

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЁТ
по лабораторной работе №3
по дисциплине «Искусственные нейронные сети»
Тема: Регрессионная модель изменения цен на дома в Бостона

Студент гр.8382

Фильцин И.В.

Преподаватель

Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы

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.

Задание

Ознакомиться с задачей регрессии

Изучить отличие задачи регрессии от задачи классификации

Создать модель

Настроить параметры обучения

Обучить и оценить модели

Ознакомиться с перекрестной проверкой

Ход работы

Исходная архитектура сети остается одной и той же на протяжении всей работы. Входной слой с 64 нейронами и функцией активации `relu`, скрытый слой с 64 нейронами и функцией активации `relu`, выходной слой без функции активации для скалярной регрессии.

Сеть компилируется с функцией потерь `mse` - mean squared error, вычисляющий квадрат разности между предсказанными и целевыми значениями. Также добавлен параметр на этапе обучения: `mae` — mean absolute error (средняя абсолютная ошибка). Это абсолютное значение разности между предсказанными и целевыми значениями.

Чтобы оценить качество сети в ходе корректировки ее параметров, можно разбить исходные данные на обучающий и проверочный наборы, как это делалось в предыдущих примерах. Однако так как у нас и без того небольшой набор данных, проверочный набор получился бы слишком маленьким. Как следствие, оценки при проверке могут сильно меняться в зависимости от того, какие данные попадут в проверочный и обучающий наборы: оценки при проверке могут иметь слишком большой разброс. Это не позволит надежно оценить качество модели.

Хорошей практикой в таких ситуациях является применение перекрестной проверки по K блокам (K-fold cross-validation). Суть ее заключается в разделении доступных данных на K блоков (обычно $K = 4$ или 5), создании K идентичных моделей и обучении каждой на $K-1$ блоках с оценкой по оставшимся блокам. По полученным K оценкам вычисляется среднее значение, которое принимается как оценка модели.

Проведем перекрестную проверку с $K = 4$ на 100 эпохах. Результаты на рис. 1, рис. 2, рис. 3, рис. 4, рис. 5.

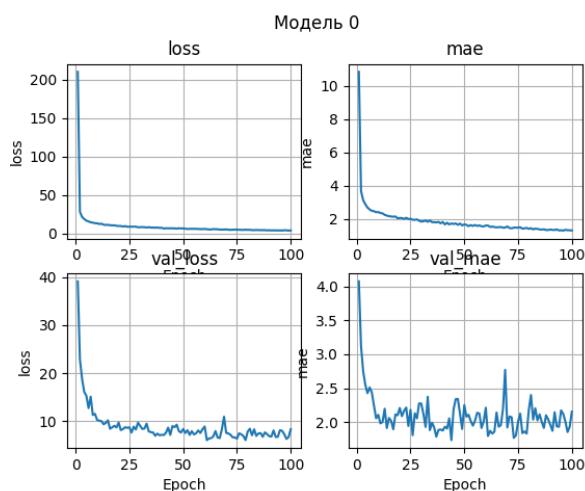


Рисунок 1 – "Модель 0 ($K = 4$, $n = 100$)"

Средняя разница между реальной и предсказанной ценой составляет 2504\$.

Попробуем сократить количество эпох, т.к. на графиках видно, что при-

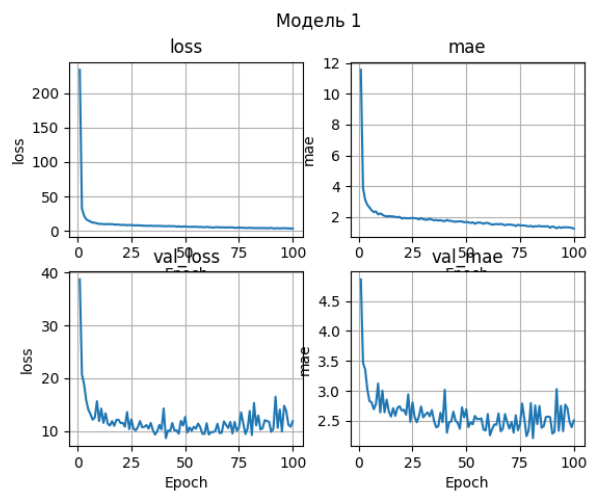


Рисунок 2 – "Модель 1 ($K = 4$, $n = 100$)"

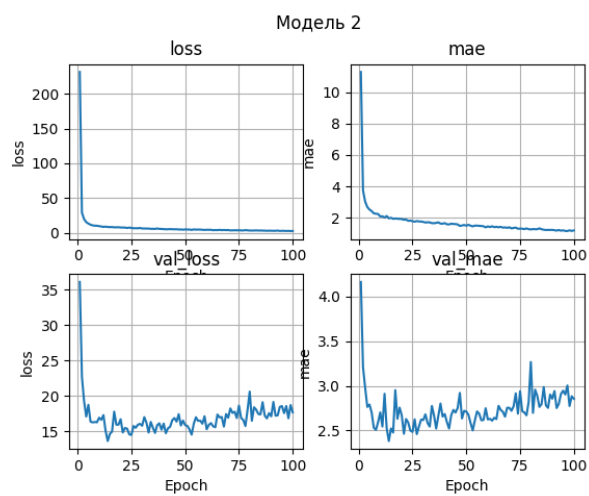


Рисунок 3 – "Модель 2 ($K = 4$, $n = 100$)"

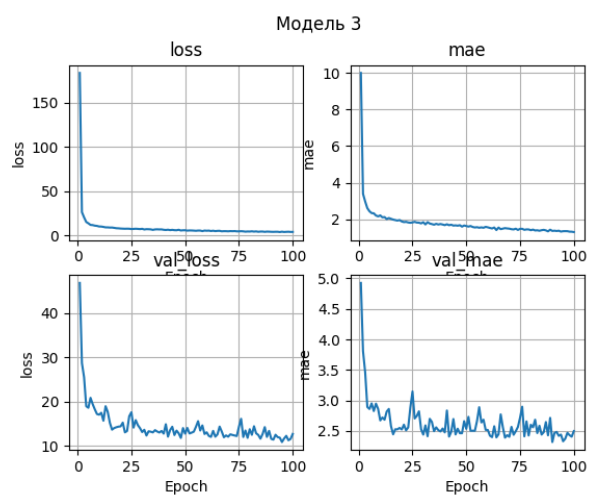


Рисунок 4 – "Модель 3 ($K = 4$, $n = 100$)"

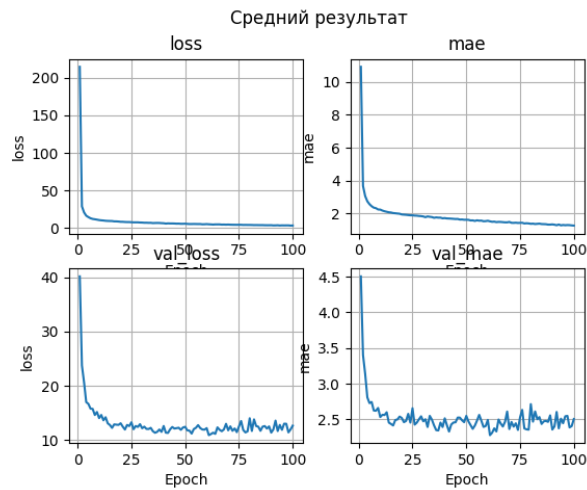


Рисунок 5 – "Среднее ($K = 4$, $n = 100$)"

мерно после 55 эпохи средняя ошибка увеличивается, что может свидетельствовать о переобучении модели. Результаты перекрестной проверки на рис. 6, рис. 7, рис. 8, рис. 9, рис. 10.

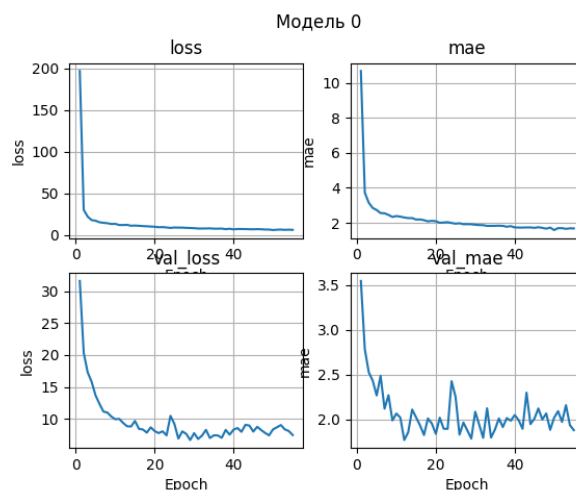


Рисунок 6 – "Модель 0 ($K = 4$, $n = 55$)"

Средняя разница между реальной и предсказанной ценой составляет 2300\$.

Попробуем увеличить количество K до 5. Результаты перекрестной проверки на рис. 11, рис. 12, рис. 13, рис. 14, рис. 15, рис. 16.

Средняя разница между реальной и предсказанной ценой составляет 2320\$, что немного хуже предыдущего результата.

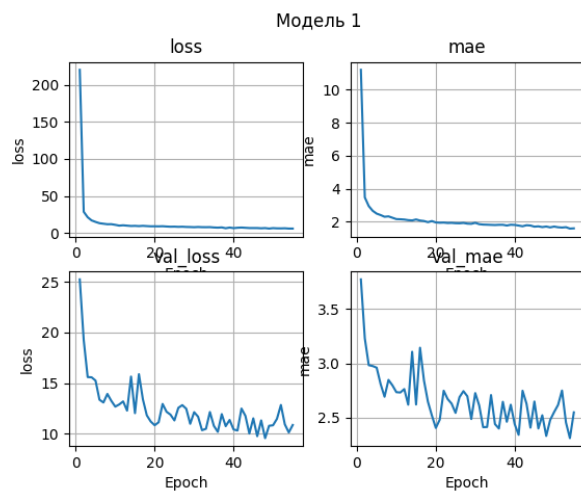


Рисунок 7 – "Модель 1 ($K = 4$, $n = 55$)"

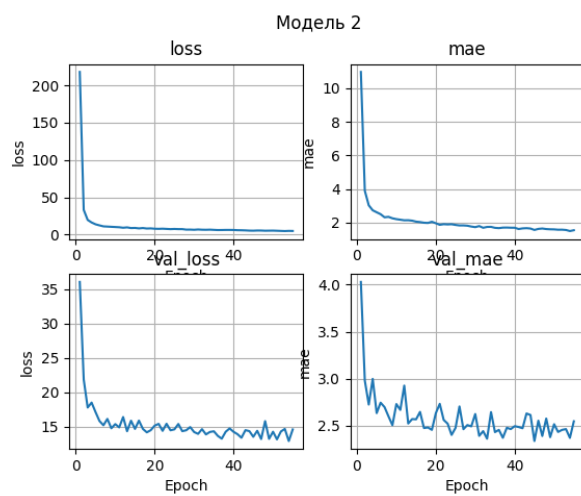


Рисунок 8 – "Модель 2 ($K = 4$, $n = 55$)"

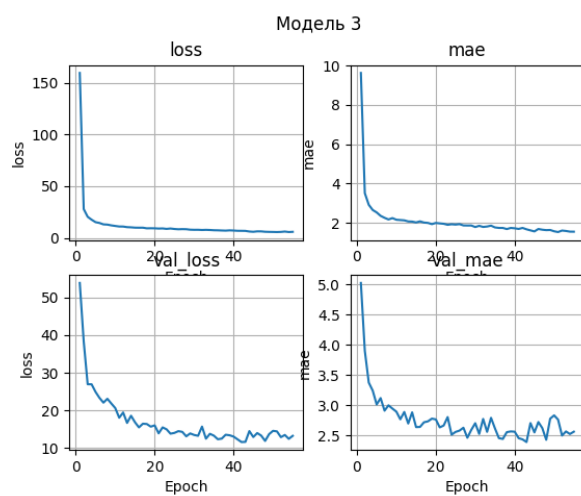


Рисунок 9 – "Модель 3 ($K = 4$, $n = 55$)"

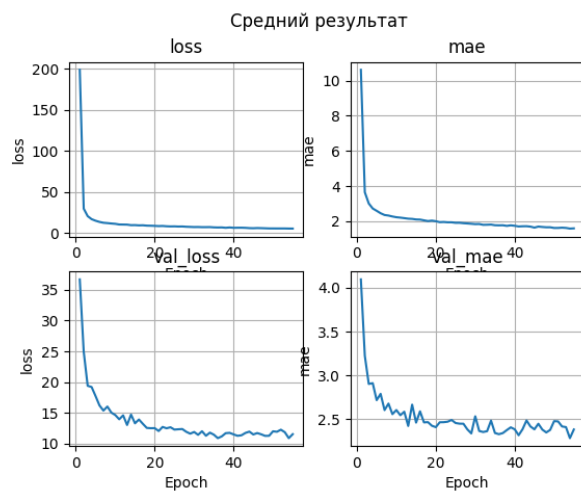


Рисунок 10 – "Среднее ($K = 4$, $n = 55$)"

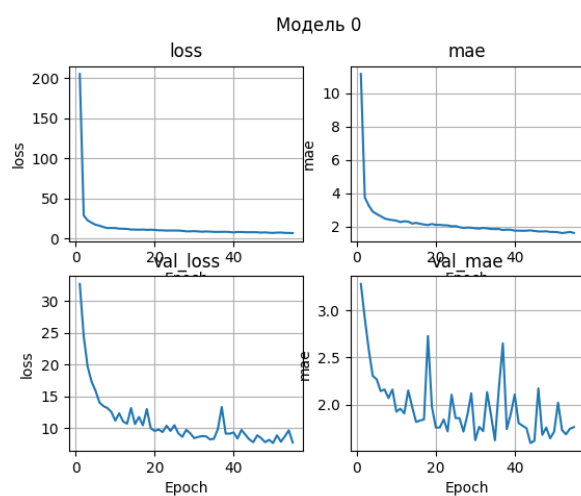


Рисунок 11 – "Модель 0 ($K = 5$, $n = 55$)"

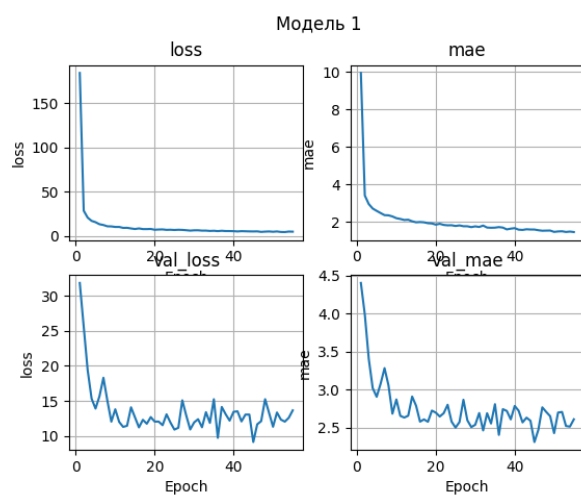


Рисунок 12 – "Модель 1 ($K = 5$, $n = 55$)"

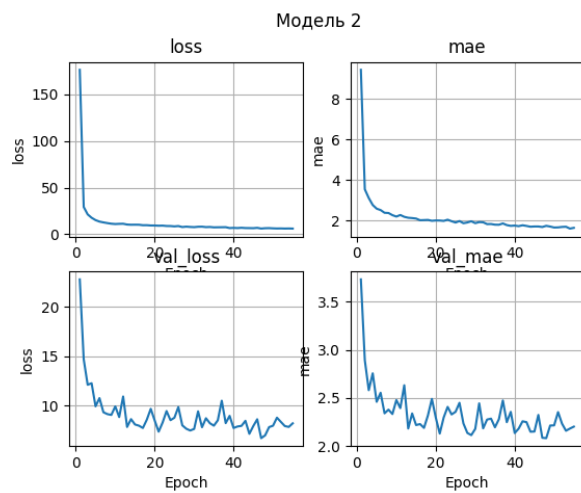


Рисунок 13 – "Модель 2 ($K = 5$, $n = 55$)"

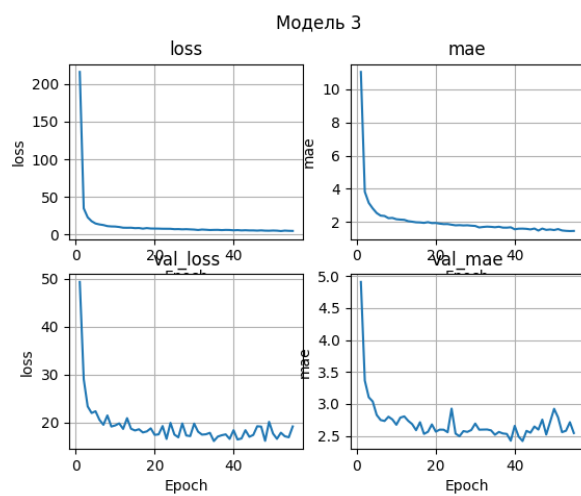


Рисунок 14 – "Модель 3 ($K = 5$, $n = 55$)"

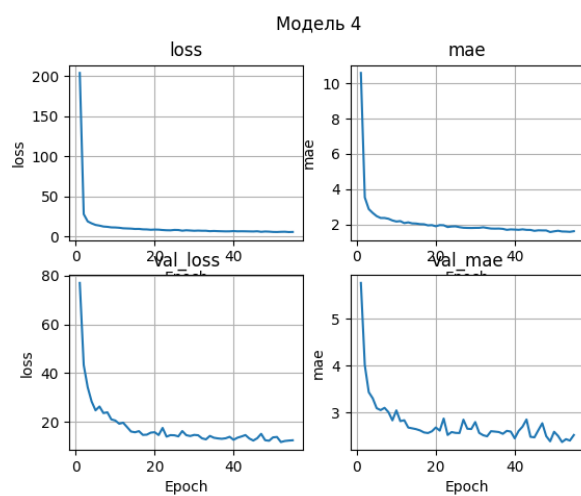


Рисунок 15 – "Модель 4 ($K = 5$, $n = 55$)"

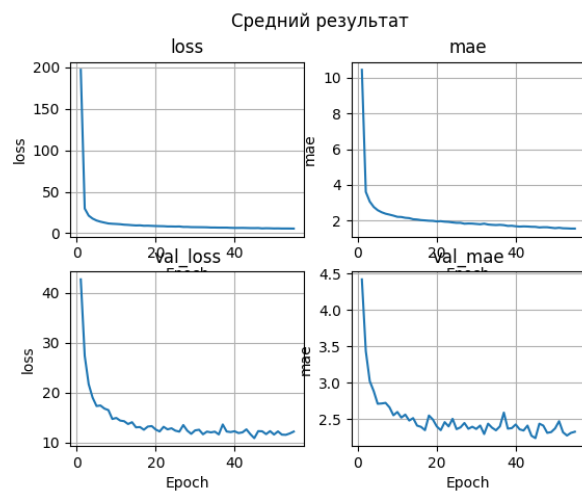


Рисунок 16 – "Среднее ($K = 5$, $n = 55$)"

Вывод

В ходе лабораторной работы была реализована регрессионная модель изменения цен на дома в Бостон. На практике было изучено влияние разного количества эпох на результаты обучения. Также была изучена перекрестная проверка

Регрессионная модель изменения цен на дома в Бостон В ходе лабораторной работы была реализована классификация между камнями и металлическими цилиндрами на основе данных об отражении сигналов радара от поверхностей. Были исследованы различных архитектуры, проведен анализ результатов.

Приложение А.

Исходный код

```
import numpy as np

from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.datasets import boston_housing
import matplotlib.pyplot as plt

def build_model():
    model = Sequential()
    model.add(Dense(64, activation='relu', input_shape=(train_data.shape[1],)))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(1))
    model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
    return model

def draw_plot(history):
    keys = ["loss", "mae", "val_loss", "val_mae"]
    y = ["loss", "mae", "loss", "mae"]

    for i in range(len(keys)):
        plt.subplot(2, 2, i + 1)
        plt.title(keys[i])
        plt.xlabel("Epoch")
        plt.ylabel(y[i])
        plt.grid()
        values = history[keys[i]]
        plt.plot(range(1, len(values) + 1), values)

(train_data, train_targets), (test_data, test_targets) = boston_housing.load_data()
mean = train_data.mean(axis=0)
train_data -= mean
std = train_data.std(axis=0)
train_data /= std
test_data -= mean
test_data /= std

k = 5
num_val_samples = len(train_data) // k
num_epochs = 55
```

```

all_scores = []

all_loss = []
all_loss_validation = []
all_mae = []
all_mae_validation = []

for i in range(k):
    print('processing fold #', i)
    val_data = train_data[i * num_val_samples: (i + 1) * num_val_samples]
    val_targets = train_targets[i * num_val_samples: (i + 1) * num_val_samples]
    partial_train_data = np.concatenate([train_data[:i * num_val_samples], train_data[(i + 1) * num_val_samples:]],
                                         axis=0)
    partial_train_targets = np.concatenate(
        [train_targets[:i * num_val_samples], train_targets[(i + 1) * num_val_samples:]], axis=0)
    model = build_model()
    h = model.fit(partial_train_data, partial_train_targets, epochs=num_epochs, batch_size=1, verbose=0,
                  validation_data=(val_data, val_targets))
    val_mse, val_mae = model.evaluate(val_data, val_targets, verbose=0)
    all_scores.append(val_mae)
    plt.suptitle('Model {}'.format(i))
    draw_plot(h.history)
    plt.show()
    all_loss.append(h.history["loss"])
    all_loss_validation.append(h.history["val_loss"])
    all_mae.append(h.history["mae"])
    all_mae_validation.append(h.history["val_mae"])

print(np.mean(all_scores))
plt.suptitle('Mean result')
draw_plot({"loss": np.mean(all_loss, axis=0), "val_loss": np.mean(all_loss_validation, axis=0), "mae": np.mean(
    all_mae, axis=0), "val_mae": np.mean(all_mae_validation, axis=0)})
plt.show()

```