Отчёт по лабораторным работам №1 и №2 по ММО студентки группы ИУ5-21М Дьяконовой Светланы

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set()

import warnings
warnings.filterwarnings('ignore')
```

Загрузка датасета

```
df = pd.read_csv('marketing_campaign.csv', sep='\t', index_col='ID')

df.head()
```

```
      Year_Birth   Education Marital_Status  Kidhome  Teenhome
Dt_Customer  \
ID

5524        1963  Graduation         Single        0         0  2018-
04-09
2174        1960  Graduation         Single        1         1  2020-
08-03
4141        1971  Graduation       Together        0         0  2019-
08-21
6182        1990  Graduation       Together        1         0  2020-
10-02
5324        1987         PhD        Married        1         0  2020-
01-19

      Recency  MntWines  MntFruits  MntMeatProducts  ...  MntGoldProds
\
ID                                                   ...

5524       58       635         88              546  ...            88

2174       38        11          1                6  ...             6

4141       26       426         49              127  ...            42

6182       26        11          4               20  ...             5

5324       94       173         43              118  ...            15


      NumDealsPurchases  NumWebPurchases  NumCatalogPurchases  \
```

```
ID
5524                  3                 8                    10
2174                  2                 1                     1
4141                  1                 8                     2
6182                  2                 2                     0
5324                  5                 5                     3

      NumStorePurchases  NumWebVisitsMonth  Complain  Z_CostContact  \
ID
5524                  4                  7         0              3
2174                  2                  5         0              3
4141                 10                  4         0              3
6182                  4                  6         0              3
5324                  6                  5         0              3

      Z_Revenue  AcceptedCmp
ID
5524         11            1
2174         11            0
4141         11            0
6182         11            0
5324         11            0

[5 rows x 22 columns]

df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 2240 entries, 5524 to 9405
Data columns (total 22 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Year_Birth           2240 non-null   int64
 1   Education            2240 non-null   object
 2   Marital_Status       2240 non-null   object
 3   Kidhome              2240 non-null   int64
 4   Teenhome             2240 non-null   int64
 5   Dt_Customer          2240 non-null   object
 6   Recency              2240 non-null   int64
 7   MntWines             2240 non-null   int64
 8   MntFruits            2240 non-null   int64
 9   MntMeatProducts      2240 non-null   int64
 10  MntFishProducts      2240 non-null   int64
 11  MntSweetProducts     2240 non-null   int64
 12  MntGoldProds         2240 non-null   int64
 13  NumDealsPurchases    2240 non-null   int64
 14  NumWebPurchases      2240 non-null   int64
 15  NumCatalogPurchases  2240 non-null   int64
 16  NumStorePurchases    2240 non-null   int64
 17  NumWebVisitsMonth    2240 non-null   int64
```

```
 18  Complain             2240 non-null   int64
 19  Z_CostContact        2240 non-null   int64
 20  Z_Revenue            2240 non-null   int64
 21  AcceptedCmp          2240 non-null   int64
dtypes: int64(19), object(3)
memory usage: 467.0+ KB
```

Описание датасета

Анализ личности клиентов — это подробный анализ идеальных клиентов компании. Это помогает бизнесу лучше понять своих клиентов и облегчает им модификацию продуктов в соответствии с конкретными потребностями, поведением и проблемами различных типов клиентов.

Люди

- ID: уникальный идентификатор клиента.
- Year_Birth: год рождения клиента.
- Образование: уровень образования клиента.
- Marital_Status: семейное положение клиента.
- Kidhome: Количество детей в семье клиента.
- Teenhome: количество подростков в семье клиента
- Dt_Customer: Дата регистрации клиента в компании
- Недавность: количество дней с момента последней покупки клиента. Пожаловаться: 1, если клиент сотрудничает

Продукты

- MntWines: сумма, потраченная на вино за последние 2 года.
- MntFruits: сумма, потраченная на фрукты за последние 2 года.
- MntMeatProducts: сумма, потраченная на мясо за последние 2 года.
- MntFishProducts: Сумма, потраченная на рыбу за последние 2 года.
- MntSweetProducts: сумма, потраченная на сладости за последние 2 года.
- MntGoldProds: Сумма, потраченная на золото за последние 2 года.

Реклама

- NumDealsPurchases: количество покупок со скидкой
- AcceptedCmp: 1, если клиент принял предложение в последней кампании, 0 в противном случае

Поведение

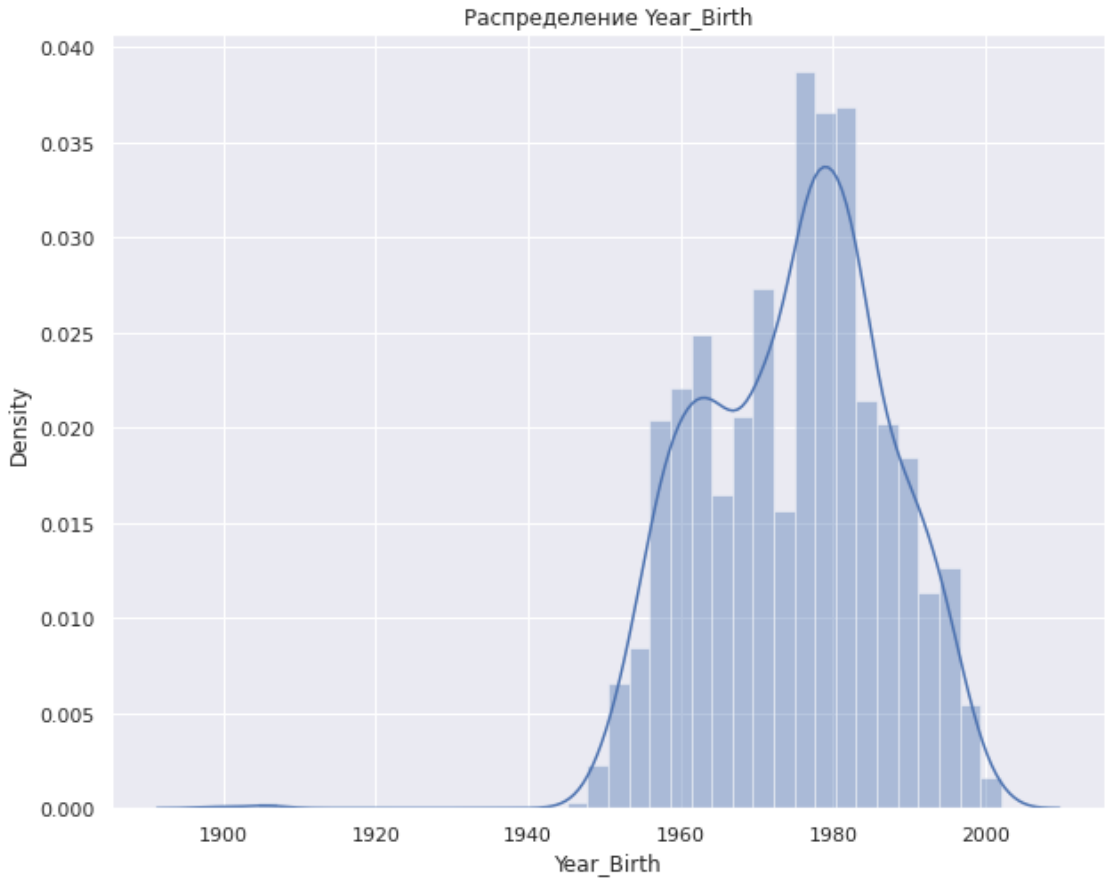
- NumWebPurchases: количество покупок, совершенных через веб-сайт компании.
- NumCatalogPurchases: количество покупок, сделанных с использованием каталога.

- NumStorePurchases: количество покупок, совершенных непосредственно в магазинах.
- NumWebVisitsMonth: количество посещений веб-сайта компании за последний месяц.
- NumDealsPurchases: количество покупок со скидкой.

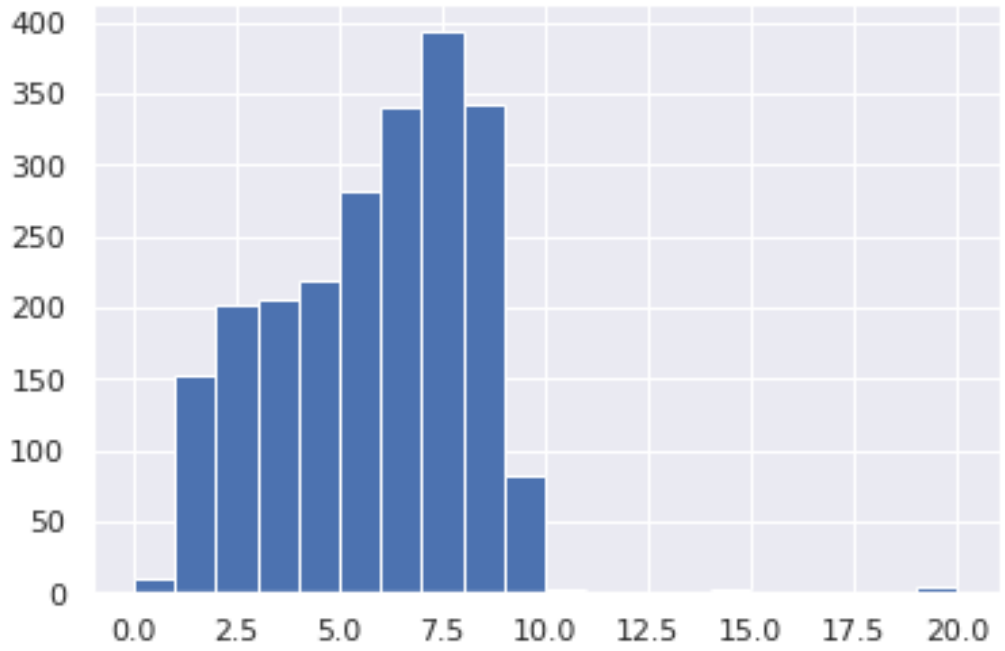
```
numerical = [var for var in df.columns if df[var].dtypes!='O']
categorical = [var for var in df.columns if df[var].dtypes=='O']
```

Лабораторная № 1

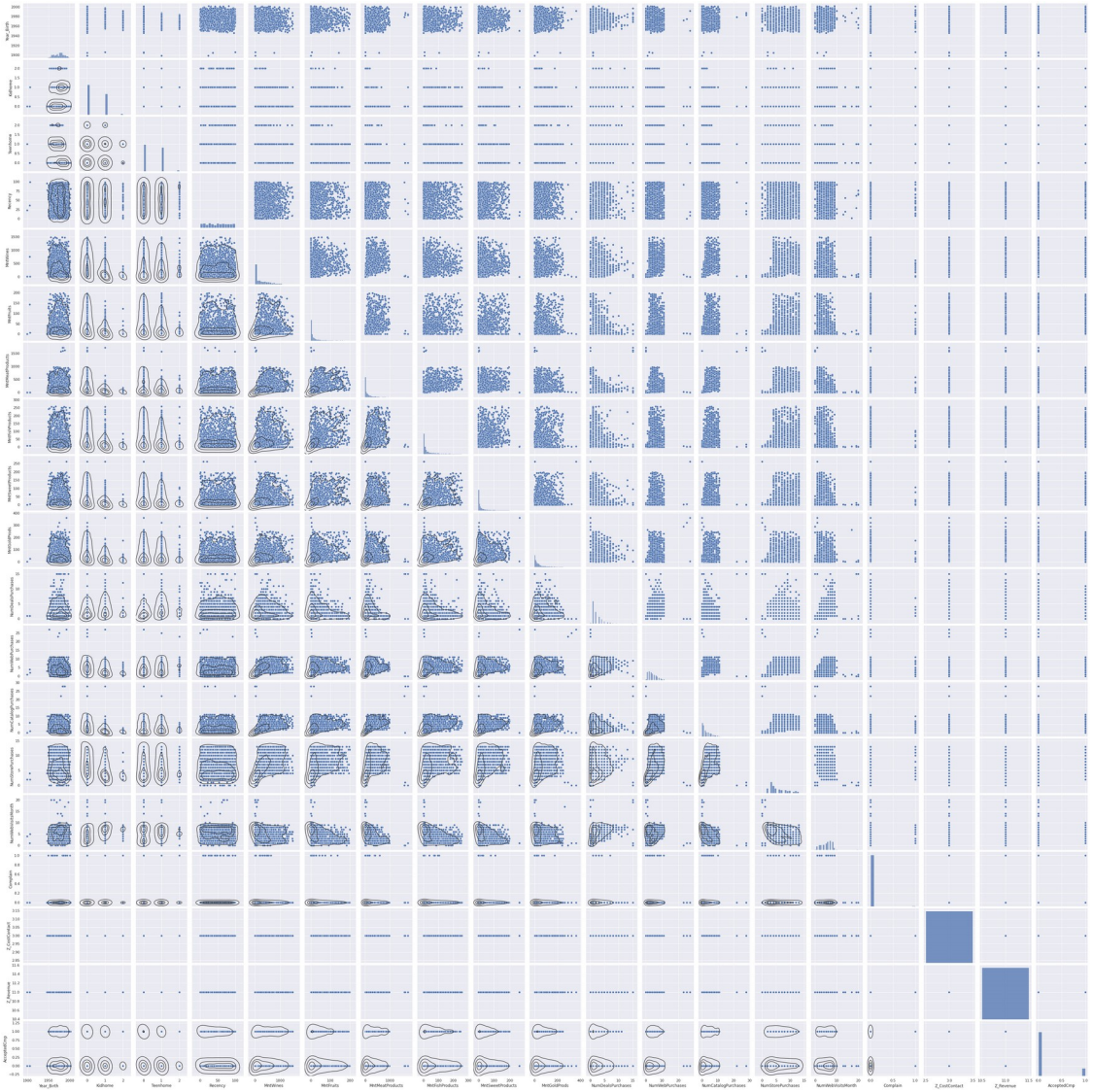
```
# Смотрим распределение таргета, чтобы понять его нормальность

f, ax = plt.subplots(figsize=(10,8))
x = df['Year_Birth']
ax = sns.distplot(x)
ax.set_title("Распределение Year_Birth")
plt.show()
```



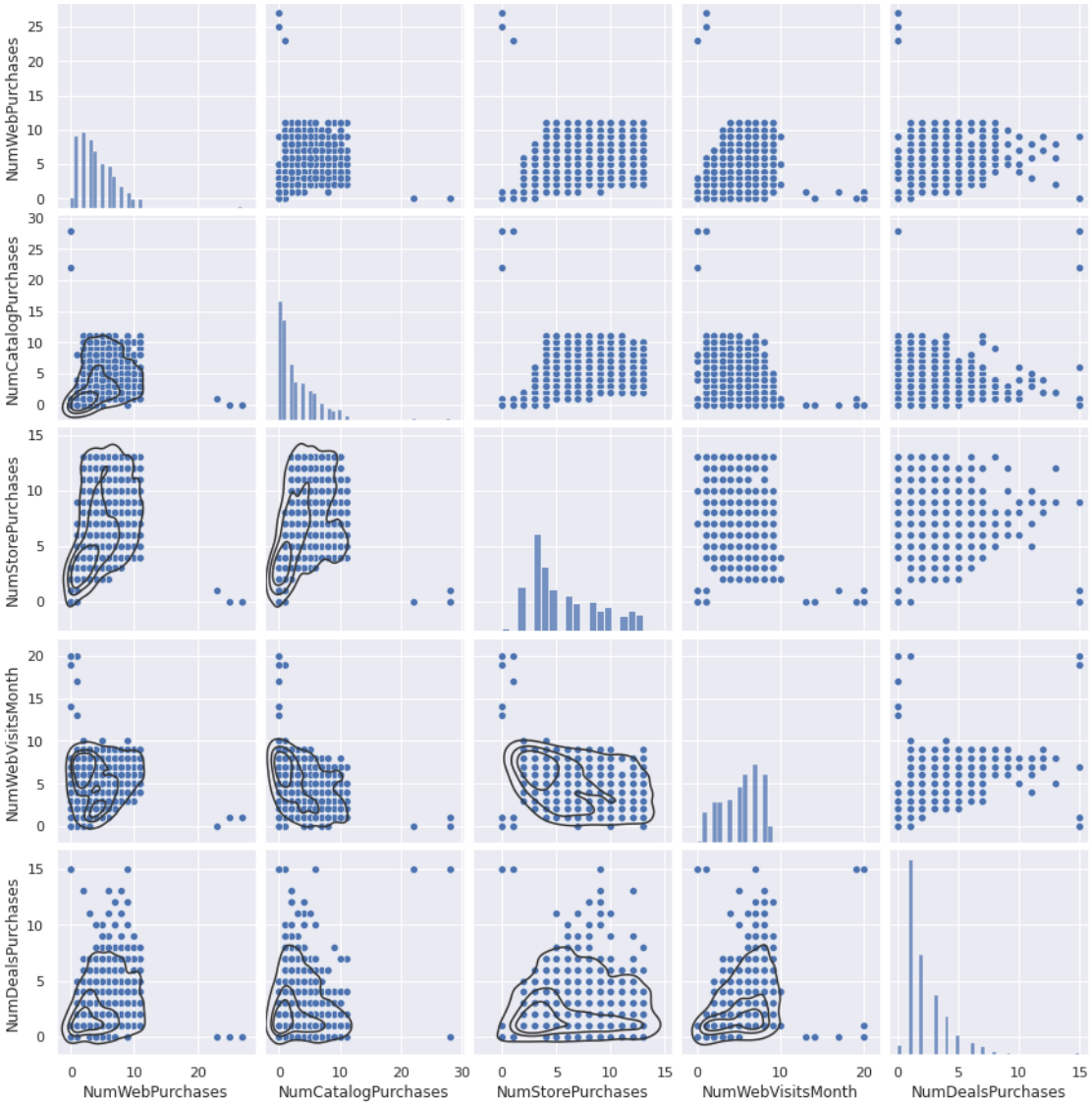

```
# Гистограмма количетсва посещений, наглядно показывает какое количество посещений встречается чаще всего

hist = df["NumWebVisitsMonth"].hist(bins=20) #groupby
```




```
# смотрим попарные зависимости числовых переменных

g = sns.pairplot(df[numerical])
g.map_lower(sns.kdeplot, levels=4, color=".2")
plt.show()
```

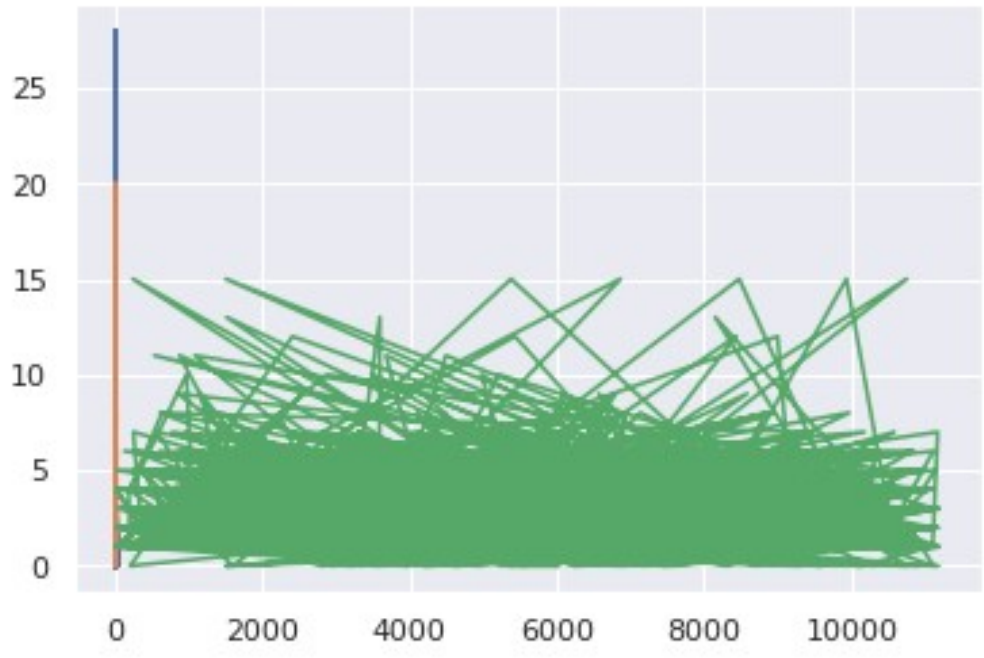


```
# т.к. попарное сравнение, приведенной выше, слишком перенасышено информацией, было принято решения построить этот график не на всех значенияъ нумерикал
df_behavior = df[["NumWebPurchases", "NumCatalogPurchases", "NumStorePurchases", "NumWebVisitsMonth", "NumDealsPurchases"]]
g = sns.pairplot(df_behavior)
g.map_lower(sns.kdeplot, levels=4, color=".2")
plt.show()
```




```
# пример неудачного графика
y1=df[["NumWebPurchases"]]
y2=df[["NumCatalogPurchases"]]
y3=df[["NumStorePurchases"]]
y4 = df[["NumWebVisitsMonth"]]
y5 = df[["NumDealsPurchases"]]

labels = ["NumWebPurchases", "NumCatalogPurchases",
"NumStorePurchases", "NumWebVisitsMonth", "NumDealsPurchases"]

# Basic stacked area chart.
plt.plot(y1, y2, y3, y4,y5)
```

```
[<matplotlib.lines.Line2D at 0x7fbbdcaec490>,
 <matplotlib.lines.Line2D at 0x7fbbdcaec750>,
 <matplotlib.lines.Line2D at 0x7fbbdcaec990>]
```

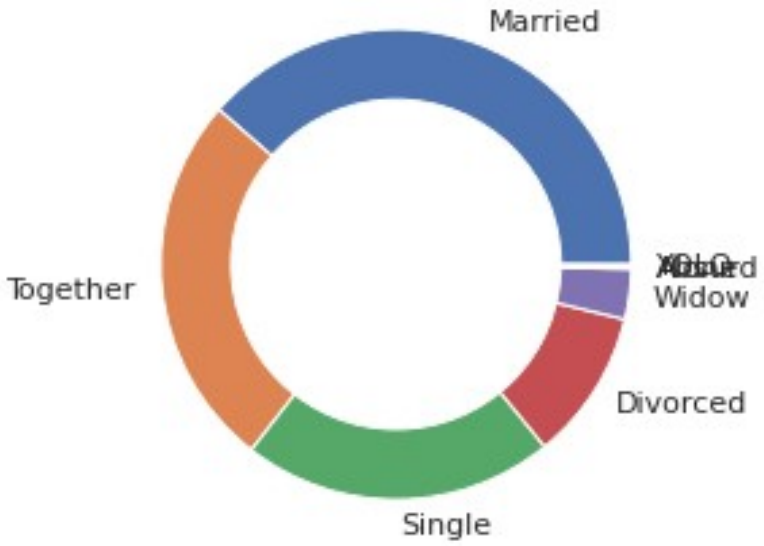



```
import plotly.express as px

df_cat = df[['Education', 'Marital_Status', 'Complain']]
fig = px.parallel_categories(df_cat)
fig.show()

df["Marital_Status"].value_counts()
```

```
Married     864
Together    580
Single      480
Divorced    232
Widow        77
Alone         3
Absurd        2
YOLO          2
Name: Marital_Status, dtype: int64
```

```
# create data
labels = ["Married", "Together", "Single", "Divorced", "Widow",
"Alone", "Absurd", "YOLO"]
size_of_groups=df["Marital_Status"].value_counts()

# Create a pieplot
plt.pie(size_of_groups, labels=labels)

# add a circle at the center to transform it in a donut chart
my_circle=plt.Circle( (0,0), 0.7, color='white')
p=plt.gcf()
p.gca().add_artist(my_circle)
```

```
plt.show()
```



Лабораторная № 2

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set()

data_gr = pd.read_csv('gran_turismo_gt7.csv')

data_gr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 425 entries, 0 to 424
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   model         425 non-null    object 
 1   category      425 non-null    object 
 2   pp            425 non-null    object 
 3   transmission  425 non-null    object 
 4   coll          425 non-null    object 
 5   price         425 non-null    object 
 6   hp            425 non-null    object 
 7   lbs           409 non-null    float64
 8   kg/kw         425 non-null    object 
 9   img_url       425 non-null    object
```

```
dtypes: float64(1), object(9)
memory usage: 33.3+ KB
```

В данных нет пропусков, поэтому он не подходит для выполнения задания с обработкой пропусков

```
data_se = pd.read_csv('sea_ears.csv')
```

```
data_se.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5700 entries, 0 to 5699
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  5700 non-null   int64  
 1   id          5700 non-null   int64  
 2   FN          5700 non-null   object 
 3   SN          5700 non-null   object 
 4   LN          5700 non-null   object 
 5   Captured    5700 non-null   object 
 6   Sex         5700 non-null   object 
 7   Length      5545 non-null   float64
 8   Diam        5079 non-null   float64
 9   Height      4931 non-null   float64
 10  Whole       5687 non-null   float64
 11  Shucke      4334 non-null   float64
 12  Viscera     5158 non-null   float64
 13  Shell       4938 non-null   float64
 14  Rings       4960 non-null   float64
dtypes: float64(8), int64(2), object(5)
memory usage: 668.1+ KB
```

```
data_se.head()
```

```
   Unnamed: 0    id        FN       SN       LN         Captured Sex
Length  \
0           0   835       Kid  College  Machine  20200621T000000   I
0.450
1           1   540  Jalapeno     Glam  Machine  20200308T000000   F
0.500
2           2  2295      Baby     Full   Killer  20200222T000000   F
0.520
3           3   858       Kid     Rock     Head  20200222T000000   F
0.595
4           4  2329       Boy    Block    Death  20201219T000000   I
0.480

    Diam  Height   Whole  Shucke  Viscera   Shell  Rings
0  0.350   0.130  0.5470  0.2450   0.1405  0.1405    8.0
1  0.375   0.140  0.6040  0.2420   0.1415  0.1790   15.0
```

```
2  0.415   0.145  0.8045  0.3325   0.1725  0.2850   10.0
3  0.480   0.150  1.1100  0.4980   0.2280  0.3300   10.0
4  0.390   0.145  0.5825  0.2315   0.1210  0.2550   15.0
```

в датасете есть пропуски, поэтому будем выполнять лабораторную работу на его основе.

```
list(zip(data_se.columns, [i for i in data_se.dtypes]))
```

```
[('Unnamed: 0', dtype('int64')),
 ('id', dtype('int64')),
 ('FN', dtype('O')),
 ('SN', dtype('O')),
 ('LN', dtype('O')),
 ('Captured', dtype('O')),
 ('Sex', dtype('O')),
 ('Length', dtype('float64')),
 ('Diam', dtype('float64')),
 ('Height', dtype('float64')),
 ('Whole', dtype('float64')),
 ('Shucke', dtype('float64')),
 ('Viscera', dtype('float64')),
 ('Shell', dtype('float64')),
 ('Rings', dtype('float64'))]
```

```
# Колонки с пропусками
hcols_with_na = [c for c in data_se.columns if
data_se[c].isnull().sum() > 0]
hcols_with_na
```

```
['Length', 'Diam', 'Height', 'Whole', 'Shucke', 'Viscera', 'Shell',
'Rings']
```

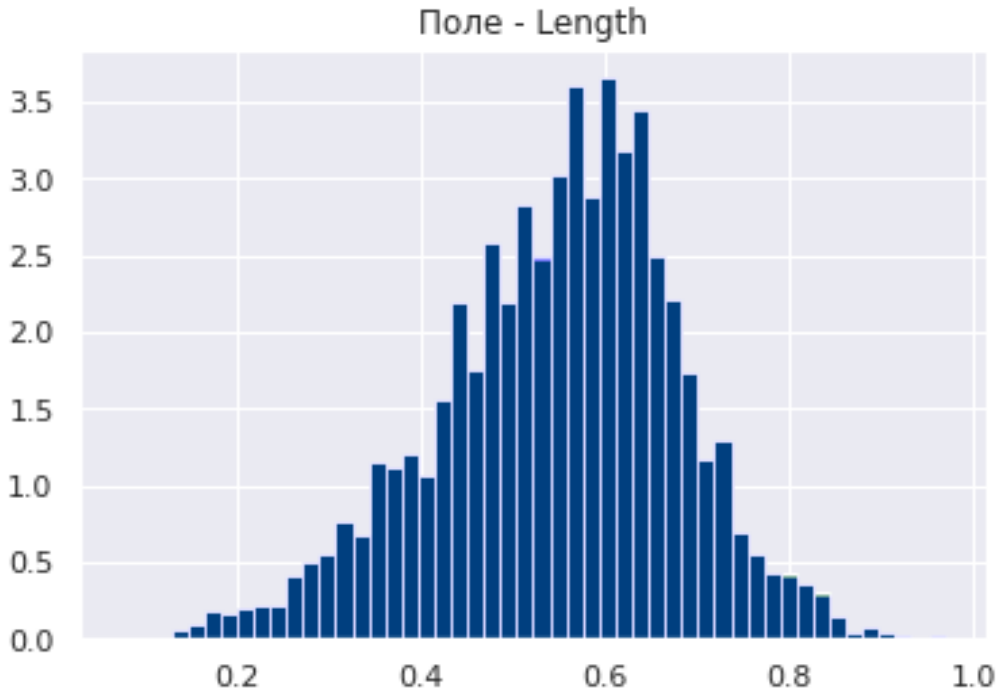
```
# Доля (процент) пропусков
[(c, data_se[c].isnull().mean()) for c in hcols_with_na]
```

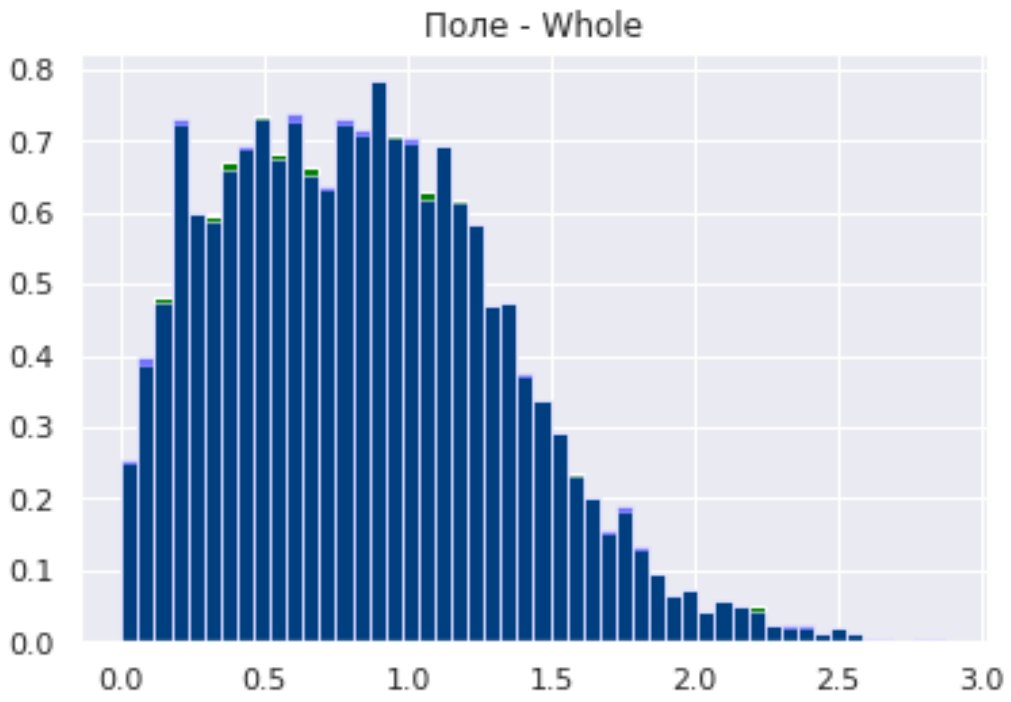
```
[('Length', 0.027192982456140352),
 ('Diam', 0.10894736842105263),
 ('Height', 0.13491228070175437),
 ('Whole', 0.0022807017543859647),
 ('Shucke', 0.23964912280701756),
 ('Viscera', 0.09508771929824561),
 ('Shell', 0.1336842105263158),
 ('Rings', 0.12982456140350876)]
```

В столбцах Length и Whole содержание пропусков меньше 5%, поэтому можно удалить строки содержащие пропущенные значения

```
# Колонки для которых удаляются пропуски
hcols_for_delete = ['Length', 'Whole']

# Удаление пропусков
```

```
new_data = data_se.dropna(axis=0, how='any', subset =
hcols_for_delete)

def plot_hist_diff(old_ds, new_ds, cols):
    """
    Разница между распределениями до и после устранения пропусков
    """
    for c in cols:
        fig = plt.figure()
        ax = fig.add_subplot(111)
        ax.title.set_text('Поле - ' + str(c))
        old_ds[c].hist(bins=50, ax=ax, density=True, color='green')
        new_ds[c].hist(bins=50, ax=ax, color='blue', density=True,
alpha=0.5)
        plt.show()

plot_hist_diff(data_se, new_data, hcols_for_delete)
```






```
new_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5533 entries, 0 to 5699
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  5533 non-null   int64  
 1   id          5533 non-null   int64  
 2   FN          5533 non-null   object 
 3   SN          5533 non-null   object 
 4   LN          5533 non-null   object 
 5   Captured    5533 non-null   object 
 6   Sex         5533 non-null   object 
 7   Length      5533 non-null   float64
 8   Diam        4967 non-null   float64
 9   Height      4852 non-null   float64
 10  Whole       5533 non-null   float64
 11  Shucke      4309 non-null   float64
 12  Viscera     5055 non-null   float64
 13  Shell       4856 non-null   float64
 14  Rings       4881 non-null   float64
dtypes: float64(8), int64(2), object(5)
memory usage: 691.6+ KB
```

Для остальных полей применим метод "заполнения значений для нескольких признаков". А конкретно воспользуемся методом ближайших соседей

```
knnimpute_cols = [
    'Length', 'Diam', 'Height', 'Whole',
    'Shucke', 'Viscera', 'Shell', 'Rings'
]

knn_data = new_data[knnimpute_cols].copy()

# Признаки с пропусками
knn_data.isnull().sum()
```

```
Length        0
Diam        566
Height      681
Whole         0
Shucke     1224
Viscera     478
Shell       677
Rings       652
dtype: int64
```

```
from sklearn.impute import KNNImputer
knnimputer = KNNImputer(
    n_neighbors=5,
    weights='distance',
    metric='nan_euclidean',
    add_indicator=False,
)
knn_data_imputed_temp = knnimputer.fit_transform(knn_data)
knn_data_imputed = pd.DataFrame(knn_data_imputed_temp,
columns=knn_data.columns)
knn_data_imputed.head()
```

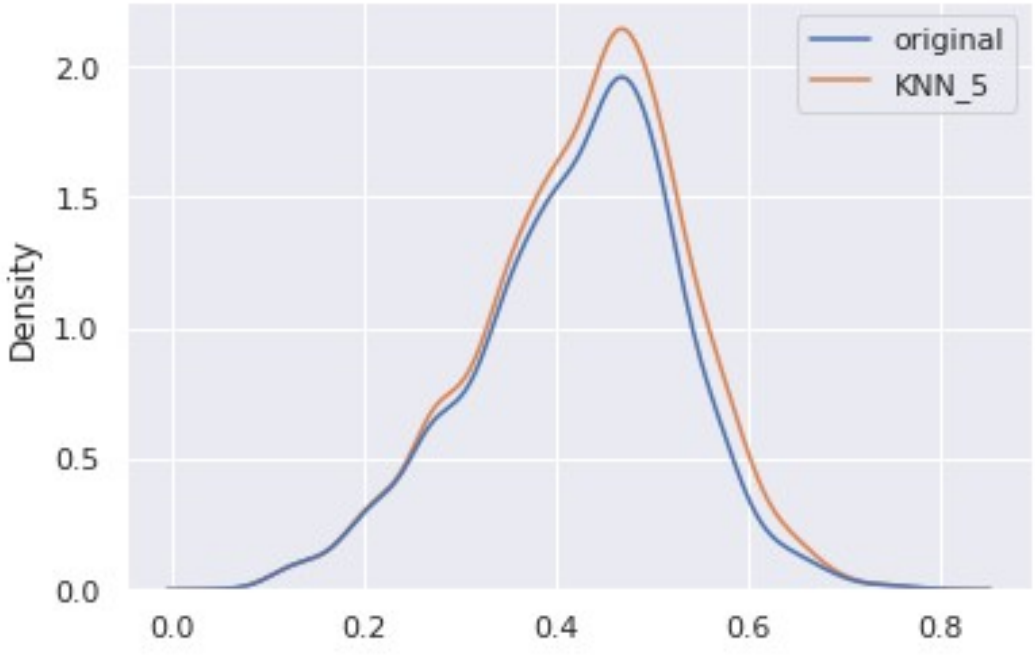
	Length	Diam	Height	Whole	Shucke	Viscera	Shell	Rings
0	0.450	0.350	0.130	0.5470	0.2450	0.1405	0.1405	8.0
1	0.500	0.375	0.140	0.6040	0.2420	0.1415	0.1790	15.0
2	0.520	0.415	0.145	0.8045	0.3325	0.1725	0.2850	10.0
3	0.595	0.480	0.150	1.1100	0.4980	0.2280	0.3300	10.0
4	0.480	0.390	0.145	0.5825	0.2315	0.1210	0.2550	15.0

```
# Пропуски заполнены
knn_data_imputed.isnull().sum()
```

```
Length     0
Diam       0
Height     0
Whole      0
Shucke     0
Viscera    0
Shell      0
Rings      0
dtype: int64
```

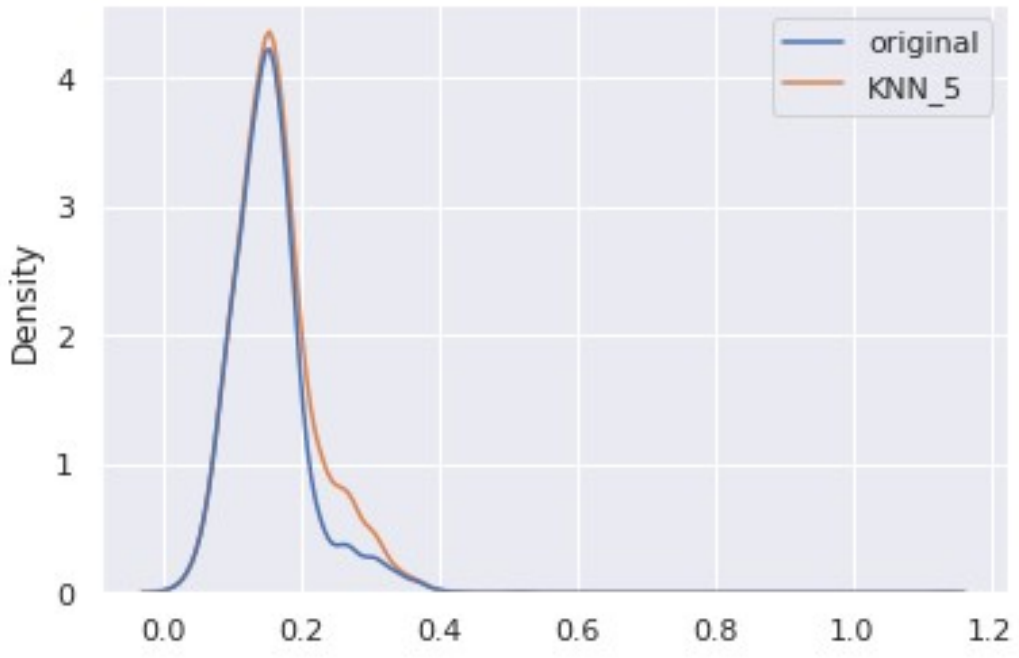
```
LotFrontage_df = pd.DataFrame({'original': knn_data['Diam'].values})
LotFrontage_df['KNN_5'] = knn_data_imputed['Diam']
sns.kdeplot(data=LotFrontage_df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f68bb87db50>
```



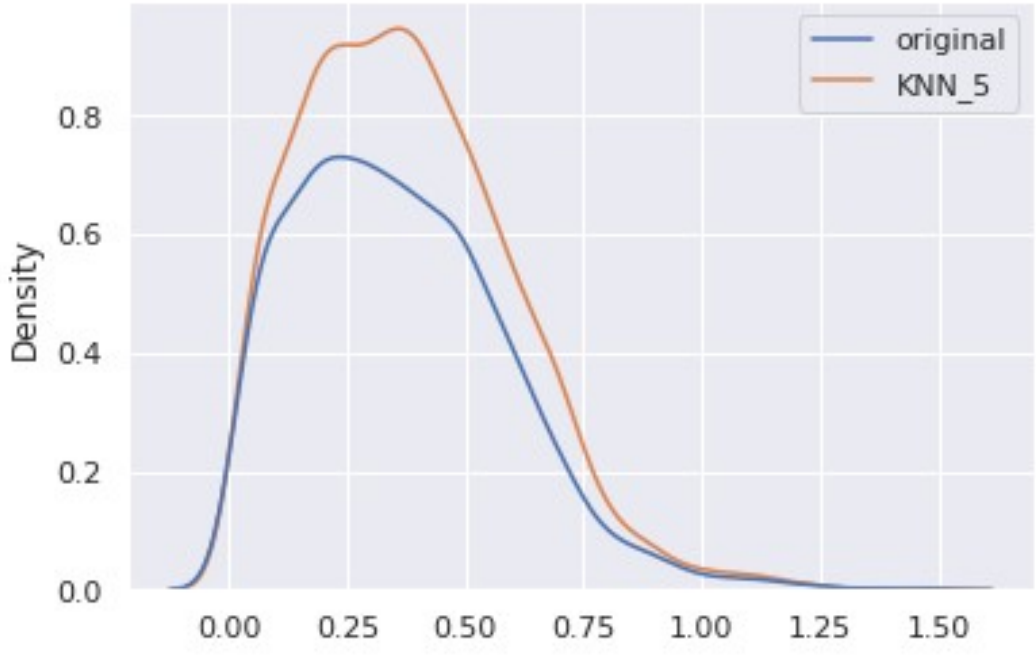

```
LotFrontage_df = pd.DataFrame({'original': knn_data['Height'].values})
LotFrontage_df['KNN_5'] = knn_data_imputed['Height']
sns.kdeplot(data=LotFrontage_df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f68b8fe6790>
```




```
LotFrontage_df = pd.DataFrame({'original': knn_data['Shucke'].values})
LotFrontage_df['KNN_5'] = knn_data_imputed['Shucke']
sns.kdeplot(data=LotFrontage_df)
```

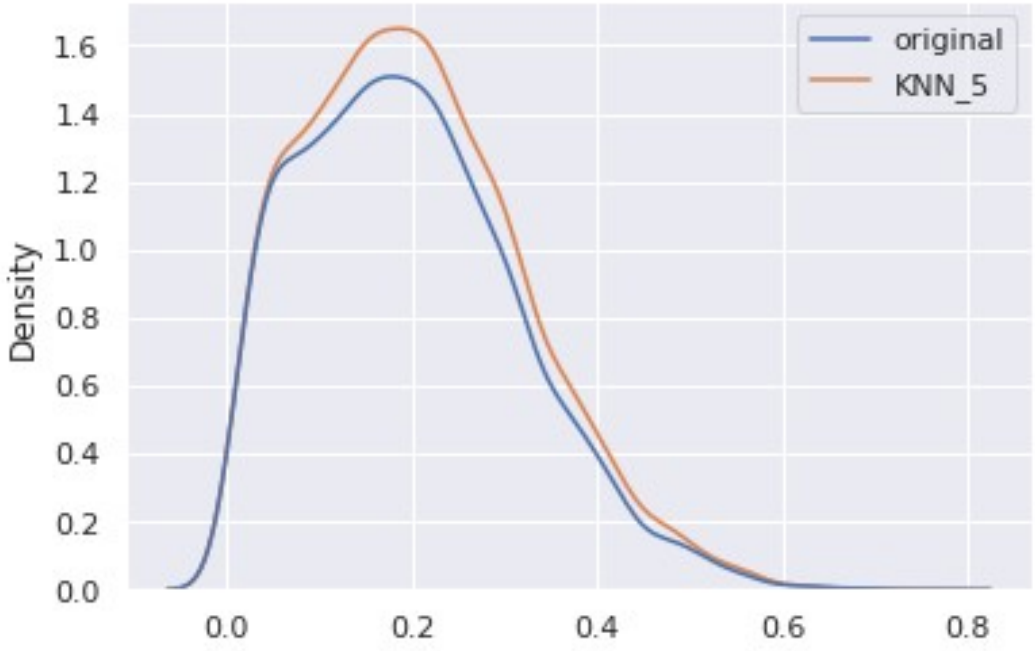
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f68b6ec7790>
```




```
LotFrontage_df = pd.DataFrame({'original':
knn_data['Viscera'].values})
```

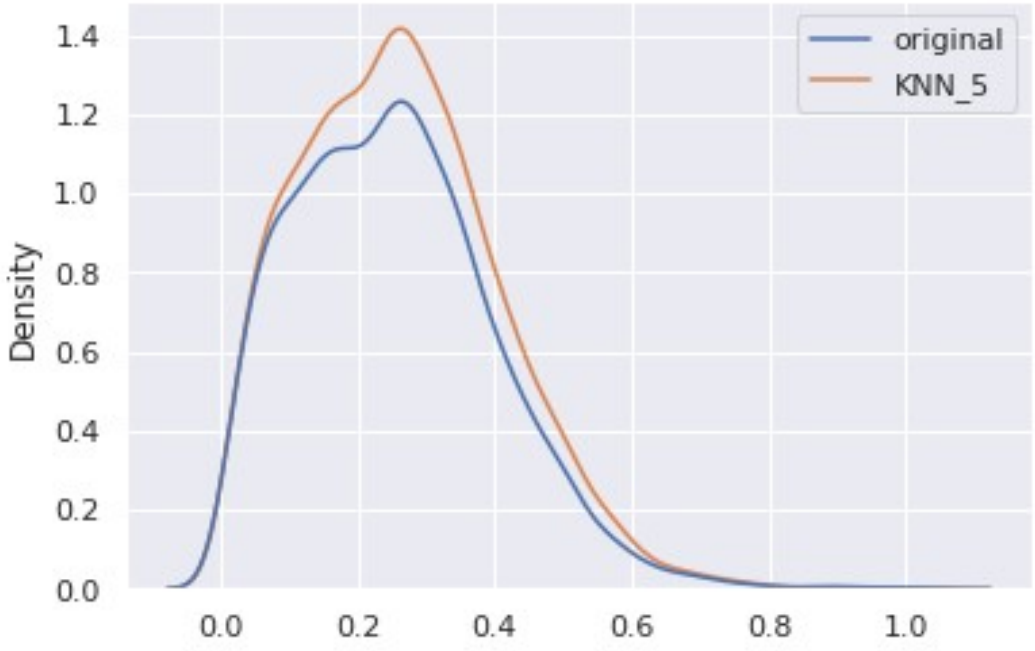
```
LotFrontage_df['KNN_5'] = knn_data_imputed['Viscera']
sns.kdeplot(data=LotFrontage_df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f68b6e98310>
```



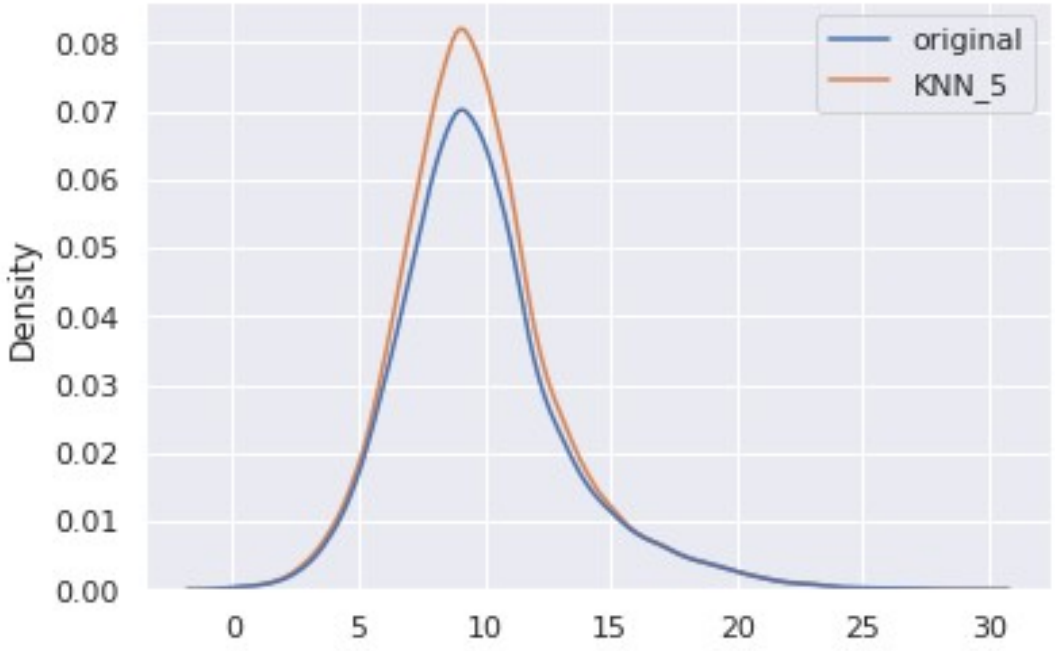

```
LotFrontage_df = pd.DataFrame({'original': knn_data['Shell'].values})
LotFrontage_df['KNN_5'] = knn_data_imputed['Shell']
sns.kdeplot(data=LotFrontage_df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f68b6e7a510>
```



```
LotFrontage_df = pd.DataFrame({'original': knn_data['Rings'].values})
LotFrontage_df['KNN_5'] = knn_data_imputed['Rings']
sns.kdeplot(data=LotFrontage_df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f68b6df4c50>
```



Кодирование категориальных

```
data_se['Sex'].unique()
```

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

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data_se_le = le.fit_transform(data_se['Sex'])
```

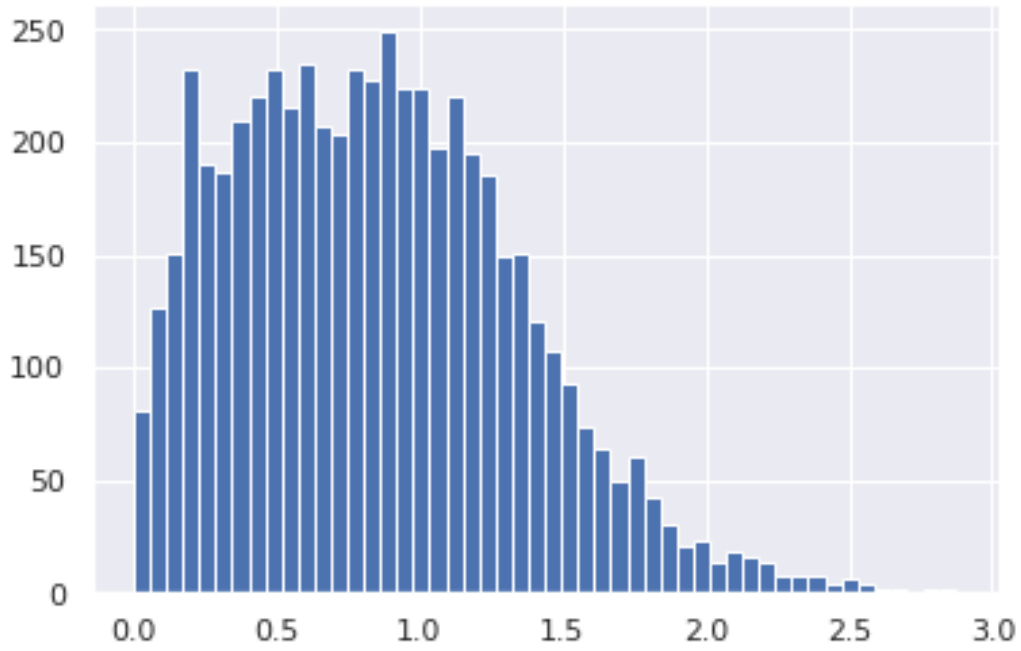
```
np.unique(data_se_le)
```

```
array([0, 1, 2])
```

Нормализация числовых признаков

```
from sklearn.preprocessing import MinMaxScaler
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(new_data[['Whole']])
```

```
plt.hist(new_data['Whole'], 50)
plt.show()
```




```
plt.hist(sc1_data, 50)
plt.show()
```