ИУ5-21М Гришин С. В.

Рубежный контроль №1

Вариант 3

В качестве набора данных мы будем использовать набор данных прогнозирования инсульта: https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction

Age: возраст пациента [лет]

Sex: пол пациента [M: Мужской, F: Женский]

ChestPainType: тип боли в груди [TA: типичная стенокардия, ATA: атипичная стенокардия, NAP: неангинальная боль, ASY: бессимптомная]

RestingBP: артериальное давление в состоянии покоя [мм рт.ст.]

Cholesterol: холестерин сыворотки [мм/дл]

FastingBS: уровень сахара в крови натощак [1: если FastingBS > 120 мг/дл, 0: иначе]

RestingECG: результаты электрокардиограммы в покое [Normal: нормальная, ST: анома- лия ST-T (инверсия T и/или элевация или депрессия ST > 0,05 мВ), LVH: вероятная или определенная гипертрофия левого желудочка по критериям Эстеса]

MaxHR: максимальная достигнутая частота сердечных сокращений [Числовое значение от 60 до 202]

ExerciseAngina: стенокардия, вызванная физической нагрузкой [Y: Да, N: Нет]

Oldpeak: oldpeak: ST [Числовое значение, измеренное в депрессии]

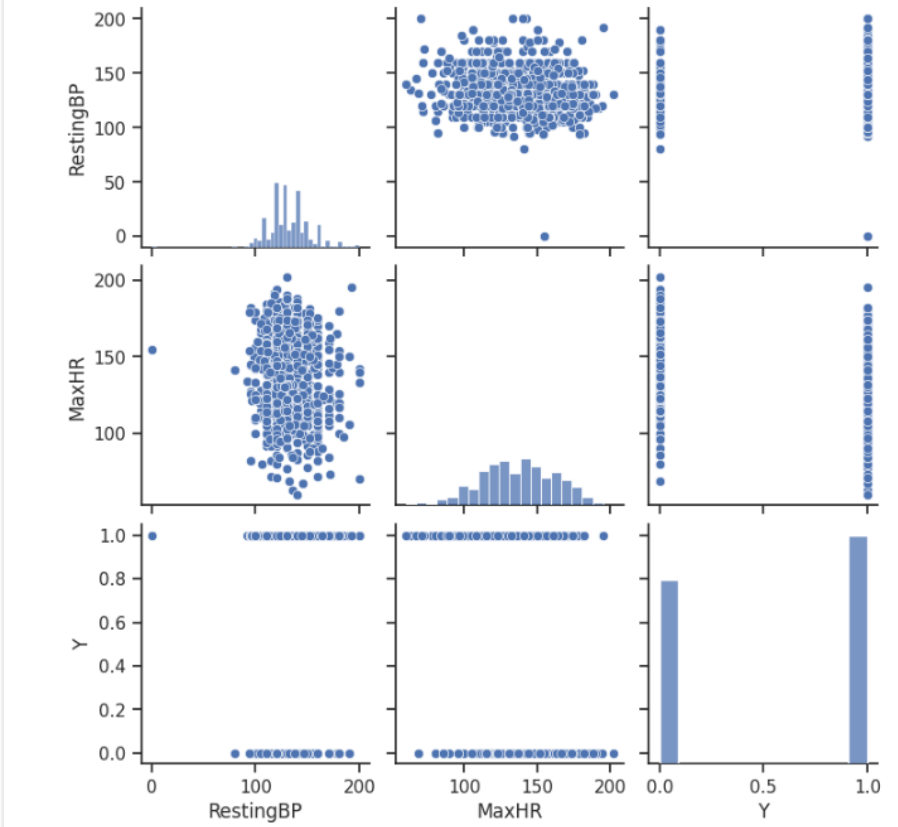
ST_Slope: наклон сегмента ST пикового упражнения [Up: восходящий, Flat: плоский, Down: нисходящий]

HeartDisease: выходной класс [1: болезнь сердца, 0: нормальный]

Диаграммы рассеяния:

```
# Парные диаграммы
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x7fdfcd15ba90>
```



Задача №3

Для набора данных проведите кодирование одного (произвольного) категориального признака с использованием метода "weight of evidence (WoE) encoding".

```
[36] from category_encoders.woe import WOEEncoder as ce_WOEEncoder
```

```
[39] ce_WOEEncoder1 = ce_WOEEncoder()
     data_WOE_ENC = ce_WOEEncoder1.fit_transform(first_data[first_data.columns.difference(['HeartDisease'])], first_data['HeartDisease'])
```

```
[41] data_WOE_ENC
```

	Age	ChestPainType	Cholesterol	ExerciseAngina	FastingBS	MaxHR	Oldpeak	RestingBP	RestingECG	ST_Slope	Sex
0	40	-2.005147	289	-0.825628	0	172	0.0	140	-0.148383	-1.605991	0.324676
1	49	-0.805730	180	-0.825628	0	156	1.0	160	-0.148383	1.350007	-1.251375
2	37	-2.005147	283	-0.825628	0	98	0.0	130	0.430163	-1.605991	0.324676
3	48	1.106462	214	1.520163	0	108	1.5	138	-0.148383	1.350007	-1.251375
4	54	-0.805730	195	-0.825628	0	122	0.0	150	-0.148383	-1.605991	0.324676
...	...	...	...	...	...	...	...	...	...	...	...
913	45	-0.464702	264	-0.825628	0	132	1.2	110	-0.148383	1.350007	0.324676
914	68	1.106462	193	-0.825628	1	141	3.4	144	-0.148383	1.350007	0.324676
915	57	1.106462	131	1.520163	0	115	1.2	130	-0.148383	1.350007	0.324676
916	57	-2.005147	236	-0.825628	0	174	0.0	130	0.040601	1.350007	-1.251375
917	38	-0.805730	175	-0.825628	0	173	0.0	138	-0.148383	-1.605991	0.324676

918 rows × 11 columns

Next steps: View recommended plots

```
[43] # Проверка для поля "Пол"
     first_data['Sex'].unique()

     array(['M', 'F'], dtype=object)
```

```
[44] data_WOE_ENC['Sex'].unique()

     array([ 0.32467585, -1.25137504])
```

```
[49] def check_woe_encoding(field):
         data_ones = first_data[first_data['HeartDisease'] == 1].shape[0]
         data_zeros = first_data[first_data['HeartDisease'] == 0].shape[0]

         for s in first_data[field].unique():
             data_filter = first_data[first_data[field]==s]
             if data_filter.shape[0] > 0:

                 filter_data_ones = data_filter[data_filter['HeartDisease'] == 1].shape[0]
                 filter_data_zeros = data_filter[data_filter['HeartDisease'] == 0].shape[0]

                 good = filter_data_ones / data_ones
                 bad = filter_data_zeros / data_zeros

                 woe = np.log(good/bad)
                 print(s, '-' , woe)
```

```
[50] check_woe_encoding('Sex')

     M - 0.32529623783380696
     F - -1.2651459127118896
```

Задача №23

Для набора данных для одного (произвольного) числового признака проведите обнаружение и удаление выбросов на основе правила трех сигм.

```
data = pd.DataFrame(first_data,
                    columns=first_data.columns)[x_col_list]
data['Y'] = first_data['HeartDisease']
data.shape

(918, 3)
```

```
data.head()
```

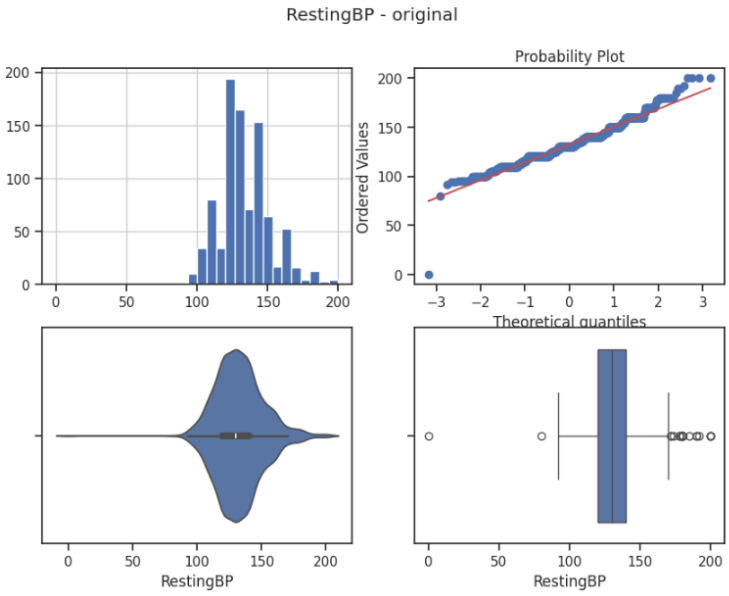
	RestingBP	MaxHR	Y
0	140	172	0
1	160	156	1
2	130	98	0
3	138	108	1
4	150	122	0

Next steps: View recommended plots

```
[28] def diagnostic_plots(df, variable, title):
         fig, ax = plt.subplots(figsize=(10,7))
         # гистограмма
         plt.subplot(2, 2, 1)
         df[variable].hist(bins=30)
         ## Q-Q plot
         plt.subplot(2, 2, 2)
         stats.probplot(df[variable], dist="norm", plot=plt)
         # ящик с усами
         plt.subplot(2, 2, 3)
         sns.violinplot(x=df[variable])
         # ящик с усами
         plt.subplot(2, 2, 4)
         sns.boxplot(x=df[variable])
         fig.suptitle(title)
         plt.show()
```

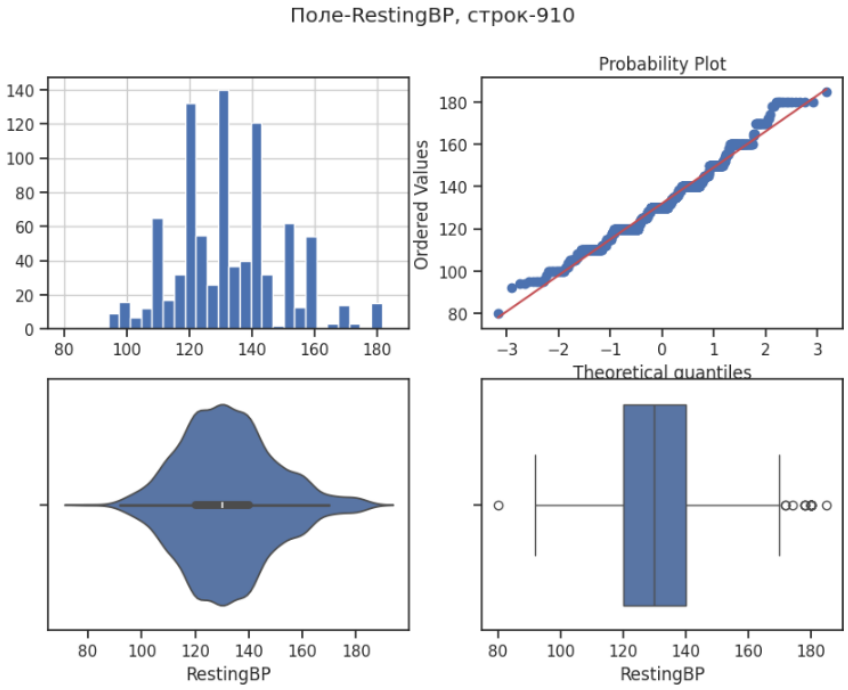
```
diagnostic_plots(data, 'RestingBP', 'RestingBP - original')
```

```
<ipython-input-28-766c933c159f>:4: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 and
  plt.subplot(2, 2, 1)
```




```
[31] # Функция вычисления верхней и нижней границы выбросов
     def get_outlier_boundaries(df, col):
       K1 = 3
       lower_boundary = df[col].mean() - (K1 * df[col].std())
       upper_boundary = df[col].mean() + (K1 * df[col].std())
       return lower_boundary, upper_boundary
```

```
for col in x_col_list:
        # Вычисление верхней и нижней границы
        lower_boundary, upper_boundary = get_outlier_boundaries(data, col)
        # Флаги для удаления выбросов
        outliers_temp = np.where(data[col] > upper_boundary, True,
                                 np.where(data[col] < lower_boundary, True, False))
        # Удаление данных на основе флага
        data_trimmed = data.loc[~(outliers_temp), ]
        title = 'Поле-{}, строк-{}'.format(col, data_trimmed.shape[0])
        diagnostic_plots(data_trimmed, col, title)
```




```
<ipython-input-28-766c933c159f>:4: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since
  plt.subplot(2, 2, 1)
```