

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

ОТЧЕТ
по лабораторной работе 3
по дисциплине «Искусственные нейронные сети»
Тема: «Регрессионная модель изменения цен на дома в Бостоне»

Студентка гр. 7383

Маркова А.В.

Преподаватель

Жукова Н.А.

Санкт-Петербург

2020

Цель работы.

Реализовать предсказать 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.

Порядок выполнения работы.

1. Ознакомиться с задачей регрессии;
2. Изучить отличие задачи регрессии от задачи классификации;
3. Создать модель;
4. Настроить параметры обучения;
5. Обучить и оценить модели;
6. 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.

Классификация – один из разделов машинного обучения, посвященный решению следующей задачи. Имеется множество объектов (ситуаций), разделённых некоторым образом на классы. Задано конечное множество

объектов, для которых известно, к каким классам они относятся. Это множество называется обучающей выборкой. Классовая принадлежность остальных объектов не известна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества.

Классифицировать объект – значит, указать номер (или наименование класса), к которому относится данный объект.

Другим распространенным типом задач машинного обучения является регрессия, которая заключается в предсказании не дискретной метки, а значения на непрерывной числовой прямой: например, предсказание температуры воздуха на завтра по имеющимся метеорологическим данным или предсказание времени завершения программного проекта по его спецификациям.

Была построена нейронная сеть, код представлен в приложении А.

Была рассмотрена модель с перекрестной проверкой по К блокам (K-fold cross-validation) при $k = 4$ и $\text{epochs} = 500$. Графики оценки mae для каждого блока приведены на рис. 1 – 4. График средних значений mae , приведен на рис. 5.

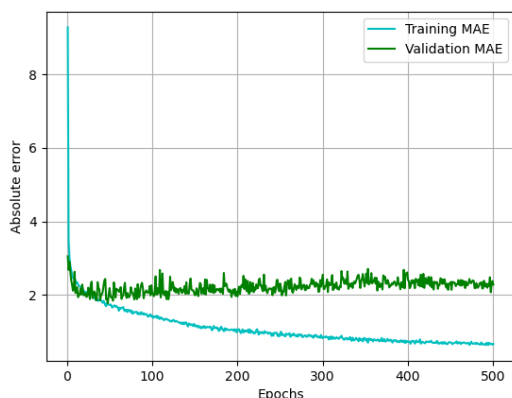


Рисунок 1 – График оценки mae для 1 блока.

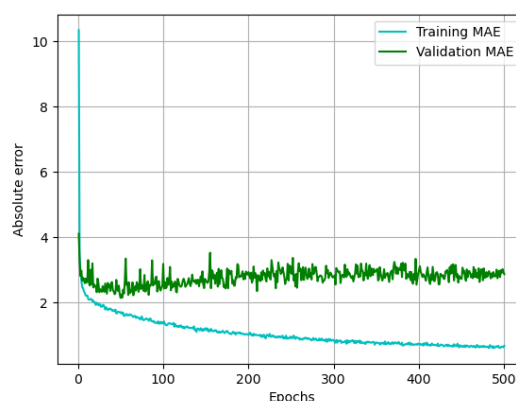


Рисунок 2 – График оценки mae для 2 блока.

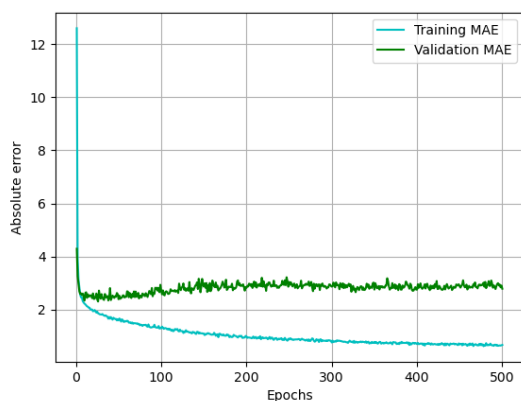


Рисунок 3 – График оценки mae для 3 блока.

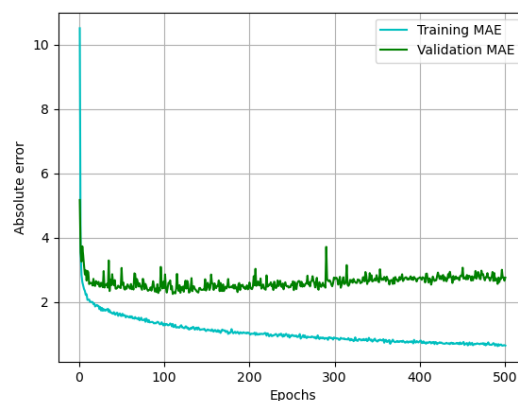


Рисунок 4 – График оценки mae для 4 блока.

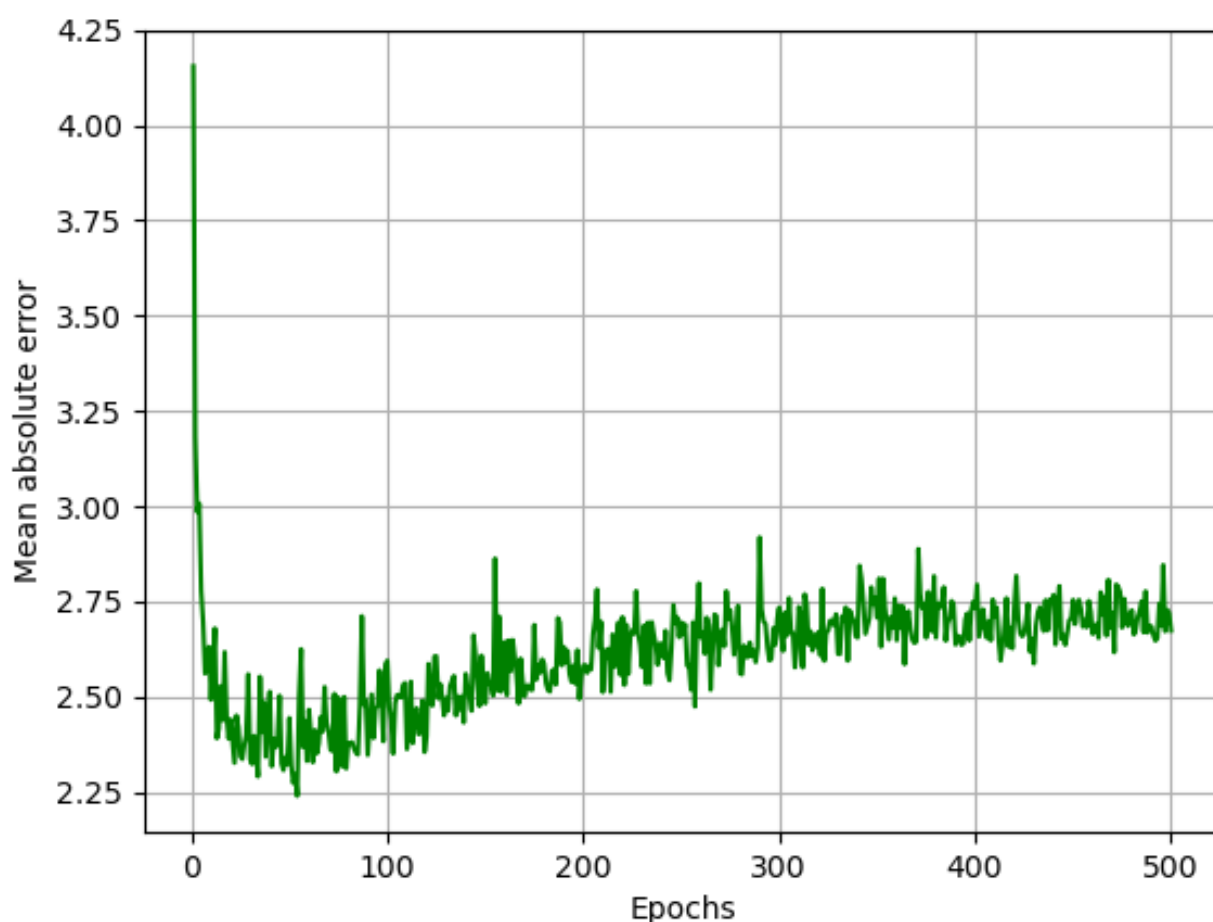


Рисунок 5 – График среднего значения mae (модель: $k = 4$ и $epochs = 500$).

Из графика (рис. 5) замечено, что mae на проверочных данных уменьшается до ≈ 50 эпох обучения, после она либо не меняется, либо увеличивается, тогда как на тестовых данных оно продолжает уменьшаться.

Это связано с переобучением нейронной сети, поэтому за оптимальное число эпох берётся 50.

При заданном числе эпох было рассмотрено среднее значение оценки mae при $k = 2, 4, 6$ и 8 блокам для перекрестной проверки. Результаты приведены на графиках, показанных на рис. 6 – 9.

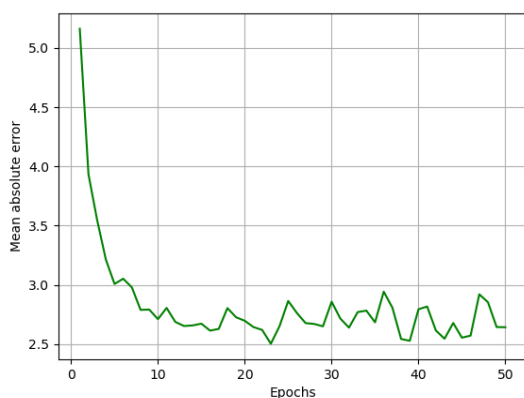


Рисунок 6 – График среднего значения mae для $k = 2$.

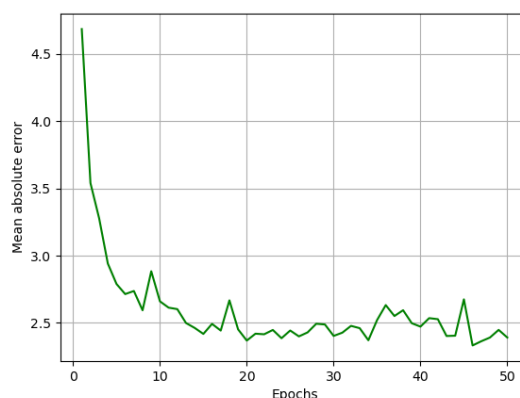


Рисунок 7 – График среднего значения mae для $k = 4$.

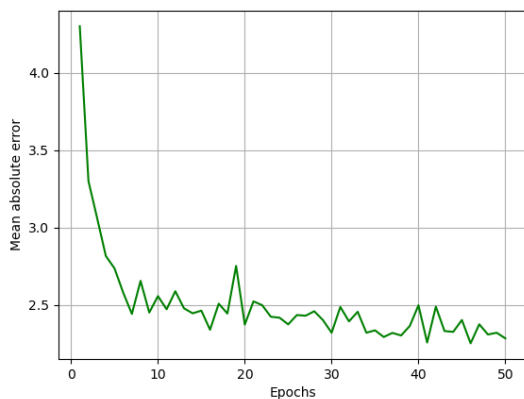


Рисунок 8 – График среднего значения mae для $k = 6$.

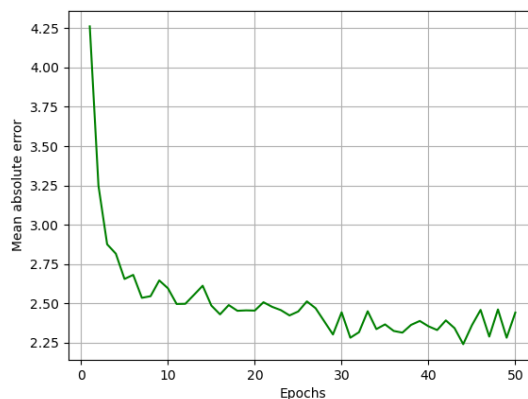


Рисунок 9 – График среднего значения mae для $k = 8$.

По графикам (рис. 6 – 9) видно, что наилучшее значение средней оценки mae достигается в модели, использующей 6 блоков (рис. 8).

Выводы.

В ходе выполнения лабораторной работы было изучено влияние количества эпох и количества блоков в перекрестной проверке по К блокам (K-fold cross-validation) на результат обучения модели искусственной нейронной сети, решающей задачу регрессии.

Также была выбрана оптимальная модель: при $k = 6$ и $epochs = 50$

ПРИЛОЖЕНИЕ А

```
# Подключение модулей
import numpy as np
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import boston_housing

# Загрузка набора данных для Бостона
(train_data, train_targets), (test_data, test_targets) =
boston_housing.load_data()

# Вывод данных для просмотра
print(train_data.shape)
print(test_data.shape)
print(test_targets)

# Нормализация данных
mean = train_data.mean(axis=0)
std = train_data.std(axis=0)
train_data -= mean
train_data /= std

test_data -= mean
test_data /= std

# Определение (создание) модели
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

# mse - среднеквадратичная ошибка
# mae - средняя абсолютная ошибка

# K-fold cross-validation
k = 6
num_val_samples = len(train_data) // k
num_epochs = 50
all_scores = []
mae_histories = []

for i in range(k):
    print('processing fold #', i)
    # Подготовка проверочных данных: данных из блока с номером k
    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]

    # Подготовка обучающих данных: данных из остальных блоков
    partial_train_data = np.concatenate([train_data[:i *
num_val_samples], train_data[(i + 1) * num_val_samples:]], axis=0)
    partial_train_target = np.concatenate([train_targets[: i *
num_val_samples], train_targets[(i + 1) * num_val_samples:]], axis=0)

    # Конструирование модели Keras (уже скомпилированной)
    model = build_model()

    # Обучение модели(в режиме без вывода сообщений, verbose = 0)

```



```

        history = model.fit(partial_train_data, partial_train_target,
epochs=num_epochs, batch_size=1, validation_data=(val_data,
val_targets))

        mae = history.history['mean_absolute_error']
        v_mae = history.history['val_mean_absolute_error']
        x = range(1, num_epochs + 1)
        mae_histories.append(v_mae)
        plt.figure(i + 1)
        plt.plot(x, mae, 'c', label='Training MAE')
        plt.plot(x, v_mae, 'g', label='Validation MAE')
        plt.title('Absolute error')
        plt.ylabel('Absolute error')
        plt.xlabel('Epochs')
        plt.legend()
        plt.grid()

# Создание истории последовательных средних оценок проверки по K
блокам

        average_mae_history = [np.mean([x[i] for x in mae_histories]) for
i in range(num_epochs)]

# Сохранение результатов в файл
plt.figure(0)
plt.plot(range(1, num_epochs + 1), average_mae_history, 'g')
plt.xlabel('Epochs')
plt.ylabel("Mean absolute error")
plt.grid()
figs = [plt.figure(n) for n in plt.get_fignums()]
for i in range(len(figs)):
        figs[i].savefig("./%d.png" %(i), format='png')

```