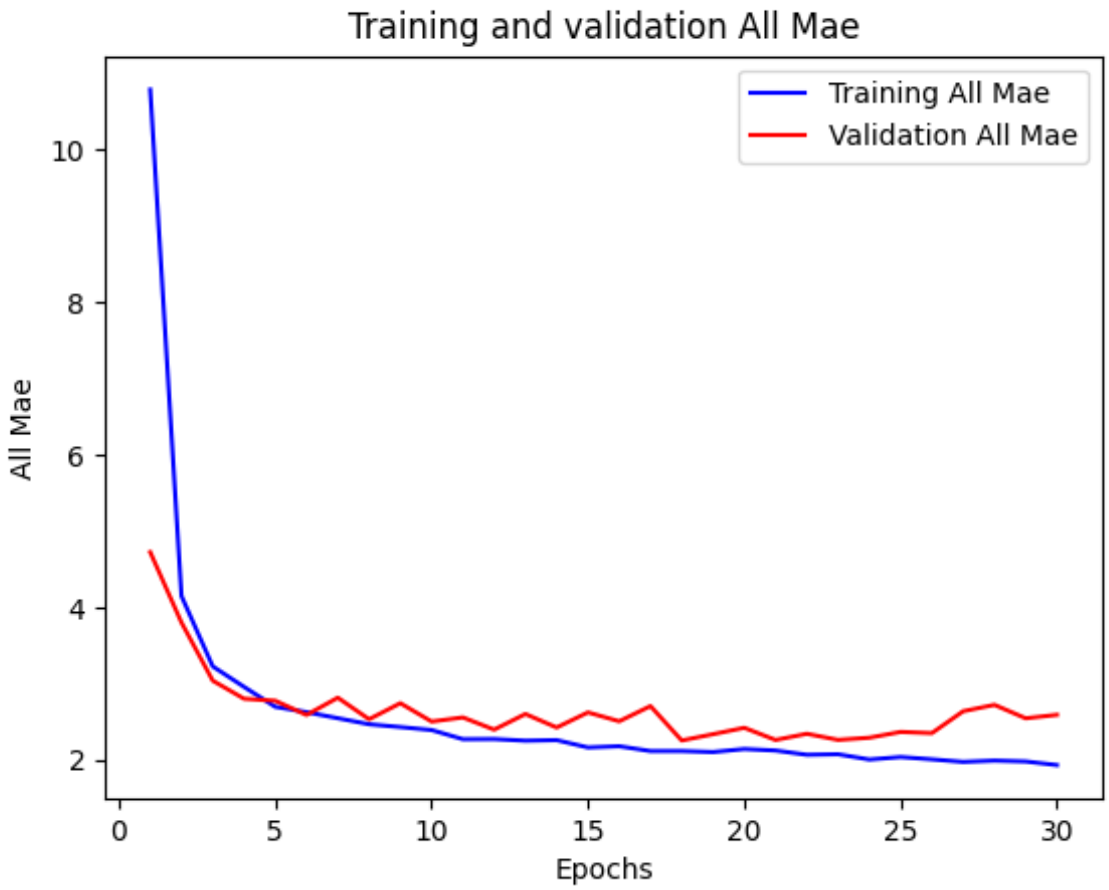

Рис. 19: Mae для модели в среднем

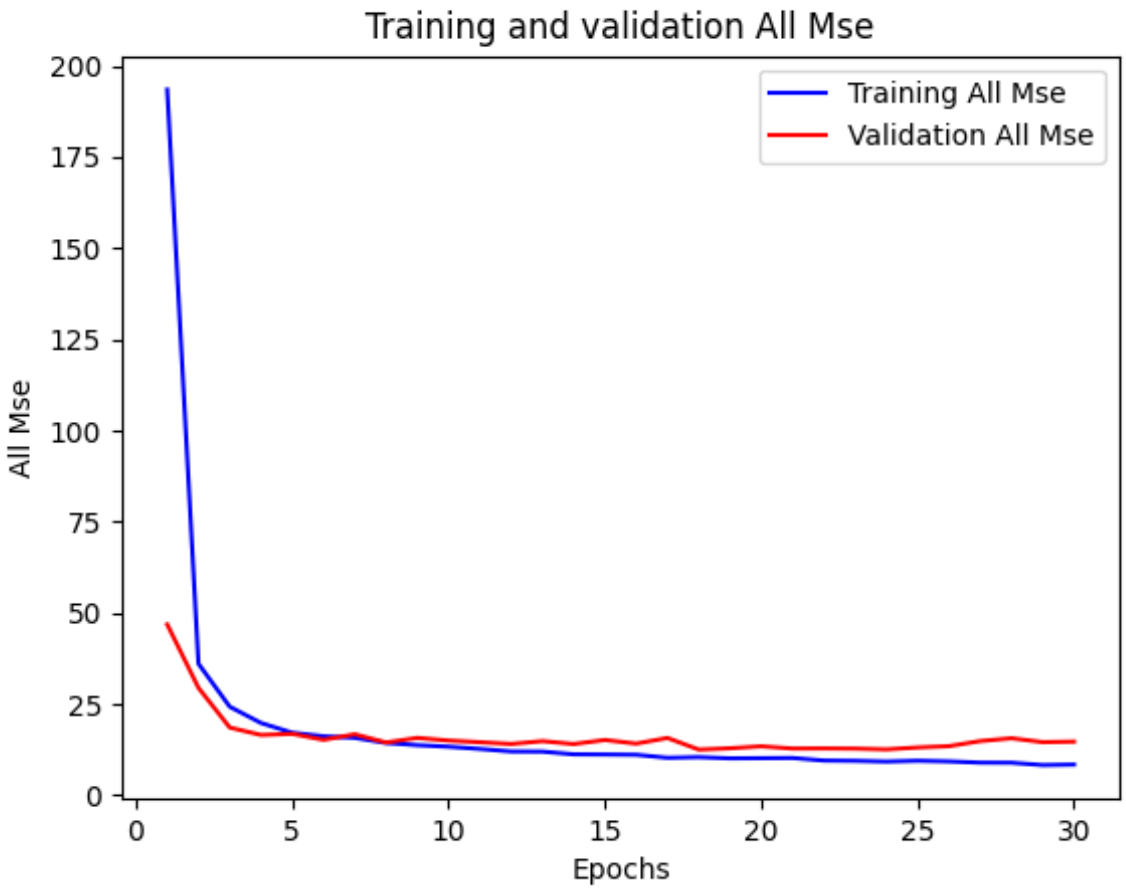

Рис. 20: Mse для модели в среднем

Изучение влияния количества блоков на результат обучения модели

Так как количество данных невелико, была применена кросс-валидация: разбиение обучающей выборки на блоки. Обучение производится на k-1 блоках и валидация на оставшемся. Разбиение по блокам меняется на каждой итерации. За конечный результат принимается усреднённое значение. Чтобы так же задействовать валидационные данные, они добавляются к каждому валидационному блоку.

Параметры модели:

Количество эпох: 30

Количество блоков: 5

Результаты обучения для моделей представлены на рис.21-30. Усредненные результаты для всех моделей приведены на рис.31.

Средняя оценка: 2.7992049090067543

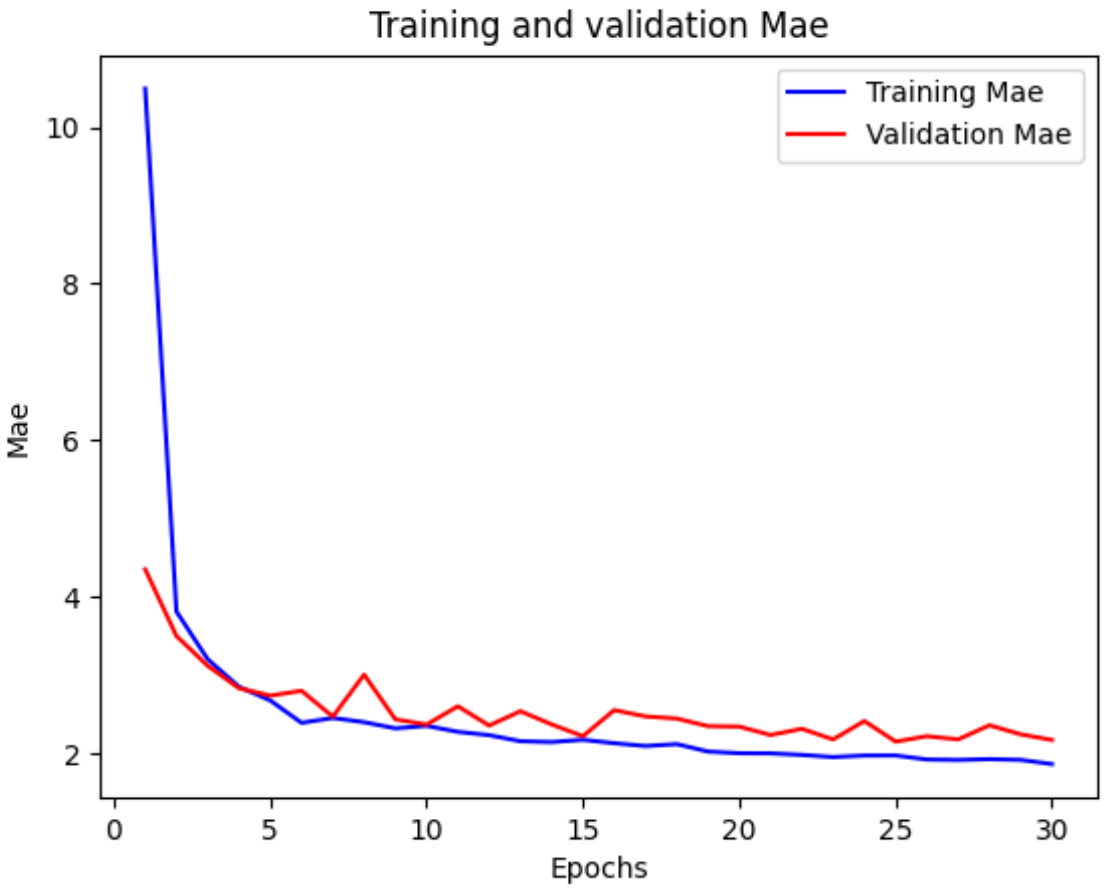

Рис. 21: Mae для модели 1

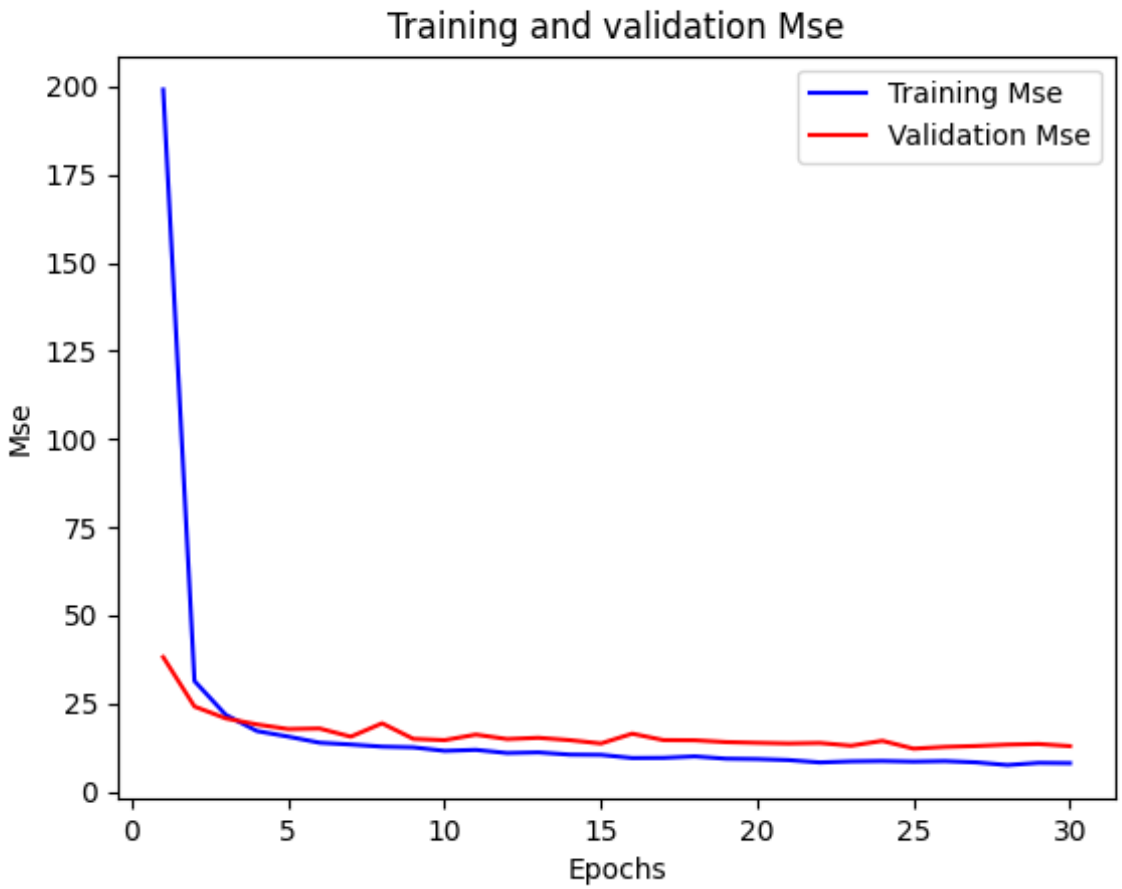

Рис. 22: Mse для модели 1

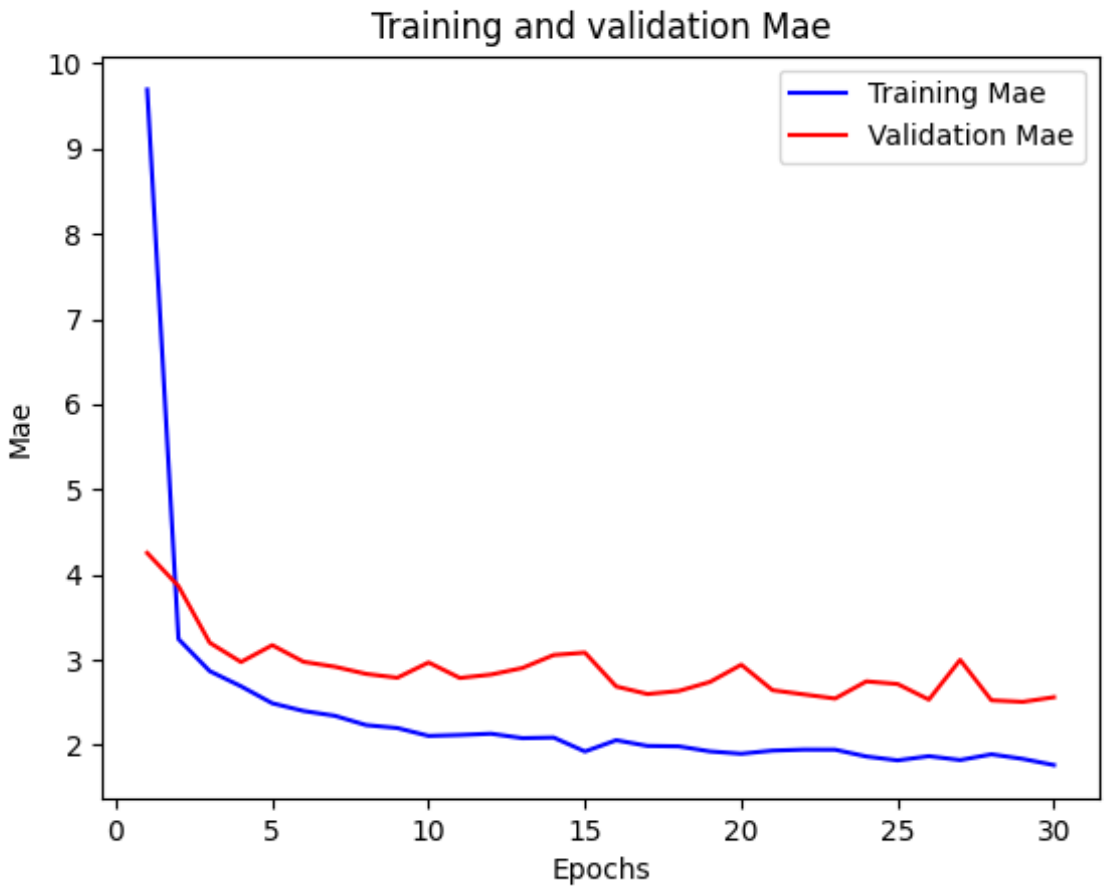

Рис. 23: Mae для модели 2

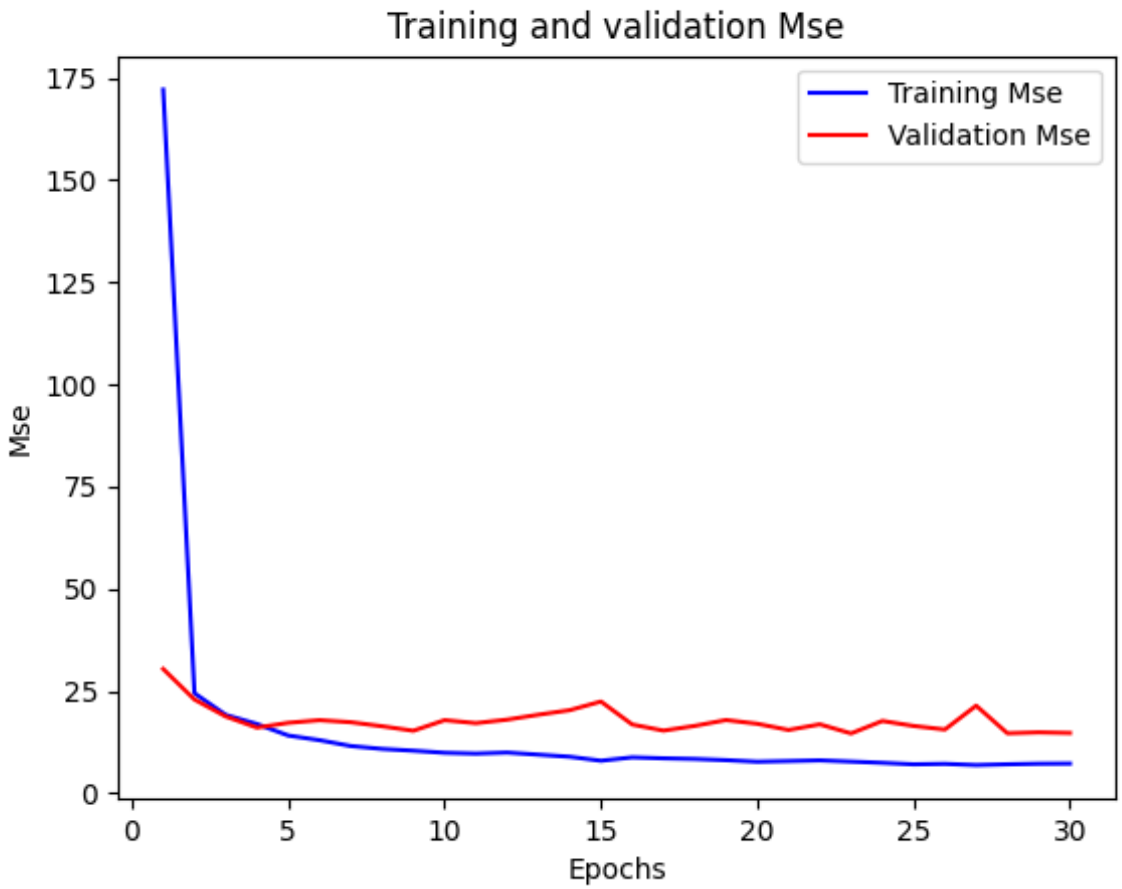

Рис. 24: Mse для модели 2

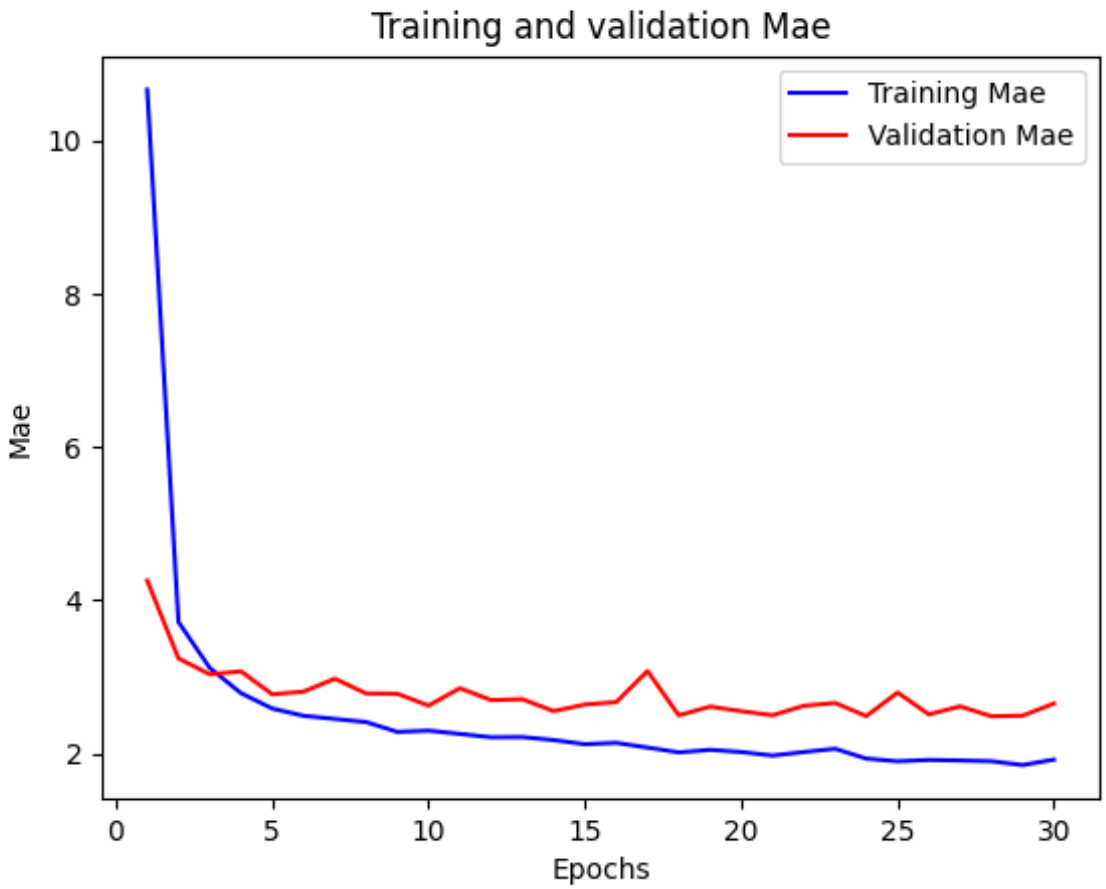

Рис. 25: Mae для модели 3

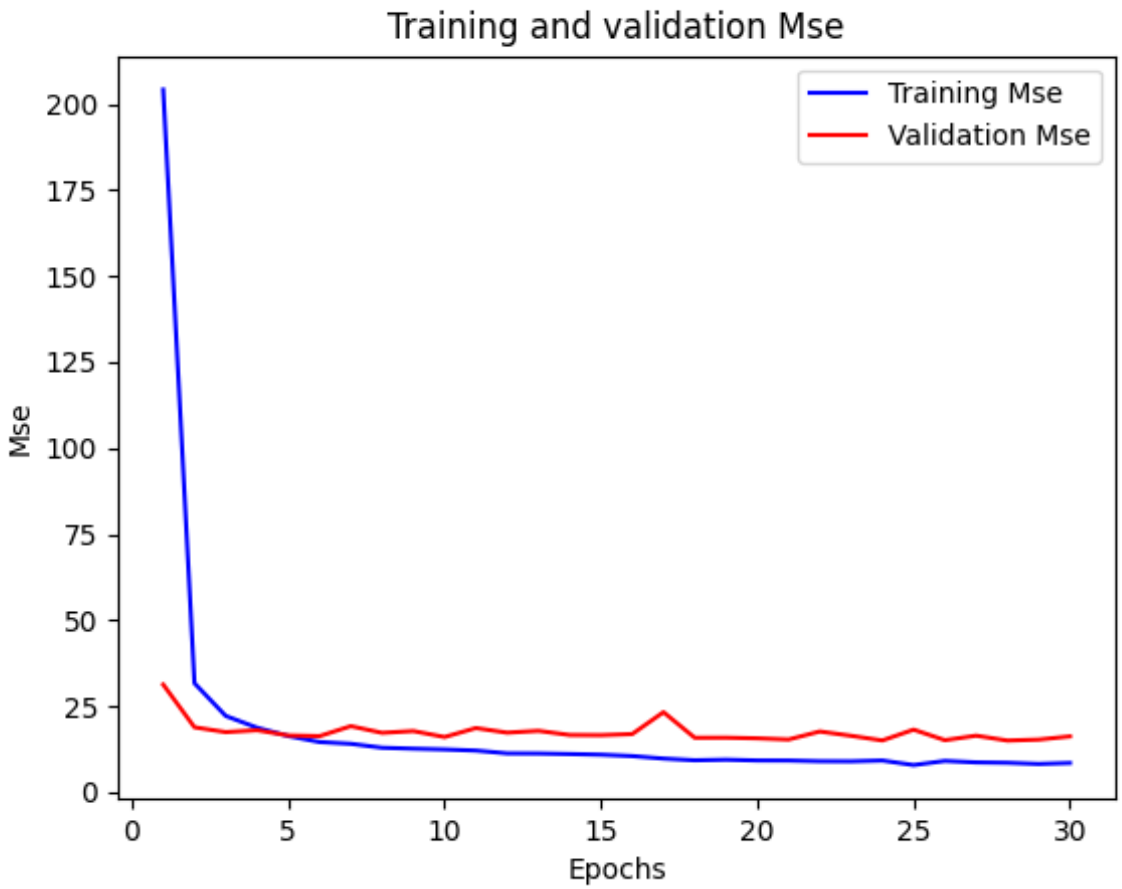

Рис. 26: Mse для модели 3

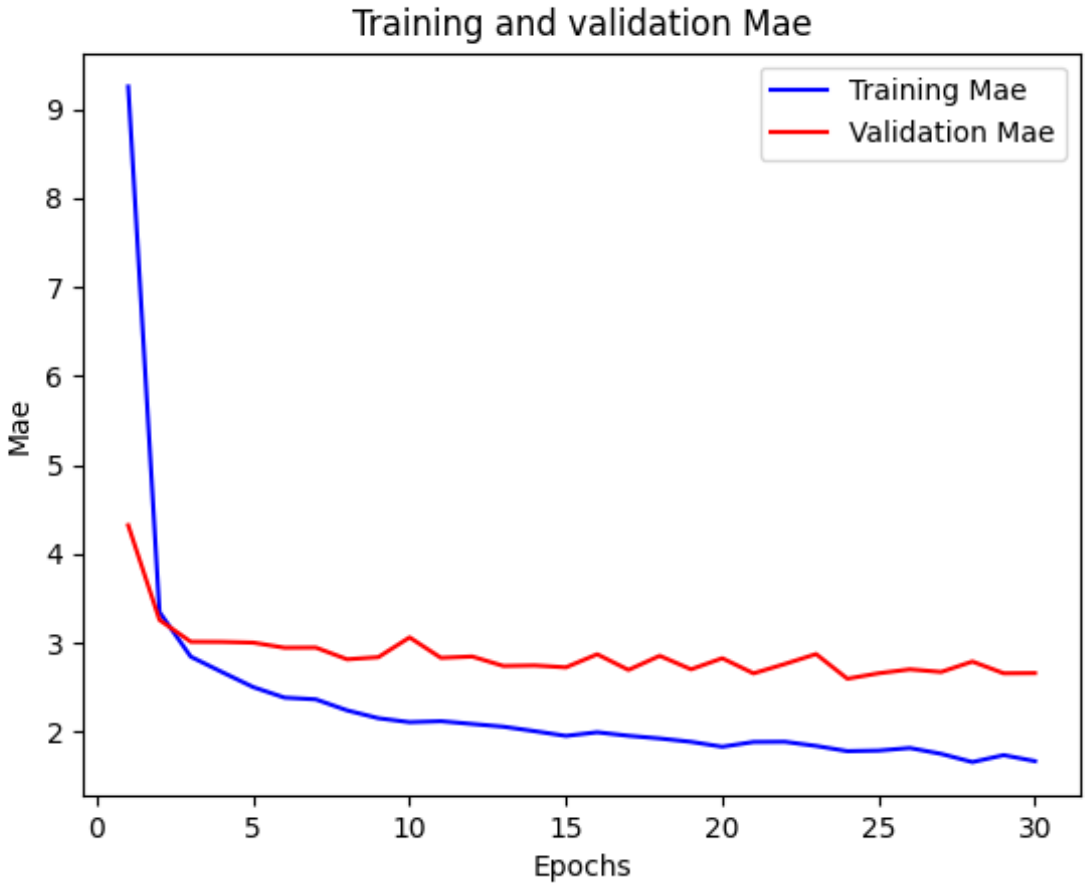

Рис. 27: Mae для модели 4

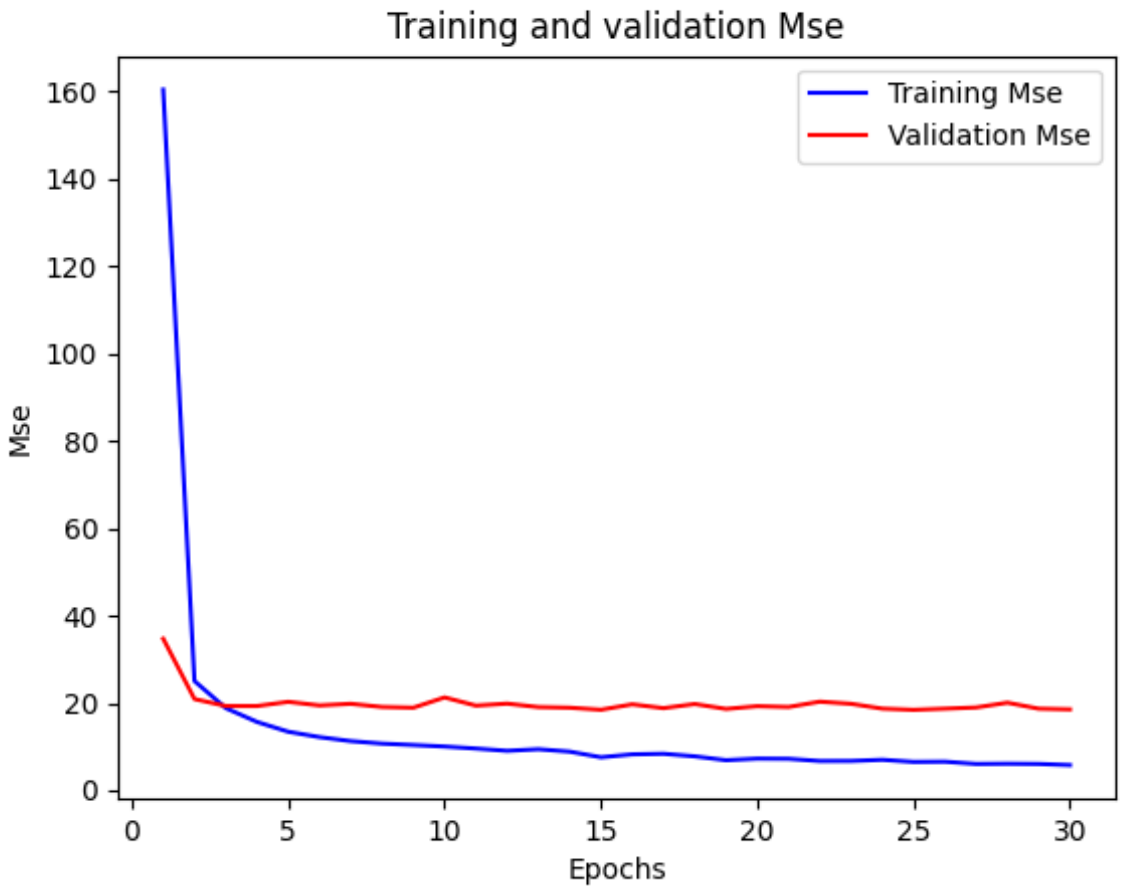

Рис. 28: Mse для модели 4

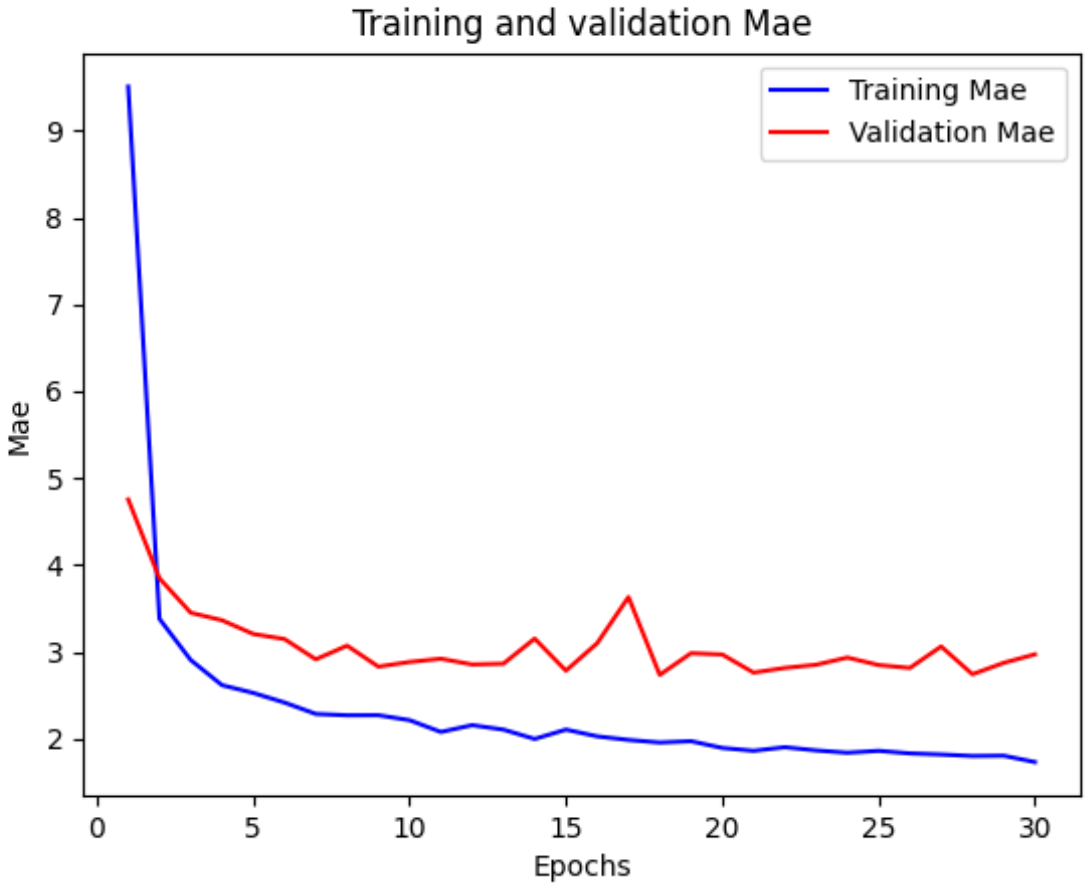

Рис. 29: Mae для модели 5

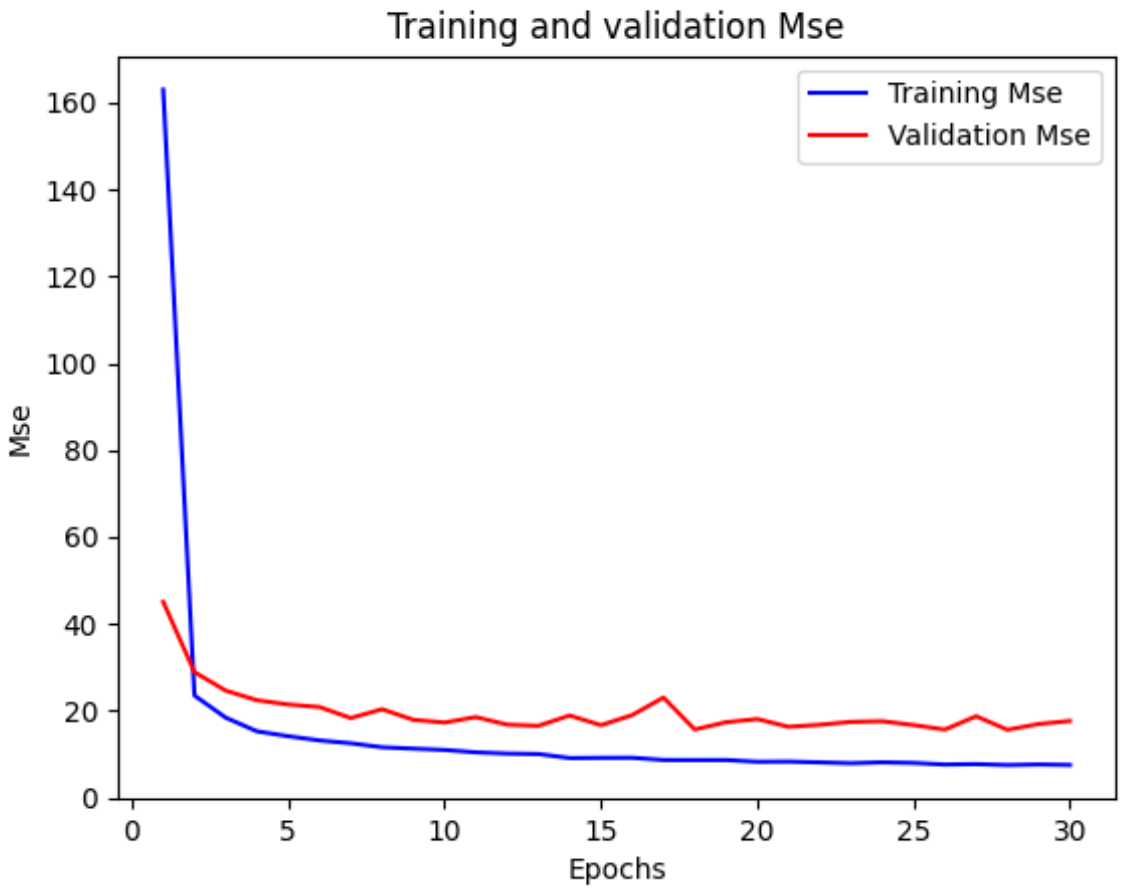

Рис. 30: Mse для модели 5

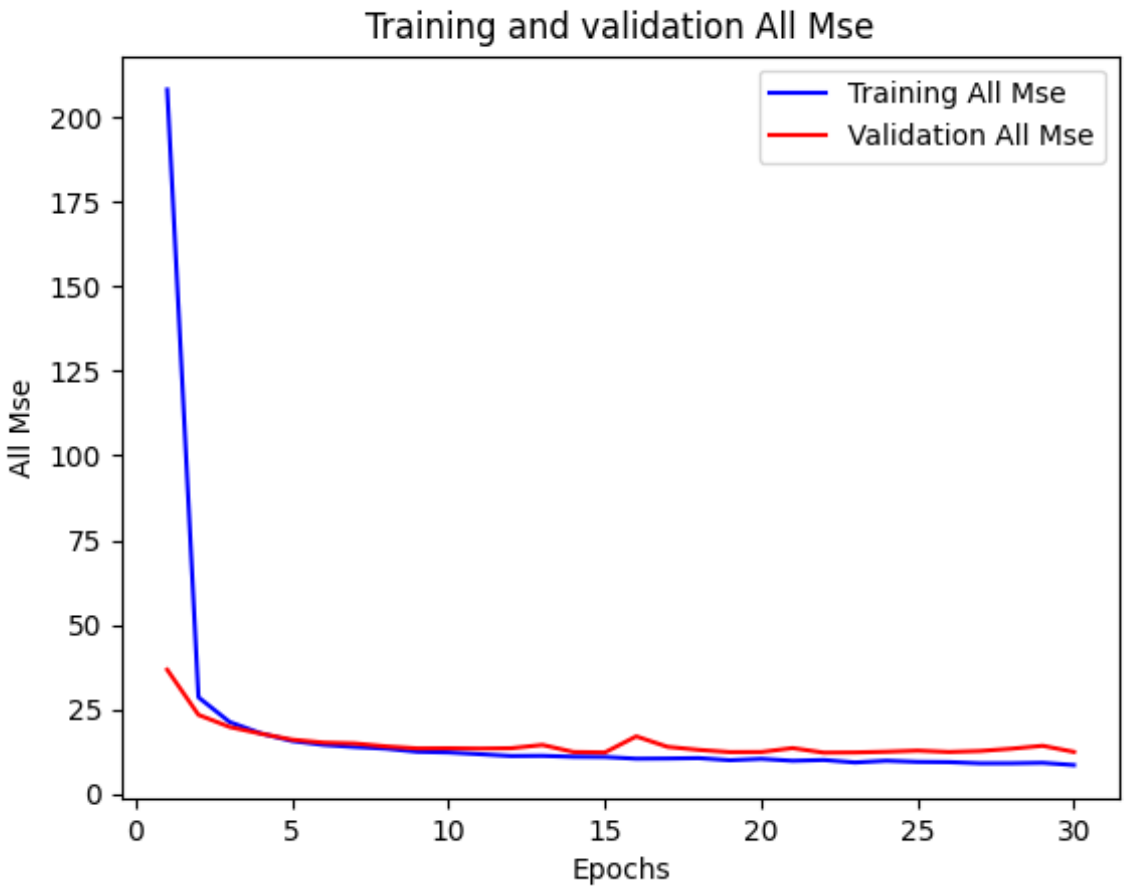

Рис. 31: Mse для модели в среднем

При увеличении количества блоков результат улучшился.

Вывод

В ходе выполнения лабораторной работы были получены практические навыки построения нейронных сетей для предсказания цен домов в Бостоне. Так же были получены навыки применения кросс-валидации.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД

```
from typing import List, Iterator, Tuple, Any

import matplotlib.pyplot as plt
import numpy as np
from tensorflow.keras import layers
from tensorflow.keras.datasets import boston_housing
from tensorflow.python.keras import models


def load_data() -> Tuple[Tuple[Any, Any], Tuple[Any, Any]]:
    (train_data, train_targets), (test_data, test_targets) = boston_housing.load_da

    mean = train_data.mean(axis=0)
    train_data -= mean

    std = train_data.std(axis=0)
    train_data /= std
    test_data -= mean
    test_data /= std

    return (train_data, train_targets), (test_data, test_targets)


def create_model(input_shape) -> models.Model:
    model = models.Sequential()
```

```
    model.add(layers.Dense(64, activation="relu", input_shape=(input_shape,)))
    model.add(layers.Dense(64, activation="relu"))
    model.add(layers.Dense(1))

    model.compile(optimizer="rmsprop", loss="mse",
                  metrics=["mae"])
    return model


def train_model(train_data: np.array, train_targets: np.array, validation_data: np.
                validation_targets: np.array, batch_size: int, epochs: int, k: int)
    num_val_samples = len(train_data) // k
    all_mse = []
    all_val_mse = []
    all_mae = []
    all_val_mae = []

    for i in range(k):
        print("processing fold #", i)
        val_data = np.concatenate([train_data[i * num_val_samples: (i + 1) * num_va
        val_targets = np.concatenate(
            [train_targets[i * num_val_samples: (i + 1) * num_val_samples], validat
        partial_train_data = np.concatenate([train_data[:i * num_val_samples], trai
                                             axis=0)
        partial_train_targets = np.concatenate(
            [train_targets[:i * num_val_samples], train_targets[(i + 1) * num_val_s
        model = create_model(train_data.shape[1])
        history = model.fit(partial_train_data, partial_train_targets, validation_d
                            epochs=epochs,
                            batch_size=batch_size, verbose=0).history
        mae = history["mae"]
        val_mae = history["val_mae"]
        mse = history["loss"]
```

```
            val_mse = history["val_loss"]
            draw_plot("Mae", range(1, epochs + 1), mae, val_mae)
            draw_plot("Mse", range(1, epochs + 1), mse, val_mse)
            all_mse.extend(mse)
            all_val_mse.extend(val_mse)
            all_mae.extend(mae)
            all_val_mae.extend(val_mae)

    return all_mse, all_val_mse, all_mae, all_val_mae


def draw_plot(data_type: str, epochs: Iterator[int], train_data_value: List[int],
              test_data_value: List[int]):
    plt.plot(epochs, train_data_value, "b", label=f"Training {data_type}")
    plt.plot(epochs, test_data_value, "r", label=f"Validation {data_type}")
    plt.title(f"Training and validation {data_type}")
    plt.xlabel("Epochs")
    plt.ylabel(f"{data_type}")
    plt.legend()
    plt.show()


def main():
    (train_data, train_targets), (test_data, test_targets) = load_data()
    k = 5
    epochs = 30

    all_mse, all_val_mse, all_mae, all_val_mae = train_model(train_data, train_targ

    draw_plot("All Mse", range(1, epochs + 1), [np.mean(all_mse[i]) for i in range(
              [np.mean(all_val_mse[i]) for i in range(epochs)])
    draw_plot("All Mae", range(1, epochs + 1), [np.mean(all_mae[i]) for i in range(
              [np.mean(all_val_mae[i]) for i in range(epochs)])
```

```
    print(np.mean(all_val_mae))


if __name__ == "__main__":
    main()
```