НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет систем управления и робототехники

Прикладной искусственный интеллект

Лабораторная работа № 2

Выполнил
студент

Нестеров Иван Алексеевич

Группа № R3237

Преподаватели:

Евстафьев Олег Александрович

Каканов Михаил Александрович

г. Санкт-Петербург

2021

Прогнозирование цен на жилье с помощью нейросетевой регрессионной модели

Необходимо по имеющимся данным о ценах на жильё предсказать окончательную цену каждого дома с учетом характеристик домов с использованием нейронной сети. Описание набора данных содержит 80 классов (набор переменых) классификации оценки типа жилья, и находится в файле `data_description.txt`.

В работе требуется дополнить раздел «Моделирование» в подразделе «Построение и обучение модели» создать и инициализировать последовательную модель нейронной сети с помощью фрэймворков тренировки нейронных сетей как: Torch или Tensorflow. Скомпилировать нейронную сеть выбрав функцию потерь и оптимизатор соответственно. Оценить точность полученных результатов. Вывести предсказанные данные о продаже.

Импорт библиотек

Импортируем необходимые библиотеки:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

Считываем набор данных

Загрузим набор данных и присвоим следующими переменные:

- `train_data` : данные, используемые для обучения модели
- `test_data` : данные, используемые для проверки модели

```
train_data = pd.read_csv('/data/notebook_files/train.csv')
test_data = pd.read_csv('/data/notebook_files/test.csv')
```

Подготовка данных

Отобразим обучающие и проверочные данные:

```
train_data.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	Mis
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN

5 rows × 81 columns

```
test_data.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	ScreenPorch	PoolArea	PoolQ
0	1461	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	AllPub	...	120	0	NaN
1	1462	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN
2	1463	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN
3	1464	60	RL	78.0	9978	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN
4	1465	120	RL	43.0	5005	Pave	NaN	IR1	HLS	AllPub	...	144	0	NaN

5 rows × 80 columns

Как можно видеть, `train_data` имеет на один столбец больше, чем `test_data`, это столбец `SalePrice`, для обучения модели перед применением ее для предсказания меток в test_data.

Проверяем нет ли тестовые данные пустых значений значений (Nan)

Построим функцию `def missing_value_checker` для проверки и подсчёта пропущеных значений в test_data. А также выведем тип данных этих значений.

```
def missing_value_checker(data):
    list = []
    for feature, content in data.items():
        if data[feature].isnull().values.any():

            sum = data[feature].isna().sum()

            type = data[feature].dtype

            print (f'{feature}: {sum}, type: {type}')

            list.append(feature)
    print(list)

    print(len(list))

missing_value_checker(test_data)
```

```
MSZoning: 4, type: object
LotFrontage: 227, type: float64
Alley: 1352, type: object
Utilities: 2, type: object
Exterior1st: 1, type: object
Exterior2nd: 1, type: object
MasVnrType: 16, type: object
MasVnrArea: 15, type: float64
BsmtQual: 44, type: object
BsmtCond: 45, type: object
BsmtExposure: 44, type: object
BsmtFinType1: 42, type: object
BsmtFinSF1: 1, type: float64
BsmtFinType2: 42, type: object
BsmtFinSF2: 1, type: float64
BsmtUnfSF: 1, type: float64
TotalBsmtSF: 1, type: float64
BsmtFullBath: 2, type: float64
BsmtHalfBath: 2, type: float64
KitchenQual: 1, type: object
Functional: 2, type: object
FireplaceQu: 730, type: object
GarageType: 76, type: object
GarageYrBlt: 78, type: float64
GarageFinish: 78, type: object
GarageCars: 1, type: float64
GarageArea: 1, type: float64
GarageQual: 78, type: object
GarageCond: 78, type: object
PoolQC: 1456, type: object
Fence: 1169, type: object
MiscFeature: 1408, type: object
SaleType: 1, type: object
['MSZoning', 'LotFrontage', 'Alley', 'Utilities', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'MasVnr
33
```

Проверяем какие признаки в таблице можно оставить, а какие удалить. Если пропущенных значений слишком много, то удалим признак. Если их небольшое количество, то заполним `mean` или `median` для чисел, новая категория `missing` для строковых объектов.

В соответствии с этим:

– удалим ['Alley', 'FireplaceQu', 'PoolQC', 'Fence', 'MiscFeature'];

– заполним числовое отсутствующее значение значением `mean`;

– заполним строковое отсутствующее значение значением `missing`.

```
test_edited = test_data.drop(['Alley','FireplaceQu','PoolQC', 'Fence', 'MiscFeature'], axis=1)
train_edited = train_data.drop(['Alley','FireplaceQu','PoolQC', 'Fence', 'MiscFeature'], axis=1)

def nan_filler(data):
    for label, content in data.items():
        if pd.api.types.is_numeric_dtype(content):
            data[label] = content.fillna(content.median())
        else:
            data[label] = content.astype("category").cat.as_ordered()
            data[label] = pd.Categorical(content).codes+1

nan_filler(test_edited)
nan_filler(train_edited)
```

Перепроверим наши данные:

```
missing_value_checker(test_edited)
```

```
[]
0
```

```
missing_value_checker(train_edited)
```

```
[]
0
```

```
train_edited.shape, test_edited.shape
```

```
((1460, 76), (1459, 75))
```

```
test_edited.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1459 entries, 0 to 1458
Data columns (total 75 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             1459 non-null   int64  
 1   MSSubClass     1459 non-null   int64  
 2   MSZoning       1459 non-null   int8   
 3   LotFrontage    1459 non-null   float64
 4   LotArea        1459 non-null   int64  
 5   Street         1459 non-null   int8   
 6   LotShape       1459 non-null   int8   
 7   LandContour    1459 non-null   int8   
 8   Utilities      1459 non-null   int8   
 9   LotConfig      1459 non-null   int8   
 10  LandSlope      1459 non-null   int8
```

```
11  Neighborhood   1459 non-null   int8
12  Condition1     1459 non-null   int8
13  Condition2     1459 non-null   int8
14  BldgType       1459 non-null   int8
15  HouseStyle     1459 non-null   int8
16  OverallQual    1459 non-null   int64
17  OverallCond    1459 non-null   int64
18  YearBuilt      1459 non-null   int64
19  YearRemodAdd   1459 non-null   int64
20  RoofStyle      1459 non-null   int8
21  RoofMatl       1459 non-null   int8
22  Exterior1st    1459 non-null   int8
23  Exterior2nd    1459 non-null   int8
24  MasVnrType     1459 non-null   int8
25  MasVnrArea     1459 non-null   float64
26  ExterQual      1459 non-null   int8
27  ExterCond      1459 non-null   int8
28  Foundation     1459 non-null   int8
29  BsmtQual       1459 non-null   int8
30  BsmtCond       1459 non-null   int8
31  BsmtExposure   1459 non-null   int8
32  BsmtFinType1   1459 non-null   int8
33  BsmtFinSF1     1459 non-null   float64
34  BsmtFinType2   1459 non-null   int8
35  BsmtFinSF2     1459 non-null   float64
36  BsmtUnfSF      1459 non-null   float64
37  TotalBsmtSF    1459 non-null   float64
38  Heating        1459 non-null   int8
39  HeatingQC      1459 non-null   int8
40  CentralAir     1459 non-null   int8
41  Electrical     1459 non-null   int8
42  1stFlrSF       1459 non-null   int64
43  2ndFlrSF       1459 non-null   int64
44  LowQualFinSF   1459 non-null   int64
45  GrLivArea      1459 non-null   int64
46  BsmtFullBath   1459 non-null   float64
47  BsmtHalfBath   1459 non-null   float64
48  FullBath       1459 non-null   int64
49  HalfBath       1459 non-null   int64
50  BedroomAbvGr   1459 non-null   int64
51  KitchenAbvGr   1459 non-null   int64
52  KitchenQual    1459 non-null   int8
53  TotRmsAbvGrd   1459 non-null   int64
54  Functional     1459 non-null   int8
55  Fireplaces     1459 non-null   int64
56  GarageType     1459 non-null   int8
57  GarageYrBlt    1459 non-null   float64
58  GarageFinish   1459 non-null   int8
59  GarageCars     1459 non-null   float64
60  GarageArea     1459 non-null   float64
61  GarageQual     1459 non-null   int8
62  GarageCond     1459 non-null   int8
63  PavedDrive     1459 non-null   int8
64  WoodDeckSF     1459 non-null   int64
65  OpenPorchSF    1459 non-null   int64
66  EnclosedPorch  1459 non-null   int64
67  3SsnPorch      1459 non-null   int64
68  ScreenPorch    1459 non-null   int64
69  PoolArea       1459 non-null   int64
70  MiscVal        1459 non-null   int64
71  MoSold         1459 non-null   int64
72  YrSold         1459 non-null   int64
73  SaleType       1459 non-null   int8
```

```
 74  SaleCondition  1459 non-null   int8
dtypes: float64(11), int64(26), int8(38)
memory usage: 476.0 KB
```

```
train_edited.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 76 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   int8
 3   LotFrontage    1460 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   int8
 6   LotShape       1460 non-null   int8
 7   LandContour    1460 non-null   int8
 8   Utilities      1460 non-null   int8
 9   LotConfig      1460 non-null   int8
 10  LandSlope      1460 non-null   int8
 11  Neighborhood   1460 non-null   int8
 12  Condition1     1460 non-null   int8
 13  Condition2     1460 non-null   int8
 14  BldgType       1460 non-null   int8
 15  HouseStyle     1460 non-null   int8
 16  OverallQual    1460 non-null   int64
 17  OverallCond    1460 non-null   int64
 18  YearBuilt      1460 non-null   int64
 19  YearRemodAdd   1460 non-null   int64
 20  RoofStyle      1460 non-null   int8
 21  RoofMatl       1460 non-null   int8
 22  Exterior1st    1460 non-null   int8
 23  Exterior2nd    1460 non-null   int8
 24  MasVnrType     1460 non-null   int8
 25  MasVnrArea     1460 non-null   float64
 26  ExterQual      1460 non-null   int8
 27  ExterCond      1460 non-null   int8
 28  Foundation     1460 non-null   int8
 29  BsmtQual       1460 non-null   int8
 30  BsmtCond       1460 non-null   int8
 31  BsmtExposure   1460 non-null   int8
 32  BsmtFinType1   1460 non-null   int8
 33  BsmtFinSF1     1460 non-null   int64
 34  BsmtFinType2   1460 non-null   int8
 35  BsmtFinSF2     1460 non-null   int64
 36  BsmtUnfSF      1460 non-null   int64
 37  TotalBsmtSF    1460 non-null   int64
 38  Heating        1460 non-null   int8
 39  HeatingQC      1460 non-null   int8
 40  CentralAir     1460 non-null   int8
 41  Electrical     1460 non-null   int8
 42  1stFlrSF       1460 non-null   int64
 43  2ndFlrSF       1460 non-null   int64
 44  LowQualFinSF   1460 non-null   int64
 45  GrLivArea      1460 non-null   int64
 46  BsmtFullBath   1460 non-null   int64
 47  BsmtHalfBath   1460 non-null   int64
 48  FullBath       1460 non-null   int64
```

```
 49  HalfBath       1460 non-null   int64
 50  BedroomAbvGr   1460 non-null   int64
 51  KitchenAbvGr   1460 non-null   int64
 52  KitchenQual    1460 non-null   int8
 53  TotRmsAbvGrd   1460 non-null   int64
 54  Functional     1460 non-null   int8
 55  Fireplaces     1460 non-null   int64
 56  GarageType     1460 non-null   int8
 57  GarageYrBlt    1460 non-null   float64
 58  GarageFinish   1460 non-null   int8
 59  GarageCars     1460 non-null   int64
 60  GarageArea     1460 non-null   int64
 61  GarageQual     1460 non-null   int8
 62  GarageCond     1460 non-null   int8
 63  PavedDrive     1460 non-null   int8
 64  WoodDeckSF     1460 non-null   int64
 65  OpenPorchSF    1460 non-null   int64
 66  EnclosedPorch  1460 non-null   int64
 67  3SsnPorch      1460 non-null   int64
 68  ScreenPorch    1460 non-null   int64
 69  PoolArea       1460 non-null   int64
 70  MiscVal        1460 non-null   int64
 71  MoSold         1460 non-null   int64
 72  YrSold         1460 non-null   int64
 73  SaleType       1460 non-null   int8
 74  SaleCondition  1460 non-null   int8
 75  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), int8(38)
memory usage: 487.7 KB
```

Разделим данные

Поскольку мы не знаем метку (Цена) тестовых данных, для оценки модели, чтобы получить лучшую модель перед прогнозированием тестовых данных, разделим данные в файле train.scv на обучающие и проверочные данные, соотношение составляет 20%.

```
X = train_edited.drop('SalePrice', axis=1)
y = train_edited['SalePrice']

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size = 0.2)
```

```
X_train.shape, test_edited.shape
```

```
((1168, 75), (1459, 75))
```

Моделирование

Построение и обучение модели

Создайте последовательную модель нейронной сети с помощью фрэймворков тренировки нейронных сетей как: Torch или Tensorflow.

```
import tensorflow as tf
from tensorflow.python import keras
from tensorflow.python.keras.layers import Dense

model = keras.Sequential(
    [
        Dense(X_train.shape[1]),
        Dense(350),
        Dense(1),
    ]
)
model2 = keras.Sequential(
    [
        Dense(X_train.shape[1]),
        Dense(350),
        Dense(1),
    ]
)
tf.random.set_seed(40) # Для обеспечения воспроизводимости результатов устанавливается функция seed
```

Скомпилируйте нейронную сеть, выбрав функцию потерь и оптимизатор соответственно.

```
model.compile(loss="mse", optimizer="adam", metrics=['mae'])
model2.compile(loss="mae", optimizer="adam", metrics=['mae'])
```

Обучите модель на обучающих данных `X_train` и `y_train` задав гиперпараметры вашей модели нейронной сети, например количество эпох (epochs), размер мини-выборки (batch_size) и другие.

```
history = model.fit(X_train, y_train, batch_size=5, epochs=30) #замените None на гиперпараметры вашей
history2 = model2.fit(X_train, y_train, batch_size=5, epochs=30)
```

```
Epoch 1/30
  1/234 [..............................] - ETA: 2:25 - loss: 19035731968.0000 - mae: 133863.6719
Epoch 2/30
  1/234 [..............................] - ETA: 0s - loss: 2550011136.0000 - mae: 38448.5547
Epoch 3/30
  1/234 [..............................] - ETA: 0s - loss: 1841714432.0000 - mae: 41661.3203
Epoch 4/30
  1/234 [..............................] - ETA: 0s - loss: 5400638976.0000 - mae: 42203.3594
Epoch 5/30
  1/234 [..............................] - ETA: 0s - loss: 724613440.0000 - mae: 22642.8184
Epoch 6/30
  1/234 [..............................] - ETA: 0s - loss: 838835904.0000 - mae: 20641.6562
Epoch 7/30
  1/234 [..............................] - ETA: 0s - loss: 258506912.0000 - mae: 14396.5371
Epoch 8/30
```

```
  1/234 [..............................] - ETA: 0s - loss: 1547636992.0000 - mae: 30303.7461
Epoch 9/30
  1/234 [..............................] - ETA: 0s - loss: 416856288.0000 - mae: 18278.6504
Epoch 10/30
  1/234 [..............................] - ETA: 0s - loss: 439315968.0000 - mae: 17194.1328
Epoch 11/30
  1/234 [..............................] - ETA: 0s - loss: 1221781504.0000 - mae: 29738.9844
Epoch 12/30
  1/234 [..............................] - ETA: 0s - loss: 99815560.0000 - mae: 9566.0283
Epoch 13/30
  1/234 [..............................] - ETA: 0s - loss: 2305953792.0000 - mae: 43948.4141
Epoch 14/30
  1/234 [..............................] - ETA: 1s - loss: 1075523840.0000 - mae: 26084.4180
Epoch 15/30
  1/234 [..............................] - ETA: 0s - loss: 2130916992.0000 - mae: 39513.0234
Epoch 16/30
  1/234 [..............................] - ETA: 0s - loss: 368485952.0000 - mae: 15882.6016
Epoch 17/30
  1/234 [..............................] - ETA: 0s - loss: 383230016.0000 - mae: 15920.5439
Epoch 18/30
  1/234 [..............................] - ETA: 0s - loss: 3755952128.0000 - mae: 33060.0078
Epoch 19/30
  1/234 [..............................] - ETA: 0s - loss: 1604503808.0000 - mae: 32015.5156
Epoch 20/30
  1/234 [..............................] - ETA: 0s - loss: 4634982400.0000 - mae: 45908.2383
Epoch 21/30
  1/234 [..............................] - ETA: 0s - loss: 962675712.0000 - mae: 24586.9258
Epoch 22/30
  1/234 [..............................] - ETA: 0s - loss: 1171059968.0000 - mae: 30356.5781
Epoch 23/30
  1/234 [..............................] - ETA: 0s - loss: 277584832.0000 - mae: 14737.4766
Epoch 24/30
  1/234 [..............................] - ETA: 0s - loss: 694513536.0000 - mae: 20504.1523
Epoch 25/30
  1/234 [..............................] - ETA: 0s - loss: 346399424.0000 - mae: 15505.2988
Epoch 26/30
  1/234 [..............................] - ETA: 0s - loss: 724792128.0000 - mae: 21476.5977
Epoch 27/30
  1/234 [..............................] - ETA: 0s - loss: 2976626944.0000 - mae: 34428.8984
Epoch 28/30
  1/234 [..............................] - ETA: 0s - loss: 1746408704.0000 - mae: 36836.7617
Epoch 29/30
  1/234 [..............................] - ETA: 0s - loss: 102470832.0000 - mae: 8150.5376
Epoch 30/30
  1/234 [..............................] - ETA: 0s - loss: 487687872.0000 - mae: 14475.2285
Epoch 1/30
  1/234 [..............................] - ETA: 1:27 - loss: 132944.8594 - mae: 132944.8594
Epoch 2/30
  1/234 [..............................] - ETA: 0s - loss: 35594.9961 - mae: 35594.9961
Epoch 3/30
  1/234 [..............................] - ETA: 0s - loss: 19540.7539 - mae: 19540.7539
Epoch 4/30
  1/234 [..............................] - ETA: 0s - loss: 33998.3164 - mae: 33998.3164
Epoch 5/30
  1/234 [..............................] - ETA: 0s - loss: 12484.9238 - mae: 12484.9238
Epoch 6/30
  1/234 [..............................] - ETA: 0s - loss: 23883.4199 - mae: 23883.4199
Epoch 7/30
  1/234 [..............................] - ETA: 0s - loss: 9477.9580 - mae: 9477.9580
Epoch 8/30
  1/234 [..............................] - ETA: 0s - loss: 28446.4121 - mae: 28446.4121
Epoch 9/30
  1/234 [..............................] - ETA: 0s - loss: 20302.9941 - mae: 20302.9941
```

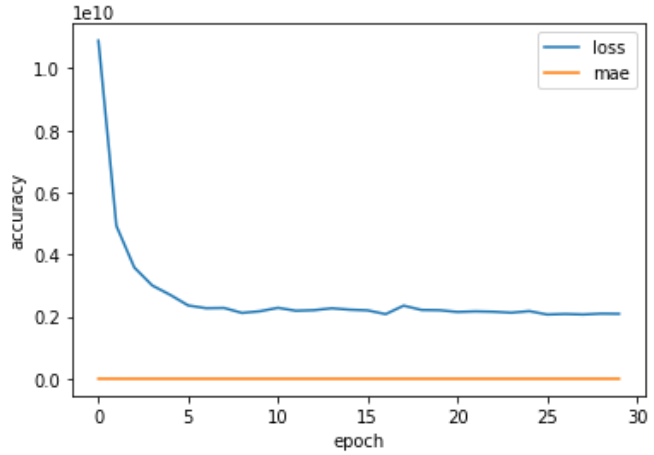
```
Epoch 10/30
  1/234 [..............................] - ETA: 0s - loss: 7293.4346 - mae: 7293.4346
Epoch 11/30
  1/234 [..............................] - ETA: 0s - loss: 25419.3633 - mae: 25419.3633
Epoch 12/30
  1/234 [..............................] - ETA: 0s - loss: 6966.8623 - mae: 6966.8623
Epoch 13/30
  1/234 [..............................] - ETA: 0s - loss: 42092.9609 - mae: 42092.9609
Epoch 14/30
  1/234 [..............................] - ETA: 0s - loss: 41454.4961 - mae: 41454.4961
Epoch 15/30
  1/234 [..............................] - ETA: 0s - loss: 37787.0859 - mae: 37787.0859
Epoch 16/30
  1/234 [..............................] - ETA: 0s - loss: 18583.0742 - mae: 18583.0742
Epoch 17/30
  1/234 [..............................] - ETA: 0s - loss: 18225.5352 - mae: 18225.5352
Epoch 18/30
  1/234 [..............................] - ETA: 0s - loss: 30722.3691 - mae: 30722.3691
Epoch 19/30
  1/234 [..............................] - ETA: 0s - loss: 33331.1562 - mae: 33331.1562
Epoch 20/30
  1/234 [..............................] - ETA: 0s - loss: 42110.1133 - mae: 42110.1133
Epoch 21/30
  1/234 [..............................] - ETA: 1s - loss: 44743.7969 - mae: 44743.7969
Epoch 22/30
  1/234 [..............................] - ETA: 0s - loss: 22340.8574 - mae: 22340.8574
Epoch 23/30
  1/234 [..............................] - ETA: 0s - loss: 13418.6777 - mae: 13418.6777
Epoch 24/30
  1/234 [..............................] - ETA: 0s - loss: 19408.4883 - mae: 19408.4883
Epoch 25/30
  1/234 [..............................] - ETA: 0s - loss: 16294.3594 - mae: 16294.3594
Epoch 26/30
  1/234 [..............................] - ETA: 0s - loss: 22073.1719 - mae: 22073.1719
Epoch 27/30
  1/234 [..............................] - ETA: 0s - loss: 35871.1250 - mae: 35871.1250
Epoch 28/30
  1/234 [..............................] - ETA: 0s - loss: 35853.4141 - mae: 35853.4141
Epoch 29/30
  1/234 [..............................] - ETA: 0s - loss: 9964.6768 - mae: 9964.6768
Epoch 30/30
  1/234 [..............................] - ETA: 0s - loss: 17638.7129 - mae: 17638.7129
```

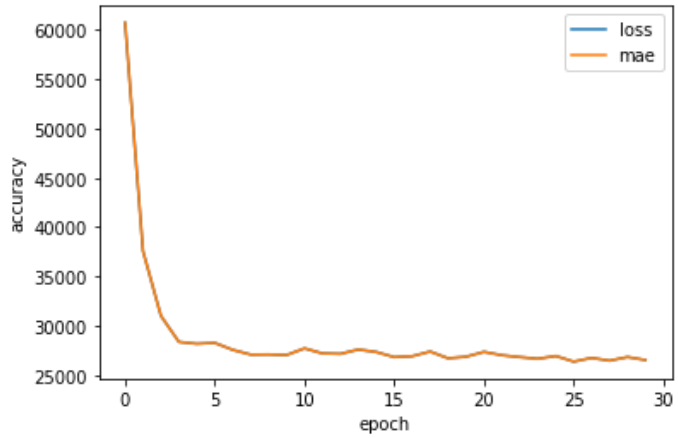
Оцените полученные результаты

```
pd.DataFrame(history.history).plot()
plt.ylabel('accuracy')
plt.xlabel('epoch')
print(history.history)

pd.DataFrame(history2.history).plot()
plt.ylabel('accuracy')
plt.xlabel('epoch')
print(history2.history)
```

```
{'loss': [10893107200.0, 4925400576.0, 3582751744.0, 3004034560.0, 2701861888.0, 2360802304.0, 227199
{'loss': [60699.0546875, 37575.70703125, 31008.03515625, 28373.830078125, 28187.349609375, 28276.3105
```






```
scores = model.evaluate(X_val, y_val, verbose=1)
```

```
 1/10 [==>...........................] - ETA: 0s - loss: 2370901760.0000 - mae: 29708.2461
```

Предсказание

```
preds = model.predict(test_edited)
preds
```

```
output = pd.DataFrame(
{
    'Id':test_data['Id'],
    'SalePrice': np.squeeze(preds)
})
output
#print (output)
```

	Id	SalePrice
0	1461	150961.609375
1	1462	63411.207031
2	1463	188760.093750
3	1464	194778.437500
4	1465	163656.078125
...	...	...
1454	2915	71402.968750
1455	2916	95280.984375
1456	2917	191611.468750
1457	2918	79910.664062
1458	2919	230057.718750

1459 rows × 2 columns

При выполнении:

Выведите отчет нейросетевой регрессионной модели, для прогнозирование цен на жилье.

Подберить разные комбинации гиперпараметров таким образом, чтобы получить лучший результат на тестовом наборе данных.

Попробуйте использовать разное количество нейронов на входном слое, например 100, 150, 200 300.

Добавьте в нейронную сеть скрытый слой с разным количеством нейронов.

Используйте разное количество эпох: 10, 15, 20, 25, 30.

Используйте разные размеры мини-выборки (batch_size): 10, 50, 100, 200.

Попробуйте использовать разные значения оптимизатора `optimizers` и функции потерь `loss`. Сравните полученые результаты.

Вопросы:

Как выше перечисленные параметры влияют на полученный вами результат?

Что такое эпоха (Epoch)? В чем отличие от итерации (Iteration)?

Что такое функция активации? Какие вам известны?

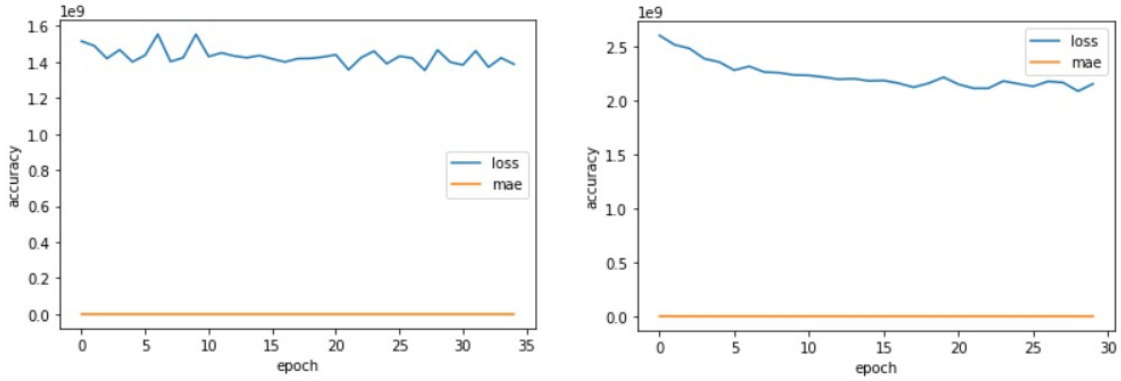
Что такое MSE(Mean Squared Error) - Средняя квадратичная ошибка? Что такое MAE(Mean Absolute Error)? Для чего используются.

Ход выполнения:

В ходе подбора и инициализации модели мною было выявлено пять наиболее влияющих на ее работу составляющих:

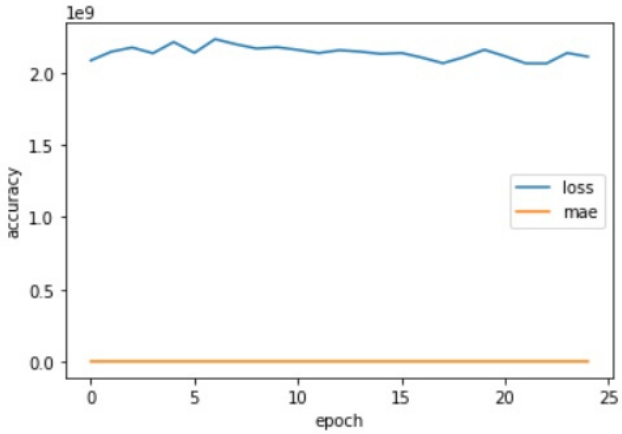
1. Количество нейронов на входном слое;
2. Количество эпох;
3. Размер мини-выборки;
4. Оптимизаторы;
5. Функции потерь;

Далее для выбора наиболее эффективной модели я начал последовательно перебирать различные комбинации настроек данных параметров. Логично предположить, что количество нейронов на входном слое влияет на сложность всей нейронной сети в целом, что также подтвердила и нагрузка, производимая на систему во время обучения, длительность работы на каждой из эпох. Также на сложность системы влияет и количество эпох: чем их больше, тем больше своеобразных «итераций цикла», которые необходимо пройти модели при обучении. Поэтому возникали опасения переобучить модель, чрезмерно усложнив параметры и слишком хорошо натренировать ее на обучающих данных, потеряв при этом предсказательную силу. Анализ результатов модели в конце лабораторной работы позволял отслеживать это. В результате пришел к такому выводу, что не рационально делать количество нейронов больше 350, а количество эпох – больше 30, так как в таком случае модель показывала совершенно не подходящие результаты на валидационной выборке (350 нейронов и 35 эпох для левого графика, и 400 нейронов и 30 эпох для правого):



Дальнейшее увеличение параметров приводило к примерно таким же результатам, в отчете показываю «граничные» значения. Уменьшение же

данных параметров тоже не давало хороших результатов, что представлено на рисунке ниже (300 нейронов и 20 эпох):



Рассмотрим оставшиеся параметры. Однако по сравнению с вышеизложенными они, на мой взгляд, не играли такой большой роли в «качестве» модели. К примеру, были проверены оптмизаторы «nadam» и «sgd», но большого влияния на сеть они не оказали, и предыдущие параметры на модель влияли все же сильнее. То же самое можно сказать и о функции потерь. Однако здесь ситуация несколько иная. Значения функции потерь кратно больше значения MAE, поэтому на первом графике отчета оранжевая линия MAE выглядит как прямая линия, но на самом деле таковой не является. Для демонстрации этого была параллельна проинициализирована и обучена схожая модель, где разница заключалась лишь в анализируемой функции потерь:

```
model.compile(loss="mse", optimizer="adam", metrics=['mae'])
model2.compile(loss="mae", optimizer="adam", metrics=['mae'])
```

При этом вывод графика результатов первой модели позволил наглядно показать убывание функции потерь согласно картинке метода локтя, а второй модели – убывание значения MAE, что можно увидеть в отчете.

Ответы на вопросы:

1. Как вышеперечисленные параметры влияют на полученный вами результат? – описано выше.

2. Что такое эпоха (Epoch)? В чем отличие от итерации (Iteration)? – Эпоха – проход данных вперед и назад по нейронной сети, то есть это цикл полного набора обучающих данных, а итерации - это количество пакетов или шагов через разделенные пакеты обучающих данных, необходимых для завершения одной эпохи.

3. Что такое функция активации? Какие вам известны? – Это функция, которая определяет выходной сигнал нейрона сети, опираясь при этом на входной сигнал (или на набор таковых). Знаю функцию единичной ступеньки, арктангенса, тождественную. Однако большинство из них устроены так, чтобы возвращать число от 0 до 1, что удобно при определении весов нейронов, поэтому множество функций похожи друг на другую.

4. Что такое MSE (Mean Squared Error) - Средняя квадратичная ошибка? Что такое MAE(Mean Absolute Error)? Для чего используются. – MAE - это мера ошибок между парными наблюдениями, выражающими одно и то же явление. Примеры Y по сравнению с X включают сравнения прогнозируемого и наблюдаемого, последующего времени и начального времени, а также один метод измерения по сравнению с альтернативным методом измерения. Средней квадратичной ошибкой называется среднее квадратичное значение из суммы квадратов ошибок отдельных измерений. MSE - статистика, среднеквадратичная ошибка, а MSE – функция риска, которая измеряет среднее значение квадратов ошибок — то есть среднюю квадратическую разницу между оценочными значениями и фактическим значением. а