Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»



**Лабораторная работа №1**

**По курсу «Методы машинного обучения»**

**ИСПОЛНИТЕЛЬ:**

Лю Цзыцзянь

Группа ИУ5И-21М

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ПРОВЕРИЛ:**

Гапанюк Ю.Е.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оглавление

[Лабораторная работа №1 2](#_Toc200472135)

[Импортируйте необходимые библиотеки и отобразите данные 3](#_Toc200472136)

[График Boxplot 4](#_Toc200472137)

[График pie 5](#_Toc200472138)

[Тепловая карта 6](#_Toc200472139)

[Столбчатая диаграмма 7](#_Toc200472140)

[Точечная диаграмма 8](#_Toc200472141)

# Лабораторная работа 2

Задание

1) Выбрать набор данных (датасет), содержащий категориальные и числовые признаки и пропуски в данных. Для выполнения следующих пунктов можно использовать несколько различных наборов данных (один для обработки пропусков, другой для категориальных признаков и т.д.) Просьба не использовать датасет, на котором данная задача решалась в лекции.

2) Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:

устранение пропусков в данных;

кодирование категориальных признаков;

нормализация числовых признаков.

# Импортируйте необходимые библиотеки и отобразите данные

In [1]:

**import** numpy **as** np

**import** pandas **as** pd

**import** seaborn **as** sns

**import** matplotlib.pyplot **as** plt

**%matplotlib** inline

sns**.**set(style**=**"ticks")

**import** plotly.express **as** px

**from** sklearn.impute **import** SimpleImputer

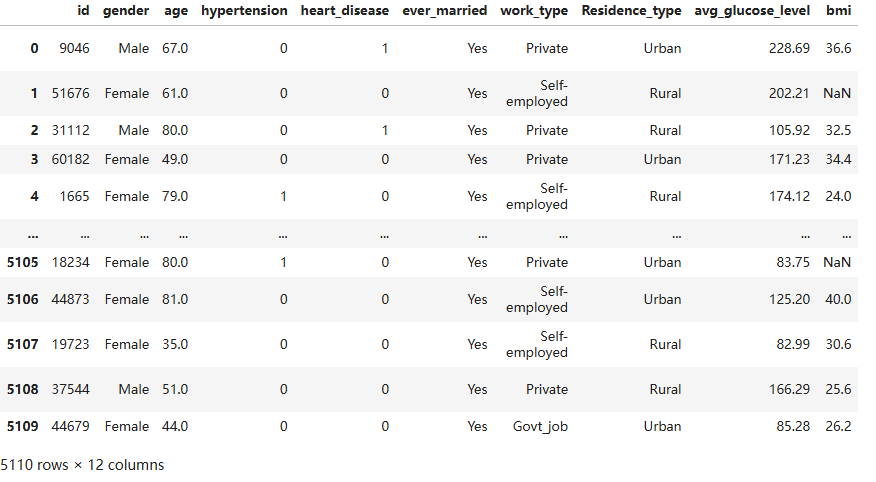
In [2]:

df **=** pd**.**read\_csv('healthcare-dataset-stroke-data.csv')

In [3]:

df

Out[3]:

In [4]:

**def** draw\_missing(df):

total **=** df**.**isnull()**.**sum()**.**sort\_values(ascending**=False**)

percent **=** (df**.**isnull()**.**sum()**/**df**.**isnull()**.**count())**.**sort\_values(ascending**=False**)**\***100

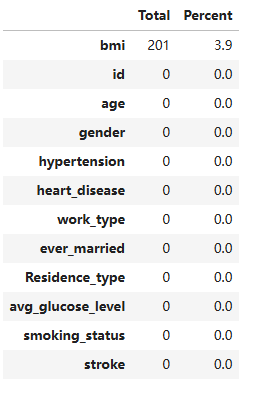
missing\_data **=** pd**.**concat([total, percent], axis**=**1, keys**=**['Total', 'Percent'])

**return** missing\_data

In [5]:

draw\_missing(df)**.**round(1)

Out[5]:



In [6]:

total\_count **=** df**.**shape[0]

print('Всего строк: {}'**.**format(total\_count))

Всего строк: 5110

# Обработка пропусков в числовых данных

In [7]:

*# Выберем числовые колонки с пропущенными значениями*

*# Цикл по колонкам датасета*

num\_cols **=** []

**for** col **in** df**.**columns:

*# Количество пустых значений*

temp\_null\_count **=** df[df[col]**.**isnull()]**.**shape[0]

dt **=** str(df[col]**.**dtype)

**if** temp\_null\_count**>**0 **and** (dt**==**'float64' **or** dt**==**'int64'):

num\_cols**.**append(col)

temp\_perc **=** round((temp\_null\_count **/** total\_count) **\*** 100.0, 2)

print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'**.**format(col, dt, temp\_null\_count, temp\_perc)

)

Колонка bmi. Тип данных float64. Количество пустых значений 201, 3.93%.

In [8]:

*# Фильтр по колонкам с пропущенными значениями*

data\_num **=** df[num\_cols]

data\_num

Out[8]:

|  | **bmi** |
| --- | --- |
| **0** | 36.6 |
| **1** | NaN |
| **2** | 32.5 |
| **3** | 34.4 |
| **4** | 24.0 |
| **...** | ... |
| **5105** | NaN |
| **5106** | 40.0 |
| **5107** | 30.6 |
| **5108** | 25.6 |
| **5109** | 26.2 |

5110 rows × 1 columns

In [9]:

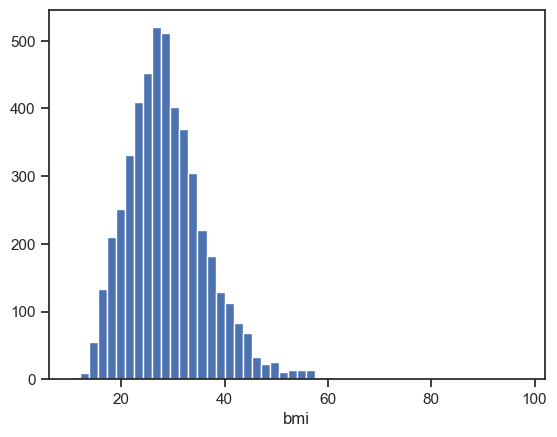
*# Гистограмма по признакам*

**for** col **in** data\_num:

plt**.**hist(df[col], 50)

plt**.**xlabel(col)

plt**.**show()



In [10]:

data\_num\_bmi **=** data\_num[['bmi']]

data\_num\_bmi**.**head()

Out[10]:

|  | **bmi** |
| --- | --- |
| **0** | 36.6 |
| **1** | NaN |
| **2** | 32.5 |
| **3** | 34.4 |
| **4** | 24.0 |

In [11]:

**from** sklearn.impute **import** SimpleImputer

**from** sklearn.impute **import** MissingIndicator

In [12]:

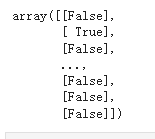
*# Фильтр для проверки заполнения пустых значений*

indicator **=** MissingIndicator()

mask\_missing\_values\_only **=** indicator**.**fit\_transform(data\_num\_bmi)

mask\_missing\_values\_only

Out[12]:



In [13]:

strategies**=**['mean', 'median', 'most\_frequent']

In [14]:

**def** test\_num\_impute(strategy\_param):

imp\_num **=** SimpleImputer(strategy**=**strategy\_param)

df["bmi"] **=** imp\_num**.**fit\_transform(df[['bmi']])

In [15]:

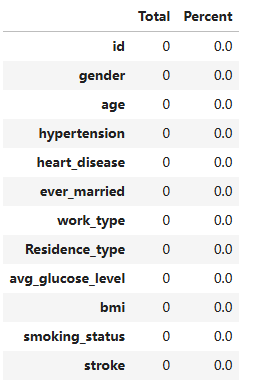
test\_num\_impute(strategies[1])

Проверим, что пропусков нет

In [16]:

draw\_missing(df)**.**round(1)

Out[16]:



Преобразование категориальных признаков в числовые

Использование LabelEncoder

In [17]:

**from** sklearn.preprocessing **import** LabelEncoder

In [18]:

le **=** LabelEncoder()

cat\_enc\_le **=** le**.**fit\_transform(df['work\_type'])

In [19]:

df['work\_type']**.**unique()

Out[19]:

array(['Private', 'Self-employed', 'Govt\_job', 'children', 'Never\_worked'],

dtype=object)

In [20]:

np**.**unique(cat\_enc\_le)

Out[20]:

array([0, 1, 2, 3, 4])

# Кодирование категорий наборами бинарных значений

In [21]:

**from** sklearn.preprocessing **import** OneHotEncoder

In [22]:

cat\_enc**=** df[['work\_type']]

In [23]:

ohe **=** OneHotEncoder()

cat\_enc\_ohe **=** ohe**.**fit\_transform(df[['work\_type']])

In [24]:

cat\_enc\_ohe**.**shape

Out[24]:

(5110, 5)

In [25]:

cat\_enc\_ohe**.**todense()[0:10]

Out[25]:

matrix([[0., 0., 1., 0., 0.],

[0., 0., 0., 1., 0.],

[0., 0., 1., 0., 0.],

[0., 0., 1., 0., 0.],

[0., 0., 0., 1., 0.],

[0., 0., 1., 0., 0.],

[0., 0., 1., 0., 0.],

[0., 0., 1., 0., 0.],

[0., 0., 1., 0., 0.],

[0., 0., 1., 0., 0.]])

In [26]:

cat\_enc**.**head(10)

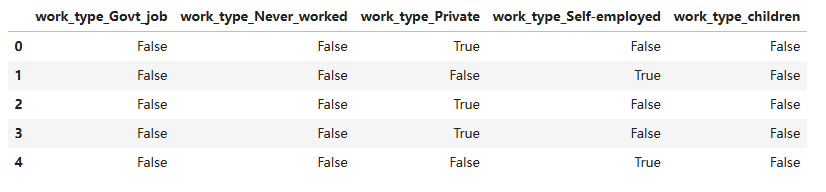
Out[26]:

|  | **work\_type** |
| --- | --- |
| **0** | Private |
| **1** | Self-employed |
| **2** | Private |
| **3** | Private |
| **4** | Self-employed |
| **5** | Private |
| **6** | Private |
| **7** | Private |
| **8** | Private |
| **9** | Private |

In [27]:

pd**.**get\_dummies(cat\_enc)**.**head()

Out[27]:

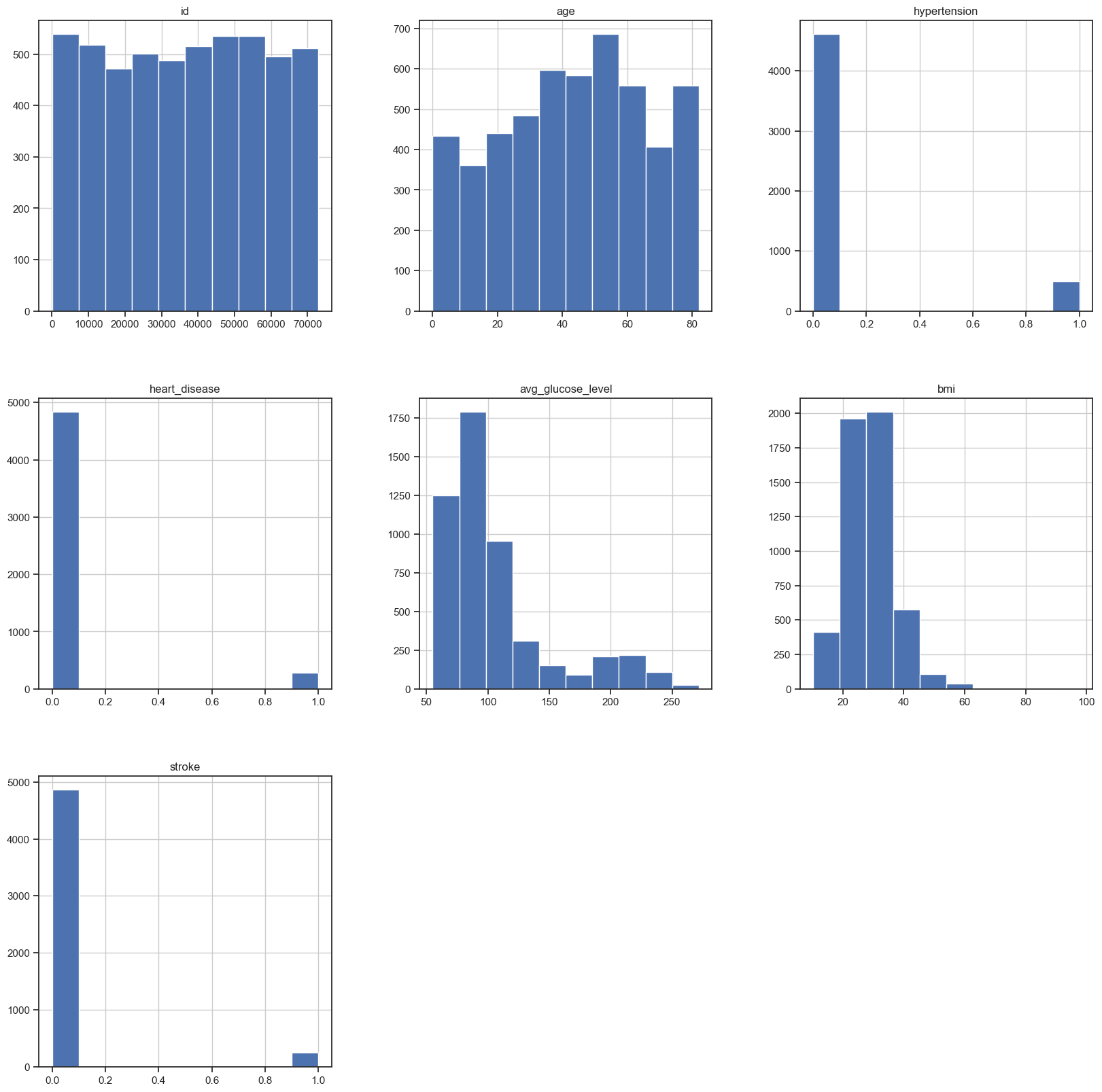


# Нормализация числовых признаков

In [28]:

df**.**hist(figsize**=**(20,20))

plt**.**show()



In [29]:

**import** scipy.stats **as** stats

In [30]:

**def** diagnostic\_plots(df, variable):

plt**.**figure(figsize**=**(15,6))

*# гистограмма*

plt**.**subplot(1, 2, 1)

df[variable]**.**hist(bins**=**30)

*## Q-Q plot*

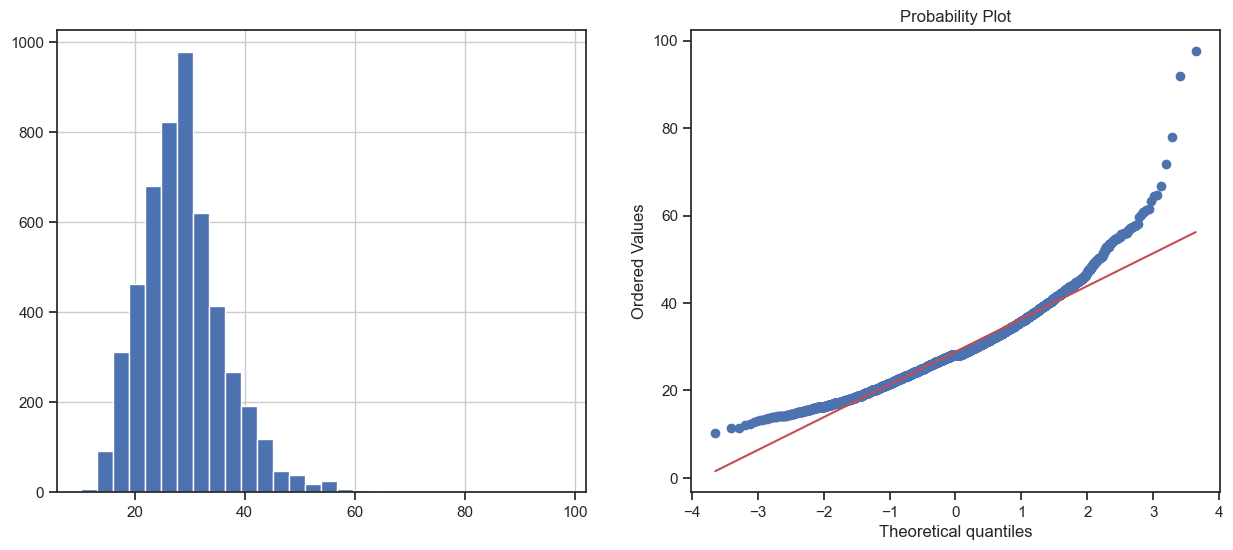
plt**.**subplot(1, 2, 2)

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

plt**.**show()

In [31]:

diagnostic\_plots(df, 'bmi')



In [32]:

df['bmi\_boxcox'], param **=** stats**.**boxcox(df['bmi'])

print('Оптимальное значение λ = {}'**.**format(param))

diagnostic\_plots(df, 'bmi\_boxcox')

Оптимальное значение λ = 0.0024460113998946375

