Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»



**Лабораторная работа №3**

**По курсу «Методы машинного обучения»**

**ИСПОЛНИТЕЛЬ:**

Лю Цзыцзянь

Группа ИУ5И-21М

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

**ПРОВЕРИЛ:**

Гапанюк Ю.Е.

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

Оглавление

[Лабораторная работа 3 3](#_Toc200473731)

[Импортируйте необходимые библиотеки и отобразите данные 4](#_Toc200473732)

[Масштабирование признаков 5](#_Toc200473733)

[MinMax 6](#_Toc200473734)

[Масштабирование данных на основе Z-оценки 8](#_Toc200473735)

[Масштабирование по максимальному значению 9](#_Toc200473736)

[Обработка выбросов для числовых признаков 10](#_Toc200473737)

[Обработка по крайней мере одного нестандартного признака (который не является числовым или категориальным) 14](#_Toc200473738)

[Один метод из группы методов фильтрации (filter methods) 17](#_Toc200473739)

[Один метод из группы методов вложений (embedded methods) 19](#_Toc200473740)

# Лабораторная работа 3

Обработка признаков (часть 2).

Цель лабораторной работы: изучение продвинутых способов предварительной обработки данных для дальнейшего формирования моделей. Требования к отчету:

Отчет по лабораторной работе должен содержать:

титульный лист;

описание задания;

текст программы;

экранные формы с примерами выполнения программы.

Задание:

1. Выбрать один или несколько наборов данных (датасетов) для решения следующих задач. Каждая задача может быть решена на отдельном датасете, или несколько задач могут быть решены на одном датасете. Просьба не использовать датасет, на котором данная задача решалась в лекции.
2. Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:
3. 2.1) масштабирование признаков (не менее чем тремя способами);
4. 2.2) обработку выбросов для числовых признаков (по одному способу для удаления выбросов и для замены выбросов);
5. 2.3) обработку по крайней мере одного нестандартного признака (который не является числовым или категориальным);

2.4) отбор признаков:

\* один метод из группы методов фильтрации (filter methods);

* + один метод из группы методов обертывания (wrapper methods);
  + один метод из группы методов вложений (embedded methods).

# Импортируйте необходимые библиотеки и отобразите данные

In [4]:

**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

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

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.feature\_selection **import** SelectFromModel

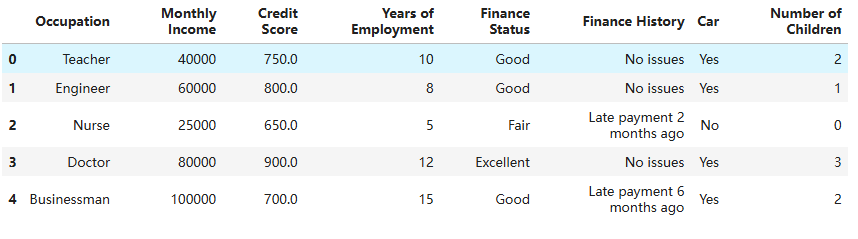
In [5]:

df **=** pd**.**read\_csv('Car Ownership.csv')

In [6]:

df**.**head()

Out[6]:



# Масштабирование признаков

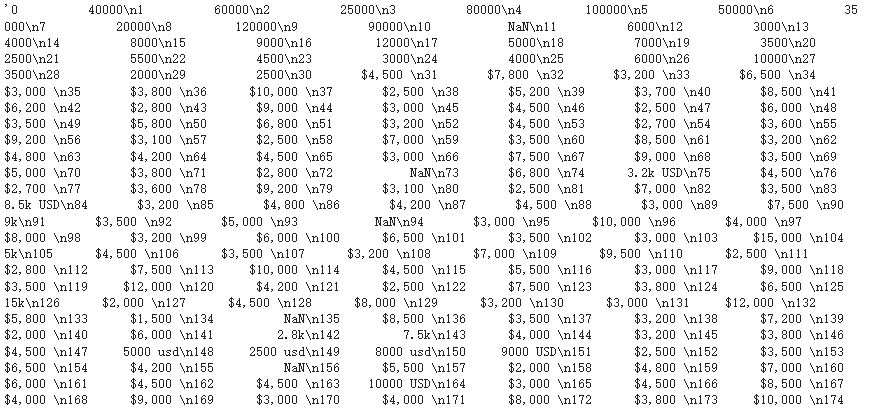
In [7]:

**from** sklearn.preprocessing **import** MinMaxScaler, StandardScaler, Normalizer, MaxAbsScaler

In [8]:

df['Monthly Income']**.**to\_string()

Out[8]:



In [9]:

df['Monthly Income'] **=** df['Monthly Income']**.**str**.**replace('$', ' ')

df['Monthly Income'] **=** df['Monthly Income']**.**str**.**replace(',', '')

df['Monthly Income'] **=** df['Monthly Income']**.**str**.**replace('USD', ' ')

df['Monthly Income'] **=** df['Monthly Income']**.**str**.**replace('k', ' ')

df['Monthly Income'] **=** df['Monthly Income']**.**str**.**replace('usd', ' ')

In [10]:

df**.**at [434, 'Monthly Income'] **=** 4000

In [11]:

df['Monthly Income'] **=** df['Monthly Income']**.**astype('float')

# MinMax

In [12]:

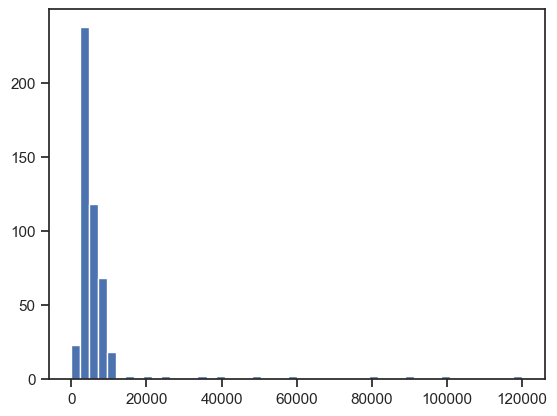
sc1 **=** MinMaxScaler()

sc1\_data **=** sc1**.**fit\_transform(df[['Monthly Income']])

In [13]:

plt**.**hist(df['Monthly Income'], 50)

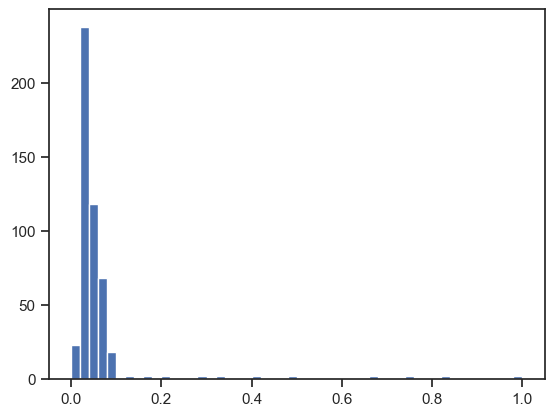
plt**.**show()



In [14]:

plt**.**hist(sc1\_data, 50)

plt**.**show()



# Масштабирование данных на основе Z-оценки

In [15]:

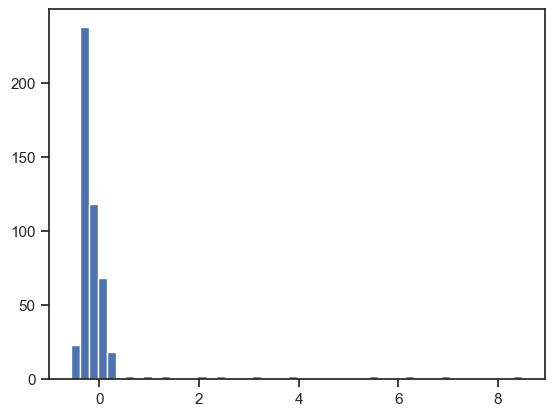
sc2 **=** StandardScaler()

sc2\_data **=** sc2**.**fit\_transform(df[['Monthly Income']])

In [16]:

plt**.**hist(sc2\_data, 50)

plt**.**show()



# Масштабирование по максимальному значению

In [17]:

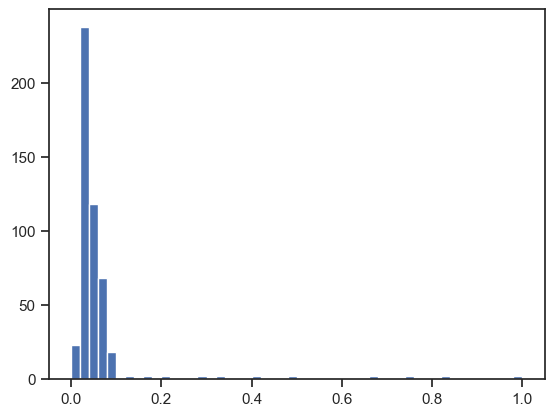
cs51 **=** MaxAbsScaler()

data\_cs51\_scaled\_temp **=** cs51**.**fit\_transform(df[['Monthly Income']])

In [18]:

plt**.**hist(data\_cs51\_scaled\_temp, 50)

plt**.**show()



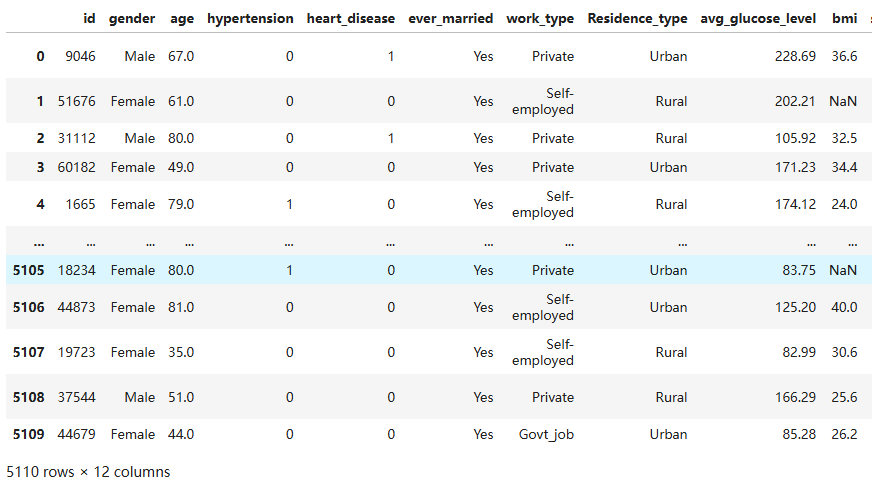
# Обработка выбросов для числовых признаков

In [19]:

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

In [20]:

df

Out[20]:In [21]:

K2 **=** 1.5

IQR **=** df['avg\_glucose\_level']**.**quantile(0.75) **-** df['avg\_glucose\_level']**.**quantile(0.25)

lower\_boundary **=** df['avg\_glucose\_level']**.**quantile(0.25) **-** (K2 **\*** IQR)

upper\_boundary **=** df['avg\_glucose\_level']**.**quantile(0.75) **+** (K2 **\*** IQR)

In [22]:

lower\_boundary, upper\_boundary

Out[22]:

(np.float64(21.977500000000006), np.float64(169.35750000000002))

In [23]:

**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)

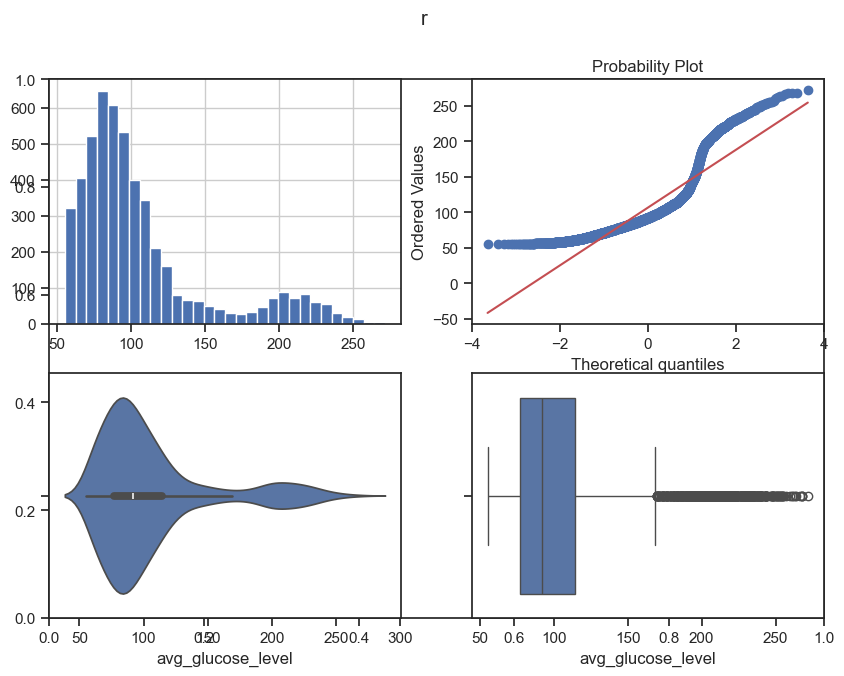
sns**.**boxplot(x**=**df[variable])

fig**.**suptitle(title)

plt**.**show()

In [24]:

diagnostic\_plots(df[['avg\_glucose\_level']], 'avg\_glucose\_level' , "r")



In [25]:

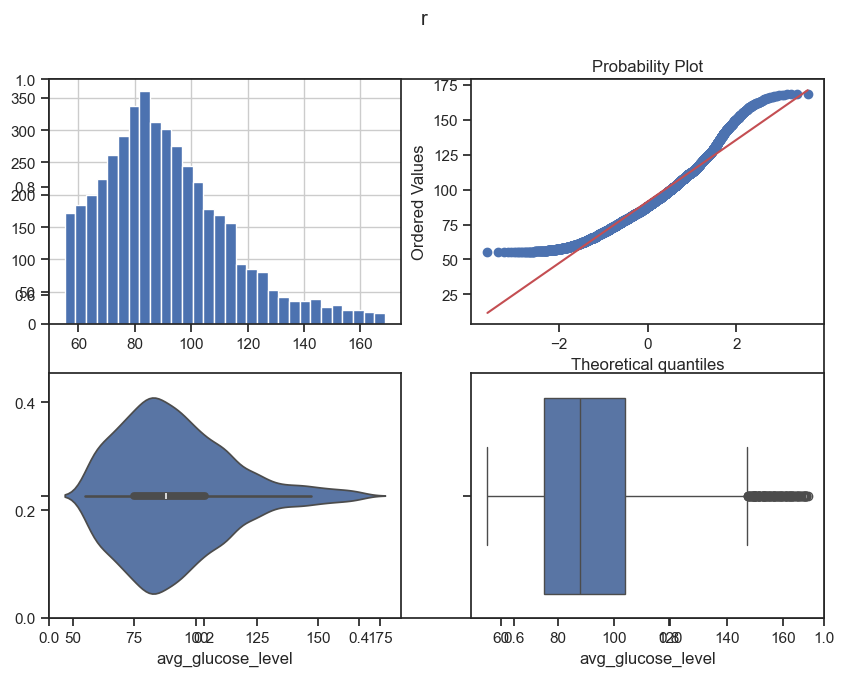
outliers\_temp **=** np**.**where(df['avg\_glucose\_level'] **>** upper\_boundary, **True**,

np**.**where(df['avg\_glucose\_level'] **<** lower\_boundary, **True**, **False**))

*# Удаление данных на основе флага*

data\_trimmed **=** df**.**loc[**~**(outliers\_temp), ]

diagnostic\_plots(data\_trimmed, 'avg\_glucose\_level' , "r")



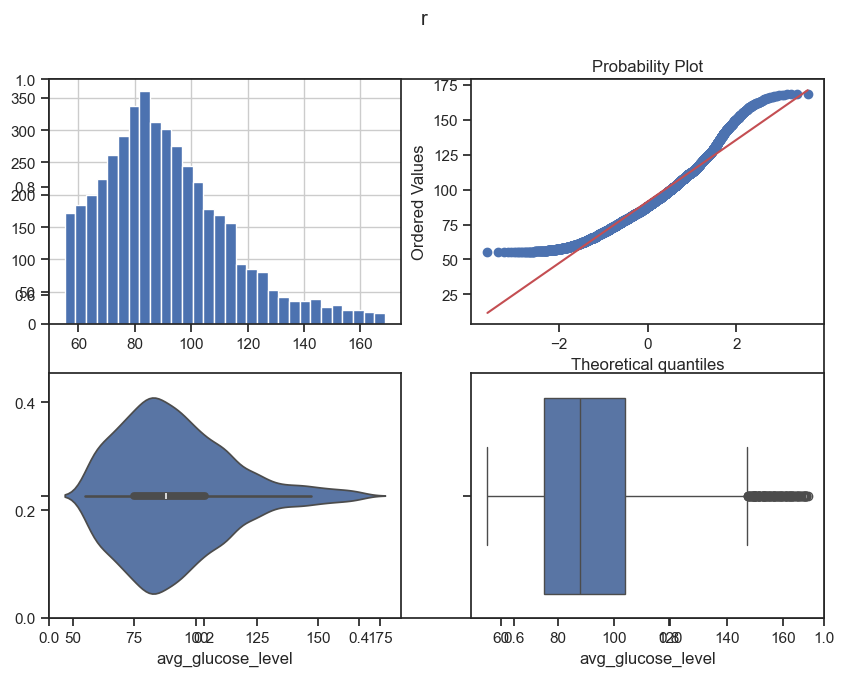
In [27]:

df['avg\_glucose\_level'] **=** np**.**where(df['avg\_glucose\_level'] **>** upper\_boundary, upper\_boundary,

np**.**where(df['avg\_glucose\_level'] **<** lower\_boundary, lower\_boundary, df['avg\_glucose\_level']))

*# diagnostic\_plots(data\_update, 'avg\_glucose\_level', "r")*

diagnostic\_plots(data\_trimmed, 'avg\_glucose\_level', "r")



# Обработка по крайней мере одного нестандартного признака (который не является числовым или категориальным)

In [28]:

df **=** pd**.**read\_csv('seattle-weather.csv')

In [29]:

df

Out[29]:

|  | **date** | **precipitation** | **temp\_max** | **temp\_min** | **wind** | **weather** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | 2012-01-01 | 0.0 | 12.8 | 5.0 | 4.7 | drizzle |
| **1** | 2012-01-02 | 10.9 | 10.6 | 2.8 | 4.5 | rain |
| **2** | 2012-01-03 | 0.8 | 11.7 | 7.2 | 2.3 | rain |
| **3** | 2012-01-04 | 20.3 | 12.2 | 5.6 | 4.7 | rain |
| **4** | 2012-01-05 | 1.3 | 8.9 | 2.8 | 6.1 | rain |
| **...** | ... | ... | ... | ... | ... | ... |
| **1456** | 2015-12-27 | 8.6 | 4.4 | 1.7 | 2.9 | rain |
| **1457** | 2015-12-28 | 1.5 | 5.0 | 1.7 | 1.3 | rain |
| **1458** | 2015-12-29 | 0.0 | 7.2 | 0.6 | 2.6 | fog |
| **1459** | 2015-12-30 | 0.0 | 5.6 | -1.0 | 3.4 | sun |
| **1460** | 2015-12-31 | 0.0 | 5.6 | -2.1 | 3.5 | sun |

1461 rows × 6 columns

In [30]:

df['date'] **=** pd**.**to\_datetime(df['date'])

In [31]:

*# День*

df['day'] **=** df['date']**.**dt**.**day

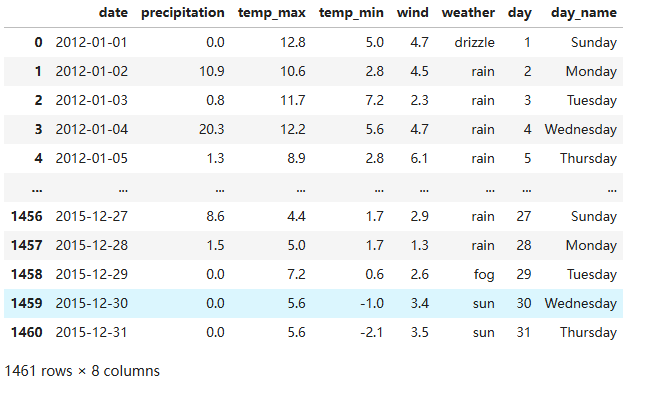
In [32]:

df['day\_name'] **=** df['date']**.**dt**.**day\_name()

In [33]:

df

Out[33]:



# Один метод из группы методов фильтрации (filter methods)

Удаление константных и псевдоконстантных (почти константных) признаков

In [34]:

**from** sklearn.feature\_selection **import** VarianceThreshold

In [35]:

selector\_1211 **=** VarianceThreshold(threshold**=**5.0)

selector\_1211**.**fit(df[['precipitation', 'temp\_max', 'temp\_min', 'wind' ]])

*# Значения дисперсий для каждого признака*

selector\_1211**.**variances\_

Out[35]:

array([44.59445204, 53.98197014, 25.21330161, 2.06592588])

In [36]:

selector\_1211**.**transform(df[['precipitation', 'temp\_max', 'temp\_min', 'wind' ]])

Out[36]:

array([[ 0. , 12.8, 5. ],

[10.9, 10.6, 2.8],

[ 0.8, 11.7, 7.2],

...,

[ 0. , 7.2, 0.6],

[ 0. , 5.6, -1. ],

[ 0. , 5.6, -2.1]])

Один метод из группы методов обертывания (wrapper methods);

Алгоритмы полного перебора

In [37]:

**!**pip install mlxtend

In [38]:

**from** mlxtend.feature\_selection **import** ExhaustiveFeatureSelector **as** EFS

**from** sklearn.neighbors **import** KNeighborsClassifier

**from** sklearn.neighbors **import** KNeighborsRegressor

knn **=** KNeighborsClassifier(n\_neighbors**=**3)

In [39]:

X **=** df[['precipitation', 'temp\_max', 'temp\_min', 'wind' ]]

y **=** df['weather']

In [40]:

efs1 **=** EFS(knn,

min\_features**=**2,

max\_features**=**4,

scoring**=**'accuracy',

print\_progress**=True**,

cv**=**4)

efs1 **=** efs1**.**fit(X, y)

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: 11/11

Best accuracy score: 0.77

Best subset (indices): (0, 3)

Best subset (corresponding names): ('precipitation', 'wind')

Features: 11/11

Best accuracy score: 0.75

Best subset (indices): (0, 3)

Best subset (corresponding names): ('precipitation', 'wind')

# Один метод из группы методов вложений (embedded methods)

Логистическая регрессия

In [41]:

*# Используем L1-регуляризацию*

e\_lr1 **=** LogisticRegression(C**=**1000, solver**=**'liblinear', penalty**=**'l1', max\_iter**=**500, random\_state**=**1)

e\_lr1**.**fit(X, y)

*# Коэффициенты регрессии*

e\_lr1**.**coef\_

Out[41]:

array([[-24.18007186, -0.07968546, 0.04760324, -0.42044543],

[-26.90205339, -0.11487053, 0.11585446, -0.35831287],

[ 0.52793365, -0.32412514, 0.3854563 , 0.05725867],

[ 0.08346434, -0.2934803 , -0.24300255, 0.41556821],

[-35.47785601, 0.14798528, -0.13157143, 0.28018094]])

In [42]:

sel\_e\_lr1 **=** SelectFromModel(e\_lr1)

sel\_e\_lr1**.**fit(X, y)

sel\_e\_lr1**.**get\_support()

Out[42]:

array([ True, True, True, True])