РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра математического моделирования и искусственного интеллекта

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7

Дисциплина: Интеллектуальный анализ данных

Студент: Генералов Даниил

Группа: НПИбд-01-21

Москва 2024

- Считайте заданный набор данных из репозитария UCI, включая указанный в индивидуальном задании столбец с метками классов и столбец с откликом (зависимой переменной).
- Преобразуйте в числовые признаки неправильно распознанные признаки с числовыми значениями. Если в столбцах с метками классов и откликом имеются пропущенные значения, то удалите записи с пропущенными значениями. Оставьте в наборе данных только числовые признаки.
- Если в наборе данных остались пропущенные значения, то замените пропущенные значения, используя метод, указанный в индивидуальном задании. Если пропущенные значения в наборе данных отсутствуют, то определите и удалите точки с выбросами в соответствии с методом, указанным в индивидуальном задании. Выберите параметры методов таким образом, чтобы выбросы составляли не менее 5% всех точек набора данных.
- Масштабируйте признаки набора данных на интервал [0, 1]. Используя метод снижения размерности данных, указанный в индивидуальном задании, оставьте в наборе данных три признака (кроме метки класса и откликов), принимающих более 50 различных значений.
- Визуализируйте набор данных в виде точек в трехмерном пространстве, отображая точки разных классов разными цветами. В

качестве подписей осей используйте названия признаков. В подписи рисунка укажите название набора данных. Создайте легенду набора данных.

- Разбейте набор данных на обучающую и тестовую выборки. Постройте регрессоры на базе моделей регрессии, указанных в индивидуальном задании, для каждого из трех признаков. Определите оптимальные параметры регрессоров при помощи GridSearchCV.

- Для каждого из трех признаков визуализируйте на плоскости набор данных одним цветом и линии регрессии для регрессоров с оптимальными параметрами, определенными в п. 6 (всего три рисунка). Регрессоры, имеющие максимальное значение показателя качества регрессии, указанного в индивидуальном задании, выделите красным цветом. В качестве подписи оси X используйте название признака, в качестве подписи оси Y – название столбца с откликами. Создайте легенду для линий регрессии.

- Постройте на одном рисунке кривые обучения (зависимость показателя качества регрессии, указанного в индивидуальном задании, от количества точек в обучающей выборке) для трех лучших регрессоров для каждого из трех признаков по показателю качества, указанному в индивидуальном задании. Кривые для регрессора с максимальным показателем качества визуализируйте красным цветом (кривую для обучающей выборки сплошной линией, кривую для тестовой выборки линией из точек). Подпишите корректно оси и создайте легенду для кривых обучения.

Вариант 13

Horse Colic Data Set

Название файла: horse-colic.data

Ссылка: http://archive.ics.uci.edu/ml/datasets/Horse+Colic

Класс: outcome (столбец No 23)

Зависимая переменная: total protein (столбец No 20)

Метод обработки пропущенных значений – медиана класса

Метод обработки выбросов – алгоритм кластеризации OPTICS

Метод снижения размерности данных – отбор на основе важности признаков (ExtraTreesClassifier)

Регрессоры:

- Полиномиальная регрессия (PolynomialFeatures+LinearRegression), параметр degree в диапазоне от 2 до 5

- гребневая регрессия (Ridge), параметр alpha в диапазоне от 0.1 до 1

- регрессии на основе метода опорных векторов (SVR), параметр degree в диапазоне от 1 до 5

Показатели качества регрессии:

- Для определения лучшего регрессора MAPE

- Для визуализации кривой обучения MaxErr

1. открыть базу данных и прочитать значения

In [1]:
```
from ucimlrepo import fetch_ucirepo

# fetch dataset
horse_colic = fetch_ucirepo(id=47)
```

In [2]:
```
horse_colic['data'].keys()
```

Out[2]:
```
dict_keys(['ids', 'features', 'targets', 'original', 'headers'])
```

In [3]:
```
horse_colic['data']['features']
```

Out[3]:

	surgery	age	hospital_number	rectal_temperature	pulse	respiratory_ra
0	2.0	1	530101	38.5	66.0	28
1	1.0	1	534817	39.2	88.0	20
2	2.0	1	530334	38.3	40.0	24
3	1.0	9	5290409	39.1	164.0	84
4	2.0	1	530255	37.3	104.0	35
...	...	...	...	...	...	
363	2.0	1	529695	38.6	60.0	30
364	2.0	1	528452	37.8	42.0	40
365	1.0	1	534783	38.0	60.0	12
366	2.0	1	528926	38.0	42.0	12
367	2.0	1	530670	37.6	88.0	30

368 rows × 27 columns

In [4]:
```
horse_colic['data']['features']['outcome'].unique()
```

Out[4]:
```
array([ 2.,  3.,  1., nan])
```

In [5]:
```
horse_colic['data']['features']['total_protein'].unique()
```

Out[5]:
```
array([ 8.4, 85. ,  6.7,  7.2,  7.4,  nan,  7. ,  8.3,  6.2,  6. ,  7.8,
        6.1, 81. ,  6.8,  8.7, 70. , 65. ,  5.5, 76. ,  7.5,  8.2,  6.6,
        8.6, 80. ,  6.5,  8.5, 67. , 69. ,  9.1,  7.7,  6.4,  8.1,  5.9,
        8. , 82. , 72. , 74. ,  6.3,  7.6,  4.9, 57. , 68. , 77. ,  8.9,
        5.7,  4.5, 61. , 86. , 60. , 66. ,  5.3,  7.3, 64. , 58. , 56. ,
       75. , 10.2, 62. ,  7.9, 73. , 71. , 63. , 46. ,  5.8, 53. ,  4.7,
       59. , 55. , 89. , 51. ,  4. , 11. ,  8.8, 50. , 54. ,  6.9,  4.6,
       13. ,  7.1,  3.3,  9. ,  5. ,  3.5, 36. , 79. ])
```

In [6]:
```
data = horse_colic['data']['features']
```

2. предобработка данных

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 368 entries, 0 to 367
Data columns (total 27 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   surgery                         366 non-null    float64
 1   age                             368 non-null    int64  
 2   hospital_number                 368 non-null    int64  
 3   rectal_temperature              299 non-null    float64
 4   pulse                           342 non-null    float64
 5   respiratory_rate                297 non-null    float64
 6   temperature_of_extremities      303 non-null    float64
 7   peripheral_pulse                285 non-null    float64
 8   mucous_membranes                320 non-null    float64
 9   capillary_refill_time           330 non-null    float64
 10  pain                            305 non-null    float64
 11  peristalsis                     316 non-null    float64
 12  abdominal_distension            303 non-null    float64
 13  nasogastric_tube                237 non-null    float64
 14  nasogastric_reflux              235 non-null    float64
 15  nasogastric_reflux_ph           69 non-null     float64
 16  rectal_examination_feces        240 non-null    float64
 17  abdomen                         225 non-null    float64
 18  packed_cell_volume              331 non-null    float64
 19  total_protein                   325 non-null    float64
 20  abdominocentesis_appearance     174 non-null    float64
 21  abdominocentesis_total_protein  133 non-null    float64
 22  outcome                         366 non-null    float64
 23  lesion_site                     368 non-null    int64  
 24  lesion_type                     368 non-null    int64  
 25  lesion_subtype                  368 non-null    int64  
 26  cp_data                         368 non-null    int64  
dtypes: float64(21), int64(6)
memory usage: 77.8 KB
```

In [8]:
```
data = data[data['outcome'].notna()]
```

In [9]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 366 entries, 0 to 367
Data columns (total 27 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   surgery                         366 non-null    float64
 1   age                             366 non-null    int64  
 2   hospital_number                 366 non-null    int64  
 3   rectal_temperature              297 non-null    float64
 4   pulse                           340 non-null    float64
 5   respiratory_rate                295 non-null    float64
 6   temperature_of_extremities      302 non-null    float64
 7   peripheral_pulse                283 non-null    float64
 8   mucous_membranes                318 non-null    float64
 9   capillary_refill_time           328 non-null    float64
 10  pain                            303 non-null    float64
 11  peristalsis                     315 non-null    float64
 12  abdominal_distension            302 non-null    float64
 13  nasogastric_tube                236 non-null    float64
 14  nasogastric_reflux              234 non-null    float64
 15  nasogastric_reflux_ph           69 non-null     float64
 16  rectal_examination_feces        239 non-null    float64
 17  abdomen                         223 non-null    float64
 18  packed_cell_volume              330 non-null    float64
 19  total_protein                   324 non-null    float64
 20  abdominocentesis_appearance     173 non-null    float64
 21  abdominocentesis_total_protein  132 non-null    float64
 22  outcome                         366 non-null    float64
 23  lesion_site                     366 non-null    int64  
 24  lesion_type                     366 non-null    int64  
 25  lesion_subtype                  366 non-null    int64  
 26  cp_data                         366 non-null    int64  
dtypes: float64(21), int64(6)
memory usage: 80.1 KB
```

3. замена пропущенных значений

In [10]:
```
for col in data.columns:
    data.loc[:, col] = data[col].fillna(data[col].median())

data
```

Out[10]:

	surgery	age	hospital_number	rectal_temperature	pulse	respiratory_ra
0	2.0	1	530101	38.5	66.0	28
1	1.0	1	534817	39.2	88.0	20
2	2.0	1	530334	38.3	40.0	24
3	1.0	9	5290409	39.1	164.0	84
4	2.0	1	530255	37.3	104.0	35
...	...	...	...	...	...	
363	2.0	1	529695	38.6	60.0	30
364	2.0	1	528452	37.8	42.0	40
365	1.0	1	534783	38.0	60.0	12
366	2.0	1	528926	38.0	42.0	12
367	2.0	1	530670	37.6	88.0	30

366 rows × 27 columns

4. масштабирование и снижение размерности

In [11]:
```
RESPONSES = ['outcome', 'total_protein']
```

In [12]:
```
for col in data.columns:
    if data[col].nunique() < 50:
        if col not in RESPONSES:
            data = data.drop(col, axis=1)
```

In [ ]:
```
for col in data.columns:
    if col not in RESPONSES:
        coldata = data[col]
        data.loc[:, col] = (coldata - coldata.min()) / (coldata.max() - colc

data
```

```
/tmp/ipykernel_1825/1264626501.py:4: FutureWarning: Setting an item of incom
patible dtype is deprecated and will raise in a future error of pandas. Valu
e '[3.30220084e-03 4.28647458e-03 3.35083014e-03 9.96823442e-01
 3.33434210e-03 2.93779415e-03 2.61366838e-03 3.19909838e-03
 3.29176537e-03 9.98747743e-01 2.97807507e-03 2.84846651e-03
 2.87017229e-03 9.97015455e-01 4.30734553e-03 3.32975049e-03
 9.99079591e-01 2.57964874e-03 9.96838469e-01 3.72149812e-03
 3.31388857e-03 3.14337295e-03 1.54486731e-03 4.32425099e-03
 4.05167646e-03 3.18052323e-03 3.39090236e-03 2.99998956e-03
 9.94534526e-01 4.41128283e-03 3.17154873e-03 3.33162887e-03
 3.16153068e-03 3.21078610e-03 3.23854446e-03 3.03317437e-03
 3.17530550e-03 4.08402642e-03 3.07157690e-03 9.94110219e-01
 3.17634905e-03 9.96372630e-01 3.34394273e-03 4.13035992e-03
 4.40961316e-03 3.24501445e-03 3.25774573e-03 3.24376220e-03
 3.04945370e-03 3.20640321e-03 3.23228318e-03 2.80234172e-03
 3.17321840e-03 3.39465913e-03 3.16862680e-03 9.95243512e-01
 3.04569693e-03 3.20598579e-03 2.92589771e-03 3.05237564e-03
 3.37253592e-03 2.85472779e-03 2.99518925e-03 4.17711084e-03
 3.21162094e-03 4.30066683e-03 4.27958717e-03 3.05801079e-03
 3.23457898e-03 3.33872499e-03 2.90836612e-03 3.16090455e-03
 9.98669269e-01 2.96158703e-03 9.97349390e-01 4.13516024e-03
 4.24431528e-03 4.27311718e-03 3.42575683e-03 2.75162532e-03
 2.76310434e-03 4.27374331e-03 9.96896491e-01 9.94613835e-01
 3.24960606e-03 3.13585941e-03 3.15339101e-03 2.86495455e-03
 3.21913448e-03 4.40418671e-03 4.15707473e-03 3.16862680e-03
 3.33350726e-03 3.34582112e-03 3.01585148e-03 4.35764451e-03
 3.34769950e-03 4.14622184e-03 4.14413474e-03 3.28153861e-03
 4.40063865e-03 4.30358876e-03 4.31172843e-03 9.97257558e-01
 2.77249627e-03 3.36063949e-03 3.00312021e-03 3.23186576e-03
 4.05251130e-03 9.98278147e-01 2.76853079e-03 4.06085967e-03
 4.31694616e-03 2.84971876e-03 3.77137967e-03 4.05814645e-03
 4.36807998e-03 9.96838678e-01 4.11241091e-03 4.35931418e-03
 3.00604214e-03 4.23200142e-03 4.23680174e-03 3.35625659e-03
 3.24772768e-03 2.95991735e-03 3.36126561e-03 9.91007127e-01
 4.39354253e-03 2.80067204e-03 3.06489820e-03 3.42095652e-03
 3.34248176e-03 2.76435660e-03 9.96149311e-01 3.42575683e-03
 4.18358083e-03 3.77137967e-03 4.21843531e-03 3.27277281e-03
 1.81577217e-03 3.36502238e-03 3.16090455e-03 4.22427917e-03
 4.37601094e-03 4.13996055e-03 3.28132990e-03 3.12918071e-03
 4.39020318e-03 4.07275611e-03 3.15568681e-03 4.08173062e-03
 3.03150469e-03 4.35868805e-03 2.91525353e-03 3.07157690e-03
 4.10093189e-03 2.91546224e-03 2.89166936e-03 2.84617070e-03
 3.37107496e-03 4.36140128e-03 4.10385382e-03 2.97285734e-03
 4.30901520e-03 2.84617070e-03 2.92610642e-03 3.33413339e-03
 4.34240872e-03 3.90411888e-03 2.79628915e-03 4.38561157e-03
 3.22456093e-03 8.75953541e-04 2.84888393e-03 4.35180064e-03
 1.85980987e-03 4.10928027e-03 2.89521742e-03 3.33100274e-03
 2.92735868e-03 9.94613835e-01 4.29482296e-03 4.12702057e-03
 4.28981394e-03 9.94302650e-01 2.73117180e-03 4.09237480e-03
 9.98742317e-01 3.01856471e-03 9.96170182e-01 4.28042201e-03
 3.15026037e-03 4.14997861e-03 4.05272001e-03 3.20348127e-03
 1.48601125e-03 4.09195738e-03 4.14872635e-03 2.87351164e-03
 2.99748505e-03 4.13119476e-03 3.21537771e-03 3.18261033e-03
 2.89521742e-03 4.35326161e-03 3.34310789e-03 4.39521221e-03
 4.21572208e-03 3.36481367e-03 3.25878928e-03 9.97649932e-01
 2.90189613e-03 3.11060557e-03 3.10058752e-03 3.06761142e-03
```

```
 9.94534317e-01 4.37475868e-03 3.22602189e-03 4.33072099e-03
 4.33113841e-03 3.03066985e-03 2.90085258e-03 4.40648252e-03
 3.05801079e-03 2.96868315e-03 9.96616820e-01 4.36557546e-03
 4.24055851e-03 4.17439761e-03 3.28696505e-03 9.91007545e-01
 3.39090236e-03 3.01585148e-03 2.96158703e-03 2.90106129e-03
 4.06378161e-03 3.01021633e-03 4.15415280e-03 3.10580525e-03
 3.08180367e-03 3.35500433e-03 9.94878688e-01 9.97032152e-01
 3.05237564e-03 3.35750884e-03 3.31660179e-03 2.80296785e-03
 2.89897419e-03 4.33364292e-03 2.85117973e-03 9.97096852e-01
 3.39820719e-03 4.06127709e-03 9.97685412e-01 4.08903545e-03
 3.24188381e-03 4.26602106e-03 2.80818558e-03 4.31068488e-03
 3.13418974e-03 1.00000000e+00 3.01877341e-03 3.38088430e-03
 2.98684087e-03 3.25294541e-03 2.82801298e-03 4.20006887e-03
 2.83928329e-03 2.98266668e-03 4.24661108e-03 3.23854446e-03
 2.99685892e-03 4.24619366e-03 9.98232231e-01 4.28021330e-03
 2.99310215e-03 2.86766778e-03 3.10830977e-03 4.25036785e-03
 3.41135588e-03 2.76853079e-03 2.81319461e-03 3.37274463e-03
 9.95367068e-01 9.93651685e-01 9.99895645e-01 3.16173938e-03
 3.09870913e-03 4.33593872e-03 3.04945370e-03 3.28821731e-03
 4.11679381e-03 4.09550544e-03 4.09216609e-03 2.80150688e-03
 3.15297359e-03 3.40885137e-03 4.24494140e-03 2.99435441e-03
 2.85326682e-03 4.37955900e-03 4.22511401e-03 3.05696724e-03
 4.30838908e-03 2.78898432e-03 9.94613627e-01 9.93651476e-01
 9.94302858e-01 3.06385465e-03 4.30818037e-03 3.07220303e-03
 2.87768583e-03 9.97015455e-01 3.17217486e-03 3.27924280e-03
 3.28800860e-03 9.98626692e-01 4.21968756e-03 2.94071608e-03
 3.30345310e-03 3.33100274e-03 3.38651946e-03 3.19075000e-03
 3.19701128e-03 4.20549532e-03 3.20368998e-03 2.94342930e-03
 4.30338005e-03 3.20076805e-03 2.46506726e-03 3.23207447e-03
 2.92840222e-03 3.26546798e-03 2.82342137e-03 9.94951736e-01
 3.42617425e-03 4.09279222e-03 4.07734772e-03 2.79357592e-03
 9.91007336e-01 2.91963643e-03 3.05550628e-03 2.75809532e-03
 3.27694700e-03 4.10635833e-03 9.94878896e-01 4.25913365e-03
 4.21509595e-03 9.93590533e-01 4.07734772e-03 4.14851764e-03
 0.00000000e+00 2.95407349e-03 2.75204274e-03 4.17189310e-03
 4.39500350e-03 4.13912572e-03 3.40133783e-03 3.26651153e-03
 9.93590533e-01 3.21746481e-03 2.95803897e-03 4.27937846e-03
 3.05696724e-03 3.42095652e-03]' has dtype incompatible with int64, please e
xplicitly cast to a compatible dtype first.
  data.loc[:, col] = (coldata - coldata.min()) / (coldata.max() - coldata.mi
n())
/tmp/ipykernel_1825/1264626501.py:4: FutureWarning: Setting an item of incom
patible dtype is deprecated and will raise in a future error of pandas. Valu
e '[0.27487229 0.05370956 0.         0.05370956 0.10459742 0.
 0.07599124 0.05370956 0.07796157 0.         0.05166626 0.05135004
 0.10031622 0.         0.05137436 0.07801022 0.03405497 0.
 0.10228655 0.         0.05135004 0.         0.07567502 0.07567502
 0.07567502 0.         0.07567502 0.10026757 0.10459742 0.
 0.10228655 0.05368523 0.05373388 0.05370956 0.02734128 0.17297495
 0.12661153 0.07567502 0.         0.05139869 0.10228655 0.22865483
 0.07567502 0.07796157 0.07796157 0.07796157 0.05606908 0.
 0.17297495 0.10459742 0.07567502 0.17535879 0.07567502 0.07567502
 0.18000486 0.17297495 0.07569934 0.07805887 0.07796157 0.10228655
 0.05166626 0.         0.07796157 0.05373388 0.05139869 0.05370956
 0.05648261 0.07567502 0.05166626 0.07567502 0.05373388 0.
 0.05139869 0.05137436 0.2705911  0.03405497 0.07796157 0.
```

```
 0.07358307 0.05370956 0.05363658 0.07567502 0.07567502 0.07796157
 0.12663586 0.00972999 0.10228655 0.05370956 0.12162491 0.
 0.05648261 0.07796157 0.         0.07796157 0.         0.07796157
 0.07567502 0.07796157 0.         0.10459742 0.         0.07796157
 0.         0.17297495 0.1313549  0.75675018 0.07805887 0.
 0.07796157 0.05363658 0.05137436 0.07567502 0.07567502 0.07796157
 0.07796157 0.         0.07796157 0.10228655 0.07569934 0.
 0.05366091 0.05370956 0.         0.07796157 0.05363658 0.05363658
 0.75675018 0.05370956 0.05370956 0.05366091 0.05139869 0.03405497
 0.00972999 0.10228655 0.05363658 0.07358307 0.07567502 0.
 0.         0.12464121 0.03405497 0.07796157 0.         0.03405497
 0.         0.07567502 0.05373388 0.05137436 0.         0.
 0.05368523 0.05373388 0.07567502 0.10031622 0.00972999 0.
 0.05363658 0.07567502 0.         0.10228655 0.1730236  0.
 0.15103381 0.05370956 0.03405497 0.         0.07567502 0.05166626
 0.07796157 0.         0.1313549  0.05366091 0.05368523 0.08270494
 0.05370956 0.14867429 0.10228655 0.14864996 0.02702505 0.05594746
 0.07567502 0.05363658 0.         0.         0.1313549  0.51350036
 0.17535879 0.1        0.75675018 0.07621017 0.17297495 0.12432498
 0.07796157 0.         0.         0.00729749 0.03405497 0.
 0.         0.07796157 0.07805887 0.05370956 0.75675018 0.07577232
 0.10031622 0.14864996 0.05370956 0.07796157 0.         0.1313549
 0.07567502 0.05166626 0.05137436 0.         0.07567502 0.
 0.05373388 0.07569934 0.05135004 0.07796157 0.05370956 0.05363658
 0.05370956 0.07567502 0.07567502 0.05363658 0.21892484 0.07796157
 0.10228655 0.05166626 0.08027244 0.05139869 0.07572367 0.00972999
 0.05366091 0.07567502 0.         0.05373388 0.07796157 0.07796157
 0.         0.05363658 0.05370956 0.22865483 1.         0.07805887
 0.05373388 0.05363658 0.07567502 0.05139869 0.12432498 0.05370956
 0.00972999 0.07567502 0.17297495 0.17535879 0.05363658 0.
 0.27730479 0.         0.07567502 0.         0.29695938 0.05373388
 0.12663586 0.12464121 0.05166626 0.         0.75675018 0.05373388
 0.10231087 0.05166626 0.10228655 0.2705911  0.1023352  0.07567502
 0.07567502 0.         0.20189735 0.         0.20432985 0.03405497
 0.07599124 0.         0.75675018 0.05363658 0.         0.05166626
 0.20432985 0.         0.17297495 0.75675018 0.05373388 0.
 0.07796157 0.05370956 0.07796157 0.05370956 0.14867429 0.07567502
 0.07567502 0.07796157 0.05363658 0.07567502 0.07567502 0.
 0.         0.05363658 0.05370956 0.02734128 0.14867429 0.07796157
 0.07599124 0.         0.12432498 0.17297495 0.05363658 0.03405497
 0.07796157 0.07805887 0.07599124 0.14864996 0.05373388 0.
 0.         0.         0.05366091 0.00972999 0.05139869 0.
 0.12430066 0.07796157 0.05594746 0.05135004 0.05139869 0.75675018
 0.07567502 0.         0.         0.         0.05370956 0.07805887
 0.07796157 0.17297495 0.00972999 0.10231087 0.05370956 0.07567502
 0.05373388 0.03405497 0.10031622 0.1730236  0.75675018 0.
 0.17297495 0.20445147 0.10231087 0.14867429 0.05373388 0.07572367
 0.05363658 0.07796157 0.10031622 0.05373388 0.07567502 0.05137436]' has dty
pe incompatible with int64, please explicitly cast to a compatible dtype fir
st.
  data.loc[:, col] = (coldata - coldata.min()) / (coldata.max() - coldata.mi
n())
```

Out[ ]:

	hospital_number	pulse	packed_cell_volume	total_protein	outcome
0	0.003302	0.233766	0.577465	8.4	2.0
1	0.004286	0.376623	0.647887	85.0	3.0
2	0.003351	0.064935	0.408451	6.7	1.0
3	0.996823	0.870130	0.619718	7.2	2.0
4	0.003334	0.480519	0.985915	7.4	2.0
...	...	...	...	...	...
363	0.003217	0.194805	0.507042	6.0	1.0
364	0.002958	0.077922	0.450704	6.2	1.0
365	0.004279	0.194805	0.563380	65.0	3.0
366	0.003057	0.077922	0.464789	5.8	1.0
367	0.003421	0.376623	0.563380	6.0	2.0

366 rows × 6 columns

In [14]:
```
# find the most important features using ExtraTreesClassifier
from sklearn.ensemble import ExtraTreesClassifier

FEATURES = list(set(data.columns) - set(RESPONSES))

X = data[FEATURES]
y = data['outcome']

model = ExtraTreesClassifier(n_estimators=5000)
model.fit(X, y)
print(model.feature_importances_)
```

```
[0.24323298 0.22945494 0.26896993 0.25834215]
```

In [15]:
```
import numpy as np

argmin = np.argmin(model.feature_importances_)

data = data.drop(FEATURES[argmin], axis=1)

FEATURES = list(set(data.columns) - set(RESPONSES))
data
```

Out[15]:

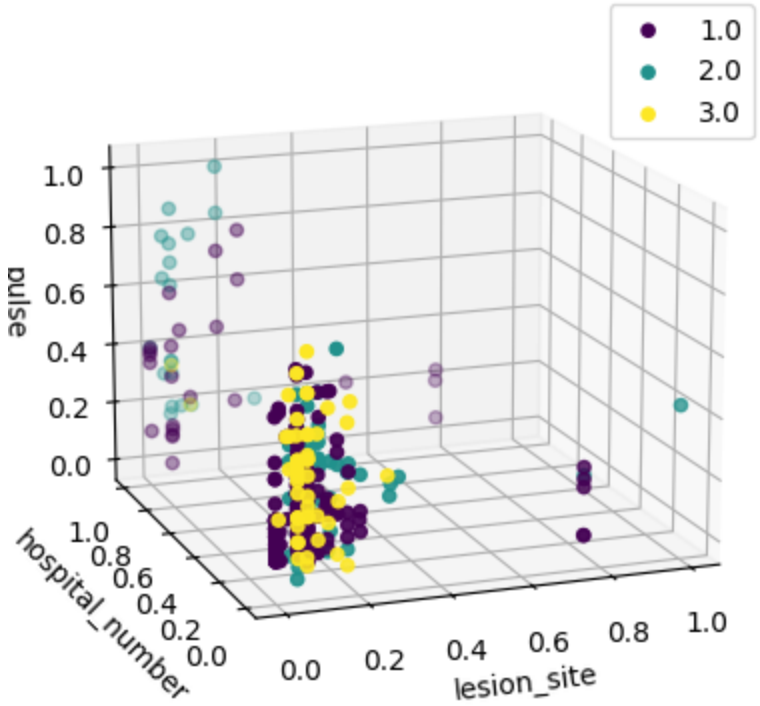
	hospital_number	pulse	total_protein	outcome	lesion_site
0	0.003302	0.233766	8.4	2.0	0.274872
1	0.004286	0.376623	85.0	3.0	0.053710
2	0.003351	0.064935	6.7	1.0	0.000000
3	0.996823	0.870130	7.2	2.0	0.053710
4	0.003334	0.480519	7.4	2.0	0.104597
...	...	...	...	...	...
363	0.003217	0.194805	6.0	1.0	0.077962
364	0.002958	0.077922	6.2	1.0	0.100316
365	0.004279	0.194805	65.0	3.0	0.053734
366	0.003057	0.077922	5.8	1.0	0.075675
367	0.003421	0.376623	6.0	2.0	0.051374

366 rows × 5 columns

5. визуализация

In [16]:
```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

CLASSES = list(set(data['outcome']))

for i in CLASSES:
    df_k = data[data['outcome'] == i]
    print(i, len(df_k))
    ax.scatter(df_k[FEATURES[0]], df_k[FEATURES[1]], df_k[FEATURES[2]], c=[i


ax.set_xlabel(FEATURES[0])
ax.set_ylabel(FEATURES[1])
ax.set_zlabel(FEATURES[2])

ax.legend()

ax.view_init(15, -110)

plt.show()
```

```
1.0 225
2.0 89
3.0 52
```



6. регрессия

In [ ]:
```
# test-train split
from sklearn.model_selection import train_test_split

X = data[FEATURES]
y = data['total_protein']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra
X_train
```

Out[ ]:

	lesion_site	hospital_number	pulse
298	0.053710	0.003409	0.454545
166	0.000000	0.002846	0.064935
117	0.102287	0.996839	0.194805
216	0.000000	0.003068	0.194805
92	0.000000	0.003334	0.129870
...	...	...	...
190	0.076210	0.003019	0.064935
254	0.075675	0.004061	0.324675
22	0.075675	0.001545	0.077922
31	0.053685	0.003332	0.506494
184	0.000000	0.004127	0.246753

274 rows × 3 columns

In [18]:
```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

from sklearn.model_selection import GridSearchCV

search_poly_log = GridSearchCV(
    Pipeline([
        ('poly', PolynomialFeatures()),
        ('linear', LinearRegression())
    ]),
    {
        'poly__degree': [1, 2, 3, 4, 5],
        'poly__interaction_only': [True, False],
    },
)

search_poly_log.fit(X_train, y_train)

search_poly_log.score(X_test, y_test)
```

Out[18]: 0.17758292628510353

In [19]:
```
# ridge regression with alpha between 0.1 and 1
from sklearn.linear_model import Ridge
search_ridge = GridSearchCV(
    Pipeline([
        ('ridge', Ridge())
    ]),
    {
        'ridge__alpha': list(np.arange(0.1, 1, 0.01)),
    },
```

```
)

search_ridge.fit(X_train, y_train)
search_ridge.score(X_test, y_test)
```

Out[19]: 0.01881088249273366

In [20]:
```
# SVR regression with degree between 1 and 5
from sklearn.svm import SVR
search_svr = GridSearchCV(
    Pipeline([
        ('svr', SVR())
    ]),
    {
        'svr__degree': [1, 2, 3, 4, 5],
    },
)

search_svr.fit(X_train, y_train)
search_svr.score(X_test, y_test)
```

Out[20]: -0.25117152233017936

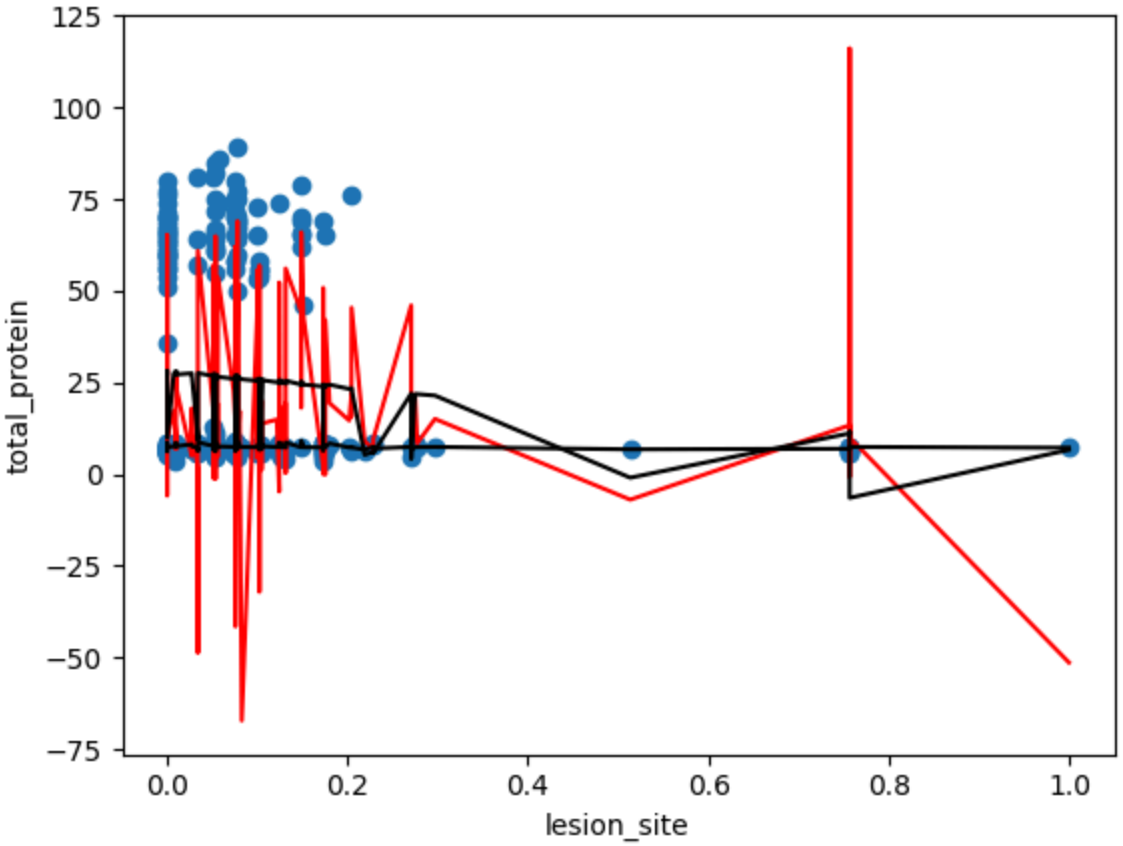
7. визуализация регрессоров

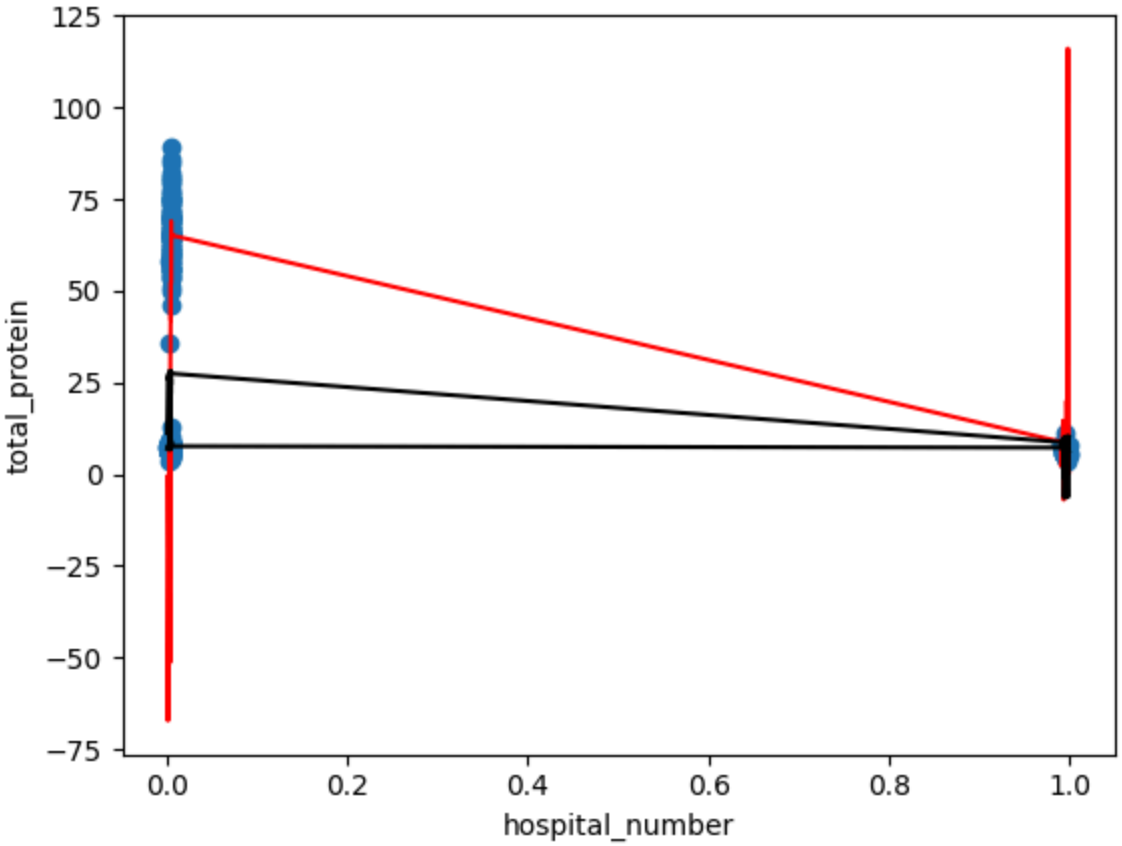
In [21]:
```
from sklearn.metrics import mean_absolute_percentage_error
polylog_mape = mean_absolute_percentage_error(search_poly_log.predict(X_test
ridge_mape = mean_absolute_percentage_error(search_ridge.predict(X_test), y_
svr_mape = mean_absolute_percentage_error(search_svr.predict(X_test), y_test

print('MAPE for polynomial regression:', polylog_mape)
print('MAPE for ridge regression:', ridge_mape)
print('MAPE for SVR:', svr_mape)

best_pred = search_poly_log
if ridge_mape < best_pred.score(X_test, y_test):
    best_pred = search_ridge

if svr_mape < best_pred.score(X_test, y_test):
    best_pred = search_svr

print(best_pred)
```

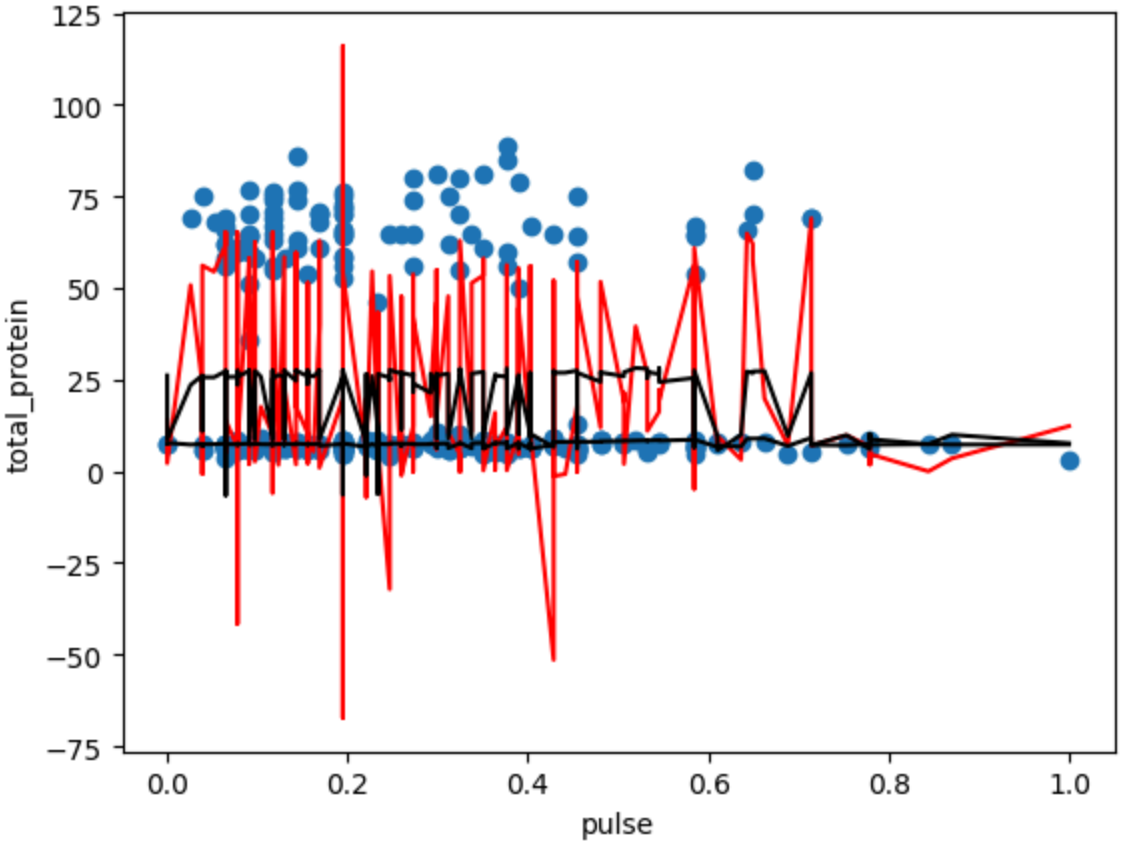
```
MAPE for polynomial regression: 3.947238830664461
MAPE for ridge regression: 0.8237882444800128
MAPE for SVR: 1.7229710708202943
GridSearchCV(estimator=Pipeline(steps=[('poly', PolynomialFeatures()),
                                       ('linear', LinearRegression())]),
             param_grid={'poly__degree': [1, 2, 3, 4, 5],
                         'poly__interaction_only': [True, False]})
```

In [22]:
```
for i in FEATURES:
    ax = plt.axes()
    ax.scatter(X[i], y)
```

```
    ax.set_xlabel(i)
    ax.set_ylabel('total_protein')
    sorted_x = X.sort_values(i)
    for pred in [search_poly_log, search_ridge, search_svr]:
        ax.plot(sorted_x[i], pred.predict(sorted_x), color='red' if pred ==
    plt.show()
```

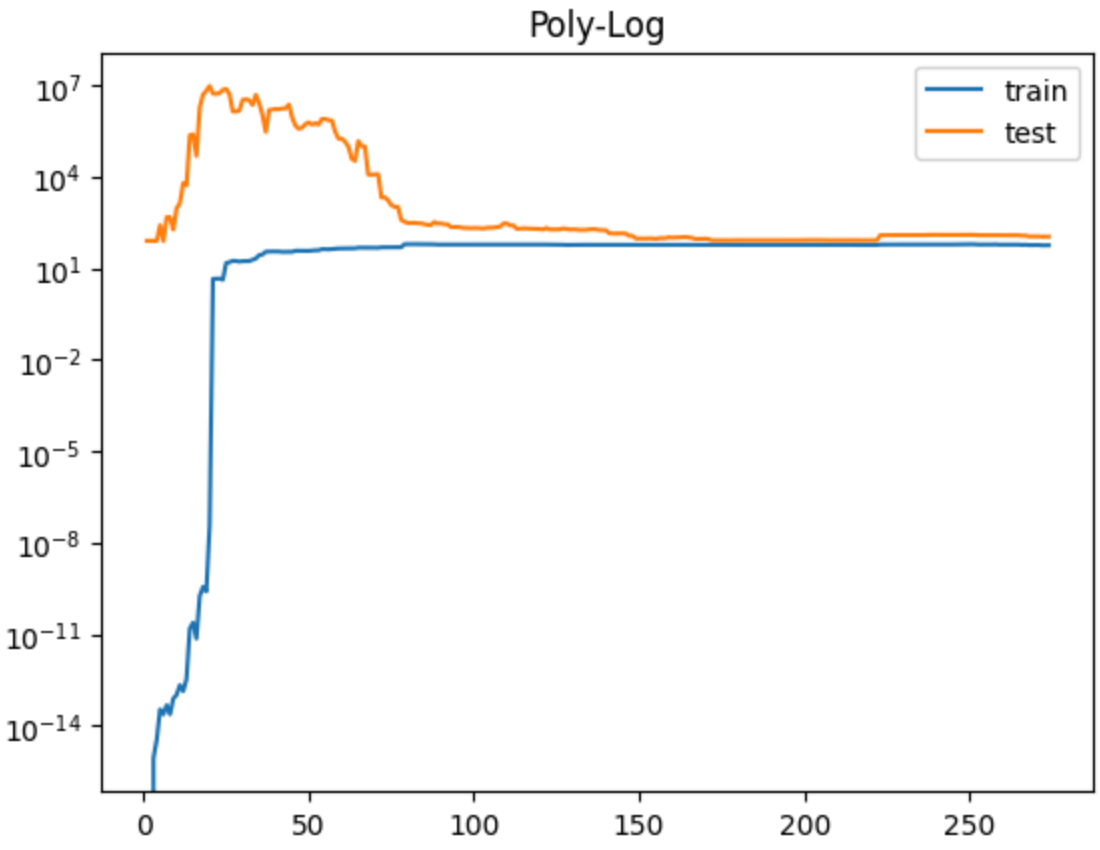




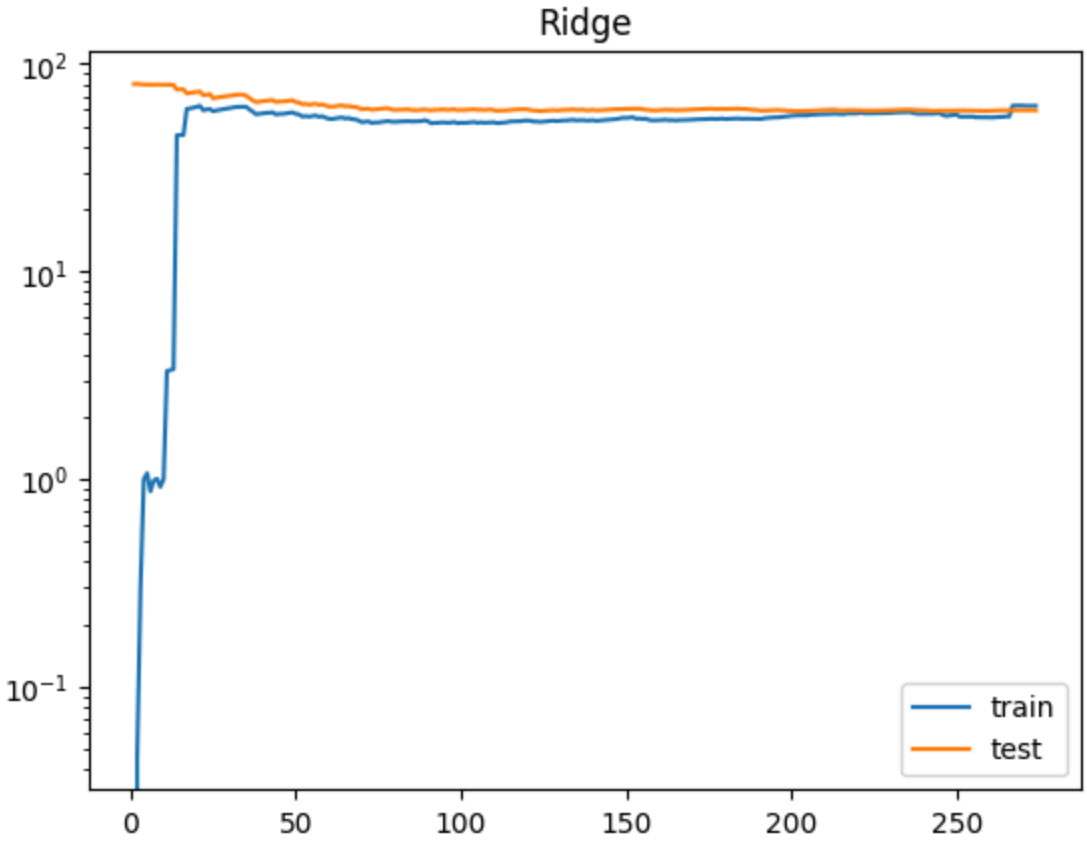


8. кривые обучения

In [23]:
```
# iterate over train split percentage
from sklearn.metrics import max_error

def plot_learning_curve(algo, X_train, X_test, y_train, y_test):
    ax = plt.subplot()
    train_score = []
    test_score = []
    for i in range(1, len(X_train)+1):
        algo.fit(X_train[:i], y_train[:i])

        y_train_predict = algo.predict(X_train[:i])
        train_score.append(max_error(y_train[:i], y_train_predict))

        y_test_predict = algo.predict(X_test)
        test_score.append(max_error(y_test, y_test_predict))

    ax.set_yscale('log')
    ax.plot([i for i in range(1, len(X_train)+1)], train_score, label="trair
    ax.plot([i for i in range(1, len(X_train)+1)], test_score, label="test")
    ax.legend()
    # plt.axis([0, len(X_train)+1, 0, 4]) # np.sqrt(test_score).max()
plt.title("Poly-Log")
plot_learning_curve(search_poly_log.best_estimator_, X_train, X_test, y_trai
```



In [24]:
```
plt.title("Ridge")
plot_learning_curve(search_ridge.best_estimator_, X_train, X_test, y_train,
```



In [25]:
```
plt.title("SVR")
plot_learning_curve(search_svr.best_estimator_, X_train, X_test, y_train, y_
```