# Задание

* 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.
* Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:
  + устранение пропусков в данных;
  + кодирование категориальных признаков;
  + нормализация числовых признаков.

# Подключение библиотек

%pip install pandas  
%pip install numpy  
%pip install seaborn

import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns

# Ход работы

## Выбор и описание датасета

Выбор пал на датасет с ценами машин

# Импорт датасета  
df = pd.read\_csv('car\_prices.csv')  
  
# Вывод первых 5 строк  
df.head(5)

year make model trim body transmission \  
0 2015 Kia Sorento LX SUV automatic   
1 2015 Kia Sorento LX SUV automatic   
2 2014 BMW 3 Series 328i SULEV Sedan automatic   
3 2015 Volvo S60 T5 Sedan automatic   
4 2014 BMW 6 Series Gran Coupe 650i Sedan automatic   
  
 vin state condition odometer color interior \  
0 5xyktca69fg566472 ca 5.0 16639.0 white black   
1 5xyktca69fg561319 ca 5.0 9393.0 white beige   
2 wba3c1c51ek116351 ca 45.0 1331.0 gray black   
3 yv1612tb4f1310987 ca 41.0 14282.0 white black   
4 wba6b2c57ed129731 ca 43.0 2641.0 gray black   
  
 seller mmr sellingprice \  
0 kia motors america inc 20500.0 21500.0   
1 kia motors america inc 20800.0 21500.0   
2 financial services remarketing (lease) 31900.0 30000.0   
3 volvo na rep/world omni 27500.0 27750.0   
4 financial services remarketing (lease) 66000.0 67000.0   
  
 saledate   
0 Tue Dec 16 2014 12:30:00 GMT-0800 (PST)   
1 Tue Dec 16 2014 12:30:00 GMT-0800 (PST)   
2 Thu Jan 15 2015 04:30:00 GMT-0800 (PST)   
3 Thu Jan 29 2015 04:30:00 GMT-0800 (PST)   
4 Thu Dec 18 2014 12:30:00 GMT-0800 (PST)

Используем только несколько из предложенных признаков:

cols\_filter = ['year', 'make', 'model', 'body',  
 'vin', 'odometer']  
data = df[cols\_filter]  
data.head()

year make model body vin odometer  
0 2015 Kia Sorento SUV 5xyktca69fg566472 16639.0  
1 2015 Kia Sorento SUV 5xyktca69fg561319 9393.0  
2 2014 BMW 3 Series Sedan wba3c1c51ek116351 1331.0  
3 2015 Volvo S60 Sedan yv1612tb4f1310987 14282.0  
4 2014 BMW 6 Series Gran Coupe Sedan wba6b2c57ed129731 2641.0

hdata = data  
list(zip(hdata.columns, [i for i in data.dtypes]))

[('year', dtype('int64')),  
 ('make', dtype('O')),  
 ('model', dtype('O')),  
 ('body', dtype('O')),  
 ('vin', dtype('O')),  
 ('odometer', dtype('float64'))]

Выведем колонки в которых присутствуют пропуски:

hcols\_with\_na = [c for c in hdata.columns if hdata[c].isnull().sum() > 0]  
hcols\_with\_na

['make', 'model', 'body', 'odometer']

hdata.shape

(418960, 6)

Выведем количество пропусках в тех колонках, которые вывелись в предыдущем блоке:

[(c, hdata[c].isnull().sum()) for c in hcols\_with\_na]

[('make', 7473), ('model', 7544), ('body', 10346), ('odometer', 88)]

Выведем ту же самую информацию, но в процентном виде :

[(c, hdata[c].isnull().mean()) for c in hcols\_with\_na]

[('make', 0.017837025014321176),  
 ('model', 0.018006492266564828),  
 ('body', 0.024694481573419896),  
 ('odometer', 0.0002100439182738209)]

Далее создадим переменную, в которой будут храниться колонки с пропусками:

hcols\_with\_na\_temp = ['make', 'model', 'body', 'odometer']

Удалим пропуски:

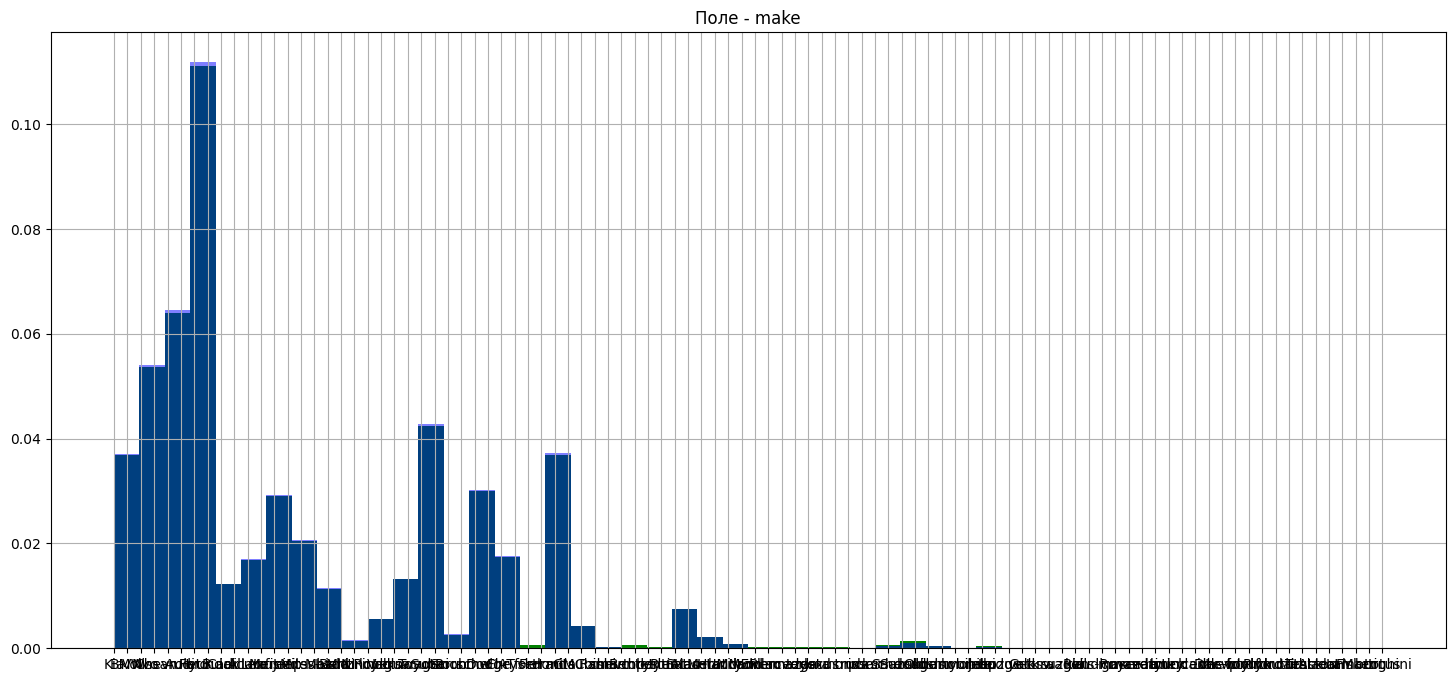
hdata\_drop = hdata[hcols\_with\_na\_temp].dropna()  
hdata\_drop.shape

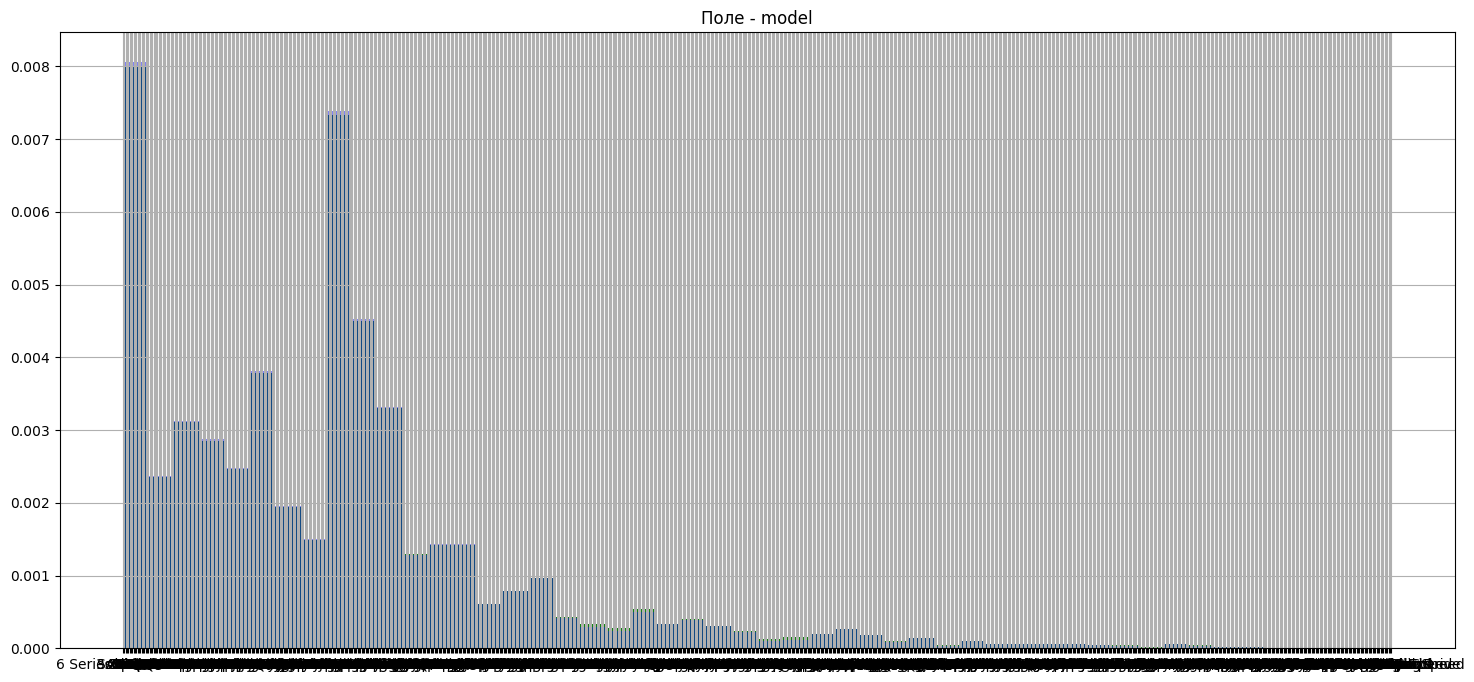
(408463, 4)

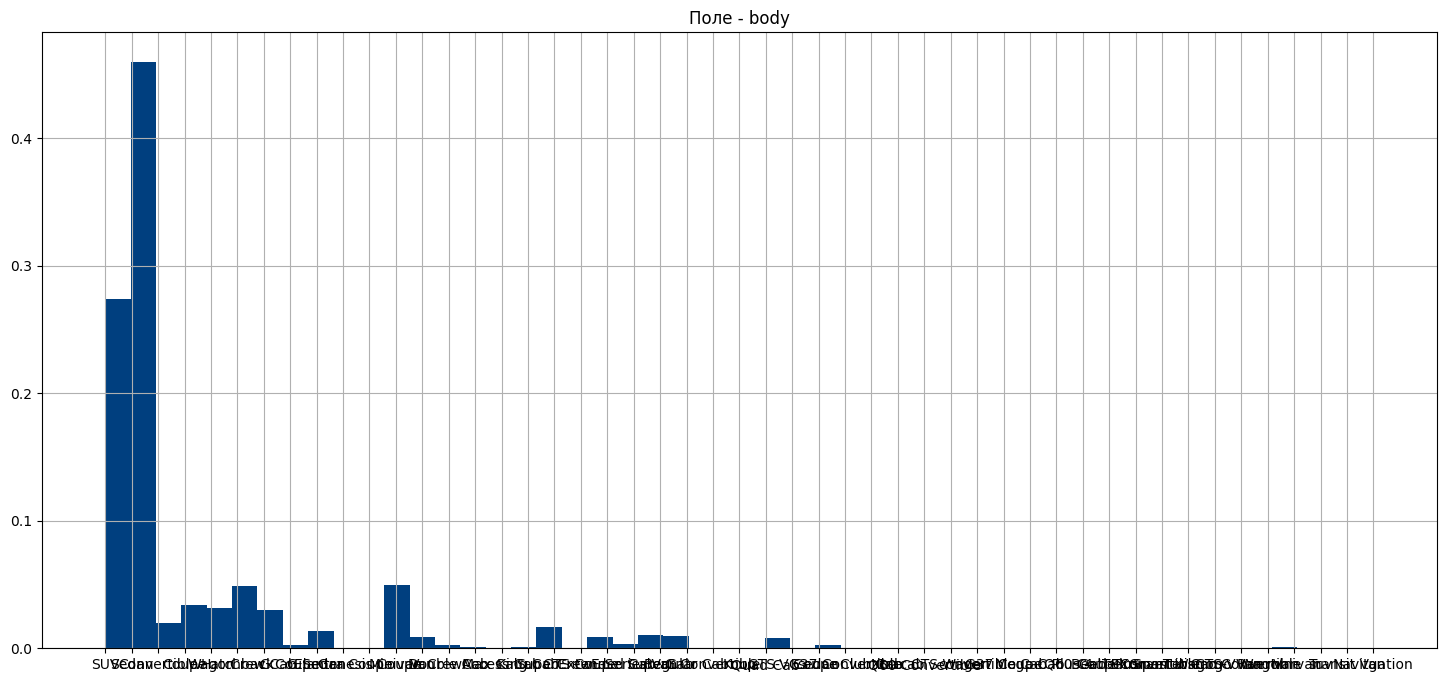
def plot\_hist\_diff(old\_ds, new\_ds, cols):  
 """  
 Разница между распределениями до и после устранения пропусков  
 """  
 for c in cols:   
 fig = plt.figure(figsize=(18, 8))  
 ax = fig.add\_subplot(111)  
 ax.title.set\_text('Поле - ' + str(c))  
 old\_ds[c].hist(bins=50, ax=ax, density=True, color='green')  
 new\_ds[c].hist(bins=50, ax=ax, color='blue', density=True, alpha=0.5)  
 plt.show()

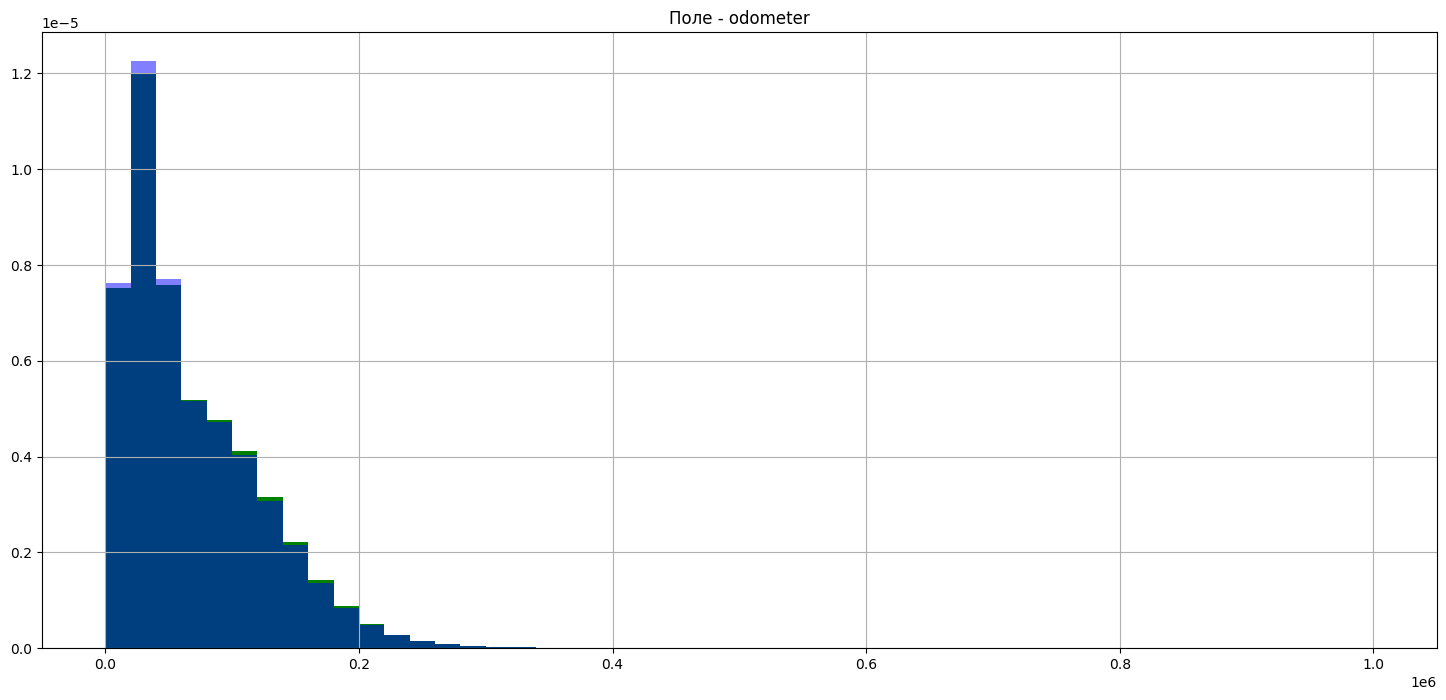
С помощью столбчатой диаграммы покажем разницу между распределениями до и после устранения признаков:

plot\_hist\_diff(hdata, hdata\_drop, hcols\_with\_na\_temp)









Лучше всего видно изменения на диаграмме поля odometer

## Кодирование категориальных признаков

* Проведём кодирование категорилального признака preferred\_device с использованием метода "one-hot encoding".
* One-hot encoding предполагает, что значение категории заменяется на отдельную колонку, которая содержит бинарные значения.
* Преимущества:
  + Простота реализации.
  + Подходит для любых моделей, так как НЕ создает фиктивное отношение порядка между значениями.
* Недостатки:
  + Расширяется признаковое пространство.

pd.get\_dummies(data[['model']]).head()

model\_1 model\_1 Series model\_1500 model\_190-Class model\_2 Series \  
0 0 0 0 0 0   
1 0 0 0 0 0   
2 0 0 0 0 0   
3 0 0 0 0 0   
4 0 0 0 0 0   
  
 model\_200 model\_200SX model\_2500 model\_3 model\_3 Series ... \  
0 0 0 0 0 0 ...   
1 0 0 0 0 0 ...   
2 0 0 0 0 1 ...   
3 0 0 0 0 0 ...   
4 0 0 0 0 0 ...   
  
 model\_wave model\_windstar model\_wrangler model\_x-trail model\_x3 \  
0 0 0 0 0 0   
1 0 0 0 0 0   
2 0 0 0 0 0   
3 0 0 0 0 0   
4 0 0 0 0 0   
  
 model\_xA model\_xB model\_xD model\_yaris model\_yukon   
0 0 0 0 0 0   
1 0 0 0 0 0   
2 0 0 0 0 0   
3 0 0 0 0 0   
4 0 0 0 0 0   
  
[5 rows x 957 columns]

# Добавление отдельной колонки, признака пустых значений  
pd.get\_dummies(data[['model']], dummy\_na=True).head()

model\_1 model\_1 Series model\_1500 model\_190-Class model\_2 Series \  
0 0 0 0 0 0   
1 0 0 0 0 0   
2 0 0 0 0 0   
3 0 0 0 0 0   
4 0 0 0 0 0   
  
 model\_200 model\_200SX model\_2500 model\_3 model\_3 Series ... \  
0 0 0 0 0 0 ...   
1 0 0 0 0 0 ...   
2 0 0 0 0 1 ...   
3 0 0 0 0 0 ...   
4 0 0 0 0 0 ...   
  
 model\_windstar model\_wrangler model\_x-trail model\_x3 model\_xA \  
0 0 0 0 0 0   
1 0 0 0 0 0   
2 0 0 0 0 0   
3 0 0 0 0 0   
4 0 0 0 0 0   
  
 model\_xB model\_xD model\_yaris model\_yukon model\_nan   
0 0 0 0 0 0   
1 0 0 0 0 0   
2 0 0 0 0 0   
3 0 0 0 0 0   
4 0 0 0 0 0   
  
[5 rows x 958 columns]

## Нормализация числовых признаков

* Проведём нормализацию числовых признаков через использование межквартильного размаха
* Межквартильный размах IQR (interquartile range, IQR) - это разность третьего квартиля и первого квартиля:

def remove\_outliers\_iqr(data, column):  
 Q1 = data[column].quantile(0.25)  
 Q3 = data[column].quantile(0.75)  
 IQR = Q3 - Q1  
 lower\_bound = Q1 - 1.5 \* IQR  
 upper\_bound = Q3 + 1.5 \* IQR  
 filtered\_data = data[(data[column] >= lower\_bound) & (data[column] <= upper\_bound)]  
 return filtered\_data

data.shape

(418960, 6)

filtered\_dataset = remove\_outliers\_iqr(data, 'odometer')  
filtered\_dataset.shape

(411973, 6)