Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)

КАФЕДРА «Системы обработки информации и управления» (ИУ5)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

«Анализ факторов, влияющих на риск сердечного заболевания»

Студент группы ИУ5-32М _____ **Рожненко М.К.**
(Подпись, дата)

Руководитель _____ **Гапанюк Ю.Е.**
(Подпись, дата)

2021 г.

Оглавление

Введение

Во второй половине XX века основную опасность для здоровья населения и проблему для здравоохранения стали представлять неинфекционные заболевания, в первую очередь болезни сердечно-сосудистой системы, которые в настоящее время являются ведущей причиной заболеваемости, инвалидизации и смертности взрослого населения. Произошло "омоложение" этих заболеваний. Они стали распространяться и среди населения развивающихся стран.

В большинстве экономически развитых стран заболевания сердечно-сосудистой системы занимают первое место среди причин заболеваемости инвалидизации и смертности, хотя их распространенность в разных регионах значительно колеблется. В Европе ежегодно умирают от сердечно-сосудистых заболеваний приблизительно 3 млн. человек, в США - 1 млн., это составляет половину всех смертей, в 2,5 раза больше, чем от всех злокачественных новообразований вместе взятых, причем 1/4 умерших от сердечно-сосудистых заболеваний составляют люди в возрасте до 65 лет.

Машинное обучение дает хорошие возможности решить эту проблему и существенно повысить точность прогнозирования сердечно-сосудистых заболеваний и их осложнений в сравнении с использованием существующих методик, за счет учета нелинейных взаимосвязей их точной настройки между факторами сердечно-сосудистого риска и проявлением заболеваний. В последнее время расчет число исследований и разработок в этой области.

Постановка задач

Целью исследования было повышение точности предсказания рисков развития сердечно-сосудистых заболеваний на основе Фрамингемской шкалы путем применения машинного обучения для разработки собственной математической модели, а также прогнозирование корреляции между рядом факторов. Были решены следующие задачи:

1. Проведен анализ данных с предварительной обработкой данных.
2. Проведение оценки рисков развития ССЗ для данного дата-сета на основе использования классического калькулятора Фрамингемской шкалы.
3. Создание модели расчета рисков ССЗ для данного дата-сета с использованием методов машинного обучения.
4. Изучить и применить методы масштабирования данных для анализа.

Модель на основе машинного обучения

В качестве методов машинного обучения использована модель искусственной нейронной сети с двумя скрытыми слоями. Популяция исследования была разделена в наборе данных на «обучающую» выборку (75% из общей извлеченной когорты), в которой были получены алгоритмы риска ССЗ и выборку «валидация» (оставшиеся 25%), которая применялась для тестирования и оценки алгоритмов. В качестве модели использована последовательная модель с одним входным, тремя скрытыми и одним выходным слоем. Для предотвращения переобучения используется исключение («dropout»). На каждом слое используется функция «dense» для полного соединения слоев друг с другом. В скрытых слоях используется функция активации «relu».

В качестве оптимизатора алгоритма, который изменяет веса и смещения во время обучения, используется «rmsprop». В качестве функции потерь («loss») используется бинарная кросс-энтропия, в качестве метрики оценки – точность. Эти алгоритмы были реализованы с помощью открытой библиотеки с исходными текстами scikit-learn, Tensorflow и Keras для языка программирования Python. Гиперпараметры каждой модели определяли с помощью алгоритма поиска GridSearchCV (из той же библиотеки scikit-learn) и 10 K-Fold перекрестной валидации на обучающей когорте для определения значений, которые привели к лучшей производительности.

Визуальный анализ данных

Первой, и наиболее заметной проблемой в полученных таблицах, является наличие строк с незаполненными полями в рамках одной записи. Эти записиявляются неинформативными - удалим их.

Следующей проблемой являются столбцы, дублирующие информацию, как следствие – неинформативные столбцы.

Обработка выбросов

Существует ГОСТ Р ИСО 16269-4-2017 Статистические методы. Статистическое представление данных. Выявление и обработка выбросов посвященный обработке выбросов. В соответствии с ГОСТ, выброс (outlier) – это элемент маломощного подмножества выборки, существенно отличающийся от остальных элементов выборки.

Выбросы негативно влияют на результат анализа, поэтому их необходимо устранять. Выбросы появляются по следующим причинам. Ошибки в измерениях, например, часть значений признака "расстояние" была измерена не в километрах,а в метрах. Технические ошибки форматирования данных.

Основные задачи обработки выбросов — это обнаружение выбросов и устранение (удаление или замена) их в зависимости от требований задачи.

По определению гистограммы распределения, выбросы — это значения накраях гистограммы (очень большие или очень маленькие по сравнению со всей выборкой). Задача обнаружения выбросов — это задача выделения элементов, находящихся на краях гистограммы.

Существует несколько подходов для определения выбросов. Подход в случае нормального распределения или распределения, похожего на нормальное.

Использование правила трех сигм

Правило, утверждающее, что для любой случайной величины ξ с конечной дисперсией вероятность того, что случайная величина отклонится от своего математического ожидания вероятность того, что случайная величина отклонится от своего математического ожидания $M[\xi]$ не менее, чем на три среднеквадратических отклонения σ, не более [1]: $\frac{}{9}$

$$P(|\xi - M[\xi]| \geq 3\sigma) \leq \frac{1}{9}$$

Для большинства случайных величин эта вероятность меньше, например, для нормального распределения:

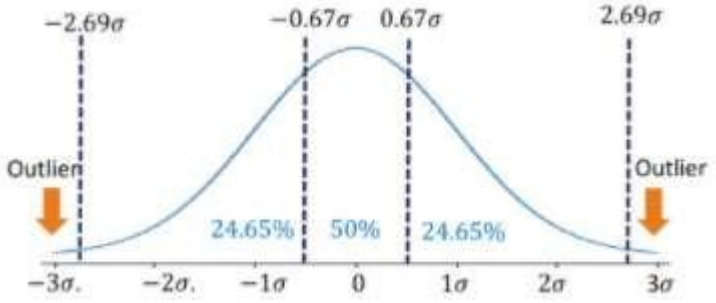


Рисунок 1 – Правило трех сигм

$$outlier < mean(x) - 3 * std(x)$$

$$outlier > mean(x) - 3 * std(x)$$

Использование 5% и 95% квантилей

95% данных располагаются выше 5% квантиля и 95% данных располагаются ниже 95% квантиля. Значения ниже 5% квантиля и выше 95% квантиля можно считать выбросами.

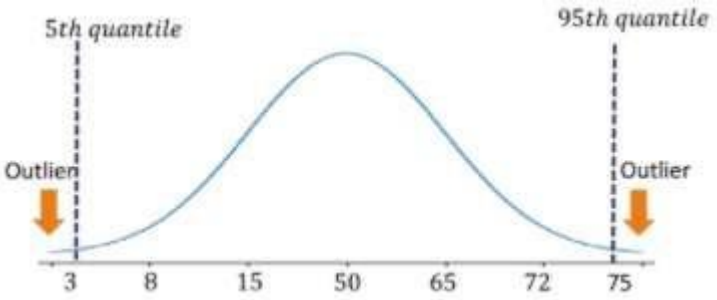


Рисунок 2 - Использование 5% и 95% квантилей

$$outlier < x.quantile(5\%)$$

$$outlier > x.quantile(95\%)$$

В нашем случае распределение данных зарплат асимметричное, в этом случае применяется подход использования межквартального размаха.

Межквартильный размах IQR (interquartile range, IQR) - это разность третьего квартиля и первого квартиля:

Тогда: $$IQR = Q3(x) - Q1(x)$$

$$outlier < Q1(x) - K * IQR$$

$$outlier > Q3(x) + K * IQR$$

Значение K обычно выбирается равным 1,5.

После обнаружения всех выбросов удалим эти объекты из данных.

Масштабирование признаков

Масштабирование - это изменение диапазона измерения признака с целью улучшения качества построения модели.

Данные необходимо масштабировать потому**,** что признаки с меньшей амплитудой оказываются "оштрафованы" по сравнению с признаками с большей амплитудой, и оказывают меньшее влияние результаты анализа.

В кластерном анализе данных часто используется подсчет расстояния между кластерами с помощью метода ближайших соседей. Ключевым шагом этого метода является вычисление расстояний между соседями. Чаще всего в методе ближайших соседей используется Евклидово расстояние:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_1)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}.$$

Если один из признаков имеет амплитуду значительно меньшую по сравнению с другими признаками, то этот признак почти не будет вносить вклад при вычислении расстояния.

В нашем случае значения всех признаков находятся в диапазоне от 0 до 1, а зарплаты указаны в десятках тысяч. Существует несколько подходов к масштабированию признаков. Рассмотрим наиболее часто применяемые.

Масштабирование данных на основе Z-оценки

$$x' = \frac{x - \mu(x)}{\sigma(\mathrm{x})},$$

где x – признак,

$\mu(x) = mean(x)$ – среднее значение,

$\sigma(\mathrm{x}) = std(x) -$ среднеквадратичное отклонение.

Особенности метода:

- Среднее значение приводится к 0.
- Среднеквадратичное отклонение приводится к 1.
- Форма исходного распределения сохраняется.
- Максимальные и минимальные значения могут варьироваться.
- Выбросы сохраняются.

MinMax-масштабирование

$$x' = \frac{x - min(x)}{\max(x) - \min(x)},$$

Особенности метода:

- Среднее значение может варьироваться.
- Среднеквадратичное отклонение может варьироваться.
- Форма исходного распределения может изменяться.
- Максимальные и минимальные значения в диапазоне [0;1].
- Выбросы сохраняются.

Масштабирование по медиане

$$x' = \frac{x - median(x)}{IQR},$$

где

$$IQR = Q3(x) - Q1(x)$$

IQR – разность между 1 и 3 квартилями.

Особенности метода:

- Медиана приводится к 0.
- Среднеквадратичное отклонение может варьироваться.
- Форма исходного распределения может изменяться.
- Максимальные и минимальные значения могут варьироваться.
- Устраняются выбросы.

Для упрощения последующей интерпретации результатов анализа будем использовать MinMax-масштабирование.

Результаты сравнительной оценки моделей

Параметры точности моделей оценивались по методу ROC анализа, основная концепция которого сводится к задаче классификации, чтобы относить ранее неизвестные моделируемые случаи ССЗ с фактическими болезнями. В результате классификации может наблюдаться четыре различных ситуации:

- истинно-положительный результат (truepositive, TP) – пациент больной, диагноз положительный;
- ложно-положительный результат (false-positive, FP) – пациент здоров, диагноз положительный;
- истинно-отрицательный результат (true-negative, TN) – пациент здоров, диагноз отрицательный;
- ложно-отрицательный результат (false-negative, FN) – пациент больной, диагноз отрицательный.

Четыре возможных выхода могут быть сформулированы и оформлены в виде матрицы сопряженности:

```
Confusion matrix
[[TP FP]
[FN TN]]
```

Значение Se=TP/(TP+FN) – доля истинно положительных случаев или способность алгоритма правильно определять больных, называется чувствительностью.

Значение Spe=TN/(TN+FP) – доля истинно отрицательных случаев или способность алгоритма не принимать здоровых за больных, называется специфичностью.

Экономический эффект от этих ошибок разный: ложноотрицательный больной придёт с запущенной болезнью, а на дообследование ложноположительного будут потрачены ресурсы.

Значение Accuracy=(TP+TN)/(TP+FP+FN+TN) – это точность модели, которая характеризует способность модели правильно определять истинно больных и истинно здоровых пациентов.

Полученные оценки для модели на основе нейронной сети:

```
Confusion matrix
[[493 18]
[124 36]]


Accuracy = 78,84%
Se = 0,79, Sp = 0,67
```

Для Фрамингемской шкалы прогнозирования риска аналогичные показатели:

```
Confusion matrix
[[134281]
[3891432]]
Accuracy = 70.0%
Se = 0,25, Sp = 0,83
```

Как показывают сравнительные оценки анализа результатов моделей,

Фрамингемская шкала способна наиболее точно определять здоровых пациентов (специфичность выше, чем у нейросети), но при этом чувствительность шкалы (Se = 0,25) низкая для определения истинно больных пациентов.

Полученная нами с помощью машинного обучения нейронная сеть показала высокую чувствительность (Se = 0,79) и повысила точность моделирования Accuracy по сравнению с базовой шкалой на +8,84%. На рисунке ниже представлены Roc-кривые зависимости показателя чувствительности модели от ее специфичности, и площадь AUC под ROC-кривой показывает качество модели. Чем выше показатель AUC, тем выше качество модели. Полученные показатели AUC: – для Фрамингемской шкалы: 0,59 неудовлетворительное качество модели); – для нейросети: 0,84 (приемлемое качество модели).

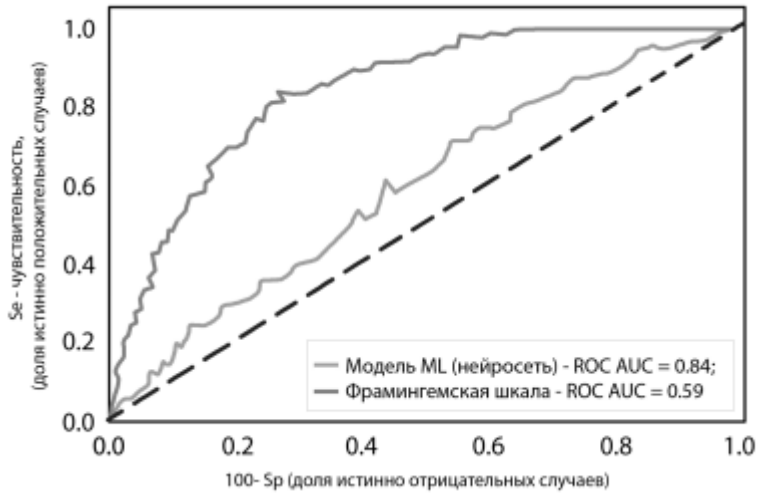


Рисунок - Roc-кривые для результатов моделирования, полученных для Фрамингемской шкалы и модели на основе машинного обучения.

Как показывают результаты оценки точности и качества, модель на основе нейронной сети улучшает результат моделирования по сравнению с Фрамингемской шкалой. Таким образом, наиболее эффективным и точным способом расчета риска ССЗ является математическая модель на основе нейронной сети с использованием данных для обучения, собранных в популяции, на которой она будет использована.

Вывод

В представленной научно-исследовательской работе были рассмотрены вопросы, связанные с исследованием и анализом данных, связанных с заболеваниями сердечно-сосудистой системы. В рамках работы были решены следующие задачи и сделаны выводы:

1. Использование алгоритмов машинного обучения, включая алгоритмы глубокого обучения, может значительно повысить точность обученных моделей прогнозирования сердечно-сосудистых рисков. Особенностью является использование дата сетов для обучения математической модели на основе данных локальной популяции, что в конечном счете тоже способствует увеличению точности прогнозирования.

2. Встраивание подобных моделей в СППВР позволяет более быстро и точно получить результат расчета сердечно-сосудистого риска.

3. Подходы машинного обучения открывают перспективу достижения улучшенной и более индивидуализированной оценки риска ССЗ. Это может помочь движению к персонализированной медицине, лучшей адаптации управления рисками к отдельным пациентам.

Список использованных источников

1. Барсегян А. А. Анализ данных и процессов: учеб. пособие. / А. А. Барсегян. – СПб.: БХВ-Петербург, 2009. – 512 с.
2. Тихонов И. А. Обзор возможностей кластерного анализа данных в программном пакете STATISTICA / И. А. Тихонов // Политехнический молодежный журнал, – 2018. –№1– С. 1 – 10.
3. Дюран Б., Оделл П. Кластерный анализ. / Б. Дюран. – “Статистика”, 1977. – 128 с.
4. Айвазян А. А., Бухштабер В. М., Енюков И. С., Мешалкин Л. Д. Прикладная статистика: Классификации и снижение размерности. / А. А. Айвазян. – М.: Финансы и статистика, 1989. – 607 с.
5. Калинина В.Н., Соловьев В.И. Введение в многомерный статистический анализ. М.: ГУУ, 2003. 66 с.
6. Методы кластерного анализа. Иерархические методы. URL: http://www.intuit.ru/studies/courses/ 6/6/lecture/182?page=2 (дата обращения 19.12.2021).
7. Smola Alex, Vishwanathan S.V.N. Introduction to machine learning. Cambridge University Press, 2008. 234 p.
8. Clinical Practice Research Datalink, reference number: CPRD00039761, https://www.cprd.com/.
9. Дьяконов А.Г. Методы решения задач классификации с категориальными признаками. Прикладная математика и информатика. Труды факультета Вычислительной математики и кибернетики МГУ имени М.В. Ломоносова. 2014. № 46. С. 103 – 127.
10. National Institute for Health and Care Excellence. Cardiovascular disease: risk assessment and reduction, including lipid modification. London, UK: National Institute for Health and Care Excellence, 2016
11. Castelli, W.P. Lipids, risk factors and ischaemic heart disease /W.P. Castelli // Atherosclerosis. – 1996. – Vol. 124. – P. 9.

Приложение А

In [15]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statistics
from sklearn.preprocessing import LabelEncoder
```

In [16]:

```
data = pd.read_csv('heart.csv')
data
```

Out[16]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	NaN	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	NaN	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130.0	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120.0	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120.0	354	0	1	163	1	0.6	2	0	2	1
...	..	...	..	...	...	...	...	...	...	...	.	...	...	...
	.		.								.			
298	51	1	0	NaN	299	0	1	173	1	1.6	2	0	3	63
299	51	1	0	140.0	299	0	1	173	1	1.6	2	0	3	64
300	51	1	0	140.0	299	0	1	173	1	1.6	2	0	3	65
301	51	1	0	NaN	299	0	1	173	1	1.6	2	0	3	66
302	51	1	0	140.0	299	0	1	173	1	1.6	2	0	3	67

303 rows × 14 columns

In [17]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trtbps    276 non-null    float64
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalachh  303 non-null    int64  
 8   exng      303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slp       303 non-null    int64  
 11  caa       303 non-null    int64  
 12  thall     303 non-null    int64  
 13  output    303 non-null    int64
```

In [18]:

```
median = data['trtbps'].median()
data['trtbps'] = data['trtbps'].fillna(median)
```

In [19]:

```
data.info()
```

```
<class
'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trtbps    303 non-null    float64
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalachh  303 non-null    int64  
 8   exng      303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slp       303 non-null    int64  
 11  caa       303 non-null    int64  
 12  thall     303 non-null    int64  
 13  output    303 non-null    int64
dtypes: float64(2), int64(12)
memory usage: 33.3 KB
```

In [20]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats
```

In [21]:

```
data.describe()
```

Out[21]:

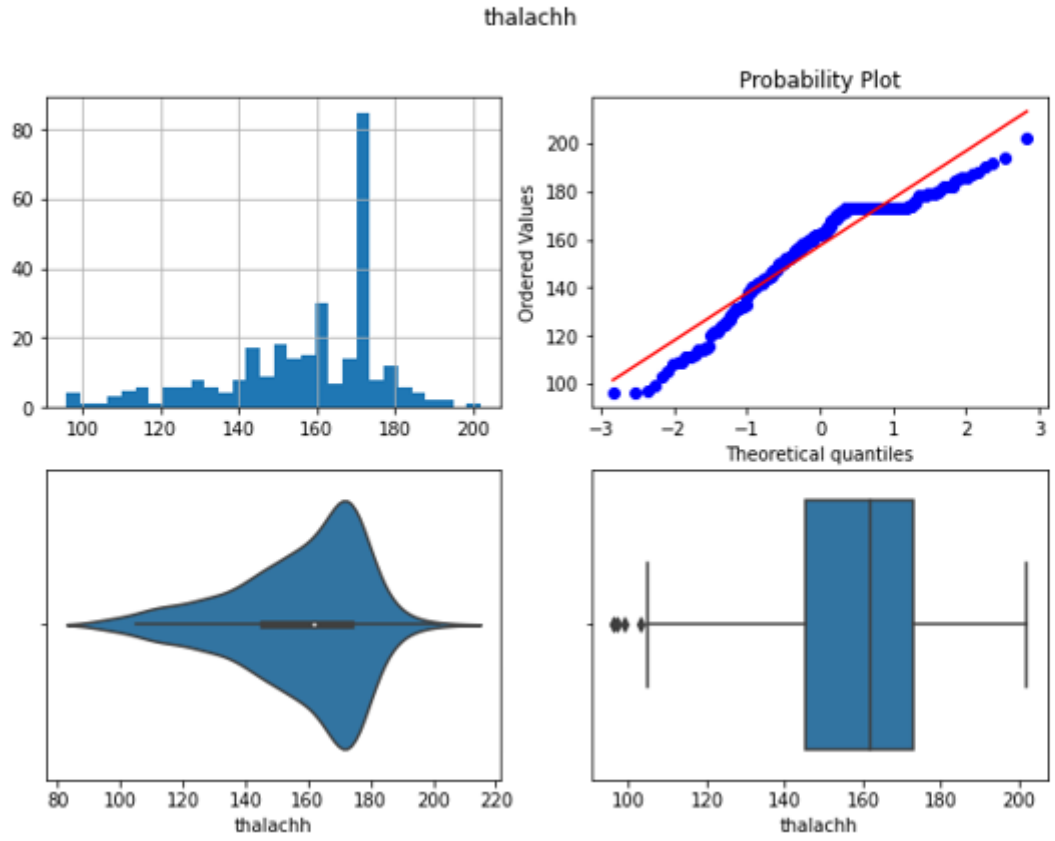
	age	sex	cp	trtbps	chol	fbs	restecg	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	30
mean	53.161716	0.726073	0.864686	132.036304	257.693069	0.115512	0.623762	15
std	8.251646	0.446710	1.008986	14.488475	51.210204	0.320167	0.492019	2
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	9
25%	50.000000	0.000000	0.000000	120.000000	220.000000	0.000000	0.000000	14
50%	51.000000	1.000000	0.000000	132.000000	257.000000	0.000000	1.000000	16
75%	59.000000	1.000000	2.000000	140.000000	299.000000	0.000000	1.000000	17
max	76.000000	1.000000	3.000000	180.000000	564.000000	1.000000	2.000000	20

```
def diagnostic_plots(df, variable, title):
    fig, ax = plt.subplots(figsize=(10,7))

    plt.subplot(2, 2, 1)
    df[variable].hist(bins=30)

    plt.subplot(2, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)

    plt.subplot(2, 2, 3)
    sns.violinplot(x=df[variable])

    plt.subplot(2, 2, 4)
    sns.boxplot(x=df[variable])
    fig.suptitle(title)
    plt.show()
```

In [22]:

In [23]:

```
diagnostic_plots(data, 'thalachh', 'thalachh')
```



In [25]:

```
def get_outlier_boundaries(df, col):
    lower_boundary = df[col].quantile(0.05)
    upper_boundary = df[col].quantile(0.95)
    print('min: {}\nmax: {}'.format(lower_boundary, upper_boundary))
    return lower_boundary, upper_boundary
```

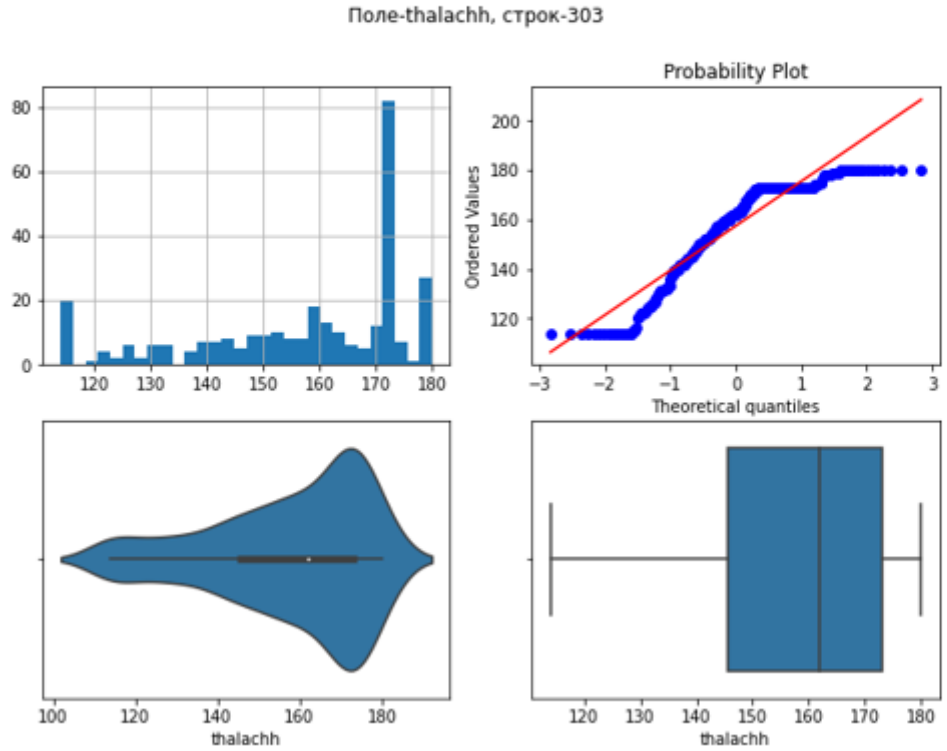
In [26]:

```
lower_boundary, upper_boundary = get_outlier_boundaries(data, 'thalachh')
```

min: 114.0
max: 180.0

In [27]:

```
data[(data.thalachh < lower_boundary)] = lower_boundary
data[(data.thalachh > upper_boundary)] = upper_boundary
title = 'Поле-{}, строк-{}'.format('thalachh', data.shape[0])
diagnostic_plots(data, 'thalachh', title)
```



In [28]:

```
data.describe()
```

Out[28]:

	age	sex	cp	trtbps	chol	fbs	restecg	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	30
mean	62.412541	14.821782	14.980198	133.811881	248.633663	14.283828	14.759076	15
std	30.687456	44.631470	44.589663	17.968910	60.032010	44.801091	44.651684	1
min	34.000000	0.000000	0.000000	94.000000	114.000000	0.000000	0.000000	11
25%	51.000000	0.000000	0.000000	120.000000	208.000000	0.000000	0.000000	14
50%	52.000000	1.000000	1.000000	132.000000	250.000000	0.000000	1.000000	16
75%	62.000000	1.000000	2.000000	140.000000	299.000000	0.000000	1.000000	17
max	180.000000	180.000000	180.000000	180.000000	564.000000	180.000000	180.000000	18

:

```
1 data['Industries']
```

```
0                    Biotechnology, Health Care, Medical
1                    Biotechnology, Health Care, Medical
2                    Biotechnology, Health Care, Medical
3                    Biotechnology, Health Care, Medical
4                    Biotechnology, Health Care, Medical
                              ...
3994                                        Venture Capital
3995    Biotechnology, Health Care, Life Science, Phar...
3996                 Non Profit, STEM Education, Women's
3997    Biopharma, Biotechnology, Health Care, Pharmac...
3998    Advanced Materials, Health Diagnostics, Pharma...
Name: Industries, Length: 3999, dtype: object
```

```
def industries_cut(col):
    new_text = []
    for text in col:
        sep = ','
        text = text.split(sep, 1)[0]
        new_text.append(text)
    #print(val,' and ', Acquisition_Status[i], ' = ', trgt[i])

    new_text = pd.Series(new_text)
    return new_text
data['first_industry'] = industries_cut(data['Industries'])
data = data.drop('Industries', 1)
data['first_industry']
```

```
0              Biotechnology
1              Biotechnology
2              Biotechnology
3              Biotechnology
4              Biotechnology
                   ...
3994         Venture Capital
3995           Biotechnology
3996              Non Profit
3997               Biopharma
3998     Advanced Materials
Name: first_industry, Length: 3999, dtype: object
```

Отбор признаков методом из группы методов фильтрации (корреляция признаков):

```
# Формирование DataFrame с сильными корреляциями
def make_corr_df(df):
    cr = data.corr()
    cr = cr.abs().unstack()
    cr = cr.sort_values(ascending=False)
    cr = cr[cr >= 0.8]
    cr = cr[cr < 1]
    cr = pd.DataFrame(cr).reset_index()
    cr.columns = ['f1', 'f2', 'corr']
    return cr
# Обнаружение групп коррелирующих признаков
```

In [27]:

```
def corr_groups(cr):
    grouped_feature_list = []
    correlated_groups = []

    for feature in cr['f1'].unique():
        if feature not in grouped_feature_list:
            # находим коррелирующие признаки
            correlated_block = cr[cr['f1'] == feature]
            cur_dups = list(correlated_block['f2'].unique()) + [feature]
            grouped_feature_list = grouped_feature_list + cur_dups
            correlated_groups.append(cur_dups)
    return correlated_groups
# Группы коррелирующих признаков
corr_groups(make_corr_df(data))
```

```
[['Total Equity Funding Amount',
  'Total Funding Amount',
  'Total Funding Amount Currency (in USD)',
  'Last Equity Funding Amount Currency (in USD)',
  'Last Equity Funding Amount',
  'Last Funding Amount',
  'Last Funding Amount Currency (in USD)',
  'Total Equity Funding Amount Currency (in USD)']]
```

```
from mlxtend.feature_selection import ExhaustiveFeatureSelector as EFS
logistic_regression = LogisticRegression()

efs1 = EFS(logistic_regression,
           min_features=2,
           max_features=10,
           scoring='accuracy',
           print_progress=True,
           cv=5)

efs1 = efs1.fit(x_train, y_train, custom_feature_names=data.columns)

print('Best accuracy score: %.2f' % efs1.best_score_)
print('Best subset (indices):', efs1.best_idx_)
print('Best subset (corresponding names):', efs1.best_feature_names_)
```

```
Features: 30547/30811
```

```
Best accuracy score: 0.78
Best subset (indices): (2, 3, 4, 5, 6, 14)
Best subset (corresponding names): ('Founded Date', 'Industry Groups', 'Number of Employees', 'Funding Status', 'Last Funding D
ate', 'first_industry')
```

```
# Используем L1-регуляризацию
e_lr1 = LogisticRegression(C=1500, solver='liblinear', penalty='l1', max_iter=2500, random_state=1)
e_lr1.fit(x_train, y_train)
# Коэффициенты регрессии
e_lr1.coef_
```

```
array([[ 0.56976329,  0.20003468, -0.57778977, -0.31884258,  0.83466091,
         1.08868251, -0.23318191, -2.7135252 , -1.32793214,  2.2416377 ,
         3.64060283,  0.9481649 , 18.24111279, -0.12290584,  0.48821289]])
```

```
# Все признаки являются "хорошими"
sel_e_lr1 = SelectFromModel(e_lr1)
sel_e_lr1.fit(x_train, y_train)
list(zip(data.columns, sel_e_lr1.get_support()))
```

In [27]:

```
[('Organization Name', True),
 ('Headquarters Location', True),
 ('Founded Date', True),
 ('Industry Groups', True),
 ('Number of Employees', True),
 ('Funding Status', True),
 ('Last Funding Date', True),
 ('Last Funding Amount', True),
 ('Number of Funding Rounds', True),
 ('Number of Investors', True),
 ('SimilarWeb - Monthly Visits', True),
 ('IPqwery - Patents Granted', True),
 ('IPqwery - Trademarks Registered', True),
 ('Website', True),
 ('first_industry', True)]
```

```
Сравнение
Логистическая регрессия
accuracy = 0.7642140468227425, balanced accuracy = 0.5012260036935301,
precision = 0.5, F1-score = 0.007042253521126761
accuracy = 0.7633779264214047, balanced accuracy = 0.49945295404814005,
precision = 0.0, F1-score = 0.0
Случайный лес
accuracy = 0.7918060200668896, balanced accuracy = 0.5707706752331735,
precision = 0.8113207547169812, F1-score = 0.25671641791044775
accuracy = 0.794314381270903, balanced accuracy = 0.5760898241693437,
precision = 0.8214285714285714, F1-score = 0.27218934911242604
Градиентный бустинг
accuracy = 0.802675585284281, balanced accuracy = 0.5950463243167745,
precision = 0.8382352941176471, F1-score = 0.32571428571428573
accuracy = 0.7959866220735786, balanced accuracy = 0.5735059049924732,
precision = 0.8958333333333334, F1-score = 0.2606060606060606
Наивный Байес
accuracy = 0.7583612040133779, balanced accuracy = 0.5047527041916912,
precision = 0.3333333333333333, F1-score = 0.04620462046204621
```

In [40]:
```
target_logistic_regression = logistic_regression.predict(X_test)
target_random_forest = random_forest.predict(X_test)
target_naive_bayes = naive_bayes.predict(X_test)
target_gradient_boosting = gradient_boosting.predict(X_test)
```

In [38]:
```
print_accuracy(target_logistic_regression, Y_test)
```

```
accuracy = 0.7642140468227425, balanced accuracy = 0.5012260036935301,
precision = 0.5, F1-score = 0.007042253521126761
```

In [39]:
```
print_accuracy(target_random_forest, Y_test)
```

```
accuracy = 0.7918060200668896, balanced accuracy = 0.5707706752331735,
precision = 0.8113207547169812, F1-score = 0.25671641791044775
```

In [40]:
```
print_accuracy(target_naive_bayes, Y_test)
```

```
accuracy = 0.7583612040133779, balanced accuracy = 0.5047527041916912,
precision = 0.3333333333333333, F1-score = 0.04620462046204621
```

In [41]:
```
print_accuracy(target_gradient_boosting, Y_test)
```

```
accuracy = 0.802675585284281, balanced accuracy = 0.5950463243167745,
precision = 0.8382352941176471, F1-score = 0.32571428571428573
```