

**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. 8382

\_\_\_\_\_

Звегинцева Е.Н.

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.

### **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.**

- 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.

Задача классификации сводится к определению класса объекта по его характеристикам. Необходимо заметить, что в этой задаче множество классов, к которым может быть отнесен объект, заранее известно.

Задача регрессии, подобно задаче классификации, позволяет определить по известным характеристикам объекта значение некоторого его параметра. В отличие от задачи классификации значением параметра является не конечное множество классов, а множество действительных чисел.

В ходе работы была создана и обучена модель искусственной нейронной сети в соответствии с условиями (код представлен в приложении).

Посмотрим на результаты нейронной сети на данных по умолчанию – на 4 блоках и 100 эпохах. Графики представлены на рис. 1, 2.

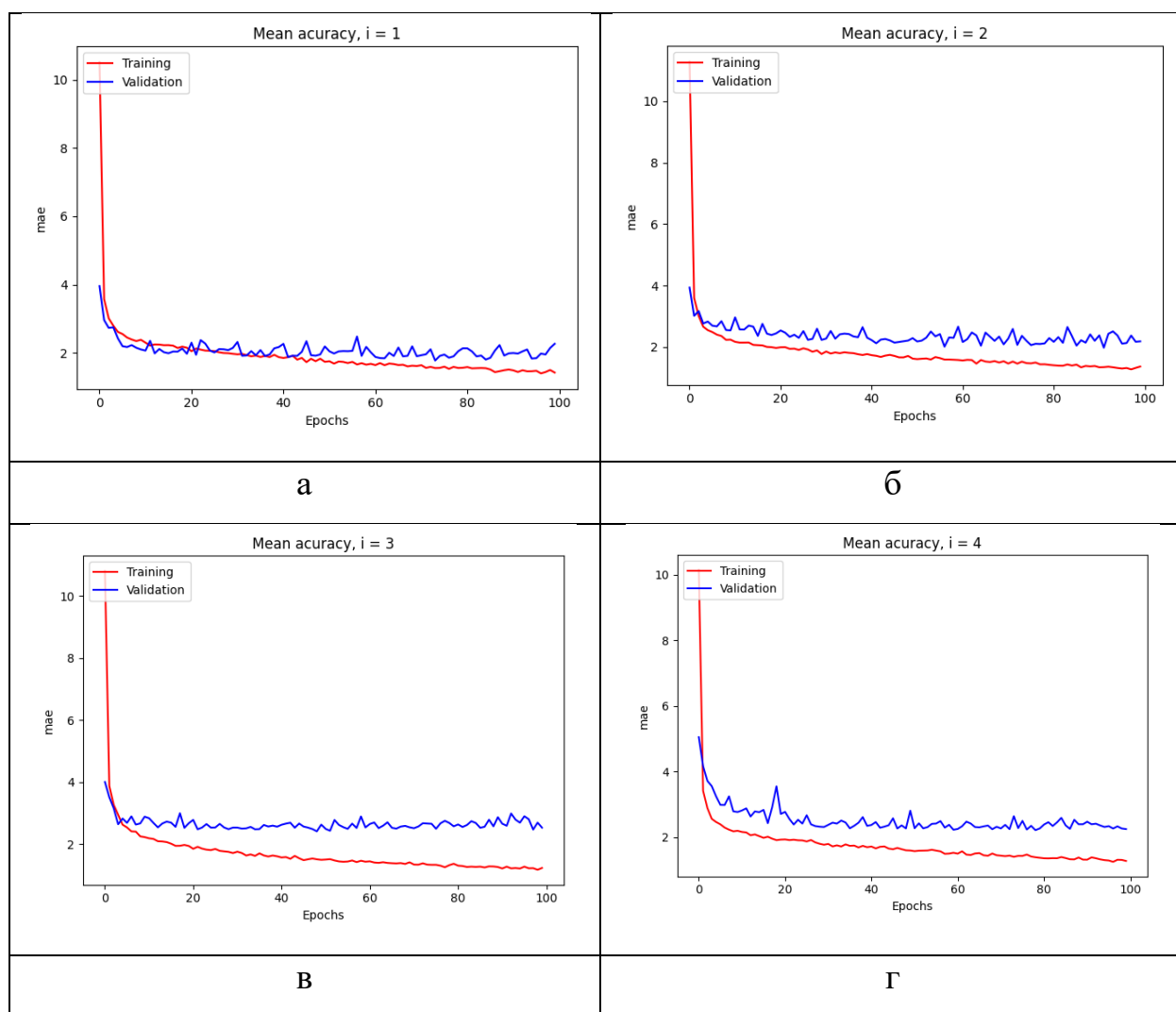


Рисунок 1 – График оценки MAE для блока а–1, б–2, в–3, г–4.

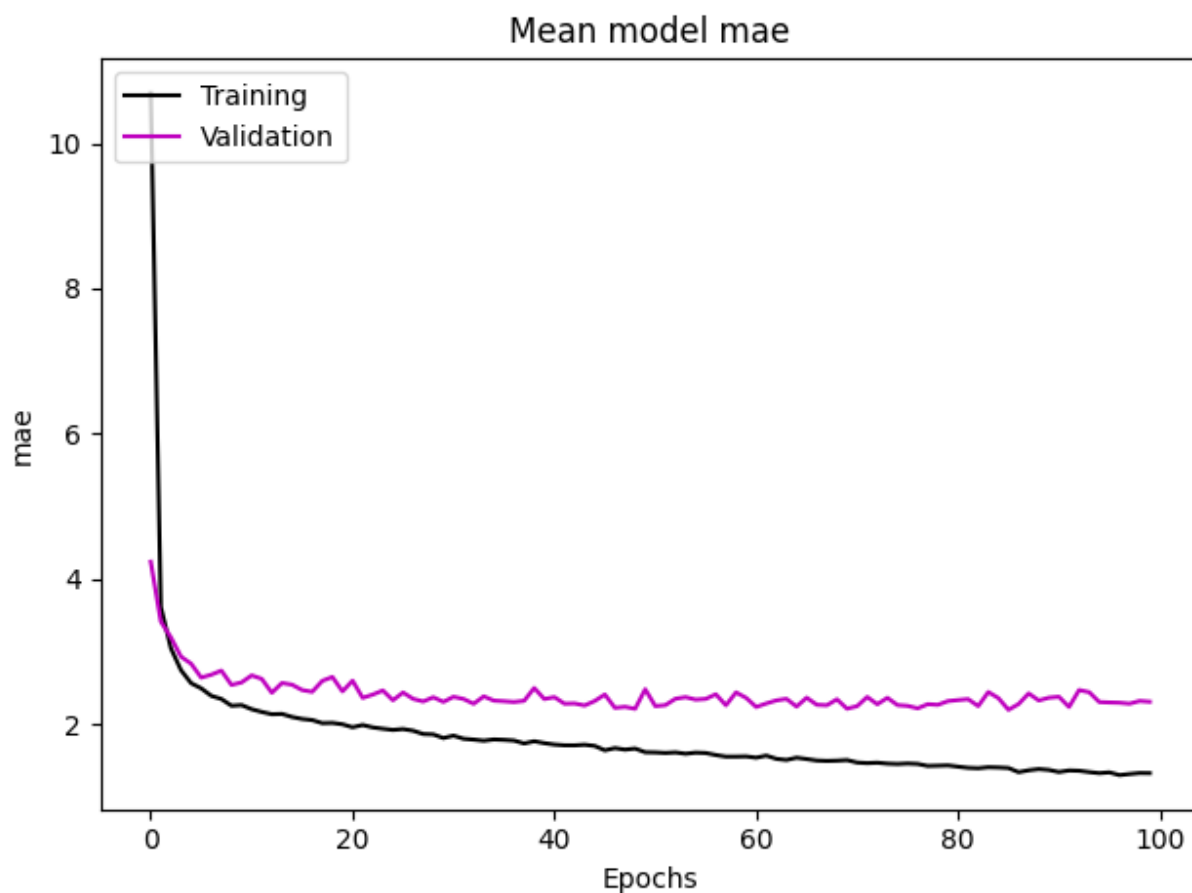
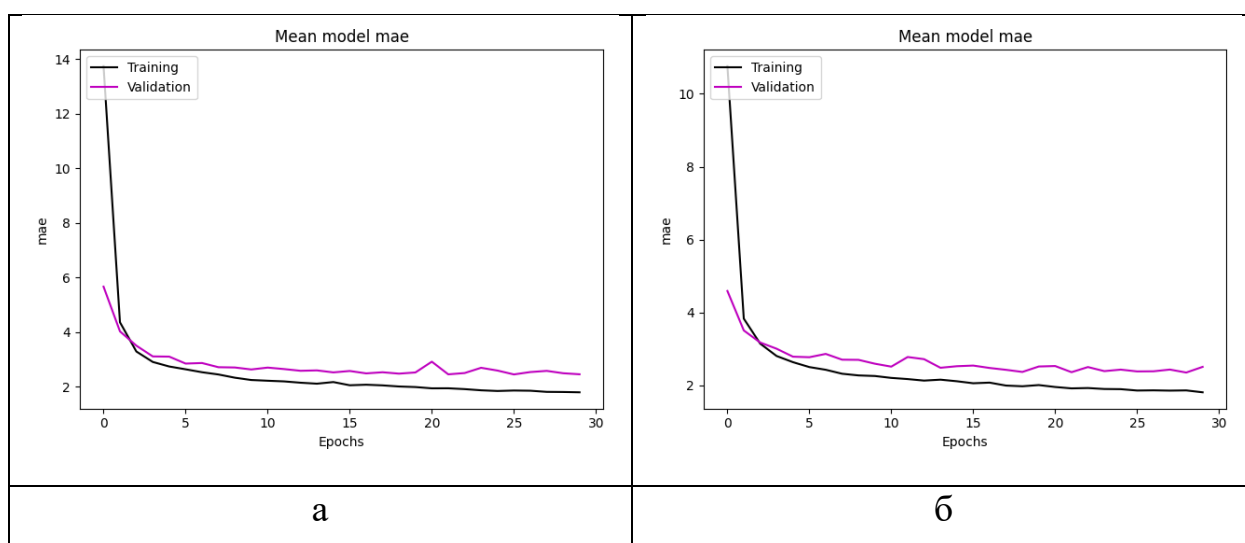


Рисунок 2 – График среднего значения MAE

Заметим, что оценки MAE на тестовых данных начинают активно возрастать после ~30 эпохи, значит следует убавить количество эпох до этого значения во избежание переобучения.

Теперь нам нужно определить оптимальное число К. Для этого рассмотрим модели с 30 эпохами на 2, 4, 6 и 8 блоках.



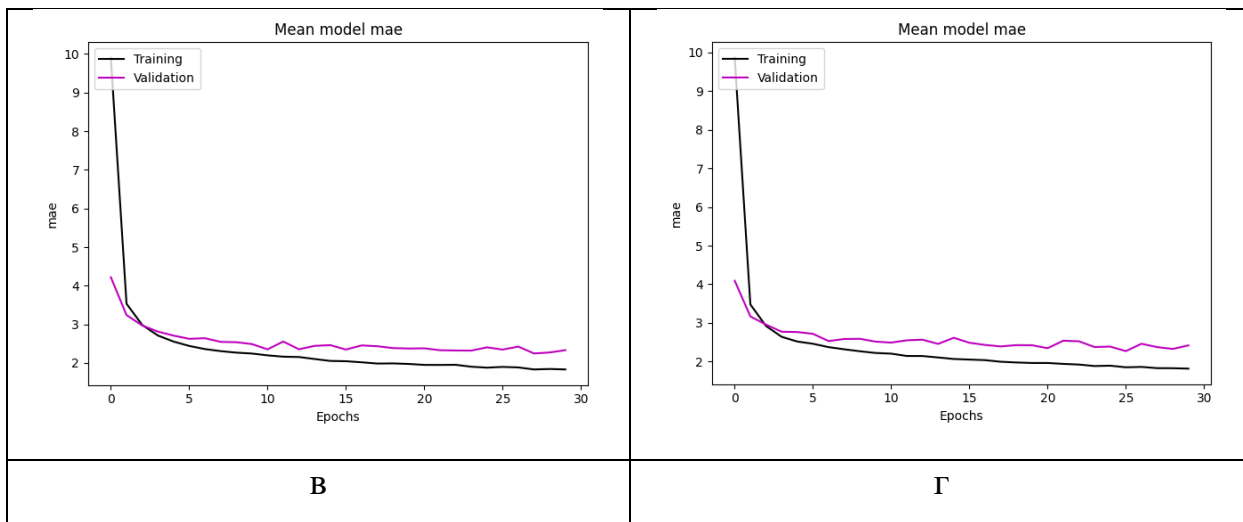
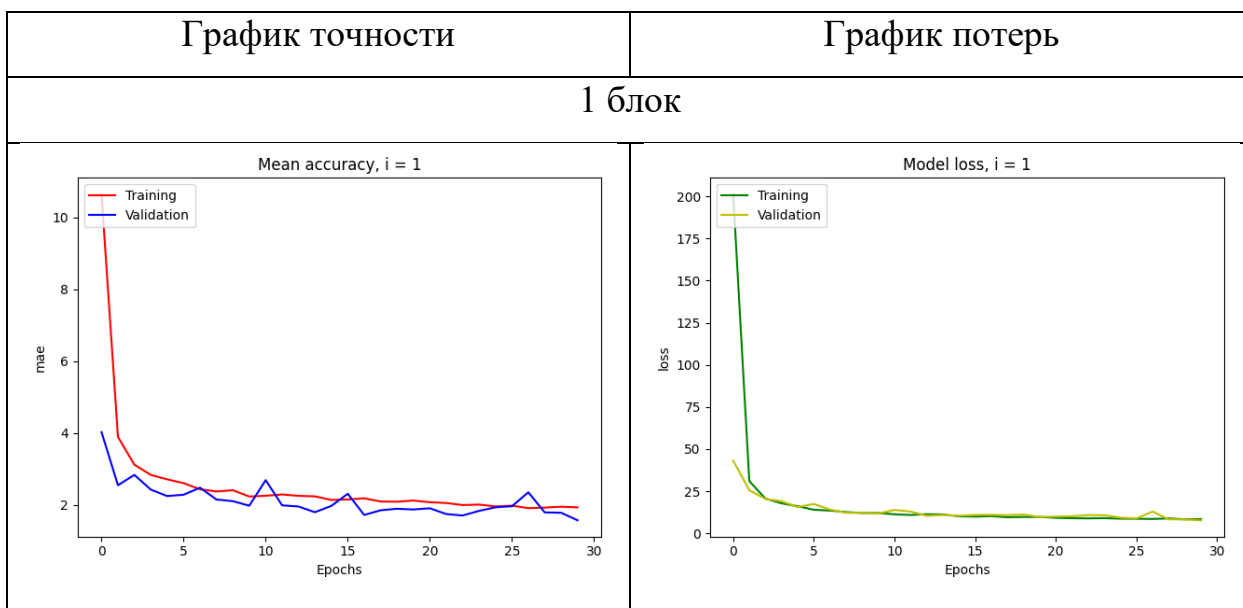


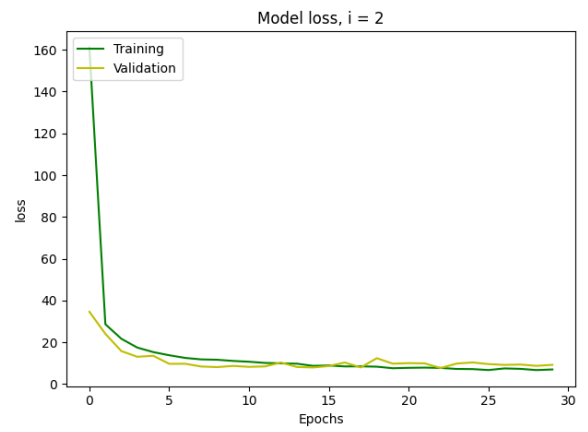
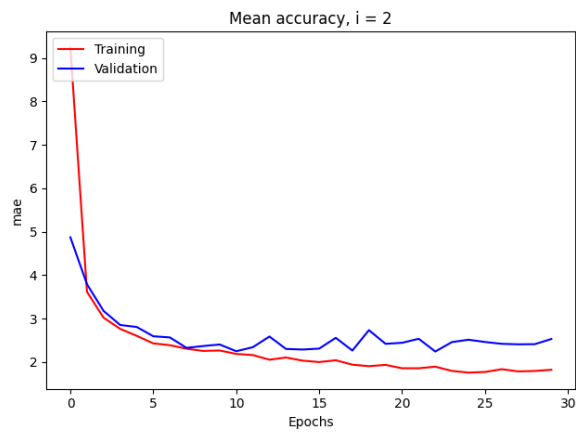
Рисунок 3 – Графики среднего значения MAE для модели с количеством блоков: а–2, б–4, в–6, г–8.

Из графиков видим, что наилучшей сходимостью и наименьшей средней ошибкой обладает модель с 6 блоками.

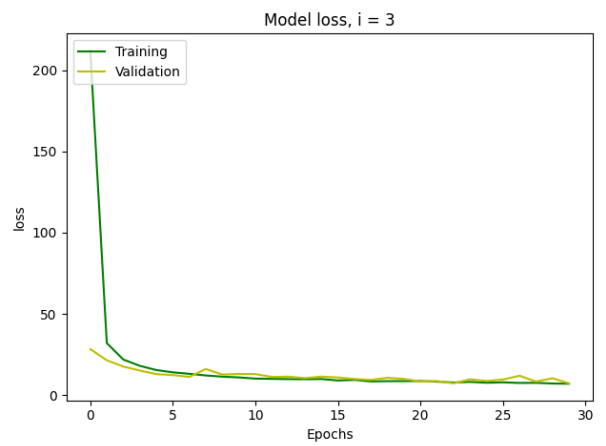
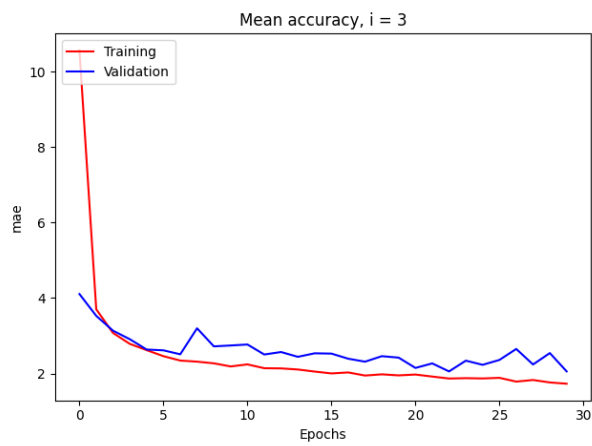
Теперь построим графики точности и ошибок обучения модели с параметрами: количество эпох обучения - 30, количество блоков – 6.



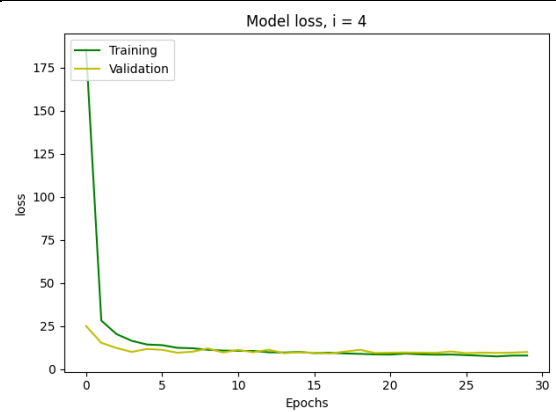
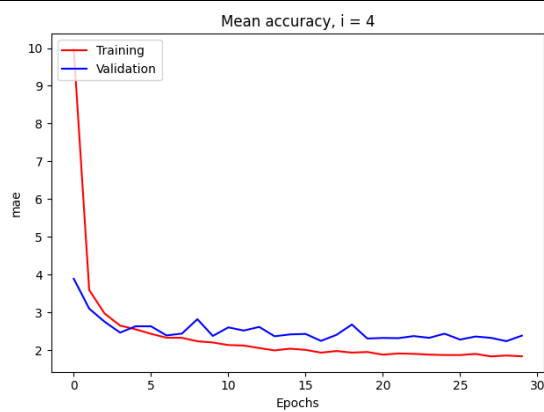
## 2 блок

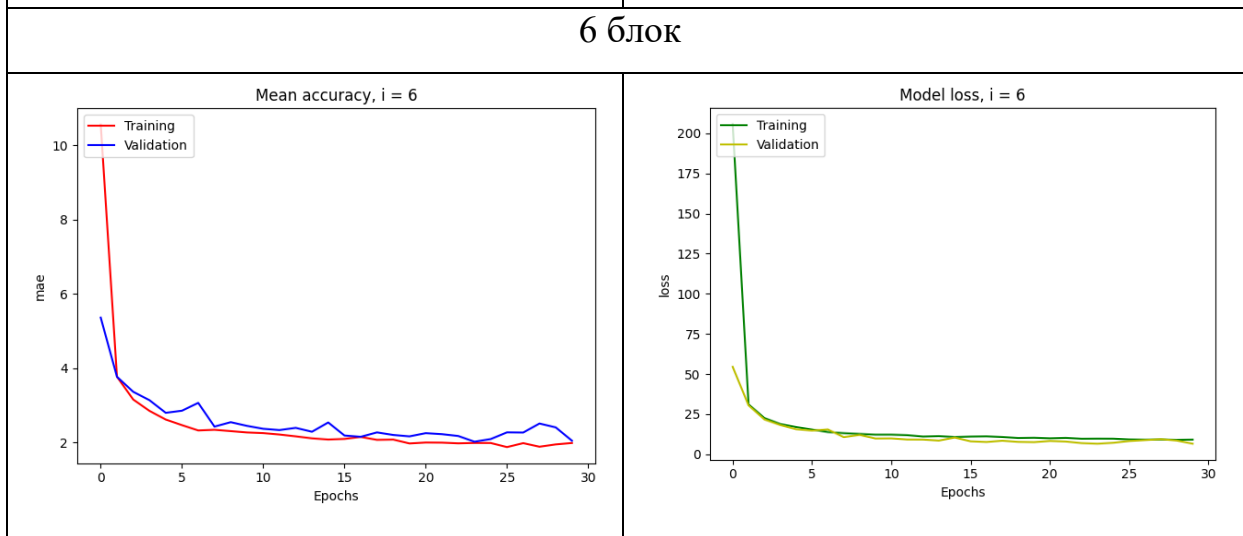
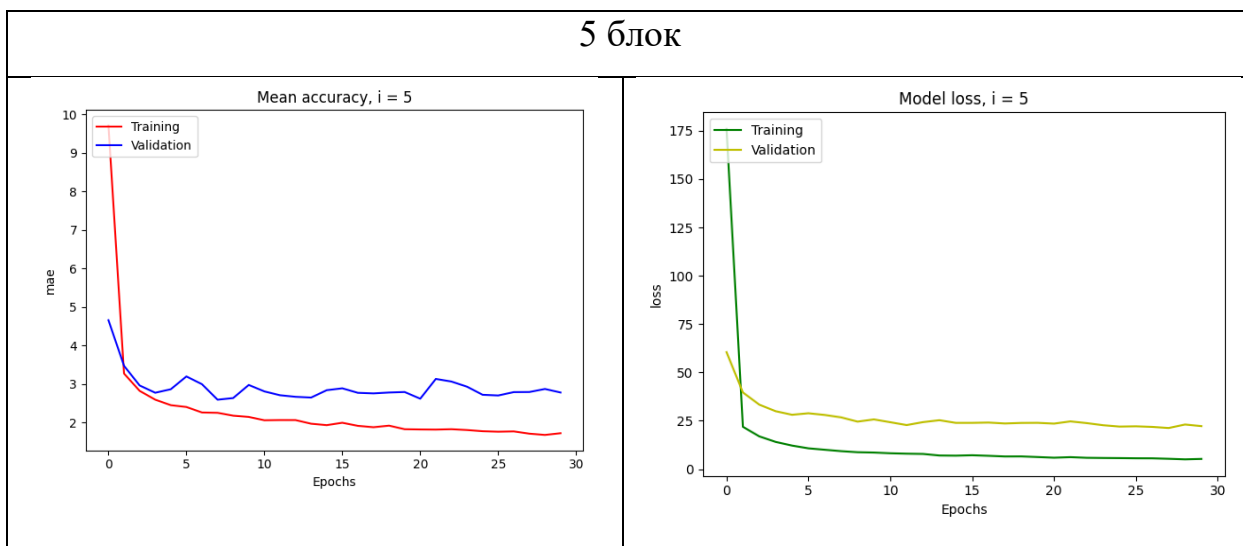


## 3 блок



## 4 блок





Графики точности и обучения усредненной модели представлены на рис. 4.

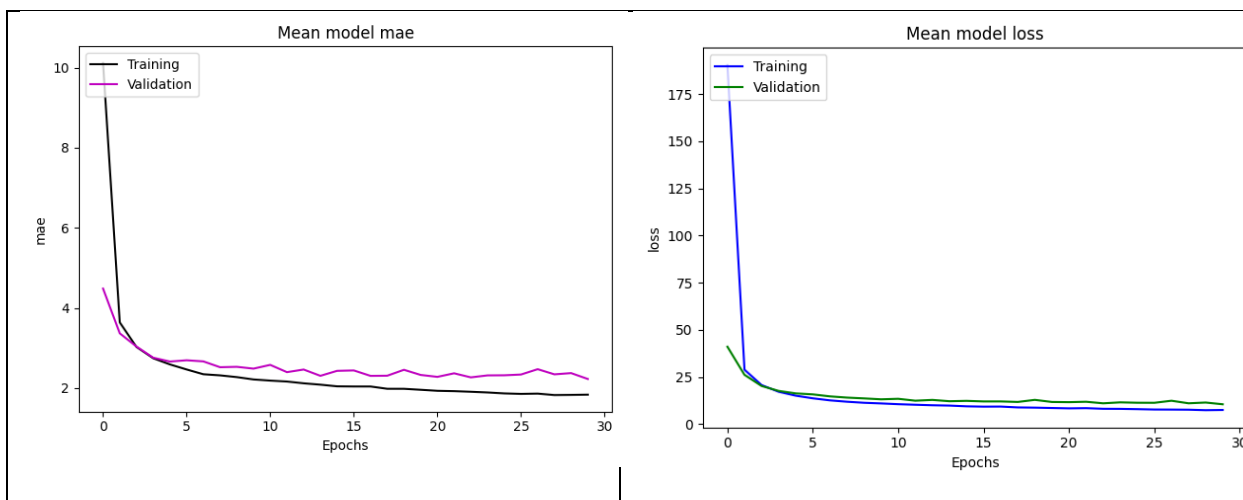


Рисунок 4 – График точности и обучения усредненной модели

### **Выводы.**

В ходе выполнения данной работы была изучена задача регрессии и ее отличие от задачи классификации с помощью библиотеки Keras. Также было изучено влияние количества эпох и числа блоков на результат обучения сети.



## ПРИЛОЖЕНИЕ

### 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

#Создание модели для обучения
def build_model():
    model = Sequential()
    model.add(Dense(64, activation='relu',
input_shape=(train_data.shape[1],)))
    model.add(Dense(64, activation='relu'))
    #одномерный слой, без ф-ции активации (не огр диапазон выходных
значений)
    # исп для предсказывания значений из любого диапазона
    model.add(Dense(1))
    model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
    return model

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

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

```

train_data /= std

test_data -= mean
test_data /= std

#Перекрестная проверка по K блокам (K-fold cross-validation)
k = 4
num_val_samples = len(train_data) // k
num_epochs = 100

mean_loss = []
mean_mae = []
mean_val_loss = []
mean_val_mae = []

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,
epochs=num_epochs, batch_size=1,
                        validation_data=(val_data, val_targets),
verbose=0)

```

```

    val_mse, val_mae = model.evaluate(val_data, val_targets,
verbose=0)

```

```

mean_val_mae.append(history.history['val_mae'])
mean_mae.append(history.history['mae'])

```

```

plt.plot(history.history['mae'], 'g')
plt.plot(history.history['val_mae'], 'b')
plt.title('Mean accuracy' + ', i = ' + str(i + 1))
plt.ylabel('mae')
plt.xlabel('Epochs')
plt.legend(['Training', 'Validation'], loc='upper left')
plt.show()

```

```

mean_val_loss.append(history.history['val_loss'])
mean_loss.append(history.history['loss'])

```

```

plt.plot(history.history['loss'], 'g')
plt.plot(history.history['val_loss'], 'b')
plt.title('Model loss' + ', i = ' + str(i + 1))
plt.ylabel('loss')
plt.xlabel('Epochs')
plt.legend(['Training', 'Validation'], loc='upper left')
plt.show()

```

```
plt.plot(np.mean(mean_mae, axis=0), 'g')
plt.plot(np.mean(mean_val_mae, axis=0), 'b')
plt.title('Mean model mae')
plt.ylabel('mae')
plt.xlabel('Epochs')
plt.legend(['Training', 'Validation'], loc='upper left')
plt.show()
```

```
plt.plot(np.mean(mean_loss, axis=0), 'g')
plt.plot(np.mean(mean_val_loss, axis=0), 'b')
plt.title('Mean model loss')
plt.ylabel('loss')
plt.xlabel('Epochs')
plt.legend(['Training', 'Validation'], loc='upper left')
plt.show()
```