

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

**ОТЧЕТ**  
**по лабораторной работе №3**  
**по дисциплине «Искусственные нейронные сети»**  
**Тема: «Бинарная классификация отраженных сигналов радара»**

Студентка гр. 7381

Преподаватель

\_\_\_\_\_

\_\_\_\_\_

Судакова П.С.

Жукова Н.А.

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.

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.**

- Ознакомиться с задачей регрессии
- Изучить отличие задачи регрессии от задачи классификации
- Создать модель
- Настроить параметры обучения
- Обучить и оценить модели
- Ознакомиться с перекрестной проверкой
- Необходимо уменьшить или увеличить количество эпох обучения и проанализировать полученные результаты

### **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.**

Задача классификации отличается от задачи регрессии тем, что результат в задаче на классификацию представляется в виде одной дискретной метки, а в задаче на регрессию - значение на непрерывной числовой прямой.

Результаты обучения нейронной сети при количестве эпох равным 100 представлены на рис. 1 - 4.

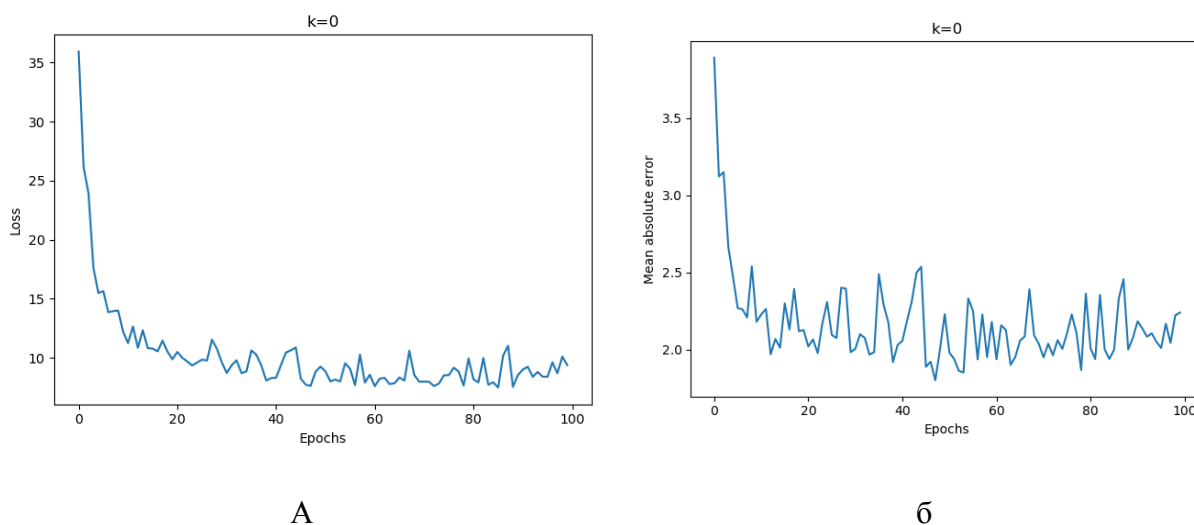


Рисунок 1 – Потери (а) и ошибки (б) модели с 100 эпохам в 0 блоке.

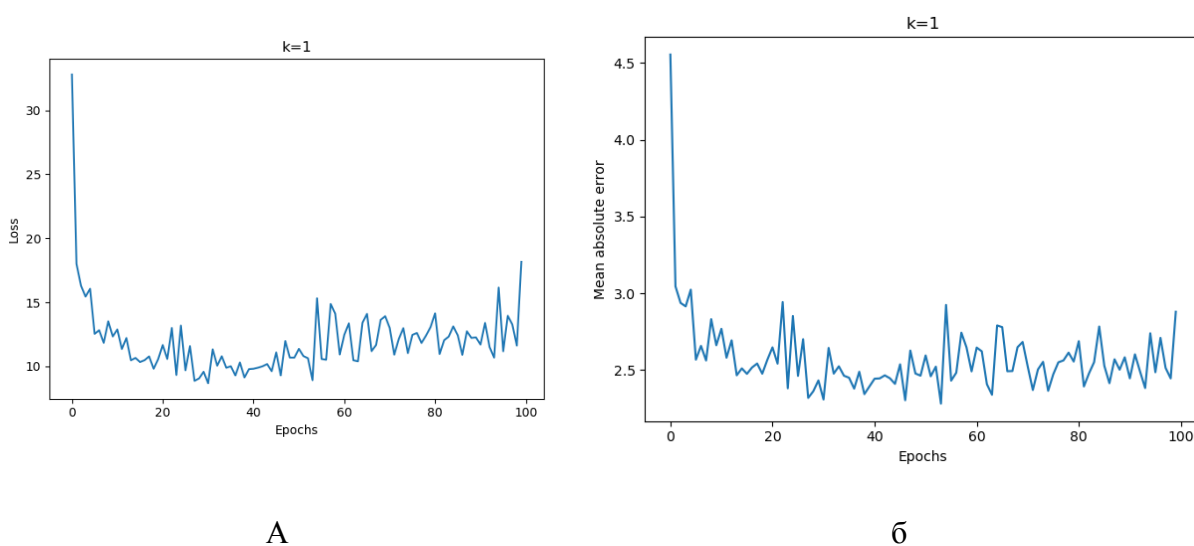
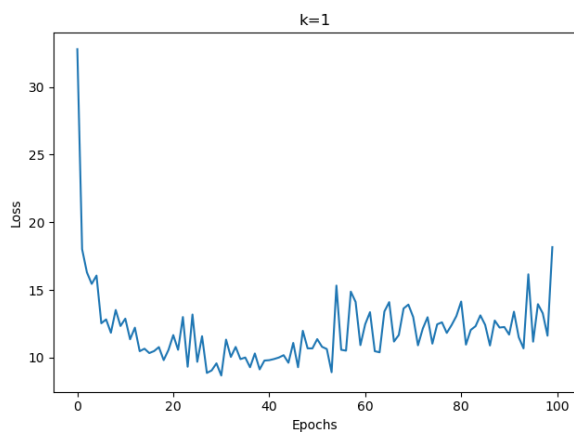
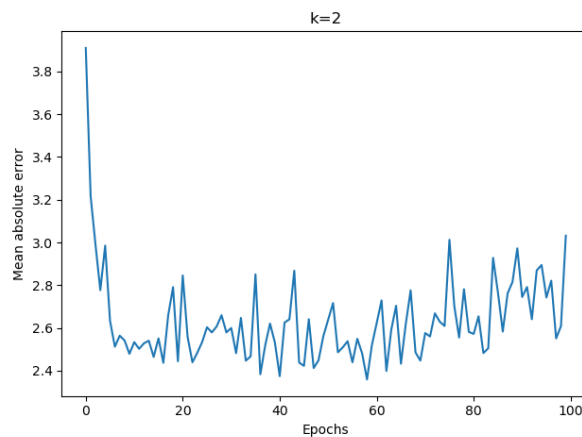


Рисунок 2 – Потери (а) и ошибки (б) модели с 100 эпохам в 1 блоке.

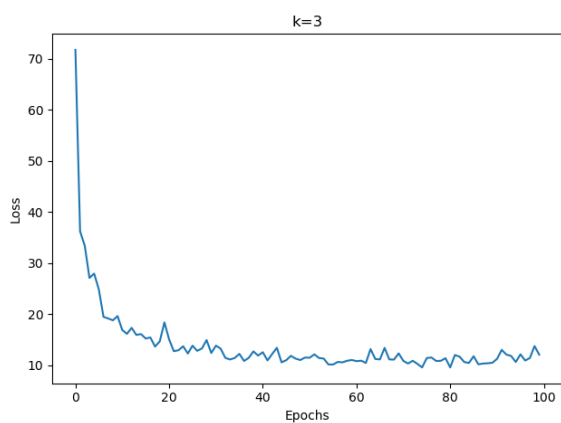


а

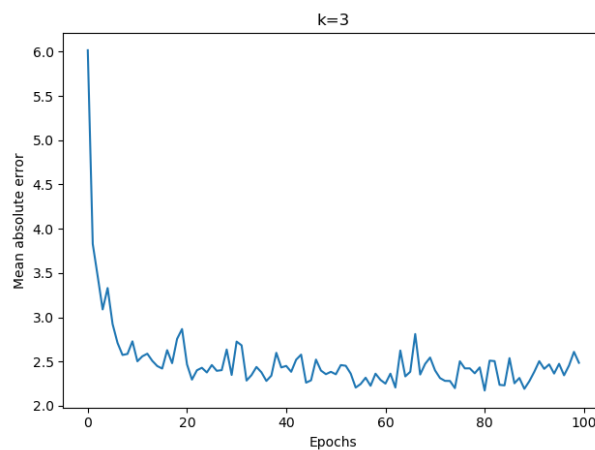


б

Рисунок 3 – Потери (а) и ошибки (б) модели с 100 эпохам во 2 блоке.



а



б

Рисунок 4 – Потери (а) и ошибки (б) модели с 100 эпохам в 3 блоке.

Для определения точки переобучения поменяем масштаб и заменим каждую оценку средним по предыдущим оценкам. Результат показан на рис. 5, из которого можно сделать вывод, что переобучение начинается на 30 эпохе.

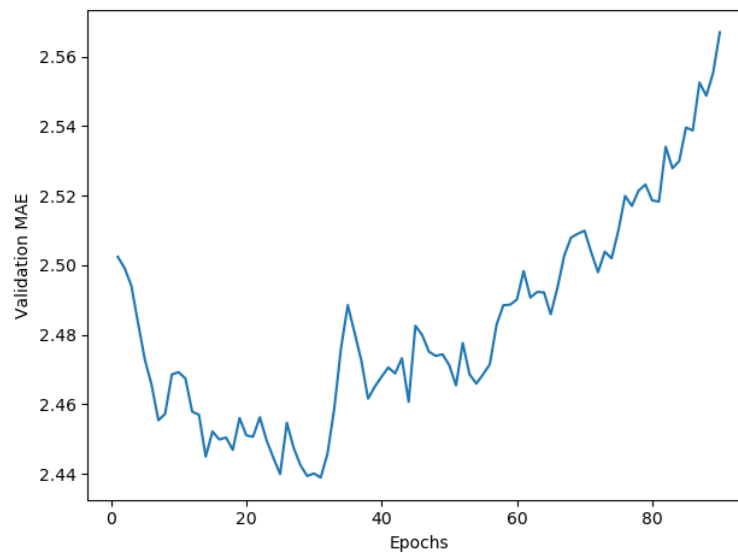


Рисунок 5

**Выводы.**

В ходе выполнения лабораторной работы была решена задача регрессии, найдена точка переобучения на 30 эпохах.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД

```
import numpy as np
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.datasets import boston_housing
import matplotlib.pyplot as plt

(train_data, train_targets), (test_data, test_targets) =
boston_housing.load_data()

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 smooth_curve(points, factor=0.9):
    smoothed_points = []
    for point in points:
        if smoothed_points:
            previous = smoothed_points[-1]
            smoothed_points.append(previous * factor + point *
(1 - factor))
        else:
            smoothed_points.append(point)
    return smoothed_points

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 = 4
num_val_samples = len(train_data) // k
num_epochs = 100

all_val_mae_history = []
all_mae_history = []
all_val_loss_history = []
```

```

all_loss_history = []

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()
    history = model.fit(partial_train_data,
partial_train_targets, validation_data=(val_data, val_targets),
epochs=num_epochs, batch_size=1, verbose=0)
    mae_history = history.history['val_mean_absolute_error']
    loss_history = history.history['val_loss']
    all_mae_history.append(mae_history)
    all_val_loss_history.append(loss_history)

    plt.plot(range(num_epochs), mae_history, label='Val mae')
    plt.title('k=' + str(i))
    plt.xlabel('Epochs')
    plt.ylabel('Mean absolute error')
    plt.show()

    plt.clf()
    plt.plot(range(num_epochs), loss_history, label='Val loss')
    plt.title('k=' + str(i))
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.show()
    plt.clf()

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

smooth_mae_history = smooth_curve(average_mae_history[10:])
plt.plot(range(1, len(smooth_mae_history) + 1),
smooth_mae_history)
plt.xlabel('Epochs')
plt.ylabel('Validation MAE')
plt.show()

model = build_model()
model.fit(train_data, train_targets, epochs=80, batch_size=16,
verbose=0)
test_mse_score, test_mae_score = model.evaluate(test_data,
test_targets)

```