Исследование параметров БПЛА

In [1]:
```
import re
import pandas as pd
import numpy as np
import missingno as msno
import seaborn as sns
import matplotlib.pyplot as plt

df=pd.read_csv('02_Database_.csv')
df
```

Out[1]:

	Name	MTOW [kg]	Payload [kg]	Wingspan [m]	Length [m]	Cruise Speed [m/s]	Max speed [m/s]	Stall Speed [m/s]	Range [km]	Endurance [h]	...
0	AAI (Textron) RQ-7A Shadow	150	NaN	3.90	3.40	58	NaN	NaN	735.0	5	...
1	AAI Aerosonde	25	3.1	2.95	1.70	31	41.0	NaN	3000.0	10	...
2	Aeromapper Avem	2	0.7	2.14	NaN	16.7	NaN	NaN	20.0	3	...
3	Aeronautics Defense Aerolight	40	8.0	4.00	2.56	25.6	51.4	NaN	150.0	4	...
4	Aeronautics Defense Orbiter 3	30	5.5	4.40	NaN	NaN	36.0	NaN	150.0	7	...
...	...	...	...	...	...	...	...	...	...	...	...
88	V-200 Skeldar	235	NaN	NaN	4.00	NaN	41.7	NaN	200.0	5+	...
89	WB Group FLYEYE mini UAV	11	4.0	3.60	1.90	NaN	33.3	NaN	30.0	2.5	...
90	WB Group FT-5 LOS Tactical UAV	85	30.0	6.40	3.10	NaN	50.0	21.1	180.0	12	...
91	XY Aviations FT-25D	90	10.0	2.50	3.00	97.2	125.0	NaN	NaN	0.45	...
92	XY Aviations H-150	105	20.0	4.90	2.92	36.1	47.2	NaN	NaN	4	...

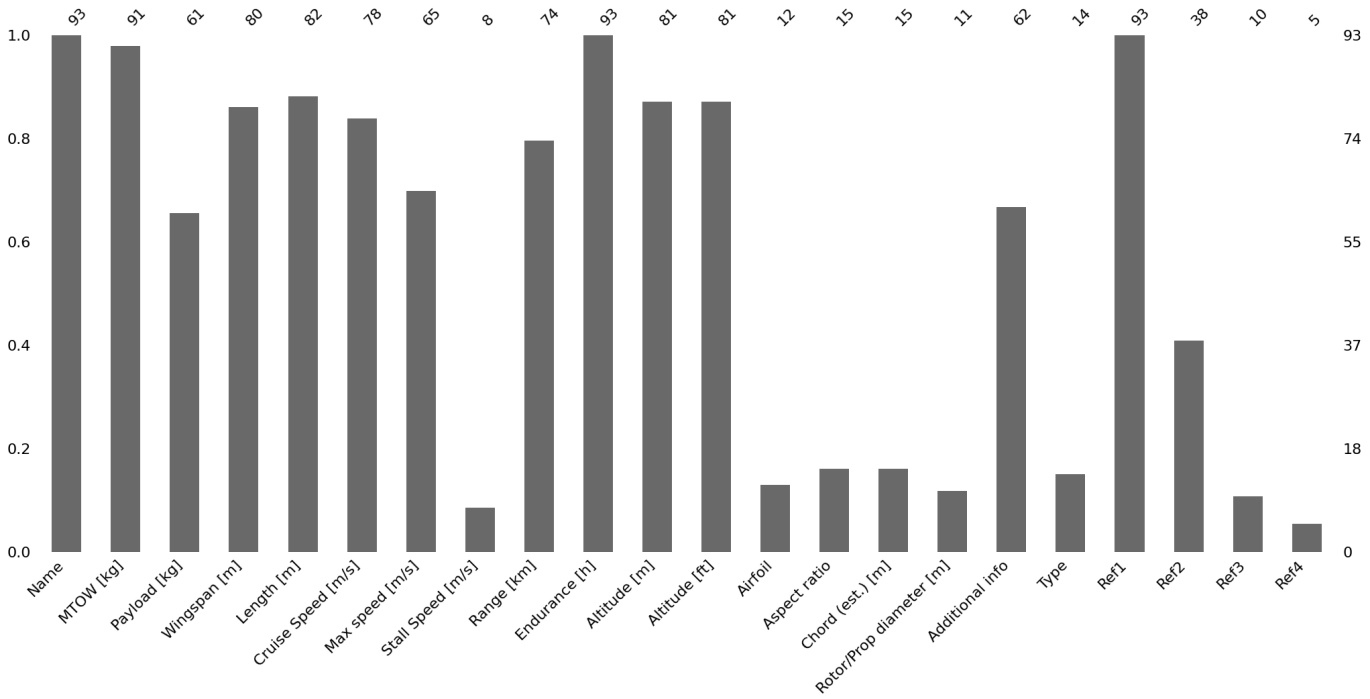
93 rows × 22 columns

In [2]:
```
df.isna().sum() # количество пропущенных значений
```

Out[2]:
```
Name                       0
MTOW [kg]                  2
Payload [kg]              32
Wingspan [m]              13
Length [m]                11
Cruise Speed [m/s]        15
Max speed [m/s]           28
Stall Speed [m/s]         85
Range [km]                19
Endurance [h]              0
Altitude [m]              12
Altitude [ft]             12
Airfoil                   81
Aspect ratio              78
Chord (est.) [m]          78
Rotor/Prop diameter [m]   82
Additional info           31
Type                      79
Ref1                       0
Ref2                      55
Ref3                      83
Ref4                      88
dtype: int64
```

In [3]:
```
msno.bar(df)
```

Out[3]: <Axes: >



In [4]:
```
#Тепловая карта
colours = ['#08e8de', '#FF0000']
sns.heatmap(df.isnull(), cmap=sns.color_palette(colours))
# Decorations
plt.title('Матрица пропущенных значений набора данных', fontsize=14)
plt.tick_params(axis='x', labelrotation=90)
plt.xticks(fontsize=7)
plt.yticks(fontsize=7)
plt.figtext(0.1, -0.2, "  Рисунок2.  Матрица пропущенных значений набора данных",font
plt.show()
```

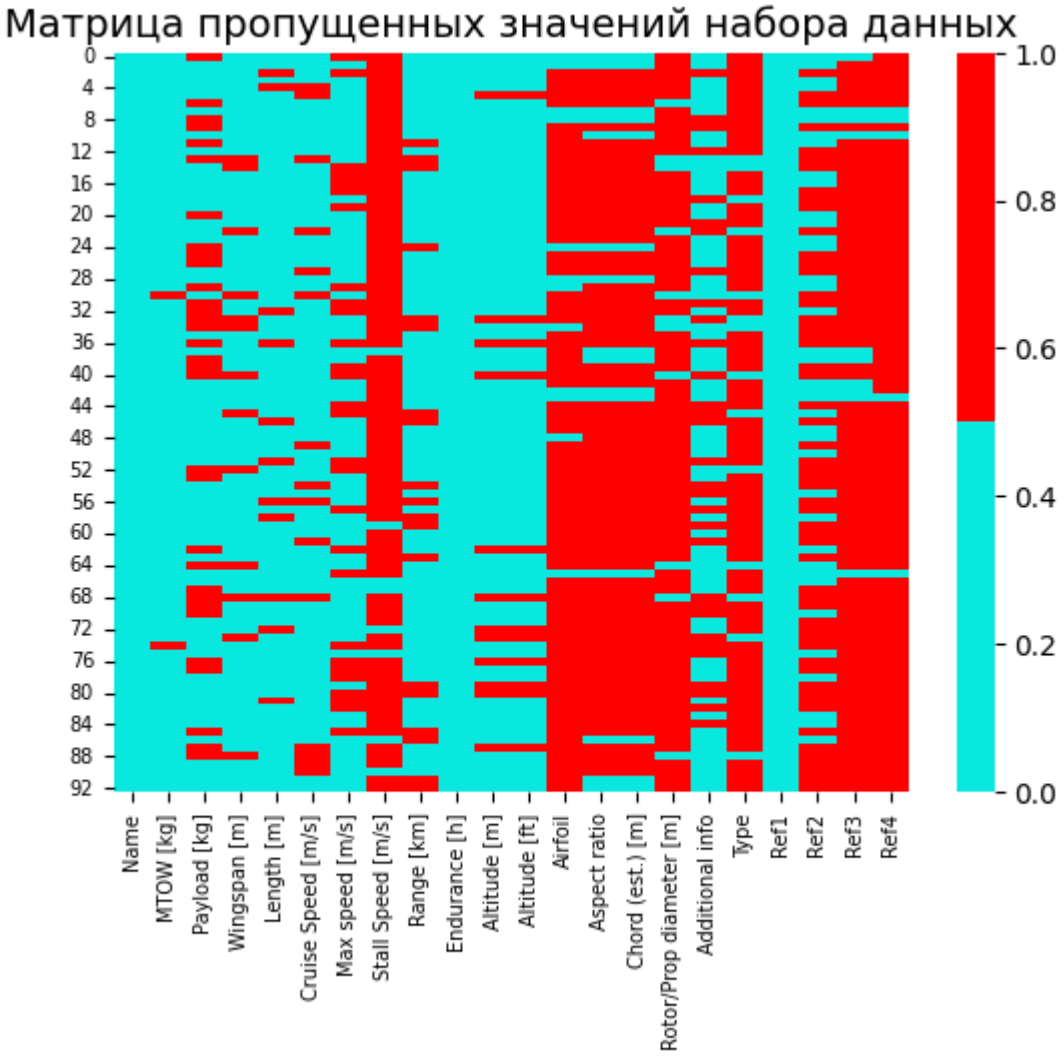


Рисунок2. Матрица пропущенных значений набора данных

In [5]:
```
df.columns
```

Out[5]:
```
Index(['Name', 'MTOW [kg]', 'Payload [kg]', 'Wingspan [m]', 'Length [m]',
       'Cruise Speed [m/s]', 'Max speed [m/s]', 'Stall Speed [m/s]',
       'Range [km]', 'Endurance [h]', 'Altitude [m]', 'Altitude [ft]',
       'Airfoil', 'Aspect ratio', 'Chord (est.) [m]',
       'Rotor/Prop diameter [m]', 'Additional info', 'Type', 'Ref1', 'Ref2',
       'Ref3', 'Ref4'],
      dtype='object')
```

In [6]:
```
df.drop(columns=['Stall Speed [m/s]','Airfoil','Aspect ratio','Chord (est.) [m]','Rot
       'Ref3', 'Ref4'],inplace=True)
```

In [7]:
```
sns.heatmap(df.isnull(), cmap=sns.color_palette(colours))
# Decorations
plt.title('Матрица пропущенных значений набора данных', fontsize=14)
plt.tick_params(axis='x', labelrotation=90)
plt.xticks(fontsize=7)
plt.yticks(fontsize=7)
plt.figtext(0.1, -0.2, "  Рисунок2.  Матрица пропущенных значений набора данных",font
plt.show()
```

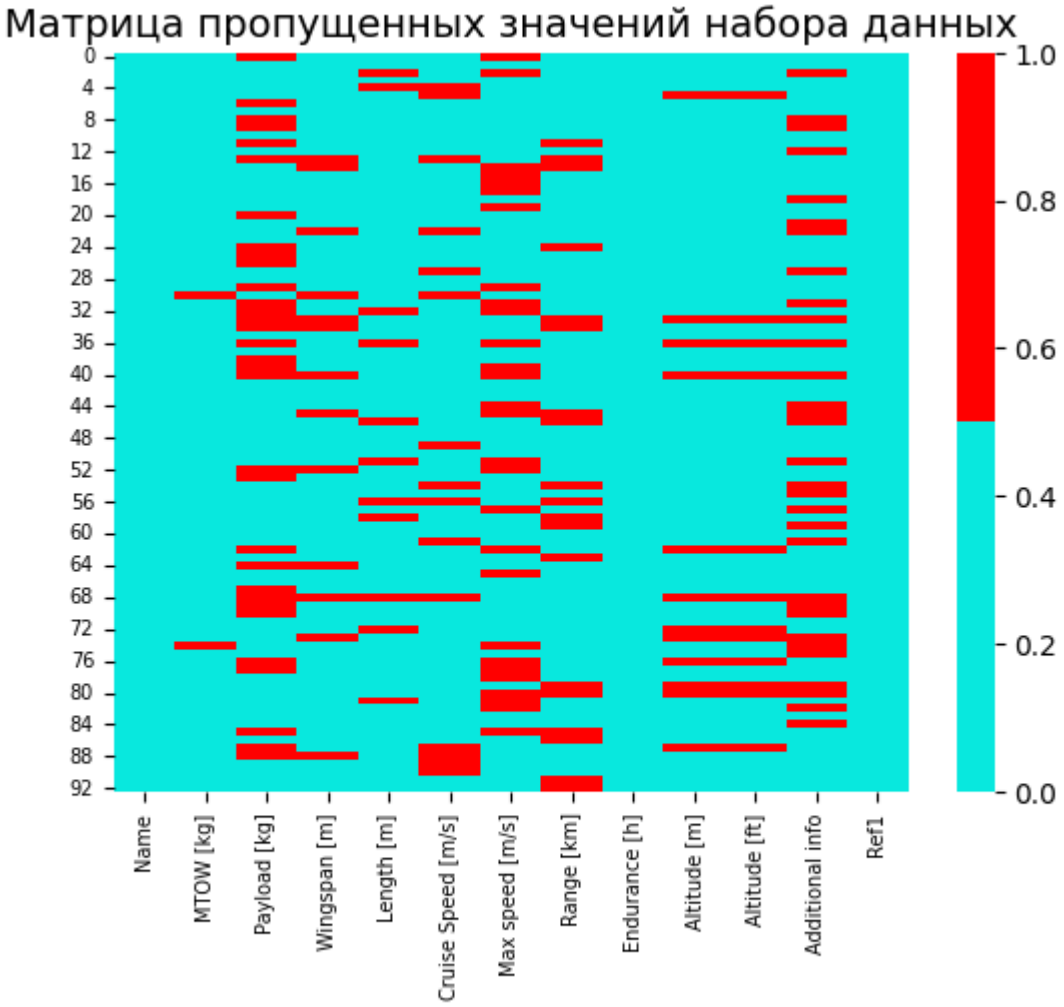


Рисунок2. Матрица пропущенных значений набора данных

In [8]:
```
# Процентное и числовое значение пропущенных данных
MissingValue = df.isnull().sum()
Percent = (df.isnull().sum()/df.isnull().count()*100)
MissingData = pd.concat([MissingValue, Percent], axis=1, keys=['Пропущенные значения'
MissingData
```

Out[8]:

	Пропущенные значения	Процент
Name	0	0.000000
MTOW [kg]	2	2.150538
Payload [kg]	32	34.408602
Wingspan [m]	13	13.978495
Length [m]	11	11.827957
Cruise Speed [m/s]	15	16.129032
Max speed [m/s]	28	30.107527
Range [km]	19	20.430108
Endurance [h]	0	0.000000
Altitude [m]	12	12.903226
Altitude [ft]	12	12.903226
Additional info	31	33.333333
Ref1	0	0.000000

In [9]:
```
df.dropna(axis='index',subset=['MTOW [kg]'],inplace=True)#удаляем строки без массыMTO
```

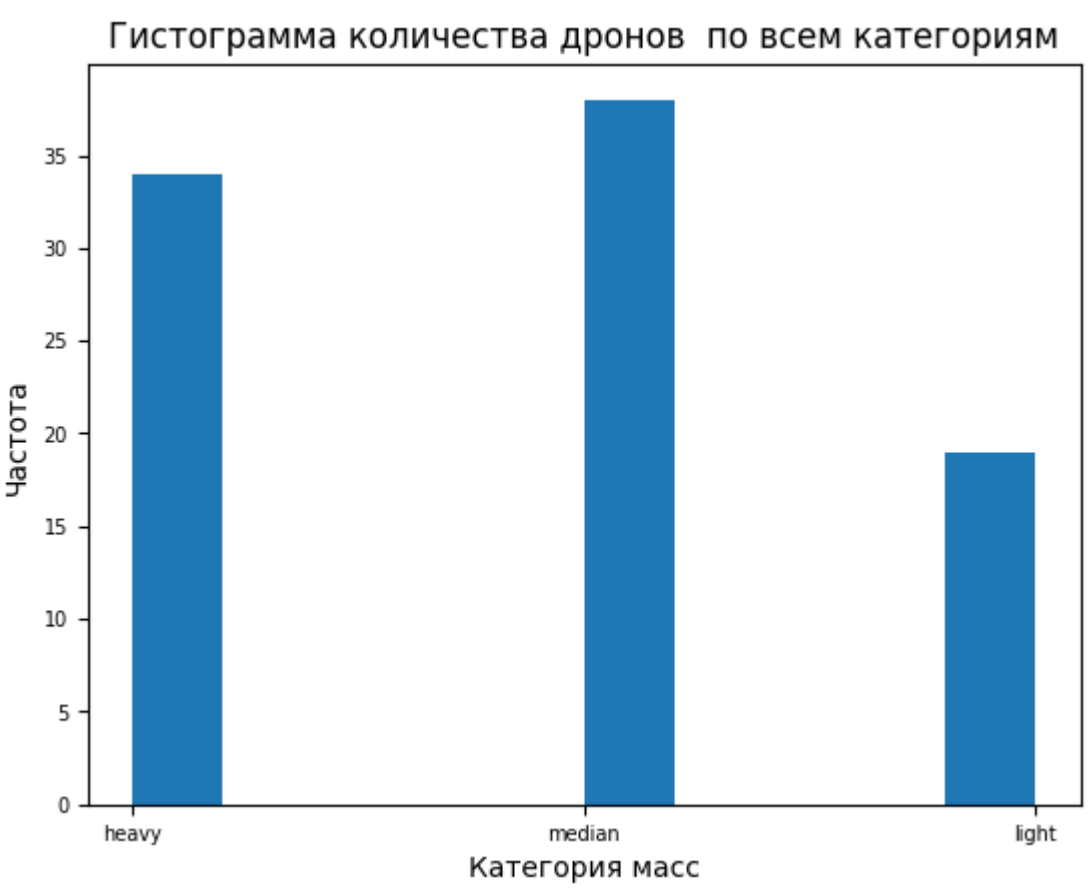
```
#заполняем 0
df['Payload [kg]'].fillna(0,inplace=True)
df['Wingspan [m]'].fillna(0,inplace=True)
df['Length [m]'].fillna(0,inplace=True)
df['Max speed [m/s]'].fillna(0,inplace=True)
#заполняем "0"
df['Cruise Speed [m/s]'].fillna("0",inplace=True)
df['Range [km]'].fillna("0",inplace=True)
df['Altitude [m]'].fillna("0",inplace=True)
df['Altitude [ft]'].fillna("0",inplace=True)
df['Additional info'].fillna("unknown",inplace=True)
#df['Cruise Speed [m/s]']= df['Cruise Speed [m/s]'].astype(float)
df.drop_duplicates(inplace=True)

#Определяем структуру данных
df['MTOW [kg]'] = df['MTOW [kg]'].str.replace(',', '.').astype(float)
# print(df['MTOW [kg]'][0:10])
# print(df['MTOW [kg]'][10:20])
# print(df['MTOW [kg]'][20:30])
# print(df['MTOW [kg]'][30:40])
# print(df['MTOW [kg]'][40:50])
# print(df['MTOW [kg]'][50:60])
# print(df['MTOW [kg]'][60:70])
# print(df['MTOW [kg]'][70:80])
# print(df['MTOW [kg]'][80:90])
# print(df['MTOW [kg]'][90:92])
# df['MTOW [kg]']
# df.dtypes #тип данных
```

In [10]:
```
df['MTOW [kg]'].describe()
```

Out[10]:
```
count       91.000000
mean       456.464495
std       1728.812690
min          0.016000
25%          5.000000
50%         24.500000
75%        154.500000
max      14628.000000
Name: MTOW [kg], dtype: float64
```

In [11]:
```
def Mass_category_(row):
    Mass = row['MTOW [kg]']
    if 0 <= Mass < 4:
        return 'light'
    elif 4 <= Mass <= 40.5:
        return 'median'
    elif  Mass > 40.5:
        return 'heavy'

#категоризуем массы
df['Mass_category_'] = df.apply(Mass_category_, axis = 1)
Mass_num={'light':0,'median':1,'heavy':2}
df['Mass_num']=df['Mass_category_'].map(Mass_num)
```

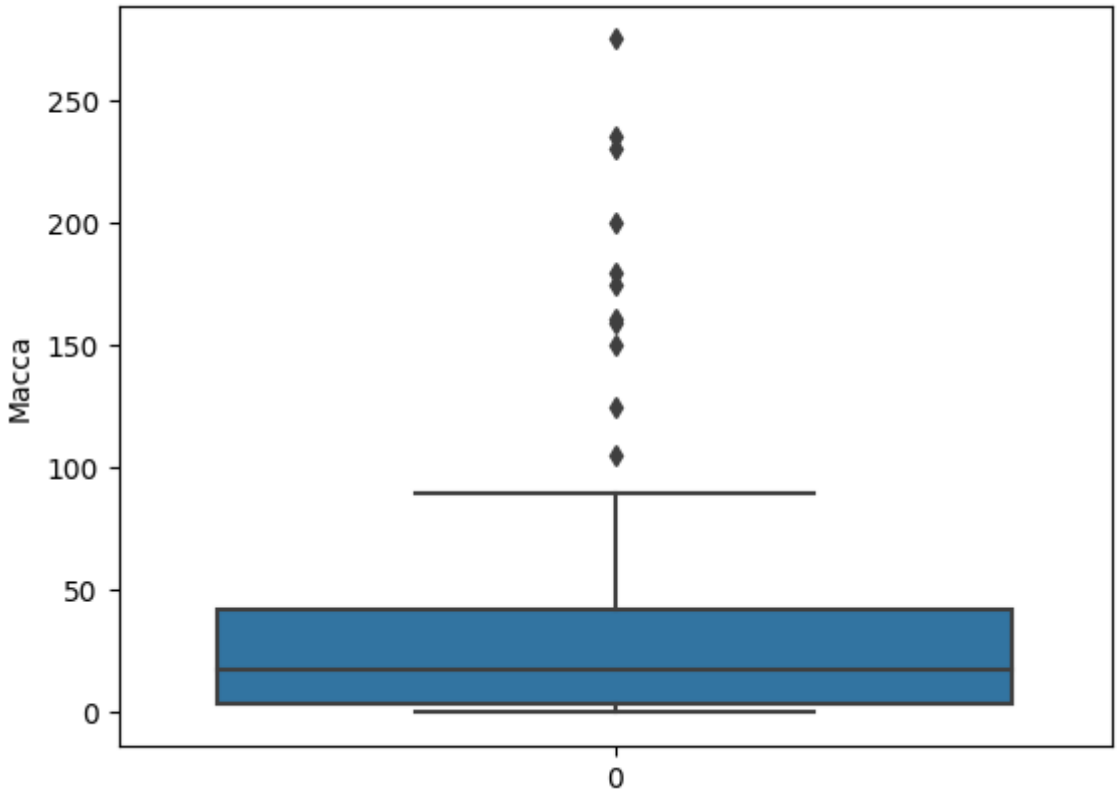
In [12]:
```
#Гистограмма количества дронов от массы
plt.hist(df['Mass_category_'])
plt.xlabel('Категория масс')
plt.ylabel('Частота')
plt.title('Гистограмма количества дронов  по всем категориям')
plt.xticks(rotation=0, ha='center')
plt.tick_params(axis='x', labelrotation=0)
plt.xticks(fontsize=7)
plt.yticks(fontsize=7)
plt.show()
```



In [13]:
```
df['Payload [kg]'].describe()
```

Out[13]:
```
count      91.000000
mean       87.418176
std       292.888113
min         0.000000
25%         0.000000
50%         2.500000
75%        20.000000
max      1701.000000
Name: Payload [kg], dtype: float64
```

In [14]:
```
Q1 = df['MTOW [kg]'].quantile(0.25)  # 1-й квартиль
Q3 = df['MTOW [kg]'].quantile(0.75)  # 3-й квартиль

IQR = Q3 - Q1  # Межквартильное расстояние

lower_bound = Q1 - 1.5 * IQR  # Нижняя граница
upper_bound = Q3 + 1.5 * IQR  # Верхняя граница

df_filtered = df[(df['MTOW [kg]'] >= lower_bound) & (df['MTOW [kg]'] <= upper_bound)]
sns.boxplot(df_filtered['MTOW [kg]'])
plt.ylabel("Масса")
plt.show()
```

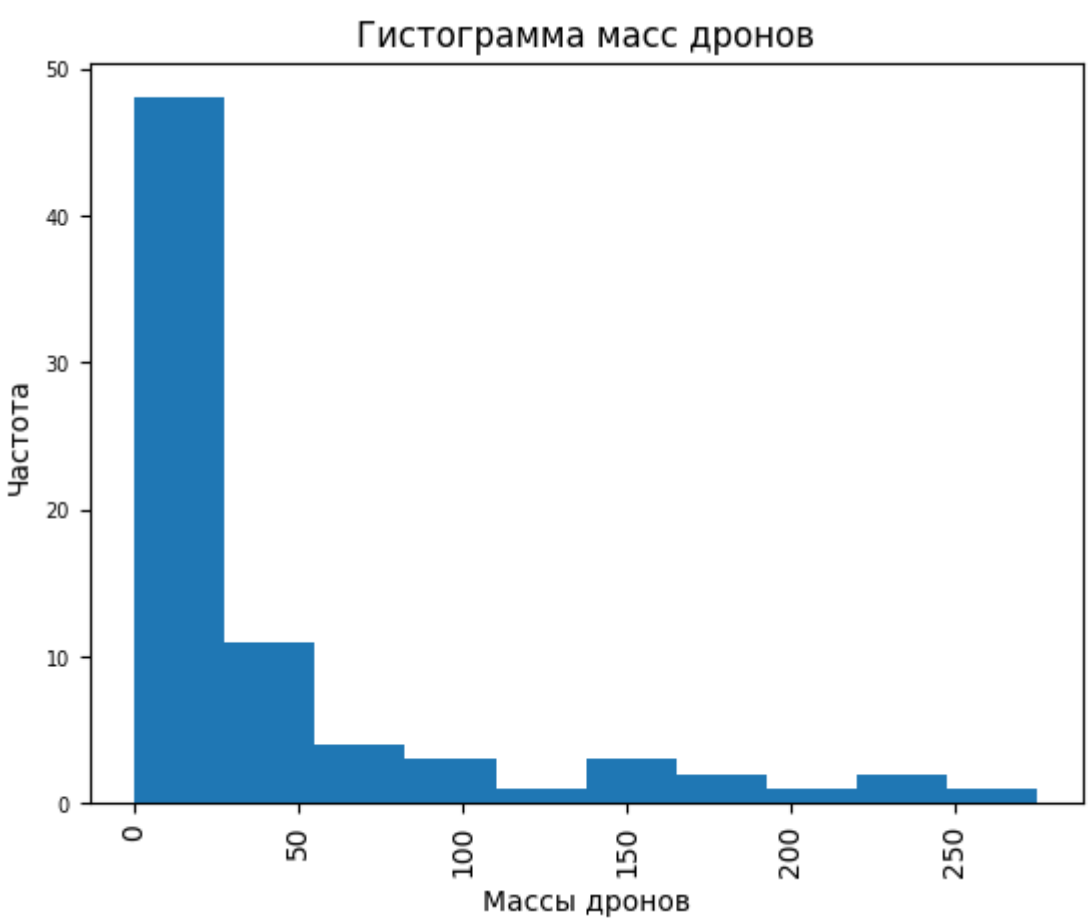


In [15]:
```
df_filtered['MTOW [kg]'].describe()
```

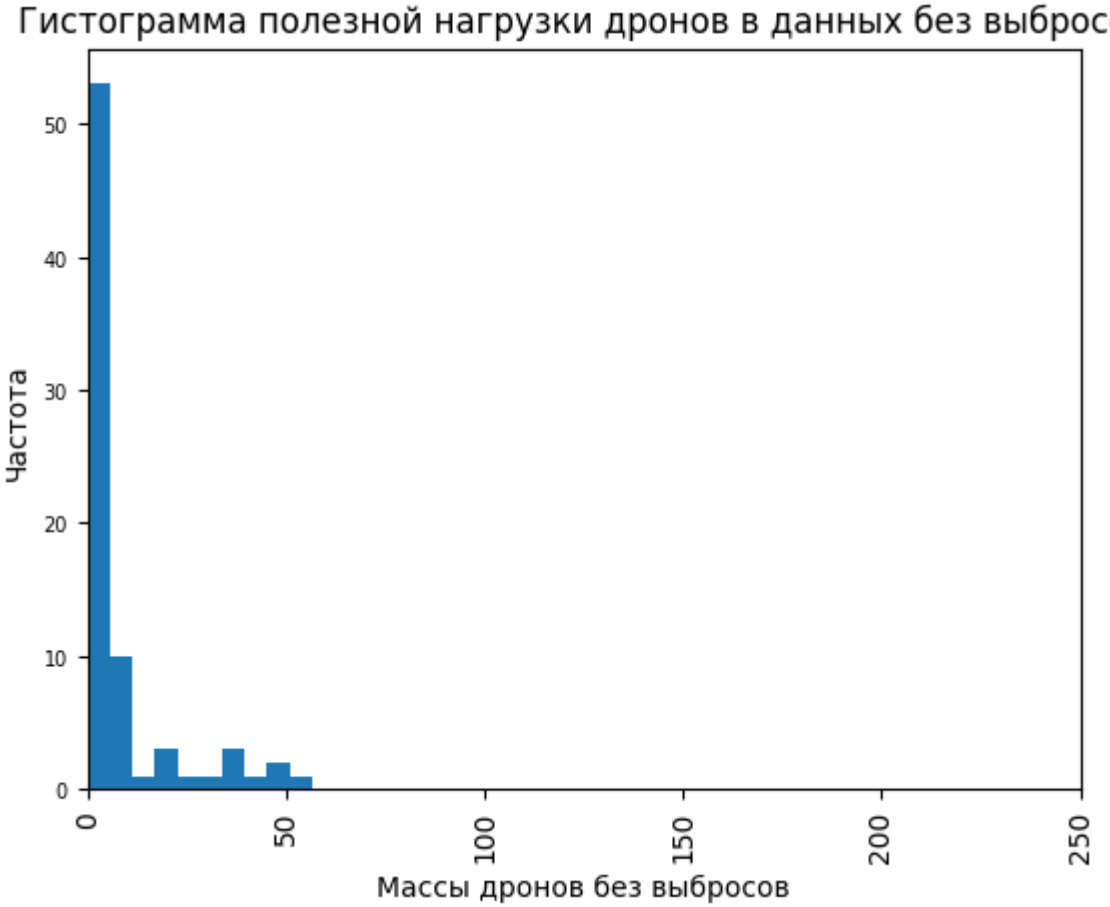
Out[15]:
```
count     76.000000
mean      43.335118
std       63.503624
min        0.016000
25%        3.875000
50%       18.000000
75%       41.750000
max      275.000000
Name: MTOW [kg], dtype: float64
```

In [16]:
```
#Гистограмма количества дронов от массы в данных
plt.hist(df[['MTOW [kg]']])
plt.xlabel('Названия дронов')
plt.ylabel('Частота')
plt.title('Гистограмма масс дронов')
plt.tick_params(axis='x', labelrotation=90)
plt.xticks(fontsize=10)
plt.yticks(fontsize=7)
plt.show()
```

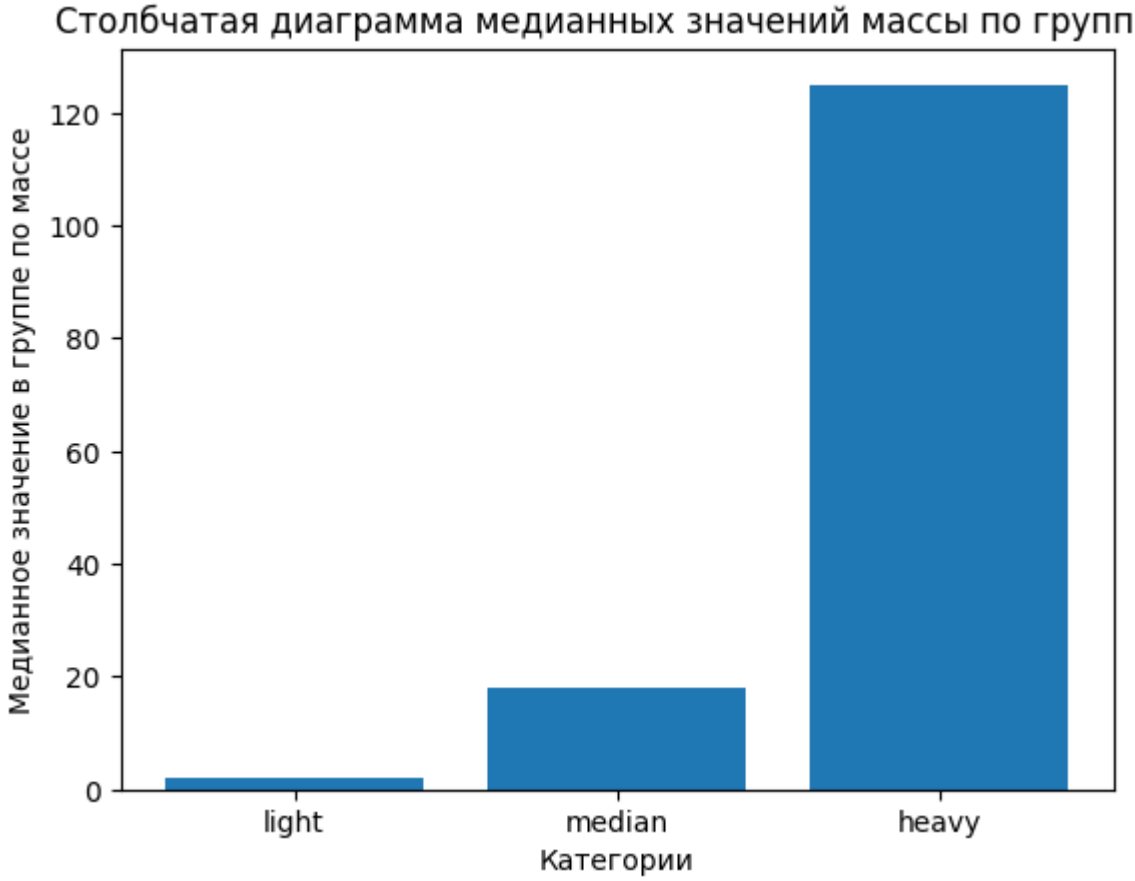
In [17]:
```
#Гистограмма количества дронов от массы в данных без выбросов
plt.hist(df_filtered[['MTOW [kg]']])
plt.xlabel('Массы дронов')
plt.ylabel('Частота')
plt.title('Гистограмма масс дронов')
plt.tick_params(axis='x', labelrotation=90)
plt.xticks(fontsize=10)
plt.yticks(fontsize=7)
plt.show()
```



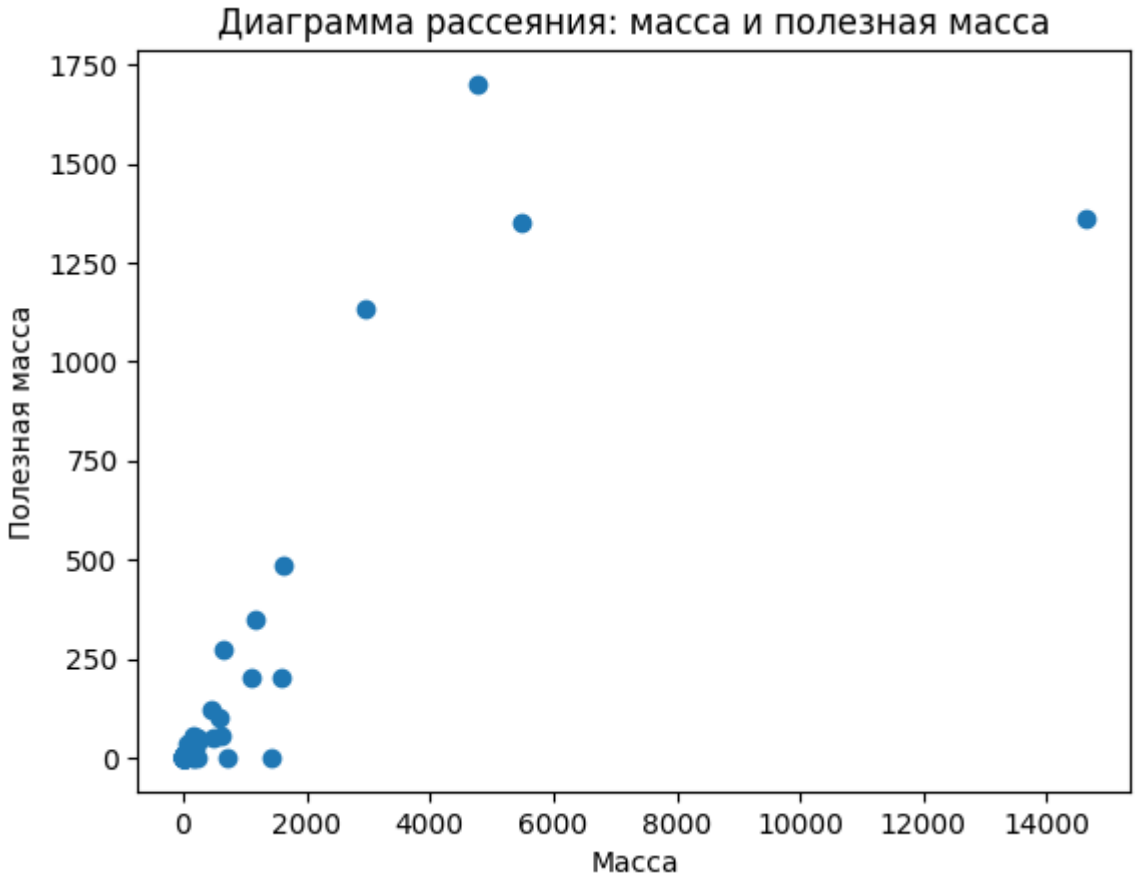
In [18]:
```
#Гистограмма количества дронов от полезной массы в данных без выбросов
plt.hist(df_filtered[['Payload [kg]']])
plt.xlabel('Массы дронов без выбросов')
plt.ylabel('Частота')
plt.title('Гистограмма полезной нагрузки дронов в данных без выбросов')
plt.tick_params(axis='x', labelrotation=90)
plt.xticks(fontsize=10)
plt.xlim(0,250)
plt.yticks(fontsize=7)
plt.show()
```



In [19]:
```
dfMedianMass=df_filtered.groupby(['Mass_category_'])['MTOW [kg]'].median()
categories = ['heavy', 'light', 'median']
#  Упорядочивание списка категорий и медианы
categories_sorted = sorted(categories, key=lambda x: dfMedianMass[categories.index(x)])

dfMed_sorted = [dfMedianMass[cat] for cat in categories_sorted]
# Построение столбчатой диаграммы
plt.bar(categories_sorted, dfMed_sorted)


# Добавление подписей
plt.xlabel('Категории')
plt.ylabel('Медианное значение в группе по массе')
plt.title('Столбчатая диаграмма медианных значений массы по группам')

# Отображение графика
plt.show()
```



In [20]:
```
plt.scatter(x = df['MTOW [kg]'], y = df['Payload [kg]']) #  Масса и Полезная масса
plt.ylabel("Полезная масса")
plt.xlabel("Масса")
plt.title('Диаграмма рассеяния: масса и полезная масса')
plt.show()
print("Корреляция полной массы и полезной нагрузки:")
df[['MTOW [kg]','Payload [kg]']].corr()# корреляция зависимости массы и полезной масс
```



Корреляция полной массы и полезной нагрузки:

Out[20]:

	MTOW [kg]	Payload [kg]
MTOW [kg]	1.000000	0.818437
Payload [kg]	0.818437	1.000000

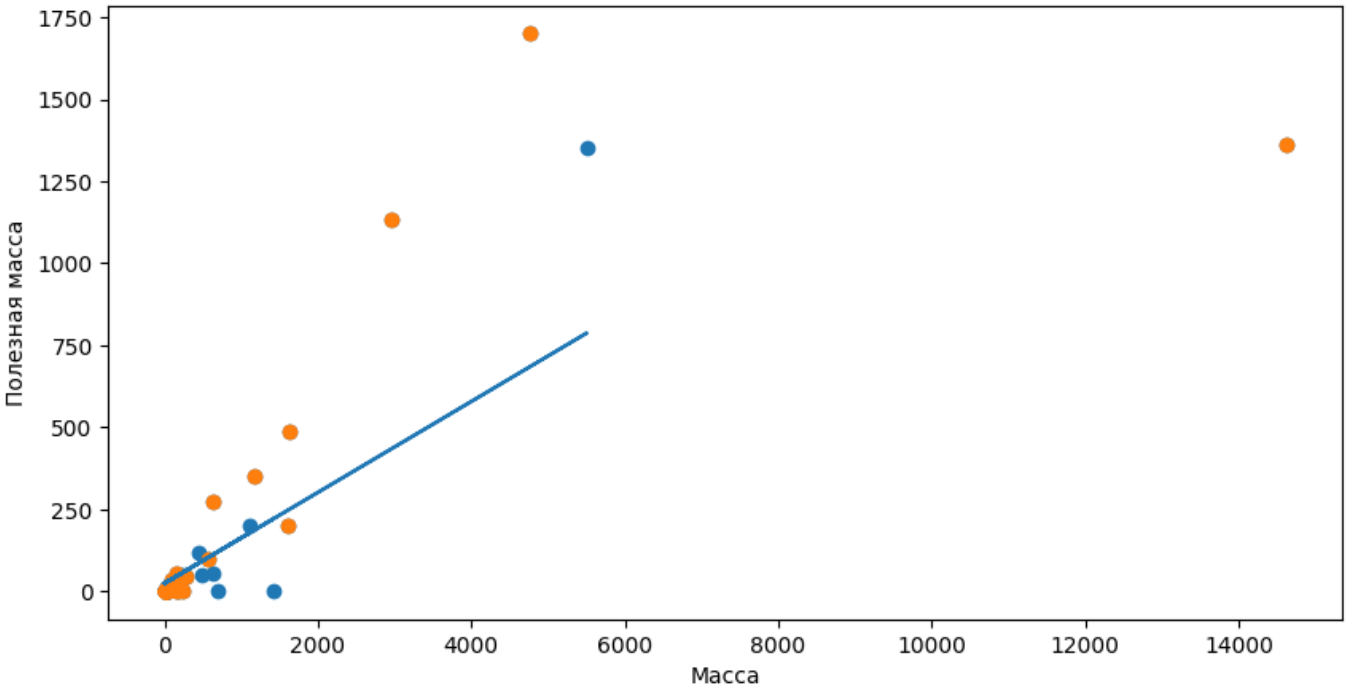
In [21]:
```
plt.scatter(x = df_filtered['MTOW [kg]'], y = df_filtered['Payload [kg]']) #  Масса и
plt.ylabel("Полезная масса")
plt.xlabel("Масса")
plt.title('Диаграмма рассеяния: масса и полезная масса (обрезаны тяжелые дроны)')
plt.ylim(0,150)
plt.show()
df_filtered[['MTOW [kg]','Payload [kg]']].corr()# корреляция зависимости массы и поле
```

Out[21]:

	MTOW [kg]	Payload [kg]
MTOW [kg]	1.000000	0.744715
Payload [kg]	0.744715	1.000000

In [22]:
```
# Линейная регрессия зависимости между массой и полезной нагрузкой
# Загружаем библотеку класса LinearRegression
from sklearn.linear_model import LinearRegression
#Подготавливаем данные
sampleX = df['MTOW [kg]'].astype(float).to_numpy()
sampleY = df['Payload [kg]'].astype(float).to_numpy()

# разделяем данные на обучающую и тестовую часть
# X_test = sampleX[0::2].reshape(-1,1)
# y_test = sampleY[0::2]
# X_train = sampleX[1::2].reshape(-1,1)
# y_train = sampleY[1::2]
X_test = sampleX[0::2].reshape(-1,1)
y_test = sampleY[0::2]
X_train = sampleX.reshape(-1,1)
y_train = sampleY
X_test1 = sampleX[1::2].reshape(-1,1)
y_test1 = sampleY[1::2]
```

```
#регриссионная модель
model =LinearRegression()
model.fit(X_train,y_train)
w = model.coef_

b = model.intercept_


plt.figure(figsize=(10,5))
plt.scatter(X_train,y_train,label='train')
plt.scatter(X_test,y_test,label='test')
# Чертим линию тренда
plt.plot(sampleX[1::2],sampleX[1::2].reshape(-1,1).dot(w)+b,label='predicted')
# plt.legend(loc='best')
plt.xlabel('Масса')
plt.ylabel('Полезная масса')
# plt.ylim(80, 100)  # Установка пределов для оси y
# Метод plt.axis()
# plt.axis(ymin=80, ymax=100)  # Установка пределов для оси y
plt.show()
# Выводим на печать линию тренда
print('Линия тренда:Полезная масса=',w,"Масса +",b)



from sklearn.metrics import mean_squared_error
y_train_predicted=model.predict(X_train)
y_test_predicted =model.predict(X_test)
y_test_predicted1 =model.predict(X_test1)
print('TrainMSE:',mean_squared_error(y_train,y_train_predicted))
print('TrainMSE:',mean_squared_error(y_test,y_test_predicted))
print('TrainMSE:',mean_squared_error(y_test1,y_test_predicted1))
```

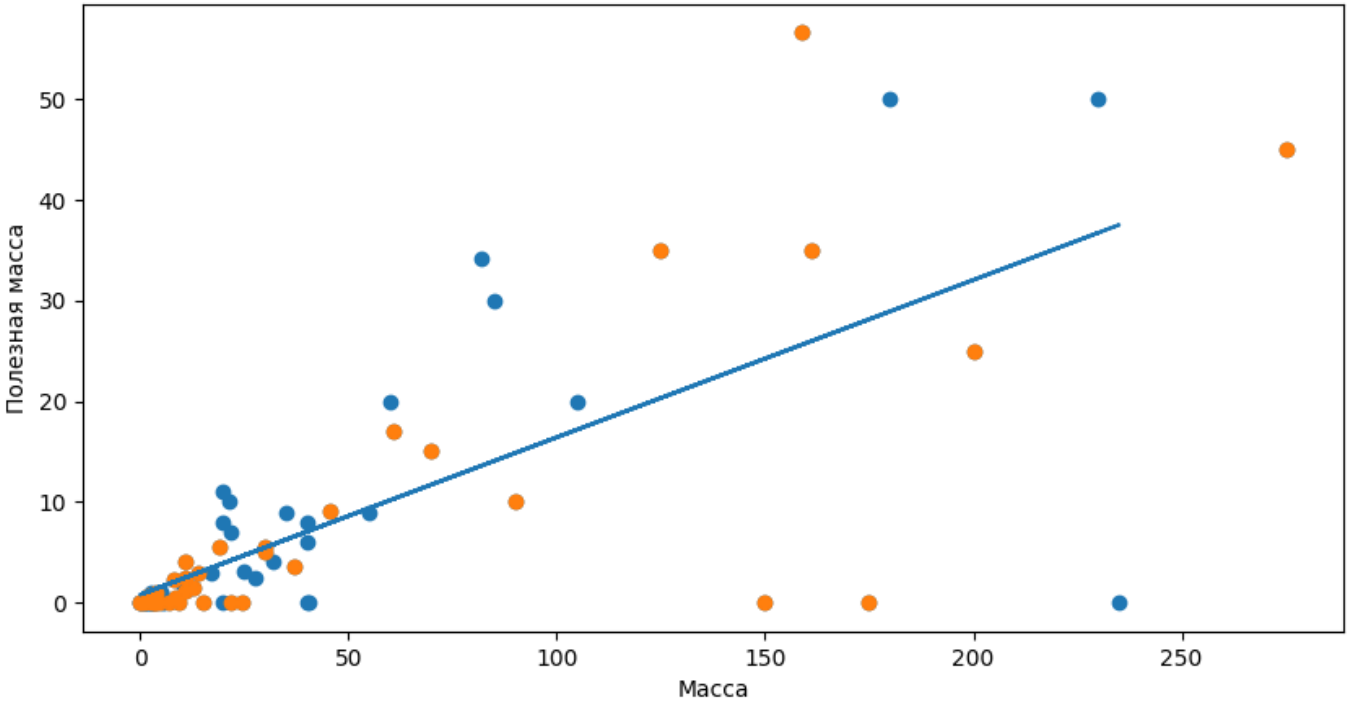



```
Линия тренда:Полезная масса= [0.13865609] Масса + 24.126595947806784
TrainMSE: 28011.135989508413
TrainMSE: 46543.59060499457
TrainMSE: 9066.849049233679
```

In [23]:

```
# Линейная регрессия зависимости между массой и полезной нагрузкой обрезаны тяжелые д
# Загружаем библотеку класса LinearRegression
from sklearn.linear_model import LinearRegression
#Подготавливаем данные
sampleX = df_filtered['MTOW [kg]'].astype(float).to_numpy()
```

```
sampleY = df_filtered['Payload [kg]'].astype(float).to_numpy()

# разделяем данные на обучающую и тестовую часть
# X_test = sampleX[0::2].reshape(-1,1)
# y_test = sampleY[0::2]
# X_train = sampleX[1::2].reshape(-1,1)
# y_train = sampleY[1::2]
X_test = sampleX[0::2].reshape(-1,1)
y_test = sampleY[0::2]
X_train = sampleX.reshape(-1,1)
y_train = sampleY
X_test1 = sampleX[1::2].reshape(-1,1)
y_test1 = sampleY[1::2]


#регриссионная модель
model =LinearRegression()
model.fit(X_train,y_train)
w = model.coef_

b = model.intercept_


plt.figure(figsize=(10,5))
plt.scatter(X_train,y_train,label='train')
plt.scatter(X_test,y_test,label='test')
# Чертим линию тренда
plt.plot(sampleX[1::2],sampleX[1::2].reshape(-1,1).dot(w)+b,label='predicted')
# plt.legend(loc='best')
plt.xlabel('Масса')
plt.ylabel('Полезная масса')
# plt.ylim(80, 100)  # Установка пределов для оси y
# Метод plt.axis()
# plt.axis(ymin=80, ymax=100)  # Установка пределов для оси y
plt.show()
# Выводим на печать линию тренда
print('Линия тренда:Полезная масса=',w,"Масса +",b)



from sklearn.metrics import mean_squared_error
y_train_predicted=model.predict(X_train)
y_test_predicted =model.predict(X_test)
y_test_predicted1 =model.predict(X_test1)
print('TrainMSE:',mean_squared_error(y_train,y_train_predicted))
print('TrainMSE:',mean_squared_error(y_test,y_test_predicted))
print('TrainMSE:',mean_squared_error(y_test1,y_test_predicted1))
```




```
Линия тренда:Полезная масса= [0.15623343] Масса + 0.7829480981573287
TrainMSE: 78.01225480584282
TrainMSE: 75.17875263250657
TrainMSE: 80.84575697917906
```