# Lab3

June 22, 2023

[ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
import scipy.stats as stats
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import RobustScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
```

[ ]:
```
data = pd.read_csv("datasets/Accident.csv")
```

```
/var/folders/fs/5xh23h99763f_blp7m50x23h0000gq/T/ipykernel_68333/3503642161.py:1
: DtypeWarning: Columns (0) have mixed types. Specify dtype option on import or
set low_memory=False.
  data = pd.read_csv("datasets/Accident.csv")
```

[ ]:
```
data.head()
```

[ ]:

| | Accident_Index | 1st_Road_Class | 1st_Road_Number | 2nd_Road_Class | \ |
|---|---|---|---|---|---|
| 0 | 200501BS00001 | A | 3218 | NaN | |
| 1 | 200501BS00002 | B | 450 | C | |
| 2 | 200501BS00003 | C | 0 | NaN | |
| 3 | 200501BS00004 | A | 3220 | NaN | |
| 4 | 200501BS00005 | Unclassified | 0 | NaN | |

| | 2nd_Road_Number | Accident_Severity | Carriageway_Hazards | Date | \ |
|---|---|---|---|---|---|
| 0 | 0.0 | Serious | NaN | 2005-01-04 | |
| 1 | 0.0 | Slight | NaN | 2005-01-05 | |
| 2 | 0.0 | Slight | NaN | 2005-01-06 | |
| 3 | 0.0 | Slight | NaN | 2005-01-07 | |
| 4 | 0.0 | Slight | NaN | 2005-01-10 | |

```
  Day_of_Week  Did_Police_Officer_Attend_Scene_of_Accident  …  \
0     Tuesday                                         1.0  …
1   Wednesday                                         1.0  …
2    Thursday                                         1.0  …
3      Friday                                         1.0  …
4      Monday                                         1.0  …

          Police_Force Road_Surface_Conditions           Road_Type  \
0  Metropolitan Police             Wet or damp  Single carriageway
1  Metropolitan Police                     Dry    Dual carriageway
2  Metropolitan Police                     Dry  Single carriageway
3  Metropolitan Police                     Dry  Single carriageway
4  Metropolitan Police             Wet or damp  Single carriageway

  Special_Conditions_at_Site Speed_limit   Time  Urban_or_Rural_Area  \
0                        NaN          30  17:42                Urban
1                        NaN          30  17:36                Urban
2                        NaN          30  00:15                Urban
3                        NaN          30  10:35                Urban
4                        NaN          30  21:13                Urban

      Weather_Conditions  Year InScotland
0  Raining no high winds  2005         No
1     Fine no high winds  2005         No
2     Fine no high winds  2005         No
3     Fine no high winds  2005         No
4     Fine no high winds  2005         No

[5 rows x 34 columns]
```

[ ]:
```
#                          (    25%)
data.dropna(axis=1, thresh=37625)
```

[ ]:
```
       Accident_Index 1st_Road_Class  1st_Road_Number  2nd_Road_Number  \
0       200501BS00001              A             3218              0.0
1       200501BS00002              B              450              0.0
2       200501BS00003              C                0              0.0
3       200501BS00004              A             3220              0.0
4       200501BS00005   Unclassified                0              0.0
…                 …              …                …                …
50164  2005070502900   Unclassified                0              0.0
50165  2005070502901   Unclassified                0              0.0
50166  2005070502902              A              537              0.0
50167  2005070502903              B             5210            574.0
50168  2005070502904              A              527              0.0
```

```
      Accident_Severity        Date Day_of_Week  \
0               Serious  2005-01-04     Tuesday
1                Slight  2005-01-05   Wednesday
2                Slight  2005-01-06    Thursday
3                Slight  2005-01-07      Friday
4                Slight  2005-01-10      Monday
…                   …         …          …
50164           Serious  2005-07-03      Sunday
50165            Slight  2005-07-03      Sunday
50166           Serious  2005-07-03      Sunday
50167            Slight  2005-07-04      Monday
50168            Slight  2005-07-04      Monday

       Did_Police_Officer_Attend_Scene_of_Accident  \
0                                              1.0
1                                              1.0
2                                              1.0
3                                              1.0
4                                              1.0
…                                              …
50164                                          1.0
50165                                          1.0
50166                                          1.0
50167                                          2.0
50168                                          1.0

                  Junction_Control                      Junction_Detail  …  \
0      Data missing or out of range  Not at junction or within 20 metres  …
1                Auto traffic signal                             Crossroads  …
2      Data missing or out of range  Not at junction or within 20 metres  …
3      Data missing or out of range  Not at junction or within 20 metres  …
4      Data missing or out of range  Not at junction or within 20 metres  …
…                              …                                    …  …
50164      Give way or uncontrolled                             Crossroads  …
50165      Give way or uncontrolled                             Crossroads  …
50166  Data missing or out of range  Not at junction or within 20 metres  …
50167      Give way or uncontrolled                             Roundabout  …
50168      Give way or uncontrolled            Private drive or entrance  …

       Pedestrian_Crossing-Physical_Facilities         Police_Force  \
0                                          1.0  Metropolitan Police
1                                          5.0  Metropolitan Police
2                                          0.0  Metropolitan Police
3                                          0.0  Metropolitan Police
4                                          0.0  Metropolitan Police
…                                          …                   …
50164                                      0.0             Cheshire
```

```
50165                                    0.0            Cheshire
50166                                    0.0            Cheshire
50167                                    0.0            Cheshire
50168                                    0.0            Cheshire

      Road_Surface_Conditions          Road_Type  Speed_limit   Time  \
0                   Wet or damp  Single carriageway          30  17:42
1                           Dry    Dual carriageway          30  17:36
2                           Dry  Single carriageway          30  00:15
3                           Dry  Single carriageway          30  10:35
4                   Wet or damp  Single carriageway          30  21:13
…                         …                  …         …    …
50164                       Dry  Single carriageway          40  18:29
50165                       Dry  Single carriageway          30  20:07
50166                       Dry  Single carriageway          60  16:04
50167               Wet or damp          Roundabout          30  16:45
50168               Wet or damp  Single carriageway          30  13:35

       Urban_or_Rural_Area     Weather_Conditions  Year  InScotland
0                    Urban  Raining no high winds  2005          No
1                    Urban     Fine no high winds  2005          No
2                    Urban     Fine no high winds  2005          No
3                    Urban     Fine no high winds  2005          No
4                    Urban     Fine no high winds  2005          No
…                      …                      …   …         …
50164                Urban     Fine no high winds  2005          No
50165                Urban     Fine no high winds  2005          No
50166                Rural     Fine no high winds  2005          No
50167                Urban     Fine no high winds  2005          No
50168                Urban     Fine no high winds  2005          No

[50169 rows x 31 columns]
```

[ ]:
```
#
def impute_na(df, variable, value):
    df[variable].fillna(value, inplace=True)
impute_na(data, 'Number_of_Vehicles', data['Number_of_Vehicles'].mean())
```

[ ]:
```
data.describe()
```

[ ]:
```
       1st_Road_Number  2nd_Road_Number  \
count     50169.000000     50072.000000
mean        865.940920       298.751578
std        1704.128271      1100.096090
min           0.000000         0.000000
25%           0.000000         0.000000
50%          61.000000         0.000000
```

```
75%         574.000000         0.000000
max        7076.000000      8228.000000

       Did_Police_Officer_Attend_Scene_of_Accident      Latitude  \
count                                 50159.000000  50156.000000
mean                                      1.205188     52.491185
std                                       0.429536      1.079310
min                                       1.000000     51.289060
25%                                       1.000000     51.509316
50%                                       1.000000     51.608980
75%                                       1.000000     53.498600
max                                       3.000000     55.047995

       Location_Easting_OSGR  Location_Northing_OSGR     Longitude  \
count           50156.000000            50156.000000  50156.000000
mean           451681.899673           289155.002193     -1.269933
std             86087.912299           119358.039886      1.256771
min            296300.000000           156110.000000     -3.602717
25%            367370.000000           180480.000000     -2.494029
50%            513030.000000           191740.000000     -0.372561
75%            531292.500000           400320.000000     -0.110122
max            559570.000000           573030.000000      0.300802

       Number_of_Casualties  Number_of_Vehicles  \
count          50169.000000        50169.000000
mean               1.302338            1.827403
std                0.743737            0.686198
min                1.000000            1.000000
25%                1.000000            1.000000
50%                1.000000            2.000000
75%                1.000000            2.000000
max               23.000000           18.000000

       Pedestrian_Crossing-Human_Control  \
count                       50156.000000
mean                            0.006579
std                             0.103746
min                             0.000000
25%                             0.000000
50%                             0.000000
75%                             0.000000
max                             2.000000

       Pedestrian_Crossing-Physical_Facilities   Speed_limit     Year
count                             50159.000000  50169.000000  50169.0
mean                                  0.905162     33.573521   2005.0
std                                   1.914236      9.766182      0.0
```

```
min                                0.000000      10.000000   2005.0
25%                                0.000000      30.000000   2005.0
50%                                0.000000      30.000000   2005.0
75%                                0.000000      30.000000   2005.0
max                                8.000000      70.000000   2005.0
```

[ ]:
```
def obj_col(column):
    return column[1] == 'object'

col_names = []
for col in list(filter(obj_col, list(zip(list(data.columns), list(data.
 ↪dtypes))))):
  col_names.append(col[0])
col_names.append('Speed_limit')
```

[ ]:
```
X_ALL = data.drop(col_names, axis=1)
```

[ ]:
```
#
#
def arr_to_df(arr_scaled):
    res = pd.DataFrame(arr_scaled, columns=X_ALL.columns)
    return res
```

### 0.1 StandardScaler

[ ]:
```
#
X_train, X_test, y_train, y_test = train_test_split(X_ALL, data['Speed_limit'],
                                                    test_size=0.2,
                                                    random_state=1)
#               DataFrame
X_train_df = arr_to_df(X_train)
X_test_df = arr_to_df(X_test)

X_train_df.shape, X_test_df.shape
```

[ ]:
```
((40135, 12), (10034, 12))
```

[ ]:
```
#      StandardScaler
cs11 = StandardScaler()
data_cs11_scaled_temp = cs11.fit_transform(X_ALL)
#       DataFrame
data_cs11_scaled =  arr_to_df(data_cs11_scaled_temp)
data_cs11_scaled
```

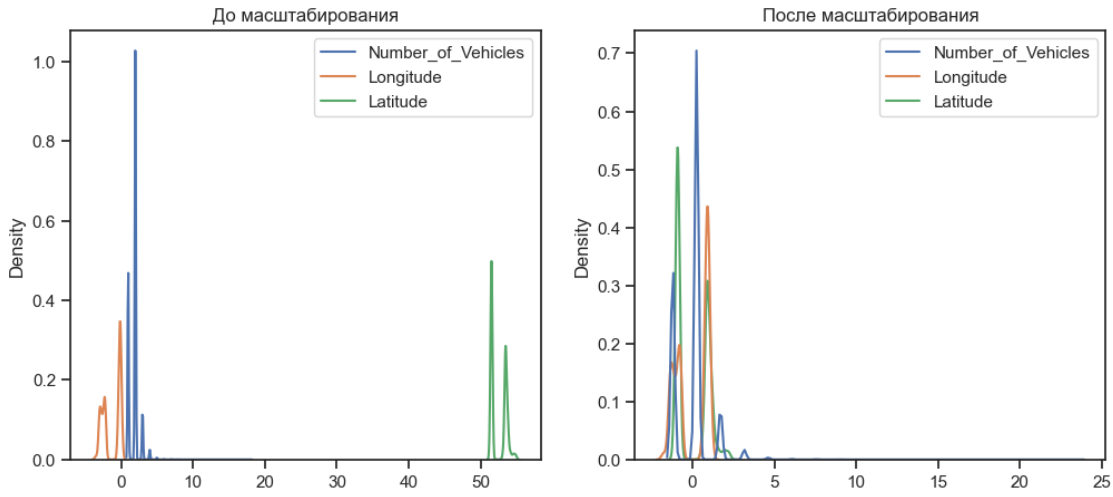
[ ]:
```
        1st_Road_Number  2nd_Road_Number  \
0              1.380226        -0.271571
1             -0.244081        -0.271571
```

```
2            -0.508148        -0.271571
3             1.381400        -0.271571
4            -0.508148        -0.271571
…                 …                …
50164        -0.508148        -0.271571
50165        -0.508148        -0.271571
50166        -0.193028        -0.271571
50167         2.549164         0.250206
50168        -0.198896        -0.271571

       Did_Police_Officer_Attend_Scene_of_Accident  Latitude  \
0                                        -0.477700 -0.928463
1                                        -0.477700 -0.899760
2                                        -0.477700 -0.894918
3                                        -0.477700 -0.934628
4                                        -0.477700 -0.922296
…                                              …       …
50164                                    -0.477700  0.638403
50165                                    -0.477700  0.556717
50166                                    -0.477700  0.709095
50167                                     1.850416  0.854118
50168                                    -0.477700  0.609283

       Location_Easting_OSGR  Location_Northing_OSGR  Longitude  \
0                   0.859573               -0.929272   0.858369
1                   0.842033               -0.900702   0.842027
2                   0.846098               -0.895759   0.846205
3                   0.873745               -0.935221   0.872141
4                   0.887219               -0.922570   0.885862
…                        …                       …          …
50164              -0.936400                0.634442  -0.926521
50165              -0.951733                0.552335  -0.941498
50166              -0.616375                0.704819  -0.598538
50167              -1.018177                0.851857  -1.012682
50168              -0.736021                0.604532  -0.721037

       Number_of_Casualties  Number_of_Vehicles  \
0                 -0.406516           -1.205791
1                 -0.406516           -1.205791
2                 -0.406516            0.251529
3                 -0.406516           -1.205791
4                 -0.406516           -1.205791
…                       …                   …
50164              6.316361            1.708849
50165              0.938059            0.251529
50166             -0.406516           -1.205791
50167              0.938059            0.251529
```

```
50168             -0.406516            0.251529

       Pedestrian_Crossing-Human_Control  \
0                                -0.06342
1                                -0.06342
2                                -0.06342
3                                -0.06342
4                                -0.06342
…                                   …
50164                            -0.06342
50165                            -0.06342
50166                            -0.06342
50167                            -0.06342
50168                            -0.06342

       Pedestrian_Crossing-Physical_Facilities  Year
0                                     0.049544   0.0
1                                     2.139172   0.0
2                                    -0.472863   0.0
3                                    -0.472863   0.0
4                                    -0.472863   0.0
…                                          …   …
50164                                -0.472863   0.0
50165                                -0.472863   0.0
50166                                -0.472863   0.0
50167                                -0.472863   0.0
50168                                -0.472863   0.0

[50169 rows x 12 columns]
```

[ ]:
```
#
def draw_kde(col_list, df1, df2, label1, label2):
    fig, (ax1, ax2) = plt.subplots(
        ncols=2, figsize=(12, 5))
    #
    ax1.set_title(label1)
    sns.kdeplot(data=df1[col_list], ax=ax1)
    #
    ax2.set_title(label2)
    sns.kdeplot(data=df2[col_list], ax=ax2)
    plt.show()
```

[ ]:
```
draw_kde(['Number_of_Vehicles', 'Longitude', 'Latitude'], data,␣
 ↪data_cs11_scaled, '          ', '            ')
```




## 0.2 "Mean Normalisation"

[ ]:
```
#
X_train, X_test, y_train, y_test = train_test_split(X_ALL, data['Speed_limit'],
                                                    test_size=0.2,
                                                    random_state=1)
#               DataFrame
X_train_df = arr_to_df(X_train)
X_test_df = arr_to_df(X_test)

X_train_df.shape, X_test_df.shape
```

[ ]:
```
((40135, 12), (10034, 12))
```

[ ]:
```
class MeanNormalisation:

    def fit(self, param_df):
        self.means = X_train.mean(axis=0)
        maxs = X_train.max(axis=0)
        mins = X_train.min(axis=0)
        self.ranges = maxs - mins

    def transform(self, param_df):
        param_df_scaled = (param_df - self.means) / self.ranges
        return param_df_scaled

    def fit_transform(self, param_df):
        self.fit(param_df)
        return self.transform(param_df)
```

[ ]:
```
sc21 = MeanNormalisation()
data_cs21_scaled = sc21.fit_transform(X_ALL)
data_cs21_scaled.describe()
```

[ ]:

| | 1st_Road_Number | 2nd_Road_Number | \ |
|---|---|---|---|
| count | 50169.000000 | 50072.000000 | |
| mean | -0.000535 | -0.000135 | |
| std | 0.240832 | 0.133702 | |
| min | -0.122912 | -0.036444 | |
| 25% | -0.122912 | -0.036444 | |
| 50% | -0.114292 | -0.036444 | |
| 75% | -0.041793 | -0.036444 | |
| max | 0.877088 | 0.963556 | |

| | Did_Police_Officer_Attend_Scene_of_Accident | Latitude | \ |
|---|---|---|---|
| count | 50159.000000 | 50156.000000 | |
| mean | -0.000419 | -0.001192 | |
| std | 0.214768 | 0.287132 | |
| min | -0.103013 | -0.320997 | |
| 25% | -0.103013 | -0.262402 | |
| 50% | -0.103013 | -0.235888 | |
| 75% | -0.103013 | 0.266813 | |
| max | 0.896987 | 0.679003 | |

| | Location_Easting_OSGR | Location_Northing_OSGR | Longitude | \ |
|---|---|---|---|---|
| count | 50156.000000 | 50156.000000 | 50156.000000 | |
| mean | 0.001293 | -0.001188 | 0.001270 | |
| std | 0.327592 | 0.286285 | 0.322451 | |
| min | -0.589985 | -0.320302 | -0.597255 | |
| 25% | -0.319541 | -0.261849 | -0.312798 | |
| 50% | 0.234742 | -0.234842 | 0.231509 | |
| 75% | 0.304236 | 0.265446 | 0.298844 | |
| max | 0.411841 | 0.679698 | 0.404275 | |

| | Number_of_Casualties | Number_of_Vehicles | \ |
|---|---|---|---|
| count | 50169.000000 | 50169.000000 | |
| mean | -0.000113 | 0.000170 | |
| std | 0.033806 | 0.040365 | |
| min | -0.013856 | -0.048501 | |
| 25% | -0.013856 | -0.048501 | |
| 50% | -0.013856 | 0.010323 | |
| 75% | -0.013856 | 0.010323 | |
| max | 0.986144 | 0.951499 | |

| | Pedestrian_Crossing-Human_Control | \ |
|---|---|---|
| count | 50156.000000 | |
| mean | -0.000025 | |

```
std                              0.051873
min                             -0.003315
25%                             -0.003315
50%                             -0.003315
75%                             -0.003315
max                              0.996685

       Pedestrian_Crossing-Physical_Facilities  Year
count                             50159.000000   0.0
mean                                  0.000120   NaN
std                                   0.239279   NaN
min                                  -0.113026   NaN
25%                                  -0.113026   NaN
50%                                  -0.113026   NaN
75%                                  -0.113026   NaN
max                                   0.886974   NaN
```

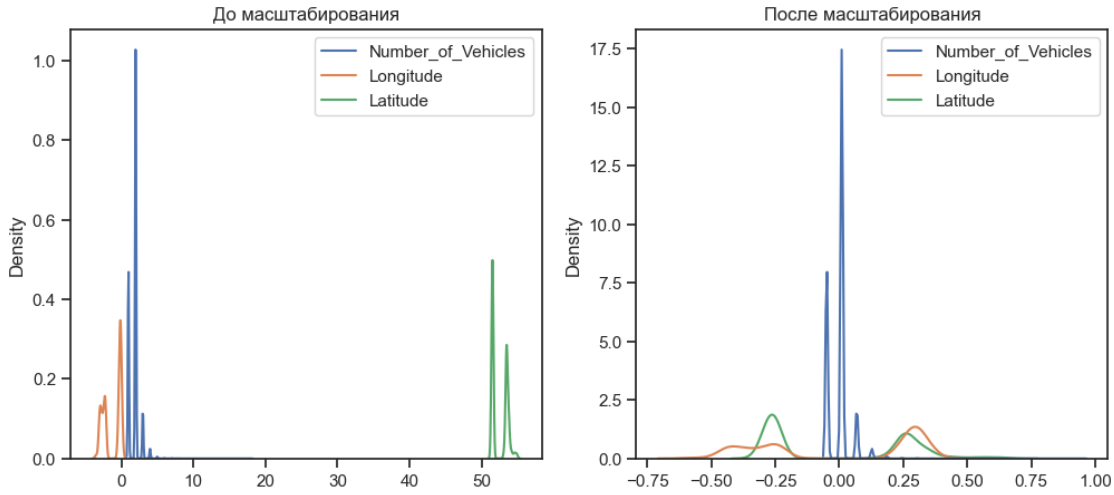
[ ]:
```
cs22 = MeanNormalisation()
cs22.fit(X_train)
data_cs22_scaled_train = cs22.transform(X_train)
data_cs22_scaled_test = cs22.transform(X_test)
```

[ ]:
```
data_cs22_scaled_train.describe()
```

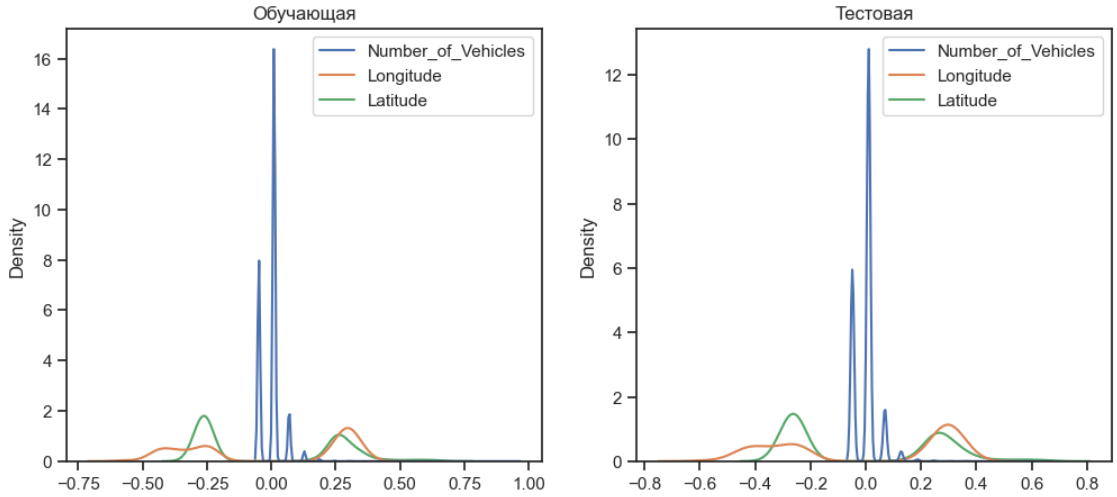
[ ]:
```
       1st_Road_Number  2nd_Road_Number  \
count     4.013500e+04     4.005800e+04
mean      4.558733e-18    -1.529889e-18
std       2.416528e-01     1.340982e-01
min      -1.229122e-01    -3.644368e-02
25%      -1.229122e-01    -3.644368e-02
50%      -1.142916e-01    -3.644368e-02
75%      -4.151033e-02    -3.644368e-02
max       8.770878e-01     9.635563e-01

       Did_Police_Officer_Attend_Scene_of_Accident      Latitude  \
count                                 4.012600e+04  4.012400e+04
mean                                 -3.652231e-17  2.606982e-15
std                                   2.154150e-01  2.873328e-01
min                                  -1.030130e-01 -3.209971e-01
25%                                  -1.030130e-01 -2.623026e-01
50%                                  -1.030130e-01 -2.354915e-01
75%                                  -1.030130e-01  2.672322e-01
max                                   8.969870e-01  6.790029e-01

       Location_Easting_OSGR  Location_Northing_OSGR     Longitude  \
count           4.012400e+04            4.012400e+04  4.012400e+04
mean            6.330850e-18           -4.830040e-17  5.954541e-17
```

```
std              3.275957e-01            2.864881e-01  3.224602e-01
min             -5.881587e-01           -3.203017e-01 -5.957253e-01
25%             -3.206257e-01           -2.616573e-01 -3.139790e-01
50%              2.340187e-01           -2.345298e-01  2.308289e-01
75%              3.039605e-01            2.658540e-01  2.985749e-01
max              4.118413e-01            6.796983e-01  4.042747e-01

       Number_of_Casualties  Number_of_Vehicles  \
count          4.013500e+04        4.013500e+04
mean           3.208817e-18        5.399664e-18
std            3.400279e-02        4.039157e-02
min           -1.385551e-02       -4.850101e-02
25%           -1.385551e-02       -4.850101e-02
50%           -1.385551e-02        1.032251e-02
75%           -1.385551e-02        1.032251e-02
max            9.861445e-01        9.514990e-01

       Pedestrian_Crossing-Human_Control  \
count                       4.012300e+04
mean                       -4.427278e-19
std                         5.225562e-02
min                        -3.314807e-03
25%                        -3.314807e-03
50%                        -3.314807e-03
75%                        -3.314807e-03
max                         9.966852e-01

       Pedestrian_Crossing-Physical_Facilities  Year
count                              4.012700e+04   0.0
mean                              -9.296357e-19   NaN
std                                2.388316e-01   NaN
min                               -1.130255e-01   NaN
25%                               -1.130255e-01   NaN
50%                               -1.130255e-01   NaN
75%                               -1.130255e-01   NaN
max                                8.869745e-01   NaN
```

[ ]:
```
draw_kde(['Number_of_Vehicles', 'Longitude', 'Latitude'], data,␣
↪data_cs21_scaled, '          ', '            ')
```




[ ]:
```
draw_kde(['Number_of_Vehicles', 'Longitude', 'Latitude'],␣
↪data_cs22_scaled_train, data_cs22_scaled_test, '     ', '     ')
```




### 0.3 MinMax-

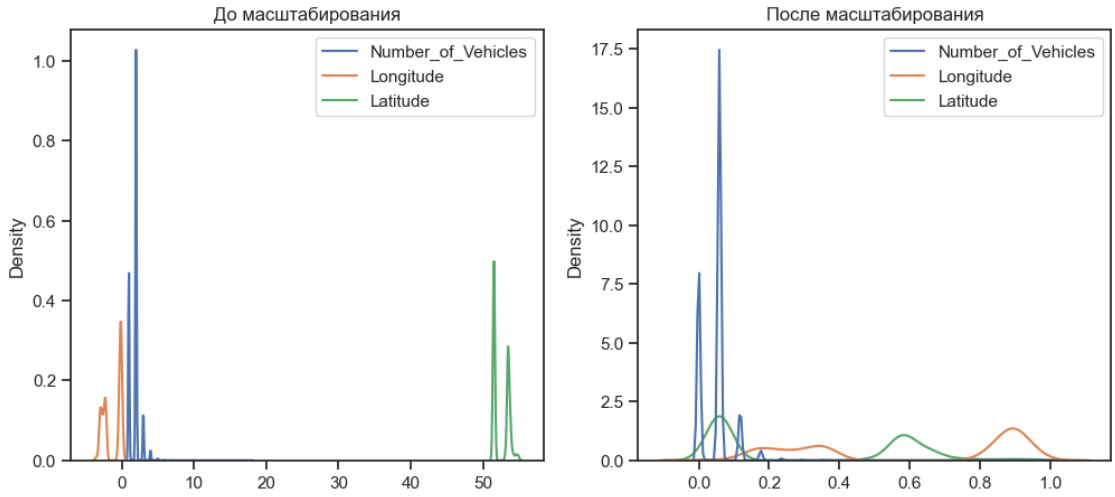
[ ]:
```
#      StandardScaler
cs31 = MinMaxScaler()
data_cs31_scaled_temp = cs31.fit_transform(X_ALL)
#       DataFrame
data_cs31_scaled = arr_to_df(data_cs31_scaled_temp)
data_cs31_scaled.describe()
```

[ ]:
```
       1st_Road_Number  2nd_Road_Number  \
count     50169.000000     50072.000000
mean          0.122377         0.036309
std           0.240832         0.133702
min           0.000000         0.000000
25%           0.000000         0.000000
50%           0.008621         0.000000
75%           0.081119         0.000000
max           1.000000         1.000000

       Did_Police_Officer_Attend_Scene_of_Accident      Latitude  \
count                                50159.000000  50156.000000
mean                                     0.102594      0.319805
std                                      0.214768      0.287132
min                                      0.000000      0.000000
25%                                      0.000000      0.058595
50%                                      0.000000      0.085109
75%                                      0.000000      0.587810
max                                      1.000000      1.000000

       Location_Easting_OSGR  Location_Northing_OSGR     Longitude  \
count           50156.000000            50156.000000  50156.000000
mean                0.590200                0.319114      0.597610
std                 0.326995                0.286285      0.321959
min                 0.000000                0.000000      0.000000
25%                 0.269951                0.058452      0.284023
50%                 0.823223                0.085460      0.827498
75%                 0.892591                0.585748      0.894730
max                 1.000000                1.000000      1.000000

       Number_of_Casualties  Number_of_Vehicles  \
count          50169.000000        50169.000000
mean               0.013743            0.048671
std                0.033806            0.040365
min                0.000000            0.000000
25%                0.000000            0.000000
50%                0.000000            0.058824
75%                0.000000            0.058824
max                1.000000            1.000000

       Pedestrian_Crossing-Human_Control  \
count                       50156.000000
mean                            0.003290
std                             0.051873
min                             0.000000
25%                             0.000000
50%                             0.000000
```

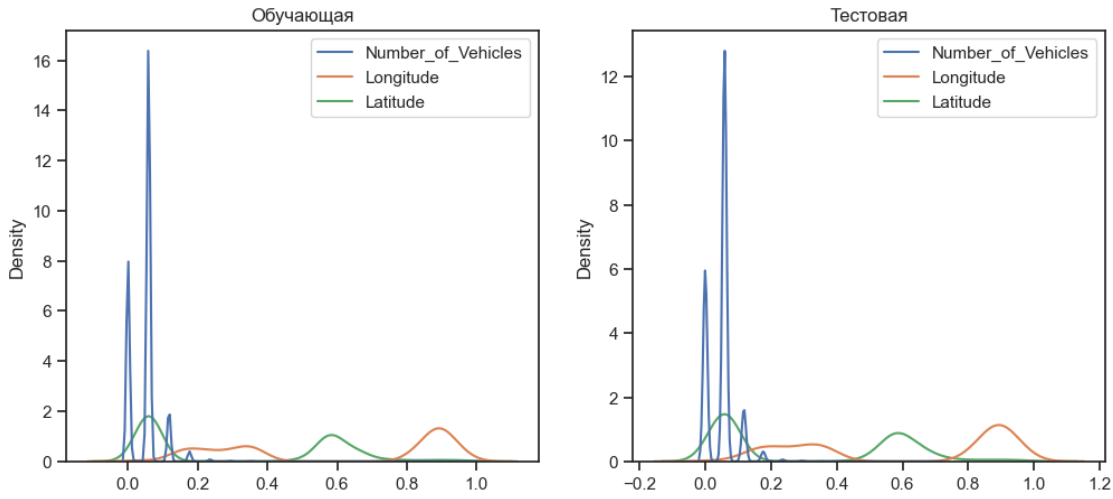
```
75%                                0.000000
max                                1.000000

       Pedestrian_Crossing-Physical_Facilities     Year
count                             50159.000000  50169.0
mean                                  0.113145      0.0
std                                   0.239279      0.0
min                                   0.000000      0.0
25%                                   0.000000      0.0
50%                                   0.000000      0.0
75%                                   0.000000      0.0
max                                   1.000000      0.0
```

[ ]:
```
cs32 = MinMaxScaler()
cs32.fit(X_train)
data_cs32_scaled_train_temp = cs32.transform(X_train)
data_cs32_scaled_test_temp = cs32.transform(X_test)
#       DataFrame
data_cs32_scaled_train = arr_to_df(data_cs32_scaled_train_temp)
data_cs32_scaled_test = arr_to_df(data_cs32_scaled_test_temp)
```

[ ]:
```
draw_kde(['Number_of_Vehicles', 'Longitude', 'Latitude'], data,␣
↪data_cs31_scaled, '           ', '            ')
```




[ ]:
```
draw_kde(['Number_of_Vehicles', 'Longitude', 'Latitude'],␣
↪data_cs32_scaled_train, data_cs32_scaled_test, '     ', '     ')
```




### 0.4

[ ]:
```
data2 = pd.read_csv("datasets/Car_Sales.csv")
```

[ ]:
```
data2.head()
```

[ ]:

| | Manufacturer | Model | Sales_in_thousands | __year_resale_value | Vehicle_type | \ |
|---|---|---|---|---|---|---|
| 0 | Acura | Integra | 16.919 | 16.360 | Passenger | |
| 1 | Acura | TL | 39.384 | 19.875 | Passenger | |
| 2 | Acura | CL | 14.114 | 18.225 | Passenger | |
| 3 | Acura | RL | 8.588 | 29.725 | Passenger | |
| 4 | Audi | A4 | 20.397 | 22.255 | Passenger | |

| | Price_in_thousands | Engine_size | Horsepower | Wheelbase | Width | Length | \ |
|---|---|---|---|---|---|---|---|
| 0 | 21.50 | 1.8 | 140.0 | 101.2 | 67.3 | 172.4 | |
| 1 | 28.40 | 3.2 | 225.0 | 108.1 | 70.3 | 192.9 | |
| 2 | NaN | 3.2 | 225.0 | 106.9 | 70.6 | 192.0 | |
| 3 | 42.00 | 3.5 | 210.0 | 114.6 | 71.4 | 196.6 | |
| 4 | 23.99 | 1.8 | 150.0 | 102.6 | 68.2 | 178.0 | |

| | Curb_weight | Fuel_capacity | Fuel_efficiency | Latest_Launch | \ |
|---|---|---|---|---|---|
| 0 | 2.639 | 13.2 | 28.0 | 2/2/2012 | |
| 1 | 3.517 | 17.2 | 25.0 | 6/3/2011 | |
| 2 | 3.470 | 17.2 | 26.0 | 1/4/2012 | |
| 3 | 3.850 | 18.0 | 22.0 | 3/10/2011 | |
| 4 | 2.998 | 16.4 | 27.0 | 10/8/2011 | |

| | Power_perf_factor |
|---|---|
| 0 | 58.280150 |
| 1 | 91.370778 |

```
2                NaN
3          91.389779
4          62.777639
```

[ ]:
```
data2.describe()
```

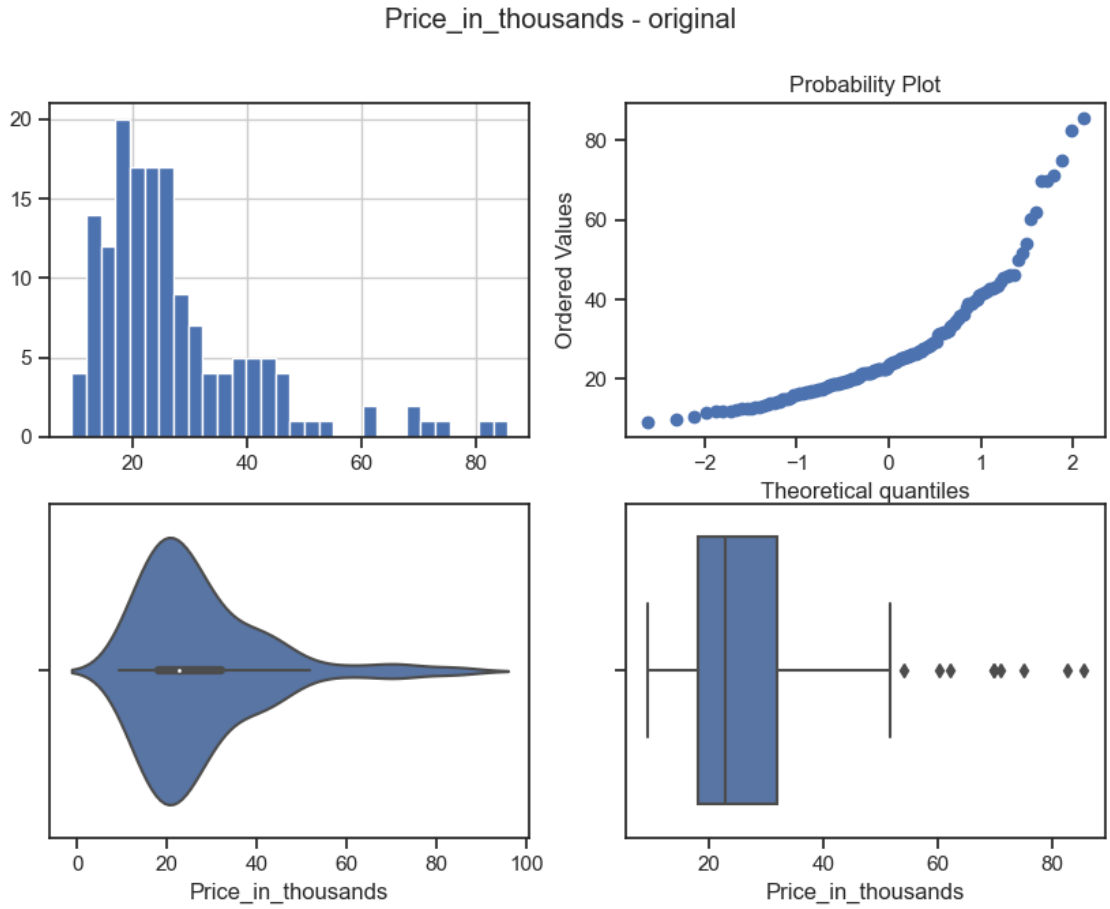
[ ]:
```
       Sales_in_thousands  __year_resale_value  Price_in_thousands  \
count          157.000000           121.000000          155.000000
mean            52.998076            18.072975           27.390755
std             68.029422            11.453384           14.351653
min              0.110000             5.160000            9.235000
25%             14.114000            11.260000           18.017500
50%             29.450000            14.180000           22.799000
75%             67.956000            19.875000           31.947500
max            540.561000            67.550000           85.500000

       Engine_size  Horsepower   Wheelbase       Width      Length  \
count   156.000000  156.000000  156.000000  156.000000  156.000000
mean      3.060897  185.948718  107.487179   71.150000  187.343590
std       1.044653   56.700321    7.641303    3.451872   13.431754
min       1.000000   55.000000   92.600000   62.600000  149.400000
25%       2.300000  149.500000  103.000000   68.400000  177.575000
50%       3.000000  177.500000  107.000000   70.550000  187.900000
75%       3.575000  215.000000  112.200000   73.425000  196.125000
max       8.000000  450.000000  138.700000   79.900000  224.500000

       Curb_weight  Fuel_capacity  Fuel_efficiency  Power_perf_factor
count   155.000000     156.000000       154.000000         155.000000
mean      3.378026      17.951923        23.844156          77.043591
std       0.630502       3.887921         4.282706          25.142664
min       1.895000      10.300000        15.000000          23.276272
25%       2.971000      15.800000        21.000000          60.407707
50%       3.342000      17.200000        24.000000          72.030917
75%       3.799500      19.575000        26.000000          89.414878
max       5.572000      32.000000        45.000000         188.144323
```

[ ]:
```
def diagnostic_plots(df, variable, title):
    fig, ax = plt.subplots(figsize=(10,7))
    #
    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)
    # violinplot
    plt.subplot(2, 2, 3)
    sns.violinplot(x=df[variable])
```

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

[ ]:
```
diagnostic_plots(data2, 'Price_in_thousands', 'Price_in_thousands - original')
```

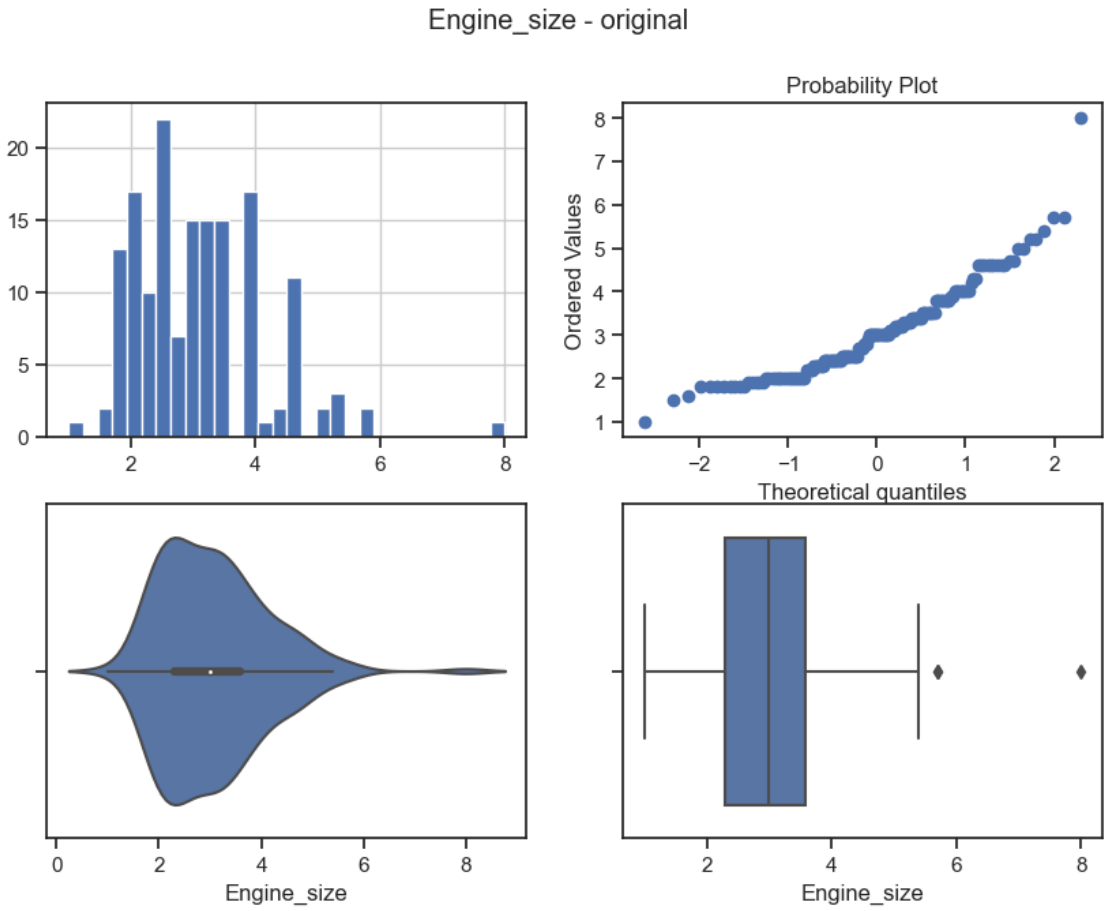
```
/var/folders/fs/5xh23h99763f_blp7m50x23h0000gq/T/ipykernel_68333/4201870494.py:4
: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated
since 3.6 and will be removed two minor releases later; explicitly call
ax.remove() as needed.
  plt.subplot(2, 2, 1)
```




[ ]:
```
diagnostic_plots(data2, 'Engine_size', 'Engine_size - original')
```

```
/var/folders/fs/5xh23h99763f_blp7m50x23h0000gq/T/ipykernel_68333/4201870494.py:4
: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated
since 3.6 and will be removed two minor releases later; explicitly call
```

```
ax.remove() as needed.
  plt.subplot(2, 2, 1)
```



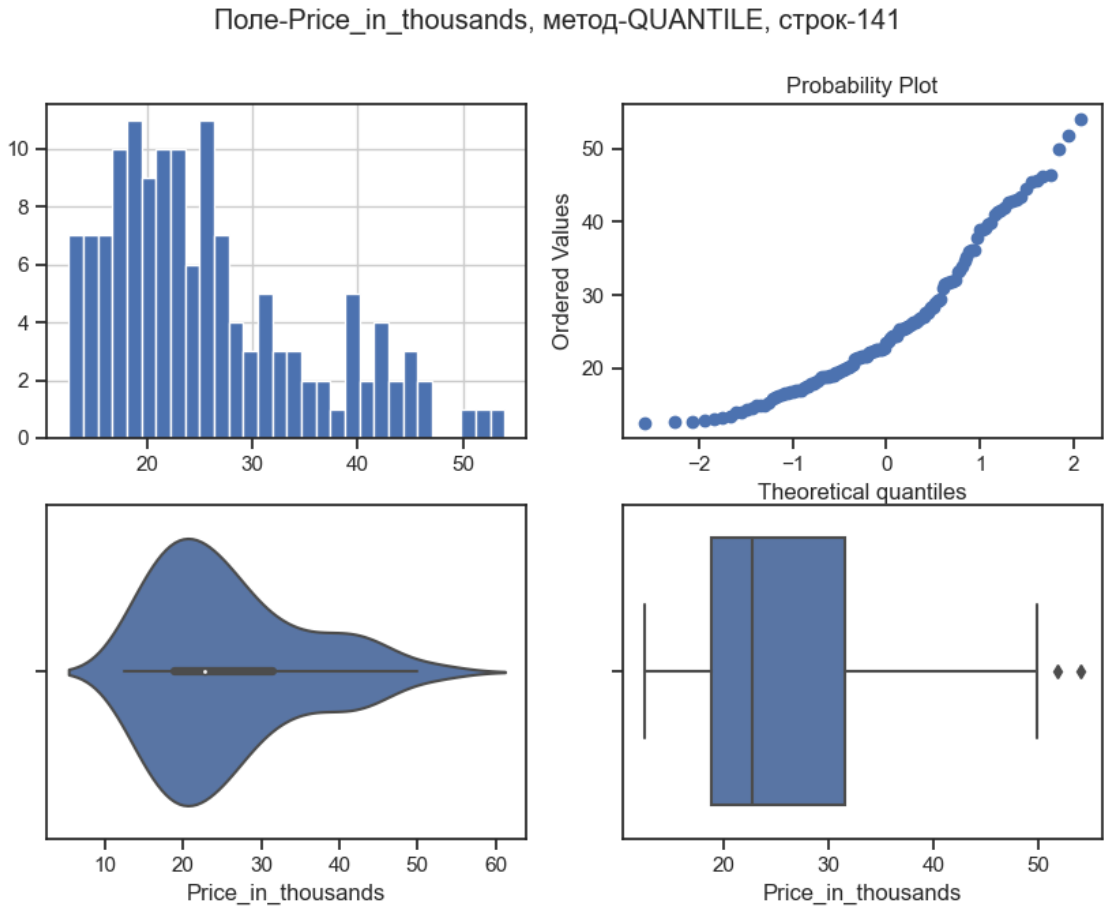

[ ]:
```
#
from enum import Enum
class OutlierBoundaryType(Enum):
    SIGMA = 1
    QUANTILE = 2
    IRQ = 3
```

[ ]:
```
#
def get_outlier_boundaries(df, col):
    lower_boundary = df[col].quantile(0.05)
    upper_boundary = df[col].quantile(0.95)
    return lower_boundary, upper_boundary
```

### 0.5 (number_of_reviews)

[ ]:
```
#
lower_boundary, upper_boundary = get_outlier_boundaries(data2,␣
 ↪"Price_in_thousands")
#
outliers_temp = np.where(data2["Price_in_thousands"] > upper_boundary, True,
                         np.where(data2["Price_in_thousands"] < lower_boundary,␣
 ↪True, False))
#
data_trimmed = data2.loc[~(outliers_temp), ]
title = '  -{},    -{},    -{}'.format("Price_in_thousands", "QUANTILE",␣
 ↪data_trimmed.shape[0])
diagnostic_plots(data_trimmed, "Price_in_thousands", title)
```

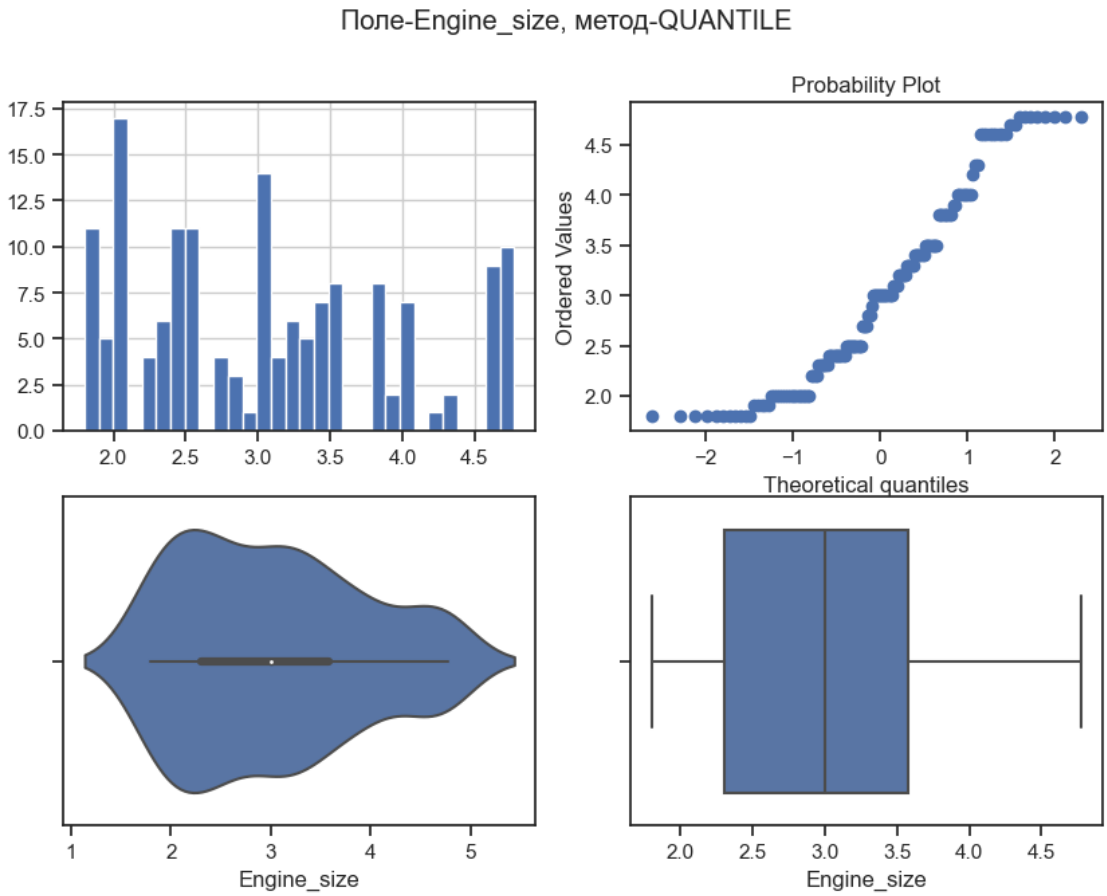
```
/var/folders/fs/5xh23h99763f_blp7m50x23h0000gq/T/ipykernel_68333/4201870494.py:4
: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated
since 3.6 and will be removed two minor releases later; explicitly call
ax.remove() as needed.
  plt.subplot(2, 2, 1)
```



### 0.6

[ ]:
```
#
lower_boundary, upper_boundary = get_outlier_boundaries(data2, "Engine_size")
#
data2["Engine_size"] = np.where(data2["Engine_size"] > upper_boundary,␣
 ↪upper_boundary,
                      np.where(data2["Engine_size"] < lower_boundary,␣
 ↪lower_boundary, data2["Engine_size"]))
title = '  -{},    -{}'.format("Engine_size", "QUANTILE")
diagnostic_plots(data2, "Engine_size", title)
```

```
/var/folders/fs/5xh23h99763f_blp7m50x23h0000gq/T/ipykernel_68333/4201870494.py:4
: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated
since 3.6 and will be removed two minor releases later; explicitly call
ax.remove() as needed.
  plt.subplot(2, 2, 1)
```



### 0.7

[ ]:
```
data2.dtypes
```

[ ]:
```
Manufacturer            object
Model                   object
Sales_in_thousands     float64
__year_resale_value    float64
Vehicle_type            object
Price_in_thousands     float64
Engine_size            float64
Horsepower             float64
Wheelbase              float64
Width                  float64
Length                 float64
Curb_weight            float64
Fuel_capacity          float64
Fuel_efficiency        float64
Latest_Launch           object
Power_perf_factor      float64
dtype: object
```

[ ]:
```
#
data2["Date"] = data2.apply(lambda x: pd.to_datetime(x["Latest_Launch"],␣
 ↪format='%m/%d/%Y'), axis=1)
```

[ ]:
```
data2.head(5)
```

[ ]:

| | Manufacturer | Model | Sales_in_thousands | __year_resale_value | Vehicle_type | \ |
|---|---|---|---|---|---|---|
| 0 | Acura | Integra | 16.919 | 16.360 | Passenger | |
| 1 | Acura | TL | 39.384 | 19.875 | Passenger | |
| 2 | Acura | CL | 14.114 | 18.225 | Passenger | |
| 3 | Acura | RL | 8.588 | 29.725 | Passenger | |
| 4 | Audi | A4 | 20.397 | 22.255 | Passenger | |

| | Price_in_thousands | Engine_size | Horsepower | Wheelbase | Width | Length | \ |
|---|---|---|---|---|---|---|---|
| 0 | 21.50 | 1.8 | 140.0 | 101.2 | 67.3 | 172.4 | |
| 1 | 28.40 | 3.2 | 225.0 | 108.1 | 70.3 | 192.9 | |
| 2 | NaN | 3.2 | 225.0 | 106.9 | 70.6 | 192.0 | |
| 3 | 42.00 | 3.5 | 210.0 | 114.6 | 71.4 | 196.6 | |
| 4 | 23.99 | 1.8 | 150.0 | 102.6 | 68.2 | 178.0 | |

| | Curb_weight | Fuel_capacity | Fuel_efficiency | Latest_Launch | \ |
|---|---|---|---|---|---|
| 0 | 2.639 | 13.2 | 28.0 | 2/2/2012 | |
| 1 | 3.517 | 17.2 | 25.0 | 6/3/2011 | |
| 2 | 3.470 | 17.2 | 26.0 | 1/4/2012 | |
| 3 | 3.850 | 18.0 | 22.0 | 3/10/2011 | |
| 4 | 2.998 | 16.4 | 27.0 | 10/8/2011 | |

```
   Power_perf_factor       Date
0          58.280150 2012-02-02
1          91.370778 2011-06-03
2                NaN 2012-01-04
3          91.389779 2011-03-10
4          62.777639 2011-10-08
```

### 0.8

### 0.9 ( )

[ ]:
```
data3 = pd.read_csv("datasets/Marketing.csv")
```

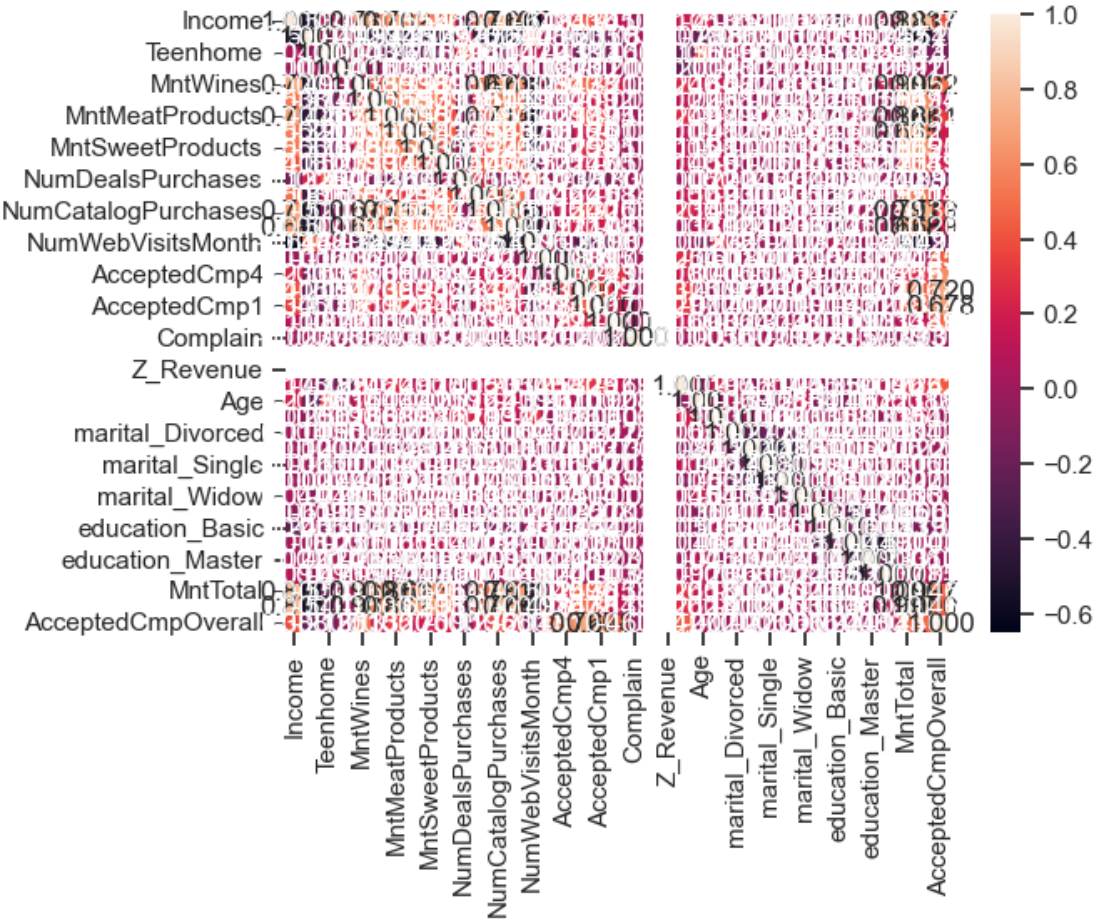
[ ]:
```
data3.head()
```

[ ]:
```
    Income  Kidhome  Teenhome  Recency  MntWines  MntFruits  MntMeatProducts  \
0  58138.0        0         0       58       635         88              546
1  46344.0        1         1       38        11          1                6
2  71613.0        0         0       26       426         49              127
3  26646.0        1         0       26        11          4               20
4  58293.0        1         0       94       173         43              118

   MntFishProducts  MntSweetProducts  MntGoldProds  …  marital_Together  \
0              172                88            88  …                 0
1                2                 1             6  …                 0
2              111                21            42  …                 1
3               10                 3             5  …                 1
4               46                27            15  …                 0

   marital_Widow  education_2n Cycle  education_Basic  education_Graduation  \
0              0                   0                0                     1
1              0                   0                0                     1
2              0                   0                0                     1
3              0                   0                0                     1
4              0                   0                0                     0

   education_Master  education_PhD  MntTotal  MntRegularProds  \
0                 0              0      1529             1441
1                 0              0        21               15
2                 0              0       734              692
3                 0              0        48               43
4                 0              1       407              392

   AcceptedCmpOverall
0                   0
1                   0
```

```
2                    0
3                    0
4                    0

[5 rows x 39 columns]
```

[ ]:
```
sns.heatmap(data3.corr(), annot=True, fmt='.3f')
```

[ ]: <Axes: >




[ ]:
```
#         DataFrame
def make_corr_df(df):
    cr = data3.corr()
    cr = cr.abs().unstack()
    cr = cr.sort_values(ascending=False)
    cr = cr[cr >= 0.3]
    cr = cr[cr < 1]
    cr = pd.DataFrame(cr).reset_index()
```

```
    cr.columns = ['f1', 'f2', 'corr']
    return cr
```

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

    for feature in cr['f1'].unique():
        if feature not in grouped_feature_list:
            #
            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
```

[ ]:
```
#
corr_groups(make_corr_df(data3))
```

[ ]:
```
[['MntTotal',
  'MntWines',
  'MntMeatProducts',
  'Income',
  'NumCatalogPurchases',
  'NumStorePurchases',
  'MntFishProducts',
  'MntSweetProducts',
  'MntFruits',
  'Kidhome',
  'NumWebPurchases',
  'NumWebVisitsMonth',
  'AcceptedCmp5',
  'AcceptedCmpOverall',
  'MntGoldProds',
  'AcceptedCmp1',
  'MntRegularProds'],
 ['AcceptedCmpOverall', 'MntWines', 'AcceptedCmp5', 'AcceptedCmp4'],
 ['education_Graduation', 'education_PhD'],
 ['marital_Married', 'marital_Single', 'marital_Together'],
 ['AcceptedCmpOverall', 'AcceptedCmp2'],
 ['education_Graduation', 'education_Master'],
 ['AcceptedCmpOverall', 'AcceptedCmp3'],
 ['AcceptedCmpOverall', 'AcceptedCmp5', 'Response'],
 ['Teenhome', 'NumWebVisitsMonth', 'NumDealsPurchases'],
 ['Teenhome', 'Age'],
 ['education_Graduation', 'education_2n Cycle']]
```

### 0.10

[ ]:
```
X3_ALL = data3.drop(['Recency'], axis=1)
```

[ ]:
```
#
X3_train, X3_test, y3_train, y3_test = train_test_split(X3_ALL,␣
 ↪data3['Recency'],
                                                        test_size=0.2,
                                                        random_state=1)
```

[ ]:
```
#       L1-
e_lr1 = LogisticRegression(C=1000, solver='liblinear', penalty='l1',␣
 ↪max_iter=500, random_state=1)
e_lr1.fit(X3_train, y3_train)
#
e_lr1.coef_
```

[ ]:
```
array([[ 2.36611132e-05, -4.37617142e-01, -1.06800247e-02, …,
        -1.63170820e-03, -8.91019230e-05, -1.06604264e+00],
       [-4.08684338e-06,  2.11252986e-01,  3.26968614e-01, …,
         2.06092631e-04, -5.67253475e-04, -3.79701869e-02],
       [ 2.97309082e-05,  2.78608379e-01,  7.84637108e-01, …,
         7.60268573e-04,  9.22122307e-04, -2.72190745e-01],
       …,
       [-3.33796349e-06,  5.87316491e-01,  8.31406329e-01, …,
         1.86837335e-04,  9.46010783e-05,  2.65448813e-02],
       [-3.64459359e-05, -1.11298763e+00,  2.33999344e-01, …,
        -2.42398207e-04,  1.69350949e-04, -6.53642523e-02],
       [ 4.67890738e-05, -1.67844004e+00,  7.82124718e-01, …,
        -4.24294712e-05, -6.97529794e-04,  1.87736876e-02]])
```

[ ]:
```
#              "     "
from sklearn.feature_selection import SelectFromModel
sel_e_lr1 = SelectFromModel(e_lr1)
sel_e_lr1.fit(X3_train, y3_train)
sel_e_lr1.get_support()
```

[ ]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True])
```

[ ]:
```
e_lr2 = LinearSVC(C=0.01, penalty="l1", max_iter=2000, dual=False)
e_lr2.fit(X3_train, y3_train)
#
e_lr2.coef_
```

```
/Users/seralekhin/BMSTU_Labs/.env/lib/python3.11/site-
```

```
packages/sklearn/svm/_base.py:1244: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
```

[ ]:
```
array([[-8.75481522e-07,  0.00000000e+00,  0.00000000e+00, …,
        -1.22148808e-05,  0.00000000e+00,  0.00000000e+00],
       [-1.70902922e-06,  0.00000000e+00,  0.00000000e+00, …,
        -3.98938091e-05,  4.26048994e-05,  0.00000000e+00],
       [-3.54264966e-06,  0.00000000e+00,  0.00000000e+00, …,
        -1.39468632e-04,  9.96158201e-05,  0.00000000e+00],
       …,
       [-2.96785441e-06,  0.00000000e+00,  0.00000000e+00, …,
         3.98224071e-05,  2.94380051e-04,  0.00000000e+00],
       [-3.02787186e-06,  0.00000000e+00,  0.00000000e+00, …,
         1.29322152e-04,  0.00000000e+00,  0.00000000e+00],
       [ 4.46651443e-07,  0.00000000e+00,  0.00000000e+00, …,
         0.00000000e+00, -1.38467919e-04,  0.00000000e+00]])
```

[ ]:
```
#           False  . .
sel_e_lr2 = SelectFromModel(e_lr2)
sel_e_lr2.fit(X3_train, y3_train)
sel_e_lr2.get_support()
```

```
/Users/seralekhin/BMSTU_Labs/.env/lib/python3.11/site-
packages/sklearn/svm/_base.py:1244: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
```

[ ]:
```
array([ True, False, False,  True,  True,  True,  True,  True,  True,
        True,  True, False,  True, False, False, False, False, False,
       False, False, False,  True, False,  True,  True, False, False,
       False, False, False, False, False, False, False, False,  True,
        True, False])
```