**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Искусственные нейронные сети»**

Тема: Регрессионная модель изменения цен на дома в Бостоне

|  |  |  |
| --- | --- | --- |
| Студентка гр. 7382 |  | Головина Е.С. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2020

**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.

**Задачи.**

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

**Требования.**

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

**Ход работы.**

1. **Знакомство с задачей регрессии**

Регрессия относится к одному из основных классов машинного обучения, т.н. *обучение с учителем*. При обучении с учителем на вход подается тренировочный набор данных, на которых нейронная сеть учится и по окончании обучения способна по новым данным (похожим на тренировочные данные) получать определенный результат.

Задача регрессии заключается в том, чтобы предсказать числовое значение по входным данным. Порождаемая функция: . Различия от задачи классификации в формате выхода.

1. **Создание модели**

Набор данных присутствует в составе Keras. Загрузка происходит следующим образом.

from tensorflow.keras.datasets import boston\_housing

(train\_data, train\_targets), (test\_data, test\_targets) = boston\_housing.load\_data()

Данные – 404 обучающих и 102 контрольных образца, каждый с 13 числовыми признаками. Из-за того, что каждый признак во входных данных (например, уровень преступности) имеет свой масштаб – необходимо нормализовать данные. В противном случае сеть сможет адаптироваться к разнородным данным, но это усложнит обучение.

Процесс нормализации: для каждого признака во входных данных (столбца в матрице входных данных) из каждого значения вычитается среднее по этому признаку, и разность делится на стандартное отклонение, в результате признак центрируется по нулевому значению и имеет стандартное отклонение, равное единице.

Такое преобразование можно сделать с помощью Numpy.

mean = train\_data.mean(axis = 0)

train\_data -= mean

std = train\_data.std(axis=0)

train\_data /= std

test\_data -=mean

test\_data /= std

Для создания модели ИНС напишем функцию build\_model():

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

Таким образом, построена нейронная сеть с тремя слоями. Сеть заканчивается одномерным слоем, не имеющим функции активации (это линейный слой). Применение функции активации могло бы ограничить диапазон выходных значений, а с линейным последним слоем, сеть способна предсказывать значения из любого диапазона.

Сеть компилируется с функцией потерь mse – mean squared error (среднеквадратичная ошибка), вычисляющей квадрат разности между предсказанными и целевыми значениями. Эта функция широко используется в задачах регрессии.

Параметр mae — mean absolute error (средняя абсолютная ошибка). Это абсолютное значение разности между предсказанными и целевыми значениями. Например, значение mae, равное 0.5, в этой задаче означает, что в среднем прогнозы отклоняются на 500 долларов США.

1. **Настройка параметров, обучение и оценка модели**

В связи с тем, что набор данных для обучения небольшой используем перекрестную проверку 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. В коде такая проверка реализуется следующим образом.

k = 4

num\_val\_samples = len(train\_data) // k

num\_epochs = 100

all\_scores = []

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()

model.fit(partial\_train\_data, partial\_train\_targets, epochs=num\_epochs, batch\_size=1, verbose=0)

val\_mse, val\_mae = model.evaluate(val\_data, val\_targets, verbose=0)

all\_scores.append(val\_mae)

print(np.mean(all\_scores))

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

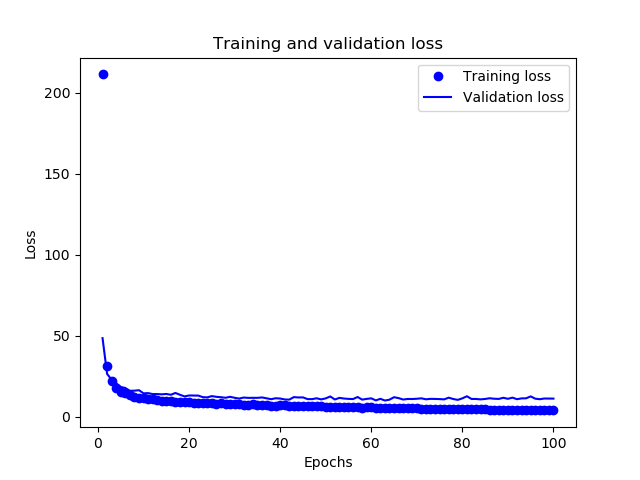
Полученные результаты:

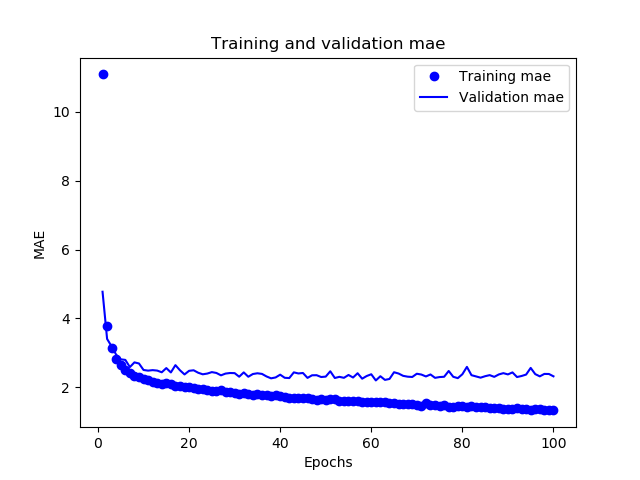
2.483508288860321

Это означает, что средняя ошибка составляет около 2400 долларов, что довольно много для цен из диапазона от 10 000 до 50 000 долларов.

1. **Выбор количества эпох**

Для того, чтобы решить в какую сторону двигаться: увеличивать или уменьшать число эпох выведем средние графики ошибки и точности по всем моделям.

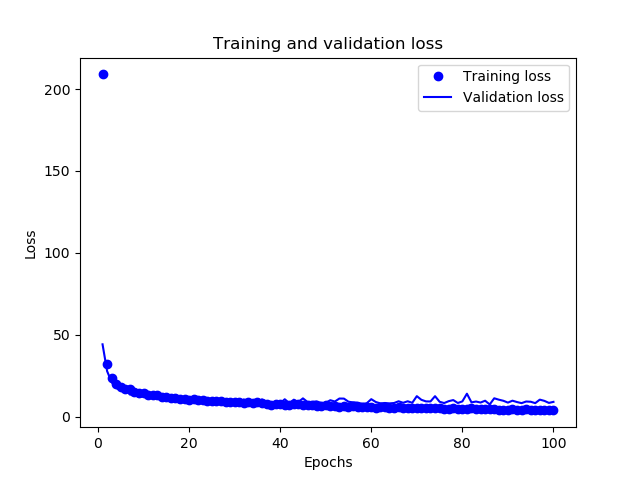
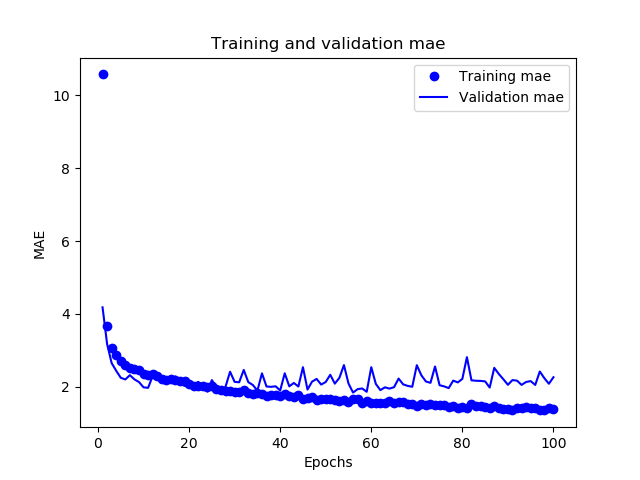


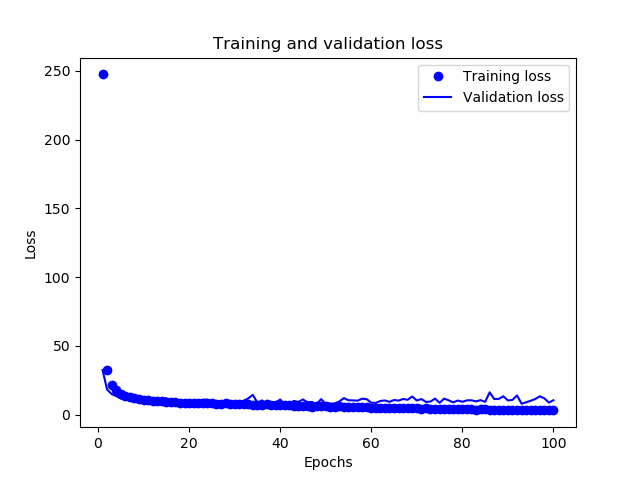
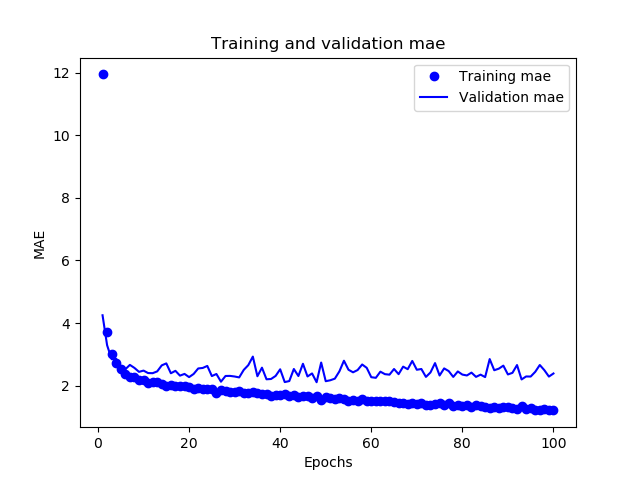


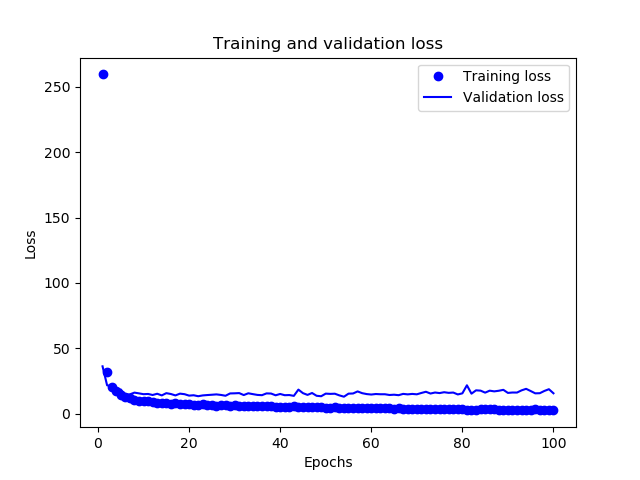
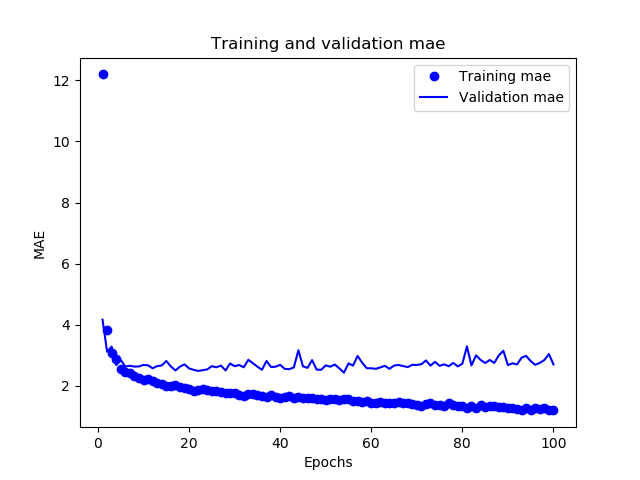
По графику ошибок можно заметить, что примерно на 40 эпохе происходит переобучение, т.к. между потерями при обучении и оценке появляется явный разрыв. Значит нужно двигаться в сторону уменьшения эпох.

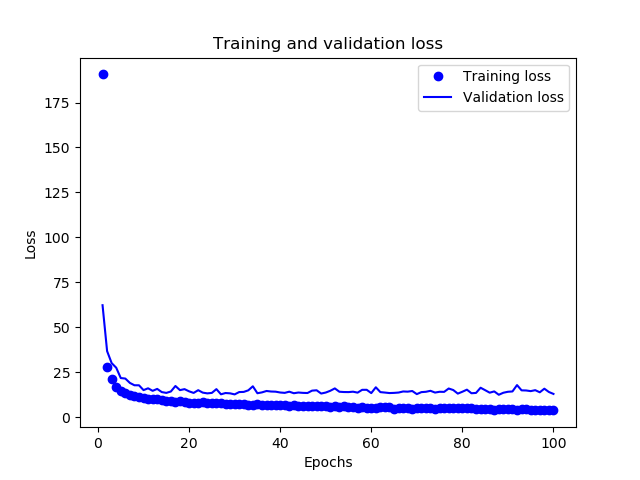
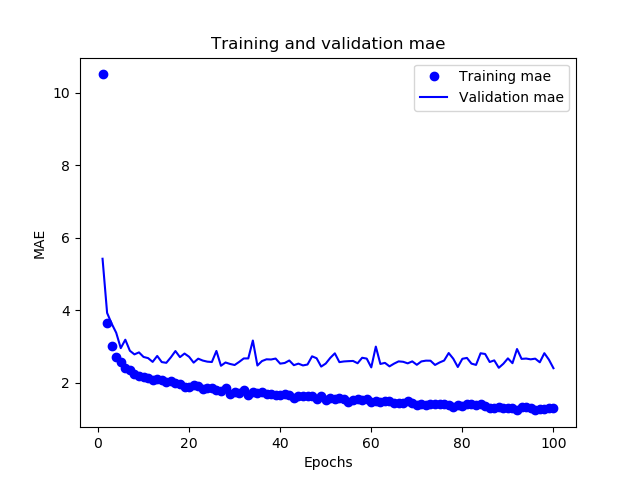
Проверив значения для эпох от 10 до 100 с шагом 10, выявили, что наименьшее значение в результате дают 50 и 60 эпоха.

Проверив значения между ними – наименьшего значение достигли на 54 эпохах. Ставим 54 эпохи и смотрим графики результатов.

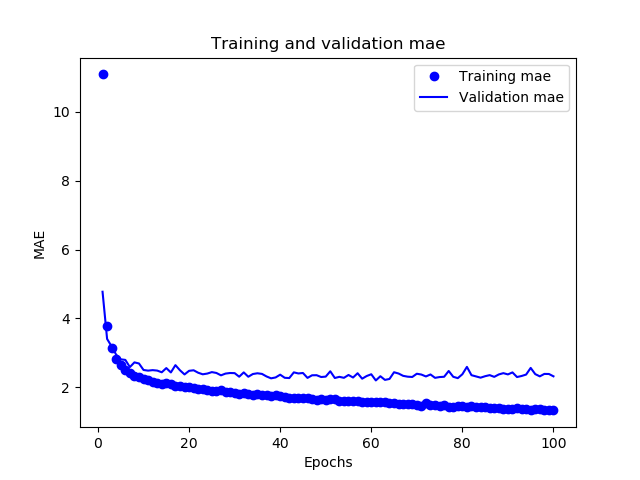


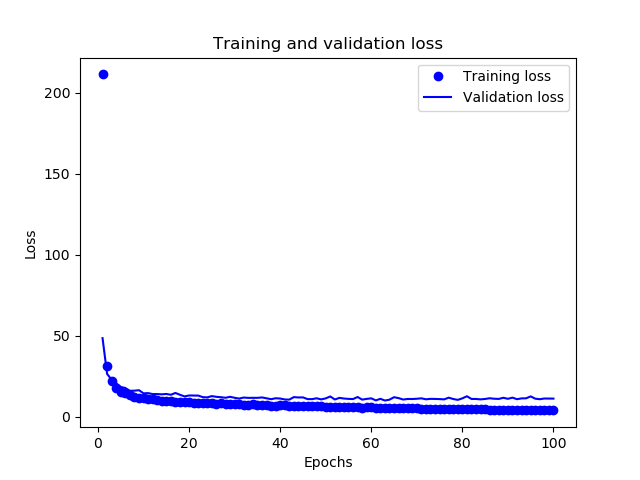






Графики средних значений:





**Вывод.**

В ходе выполнения работы реализовали решение задачи регрессии: предсказание медианной цены на дома в пригороде Бостона в середине 1970-х по набору разных, влияющих на цену данных. Для этого создали искусственную нейронную сеть с помощью библиотеки keras в python, протестировали получившуюся модель, затем, проведя ряд экспериментов, выбрали наилучшую конфигурацию модели.