**Постановка задачи.**  
<https://www.kaggle.com/hugoncosta/price-of-flats-in-moscow>  
Данные о стоимости квартир в Москве в зависимости от параметров квартиры и ее расположения.

Структура:  
price - Цена квартиры в $1000  
totsp - Общая площадь квартиры, кв.м.  
livesp - Жилая площадь квартиры, кв.м.  
kitsp - Площадь кухни, кв.м.  
dist - Расстояние от центра в км.  
metrdist - Расстояние до метро в минутах  
walk - 1 – пешком от метро, 0 – на транспорте  
brick - 1 – кирпичный, монолит ж/б, 0 – другой  
floor - 1 – этаж кроме первого и последнего, 0 – иначе  
code - Число от 1 до 8, при помощи которого мы группируем наблюдения по подвыборкам:  
1. Наблюдения сгруппированы на севере, вокруг Калужско-Рижской линии метрополитена  
2. Север, вокруг Серпуховско-Тимирязевской линии метрополитена  
3. Северо-запад, вокруг Замоскворецкой линии метрополитена  
4. Северо-запад, вокруг Таганско-Краснопресненской линии метрополитена  
5. Юго-восток, вокруг Люблинской линии метрополитена  
6. Юго-восток, вокруг Таганско-Краснопресненской линии метрополитена  
7. Восток, вокруг Калиниской линии метрополитена  
8. Восток, вокруг Арбатско-Покровской линии метрополитена

Задача: вычисление значения параметра price на основании остальных параметров.  
Опциональный вариант: вычисление значения параметра code на основании остальных параметров.

**Подход к обработке данных.**  
В таблицах price, totsp, livesp, mertdist, mertdist были выбросы. Выбросы были устранены с помощью numpy.clip().

К таблицам price, livesp применение логарифм. Так как были нули или они давали более хорошее распределение.

К таблицам ltotsp, mertdist применен квадратный корень.

Ко всем таблицах примененили MinMaxScaler() fit\_transform(data).flatten(). У нас нет нормального распределения.

walk, brick, floor – принимают только 0 и 1 поэтому были оставлены без изменений

code – было применено pandas.get\_dummies() т.к. столбец код это не содержит в себе количественную меру, а атрибутивное свойство в данном случае местоположение квартиры.

**Выбранные базовые параметры нейронной сети и алгоритма обучения.**функция активации – ReLU  
Количество нейронов входного слоя – 16  
Первый скрытый слой – 6  
Второй скрытый слой – 6  
Выходной слой – 1  
Количество эпох – 30  
Батчей – 16

**Описание проведенных экспериментов.**Первая модель работает корректно. Во второй модели добавил 3 слой. Графики сошлись, обучение проходило дольше. Параметров потребовалось больше, чем в первой модели. Все 3 слоя имеют 4 нейрона. В третьей модели всего 1 слой на 16 нейронов. Работает быстро график не очень красивый но точность не сильно отличается от первой и второй модели.

**Итоговый вывод.**Увеличение скрытых слоев нейронной сети делает алгоритм нейронной сети более сложным. Время работы сети также увеличивается с количеством слоев. Если у нас малый набор данных и большое количество слоев мы можем переобучить нейронную сеть или сделать ее решение очень специфичным. Чем больше слоев, тем больше нейронной сети требуется параметров.