Рубежный контроль №1

Покшубина Софья, ИУ5-61Б

Вариант 12

Задача №2.

Для заданного набора данных проведите обработку пропусков в данных для одного категориального и одного количественного признака. Какие способы обработки пропусков в данных для категориальных и количественных признаков Вы использовали? Какие признаки Вы будете использовать для дальнейшего построения моделей машинного обучения и почему? Для пары произвольных колонок данных построить график "Диаграмма рассеяния".

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
from sklearn.impute import SimpleImputer, MissingIndicator

df = pd.read_csv("heart.csv")

df.head()
```

```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak
slope  \
0   52    1   0       125   212    0        1      168      0      1.0
2
1   53    1   0       140   203    1        0      155      1      3.1
0
2   70    1   0       145   174    0        1      125      1      2.6
0
3   61    1   0       148   203    0        1      161      0      0.0
2
4   62    0   0       138   294    1        1      106      0      1.9
1

   ca  thal  target
0   2     3       0
1   0     3       0
2   0     3       0
3   1     3       0
4   3     2       0
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1025 non-null   int64  
 1   sex       1025 non-null   int64  
 2   cp        1025 non-null   int64  
 3   trestbps  1025 non-null   int64  
 4   chol      1025 non-null   int64  
 5   fbs       1025 non-null   int64  
 6   restecg   1025 non-null   int64  
 7   thalach   1025 non-null   int64  
 8   exang     1025 non-null   int64  
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64  
 11  ca        1025 non-null   int64  
 12  thal      1025 non-null   int64  
 13  target    1025 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

Данный датасет не имеет пропусков, поэтому заменим датасет на другой.

```
house_prices = pd.read_csv("train.csv")
```

```
house_prices.head()
```

```
   Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape
\
0   1          60       RL         65.0     8450   Pave   NaN      Reg

1   2          20       RL         80.0     9600   Pave   NaN      Reg

2   3          60       RL         68.0    11250   Pave   NaN      IR1

3   4          70       RL         60.0     9550   Pave   NaN      IR1

4   5          60       RL         84.0    14260   Pave   NaN      IR1


  LandContour Utilities  ... PoolArea PoolQC Fence MiscFeature MiscVal
MoSold  \
0         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
2
1         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
5
2         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
9
```

```
3         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
2
4         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
12

  YrSold  SaleType  SaleCondition  SalePrice
0   2008        WD         Normal     208500
1   2007        WD         Normal     181500
2   2008        WD         Normal     223500
3   2006        WD        Abnorml     140000
4   2008        WD         Normal     250000

[5 rows x 81 columns]

house_prices.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   object
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   Alley          91 non-null     object
 7   LotShape       1460 non-null   object
 8   LandContour    1460 non-null   object
 9   Utilities      1460 non-null   object
 10  LotConfig      1460 non-null   object
 11  LandSlope      1460 non-null   object
 12  Neighborhood   1460 non-null   object
 13  Condition1     1460 non-null   object
 14  Condition2     1460 non-null   object
 15  BldgType       1460 non-null   object
 16  HouseStyle     1460 non-null   object
 17  OverallQual    1460 non-null   int64
 18  OverallCond    1460 non-null   int64
 19  YearBuilt      1460 non-null   int64
 20  YearRemodAdd   1460 non-null   int64
 21  RoofStyle      1460 non-null   object
 22  RoofMatl       1460 non-null   object
 23  Exterior1st    1460 non-null   object
 24  Exterior2nd    1460 non-null   object
 25  MasVnrType     1452 non-null   object
 26  MasVnrArea     1452 non-null   float64
 27  ExterQual      1460 non-null   object
 28  ExterCond      1460 non-null   object
```

```
29  Foundation     1460 non-null   object
30  BsmtQual       1423 non-null   object
31  BsmtCond       1423 non-null   object
32  BsmtExposure   1422 non-null   object
33  BsmtFinType1   1423 non-null   object
34  BsmtFinSF1     1460 non-null   int64
35  BsmtFinType2   1422 non-null   object
36  BsmtFinSF2     1460 non-null   int64
37  BsmtUnfSF      1460 non-null   int64
38  TotalBsmtSF    1460 non-null   int64
39  Heating        1460 non-null   object
40  HeatingQC      1460 non-null   object
41  CentralAir     1460 non-null   object
42  Electrical     1459 non-null   object
43  1stFlrSF       1460 non-null   int64
44  2ndFlrSF       1460 non-null   int64
45  LowQualFinSF   1460 non-null   int64
46  GrLivArea      1460 non-null   int64
47  BsmtFullBath   1460 non-null   int64
48  BsmtHalfBath   1460 non-null   int64
49  FullBath       1460 non-null   int64
50  HalfBath       1460 non-null   int64
51  BedroomAbvGr   1460 non-null   int64
52  KitchenAbvGr   1460 non-null   int64
53  KitchenQual    1460 non-null   object
54  TotRmsAbvGrd   1460 non-null   int64
55  Functional     1460 non-null   object
56  Fireplaces     1460 non-null   int64
57  FireplaceQu    770 non-null    object
58  GarageType     1379 non-null   object
59  GarageYrBlt    1379 non-null   float64
60  GarageFinish   1379 non-null   object
61  GarageCars     1460 non-null   int64
62  GarageArea     1460 non-null   int64
63  GarageQual     1379 non-null   object
64  GarageCond     1379 non-null   object
65  PavedDrive     1460 non-null   object
66  WoodDeckSF     1460 non-null   int64
67  OpenPorchSF    1460 non-null   int64
68  EnclosedPorch  1460 non-null   int64
69  3SsnPorch      1460 non-null   int64
70  ScreenPorch    1460 non-null   int64
71  PoolArea       1460 non-null   int64
72  PoolQC         7 non-null      object
73  Fence          281 non-null    object
74  MiscFeature    54 non-null     object
75  MiscVal        1460 non-null   int64
76  MoSold         1460 non-null   int64
77  YrSold         1460 non-null   int64
78  SaleType       1460 non-null   object
```

```
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

- GarageType - категориальный
- LotFrontage - числовой

Категориальный

```
house_prices['GarageType'].unique()
```

```
array(['Attchd', 'Detchd', 'BuiltIn', 'CarPort', nan, 'Basment',
'2Types'],
      dtype=object)
```

```
house_prices['GarageType'].isnull().sum()
```

```
81
```

В столбце отсутствует тип "нет гаража", но присутствуют пропущенные значения. Скорее всего, под пропущенными значениями подразумевается отсутствие гаража на участке, поэтому заменим их константой "NoGarage":

```
house_prices['GarageType']=
house_prices['GarageType'].fillna('NoGarage')
```

Теперь пустые значения отсутствуют:

```
np.unique(house_prices['GarageType'])
```

```
array(['2Types', 'Attchd', 'Basment', 'BuiltIn', 'CarPort', 'Detchd',
       'NoGarage'], dtype=object)
```

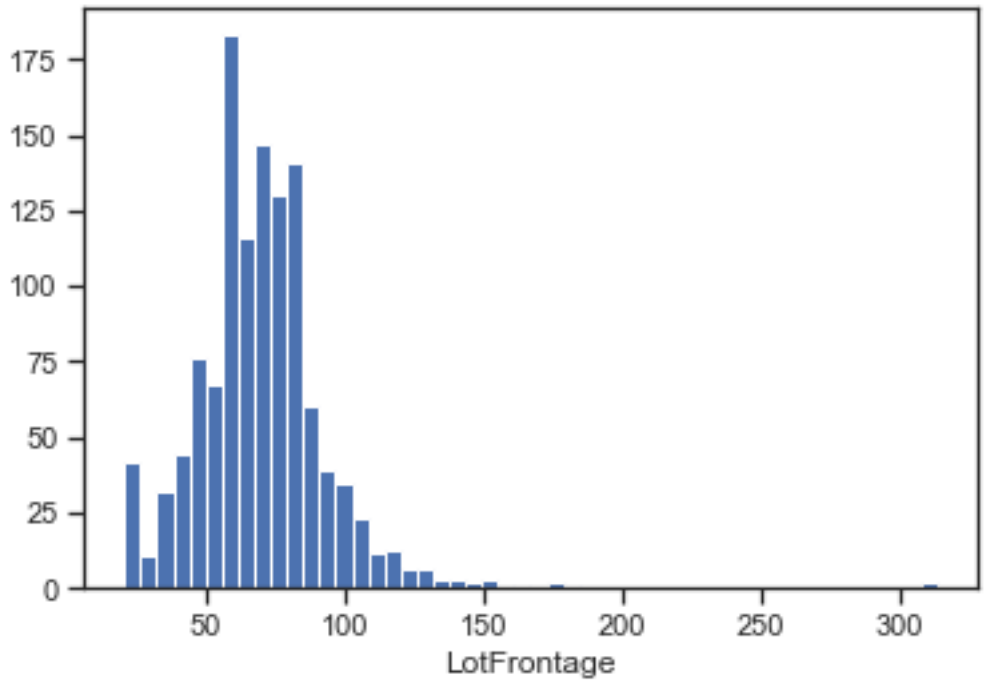
Числовой

```
num_temp_data = house_prices [['LotFrontage']]
num_temp_data
```

```
      LotFrontage
0            65.0
1            80.0
2            68.0
3            60.0
4            84.0
...           ...
1455         62.0
1456         85.0
1457         66.0
1458         68.0
1459         75.0

[1460 rows x 1 columns]
```

```
plt.hist(num_temp_data['LotFrontage'], 50)
plt.xlabel('LotFrontage')
plt.show()
```




```
num_temp_data.describe().T
```

```
              count       mean        std   min   25%   50%   75%
max
LotFrontage  1201.0  70.049958  24.284752  21.0  59.0  69.0  80.0
313.0
```

```
# Фильтр для проверки заполнения пустых значений
indicator = MissingIndicator()
mask_missing_values_only = indicator.fit_transform(num_temp_data)
mask_missing_values_only
```

```
array([[False],
       [False],
       [False],
       ...,
       [False],
       [False],
       [False]])
```

```
#Стратегии заполнения
strategies=['mean', 'median', 'most_frequent']

def test_num_impute_col(dataset, column, strategy_param):
    temp_data = dataset[[column]]
    indicator = MissingIndicator()
```

```
    mask_missing_values_only = indicator.fit_transform(temp_data)
    imp_num = SimpleImputer(strategy=strategy_param)
    data_num_imp = imp_num.fit_transform(temp_data)
    filled_data = data_num_imp[mask_missing_values_only]
    return column, strategy_param, filled_data.size, filled_data[0],
filled_data[filled_data.size-1]

test_num_impute_col(num_temp_data, 'LotFrontage', strategies[0])

('LotFrontage', 'mean', 259, 70.04995836802665, 70.04995836802665)

test_num_impute_col(num_temp_data, 'LotFrontage', strategies[1])

('LotFrontage', 'median', 259, 69.0, 69.0)

test_num_impute_col(num_temp_data, 'LotFrontage', strategies[2])

('LotFrontage', 'most_frequent', 259, 60.0, 60.0)
```

Т.к. в столбце LotFrontage данные распределены одномодально, можно заполнить пропуски медианным значением.

```
house_prices['LotFrontage'] =
house_prices['LotFrontage'].fillna(house_prices['LotFrontage'].median(
))
```

При заполнении пропусков я использовала импьютацию константным и медианным значениями.

Вернемся к первому датасету. В нем имеется 13 нецелевых признаков и 1 целевой. Целевой признак - наличие или отсутствие заболеваний сердца у конкретного пациента (столбец 'target'). Чтобы ответить на вопрос о том, какие признаки лучше всего будет использовать для построения модели машинного обучения, нам необходимо посмотреть корррелящию между столбцами датасета.

```
df.corr()
```

```
                age       sex        cp  trestbps      chol
fbs  \
age        1.000000 -0.103240 -0.071966  0.271121  0.219823  0.121243

sex       -0.103240  1.000000 -0.041119 -0.078974 -0.198258  0.027200

cp        -0.071966 -0.041119  1.000000  0.038177 -0.081641  0.079294

trestbps   0.271121 -0.078974  0.038177  1.000000  0.127977  0.181767

chol       0.219823 -0.198258 -0.081641  0.127977  1.000000  0.026917

fbs        0.121243  0.027200  0.079294  0.181767  0.026917  1.000000
```

```
restecg  -0.132696 -0.055117  0.043581 -0.123794 -0.147410 -0.104051
thalach  -0.390227 -0.049365  0.306839 -0.039264 -0.021772 -0.008866
exang     0.088163  0.139157 -0.401513  0.061197  0.067382  0.049261
oldpeak   0.208137  0.084687 -0.174733  0.187434  0.064880  0.010859
slope    -0.169105 -0.026666  0.131633 -0.120445 -0.014248 -0.061902
ca        0.271551  0.111729 -0.176206  0.104554  0.074259  0.137156
thal      0.072297  0.198424 -0.163341  0.059276  0.100244 -0.042177
target   -0.229324 -0.279501  0.434854 -0.138772 -0.099966 -0.041164

            restecg   thalach     exang   oldpeak     slope
ca  \
age      -0.132696 -0.390227  0.088163  0.208137 -0.169105  0.271551
sex      -0.055117 -0.049365  0.139157  0.084687 -0.026666  0.111729
cp        0.043581  0.306839 -0.401513 -0.174733  0.131633 -0.176206
trestbps -0.123794 -0.039264  0.061197  0.187434 -0.120445  0.104554
chol     -0.147410 -0.021772  0.067382  0.064880 -0.014248  0.074259
fbs      -0.104051 -0.008866  0.049261  0.010859 -0.061902  0.137156
restecg   1.000000  0.048411 -0.065606 -0.050114  0.086086 -0.078072
thalach   0.048411  1.000000 -0.380281 -0.349796  0.395308 -0.207888
exang    -0.065606 -0.380281  1.000000  0.310844 -0.267335  0.107849
oldpeak  -0.050114 -0.349796  0.310844  1.000000 -0.575189  0.221816
slope     0.086086  0.395308 -0.267335 -0.575189  1.000000 -0.073440
ca       -0.078072 -0.207888  0.107849  0.221816 -0.073440  1.000000
thal     -0.020504 -0.098068  0.197201  0.202672 -0.094090  0.149014
target    0.134468  0.422895 -0.438029 -0.438441  0.345512 -0.382085
```

```
              thal    target
age       0.072297 -0.229324
sex       0.198424 -0.279501
cp       -0.163341  0.434854
trestbps  0.059276 -0.138772
chol      0.100244 -0.099966
fbs      -0.042177 -0.041164
restecg  -0.020504  0.134468
thalach  -0.098068  0.422895
exang     0.197201 -0.438029
oldpeak   0.202672 -0.438441
slope    -0.094090  0.345512
ca        0.149014 -0.382085
thal      1.000000 -0.337838
target   -0.337838  1.000000
```

Можно заметить, что все признаки имеют слабую корреляцию между собой, поэтому построение модели машинного обучения с помощью данной выборки будет нецелесообразно.

Диаграмма рассеяния

```
data = pd.read_csv("train.csv")

plt.scatter(data['YrSold'], data['SalePrice'], 100)
plt.title('Изменение цен на недвижимость по годам')
plt.xlabel('Год продажи')
plt.ylabel('Цена')
plt.locator_params(axis = "x", nbins = 5)
```


Изменение цен на недвижимость по годам
Цена
700000
600000
500000
400000
300000
200000
100000
0
2006
2007
2008
2009
2010
Год продажи