In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import wbdata
import matplotlib.pyplot as plt
import scipy.stats as st
from sympy import *
import time as dt
import warnings
from IPython.display import Math,Latex
from pandas.errors import SettingWithCopyWarning

from sklearn.linear_model import LinearRegression
from statsmodels.api import *
from sklearn.metrics import r2_score
from sklearn.preprocessing import PolynomialFeatures,StandardScaler,MinMaxScaler

from scipy.stats import jarque_bera, kstest, kstwobign
```

In [2]:
```
from country_dict import *

import os
from docx import Document
from docx.shared import Inches, Pt
from docx.table import Table
from openpyxl import load_workbook
from deep_translator import GoogleTranslator
from docx.enum.text import WD_ALIGN_PARAGRAPH
##########################
import locale
locale.setlocale(locale.LC_NUMERIC, 'russian')
plt.rcParams['axes.formatter.use_locale'] = True
#########
#########
plt.rcParams["font.family"] = "Times New Roman"
plt.rcParams['mathtext.fontset'] = 'cm'
###################
init_printing(use_unicode=True,use_latex=True)
###################
def add_excel_table_to_docx(xlsx_file, docx_file, sheet_name='Sheet1',from_row=1
    """Добавляет таблицу из указанного листа Excel в документ Word.

    Args:
        xlsx_file (str): Путь к файлу Excel.
        docx_file (str): Путь к документу Word.
        sheet_name (str, optional): Название листа в Excel. По умолчанию 'Лист1'
    """

    # Загрузка данных из Excel
    workbook = load_workbook(xlsx_file)
    worksheet = workbook[sheet_name]

    # Создание нового документа Word или открытие существующего
    document = Document(docx_file)

    # Создание таблицы в документе Word
    table = document.add_table(rows=worksheet.max_row, cols=worksheet.max_column
```

```
    # Заполнение ячеек таблицы данными из Excel
    for row_idx in range(from_row, worksheet.max_row + 1):
        for col_idx in range(1, worksheet.max_column + 1):
            cell = table.cell(row_idx - 1, col_idx - 1)
            cell.text = str(worksheet.cell(row=row_idx, column=col_idx).value)

    # Сохранение документа Word
    document.save(docx_file)
###################
def add_text(docx_file,text:str,font_name='Times New Roman',font_size=12,par_all
    alligments={
        'CENTER': WD_ALIGN_PARAGRAPH.CENTER,
        'LEFT':WD_ALIGN_PARAGRAPH.LEFT,
        'RIGHT':WD_ALIGN_PARAGRAPH.RIGHT,
    }
    document = Document(docx_file)
    paragraph = document.add_paragraph()
    paragraph.alignment = alligments[par_allign]  # Выравнивание по центру (можн
    run = paragraph.add_run(text)
    run.font.size=Pt(font_size)
    run.font.name=font_name
    document.save(docx_file)
###################
def add_image_to_docx(docx_file, image_path, indent_size=-2, width=10.0):
    indent_size=Inches(indent_size)
    width=Inches(width)
    """Добавляет изображение PNG в документ DOCX с заданным отступом слева.

    Args:
        docx_file (str): Путь к существующему или новому DOCX файлу.
        image_path (str): Путь к изображению PNG.
        indent_size (Inches, optional): Размер отступа слева в дюймах. По умолча
        width (Inches, optional): Ширина изображения в дюймах. По умолчанию 3 дю
    """

    document = Document(docx_file)
    paragraph = document.add_paragraph()
    paragraph.alignment = WD_ALIGN_PARAGRAPH.CENTER  # Выравнивание по центру (м
    run = paragraph.add_run()
    run.add_picture(image_path, width=width)
    paragraph.paragraph_format.first_line_indent = indent_size
    document.save(docx_file)
###################
```

Возьмем данные из библиотеки `wbdata`

In [3]:
```
country = 'IND'
document = Document()
indicators = {
    #'NY.'Inflation rate'.DEFL.KD.ZG': 'Inflation rate',
    'FP.CPI.TOTL': 'Inflation rate',
    'SL.UEM.TOTL.ZS': 'Unemployment rate',
}
data = wbdata.get_dataframe(indicators, country=country )
```

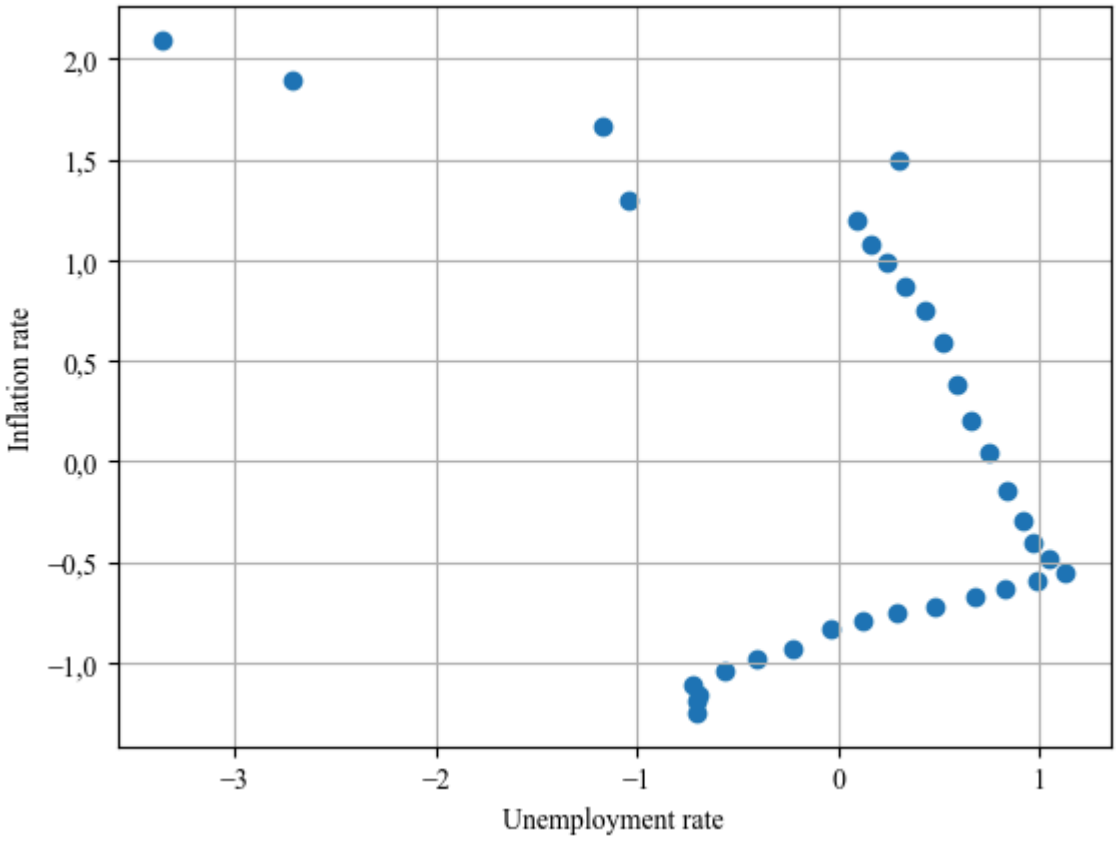
```
data = data.dropna().sort_values('date')

try:
    os.mkdir(f'{country}')
    os.chdir(f'{country}')
except:
    os.chdir(f'{country}')


try:
    data.to_excel(f'{country}_1.xlsx')
except:
    pass

display(data)


document.save(f'{country}.docx')
add_text(f'{country}.docx',f'Кривая Филлипса для страны {country_dict[country]}'
add_text(f'{country}.docx',f'\tПолучим данные о динамике безработицы(Inflation r
add_excel_table_to_docx(f'{country}_1.xlsx',f'{country}.docx')
```

date	Inflation rate	Unemployment rate
1991	26.132091	6.850
1992	29.212495	6.853
1993	31.060737	6.859
1994	34.243821	6.828
1995	37.745213	6.990
1996	41.133658	7.147
1997	44.080577	7.335
1998	49.912808	7.517
1999	52.243646	7.682
2000	54.338322	7.856
2001	56.391926	8.039
2002	58.815173	8.248
2003	61.053595	8.397
2004	63.353638	8.551
2005	66.043851	8.697
2006	69.872099	8.614
2007	74.324964	8.534
2008	80.530554	8.486
2009	89.294173	8.406
2010	100.000000	8.318
2011	108.911793	8.222
2012	119.235539	8.156
2013	131.180410	8.088
2014	139.924446	7.992
2015	146.790502	7.894
2016	154.054013	7.800
2017	159.181198	7.723
2018	165.451069	7.652
2019	171.621576	6.510
2020	182.988823	7.859
2021	192.378725	6.380
2022	205.266241	4.822

date	Inflation rate	Unemployment rate
2023	216.862025	4.172

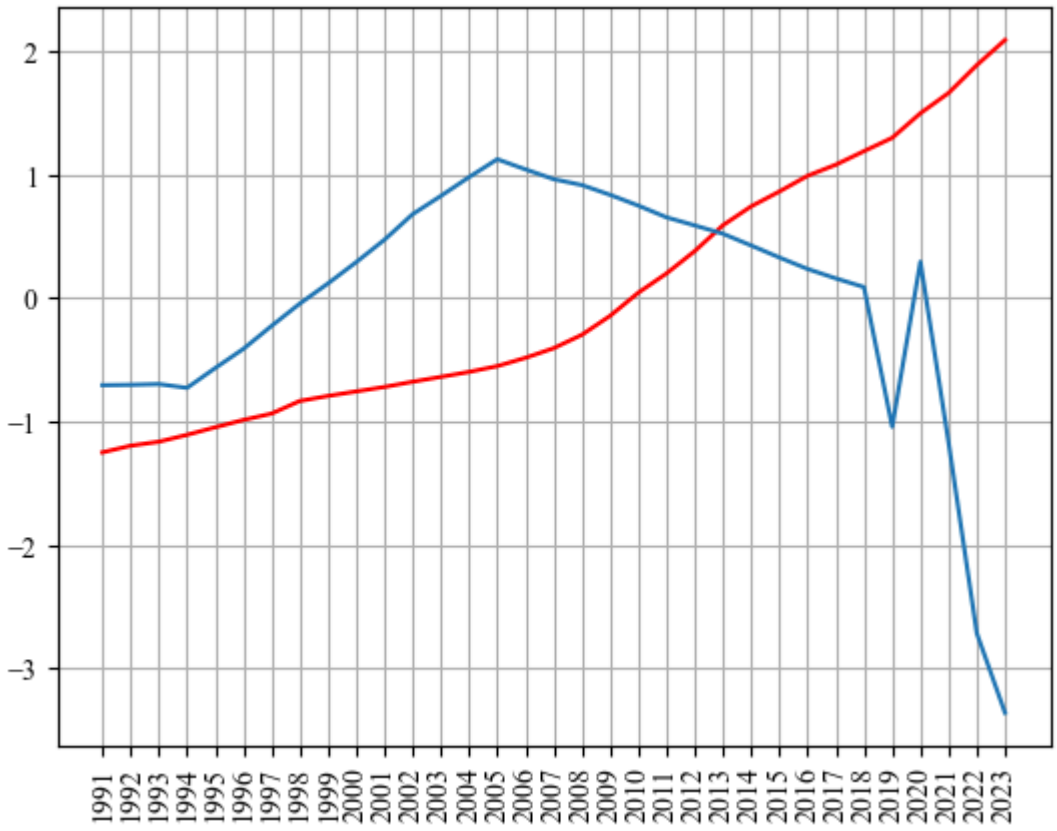
Стандартизируем полученные данные

In [4]:
```
s=StandardScaler()
x=s.fit_transform(data[['Unemployment rate']])
s=StandardScaler()
y=s.fit_transform(data[['Inflation rate']])


plt.xlabel('Unemployment rate')
plt.ylabel('Inflation rate')
plt.grid()
plt.scatter(x,y)

try:
    plt.savefig(f'{country}_2.png')
except:
    pass

add_text(f'{country}.docx','\n\nСтандартизируем полученнные данные и выведем пол
add_image_to_docx(f'{country}.docx',f'{country}_2.png',0,5)

plt.show()
```



In [5]:
```
plt.plot(data.index,y,color='r')
plt.plot(data.index,x)
```

```
plt.xticks(rotation=90)
plt.grid()

try:
    plt.savefig(f'{country}_3.png')
except:
    pass

add_text(f'{country}.docx','\n\nПроведем анализ изменения инфляции и безработицы
add_image_to_docx(f'{country}.docx',f'{country}_3.png',0,5)

plt.show()
```



In [6]:
```
maxgdp_year=int(*data[data['Inflation rate']==max(data['Inflation rate'])].index
maxunemp_year=int(*data[data['Unemployment rate']==max(data['Unemployment rate']
mingdp_year=int(*data[data['Inflation rate']==min(data['Inflation rate'])].index
minunemp_year=int(*data[data['Unemployment rate']==min(data['Unemployment rate']


add_text(f'{country}.docx',f'\tКак видно из графиков, за рассматриваемый период
```

построим модель линейной регрессии

In [7]:
```
x1 = add_constant(x)
model = OLS(y,x1).fit()

model.summary().as_text()

text = '\n\nПостроим модель линейной регрессии на получившихся данных\n\n' + mod

add_text(f'{country}.docx',text,par_allign='CENTER')
```

Построим модель гиперболической функции

In [8]:
```
x1 = add_constant(x)
x1 = x1/x
model = OLS(y,x1).fit()

model.summary().as_text()

text = '\n\nПостроим модель гиперболической функции:\n\n' + model.summary().as_t

add_text(f'{country}.docx',text,par_allign='CENTER')

model.summary()
```

Out[8]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.090
Model:	OLS	**Adj. R-squared:**	0.061
Method:	Least Squares	**F-statistic:**	3.063
Date:	Sun, 03 Nov 2024	**Prob (F-statistic):**	0.0900
Time:	20:45:16	**Log-Likelihood:**	-45.270
No. Observations:	33	**AIC:**	94.54
Df Residuals:	31	**BIC:**	97.53
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
x1	0.0578	0.033	1.750	0.090	-0.010	0.125
const	-0.0329	0.172	-0.191	0.850	-0.384	0.319

Omnibus:	3.368	**Durbin-Watson:**	0.198
Prob(Omnibus):	0.186	**Jarque-Bera (JB):**	2.969
Skew:	0.655	**Prob(JB):**	0.227
Kurtosis:	2.336	**Cond. No.**	5.25

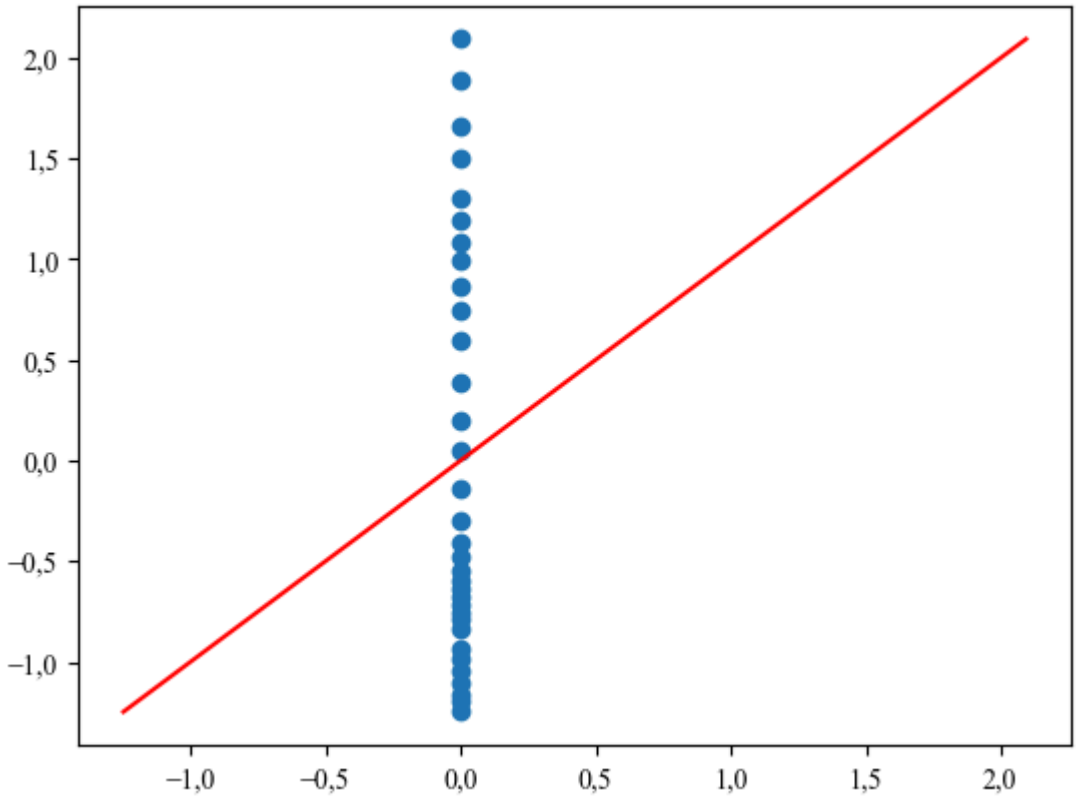
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Как видно, гиперболическая функция еще хуже описывает ситуацию

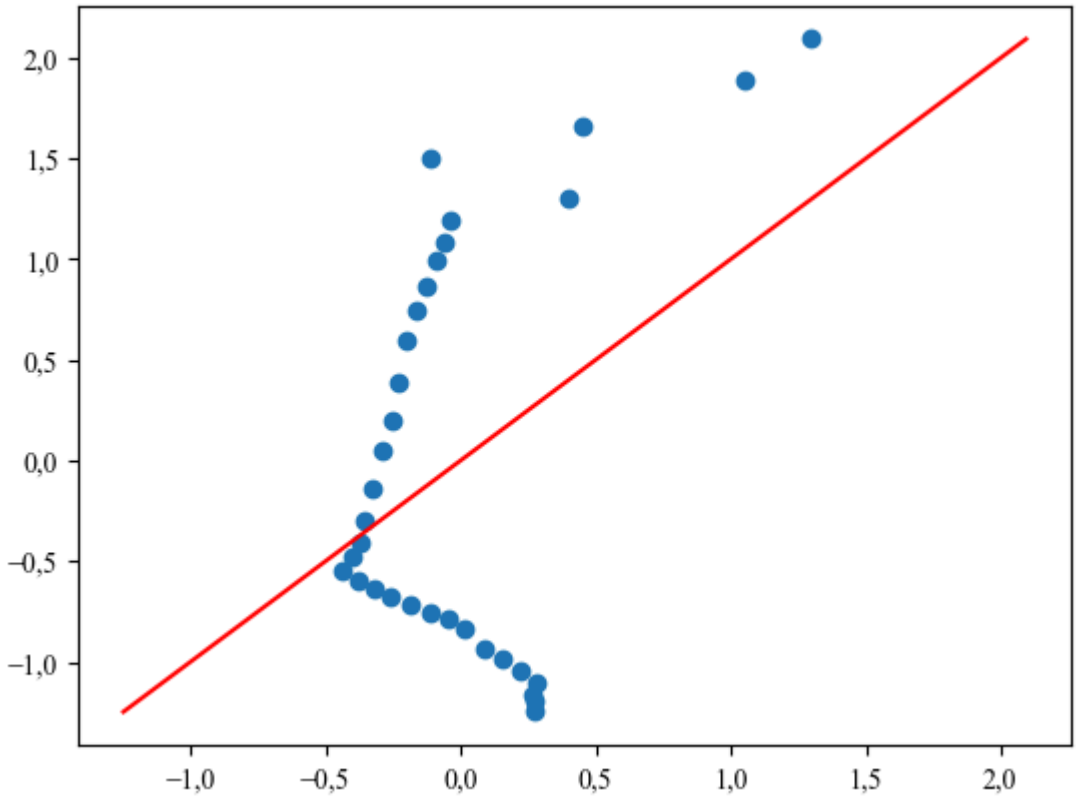
Попробуем совершить полиномиальные преобразования

In [9]:
```
add_text(f'{country}.docx','Попробуем совершить полиномиальные преобразования',p

degrees=range(27)

for deg in degrees:
        xp=x
        pf = PolynomialFeatures(degree=deg)
        xp = pf.fit_transform(xp)

        model = OLS(y,xp).fit()

        display((deg, model.rsquared))
        plt.scatter(model.predict(xp), y)
        plt.plot([np.min(y),np.max(y)],[np.min(y),np.max(y)],color='red')

        if deg%6.5==0:

                try:
                        plt.savefig(f'{country}_x_deg_{deg}.png')
                        add_text(f'{country}.docx',f'Степень полинома = {deg},\n
                        add_image_to_docx(f'{country}.docx',f'{country}_x_deg_{d
                except:
                        pass
        plt.show()
```

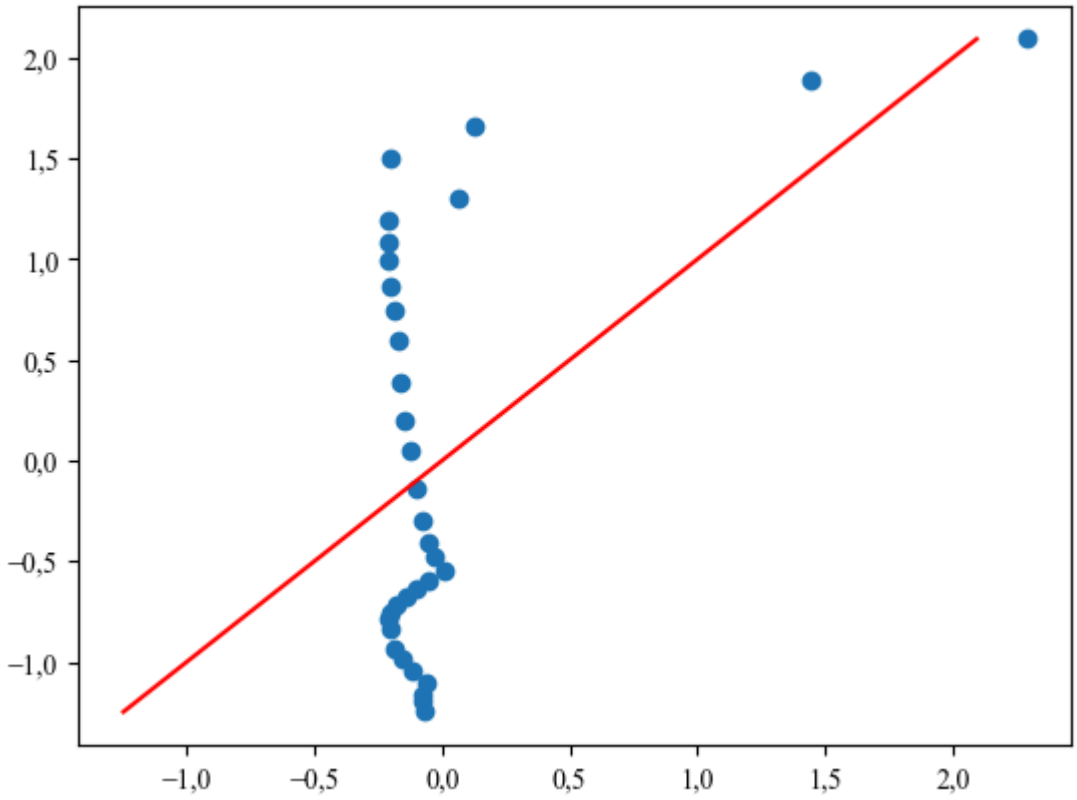
$(0,\ 0.0)$



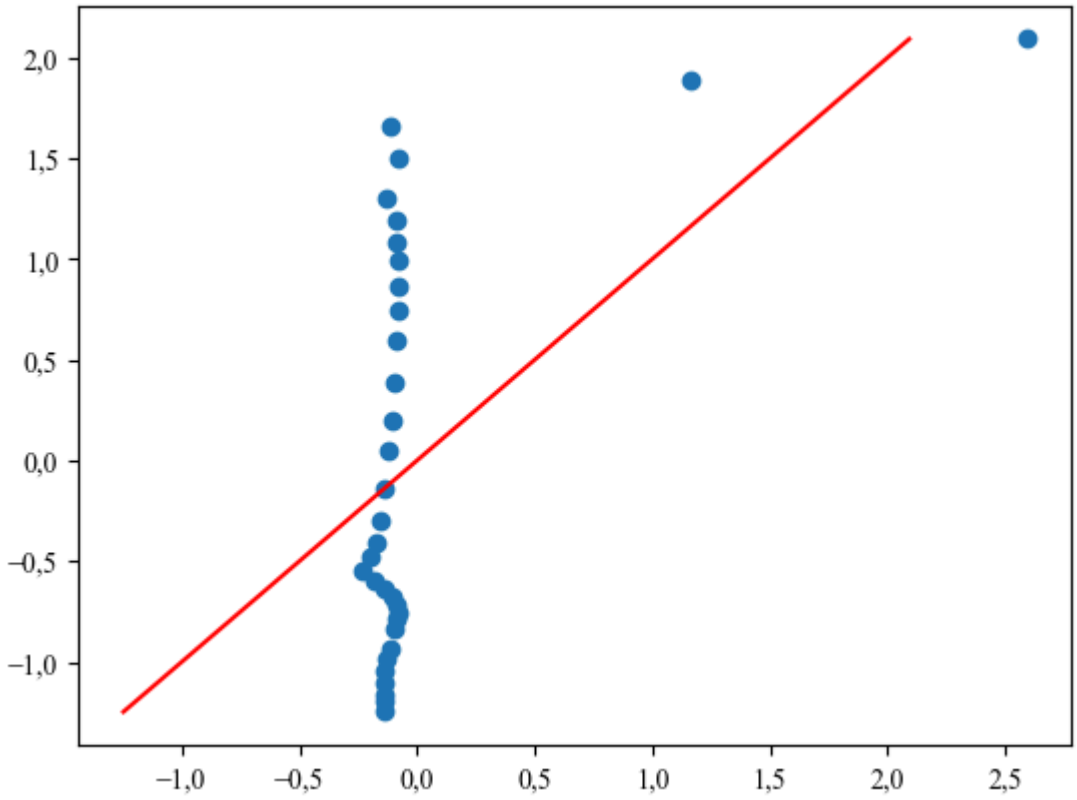
$(1,\ 0.149425381139715)$



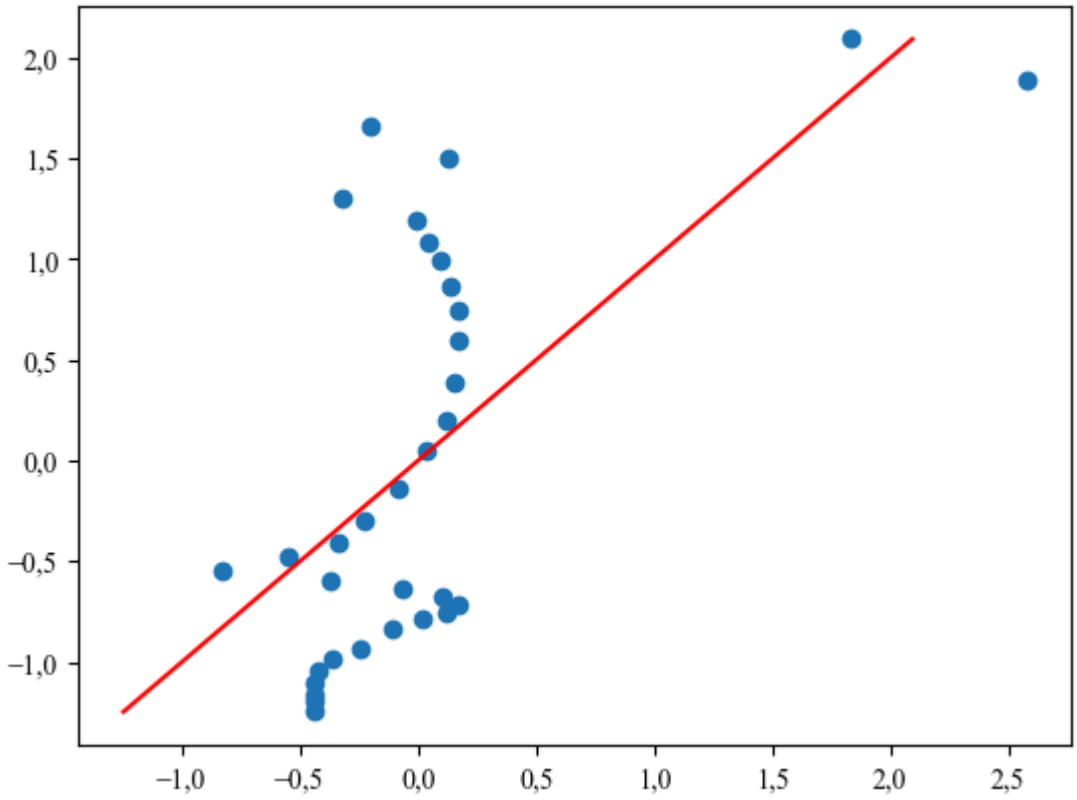
$(2,\ 0.241969035743383)$



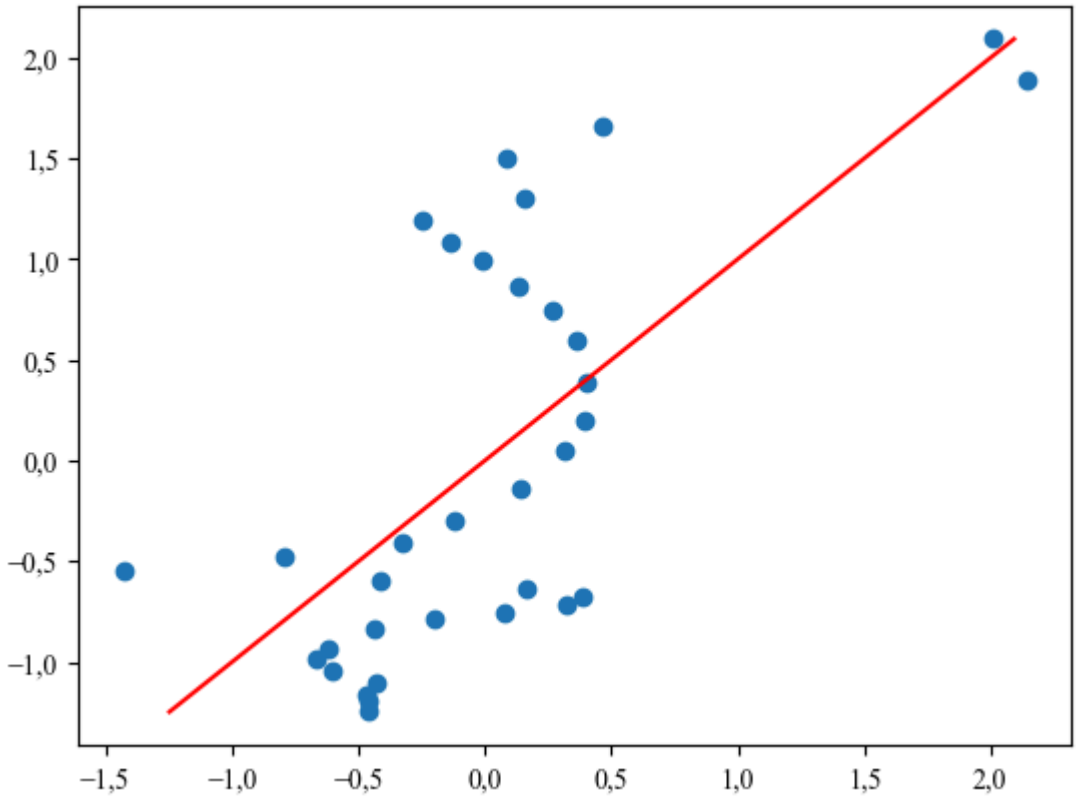
$(3,\ 0.259375634204206)$



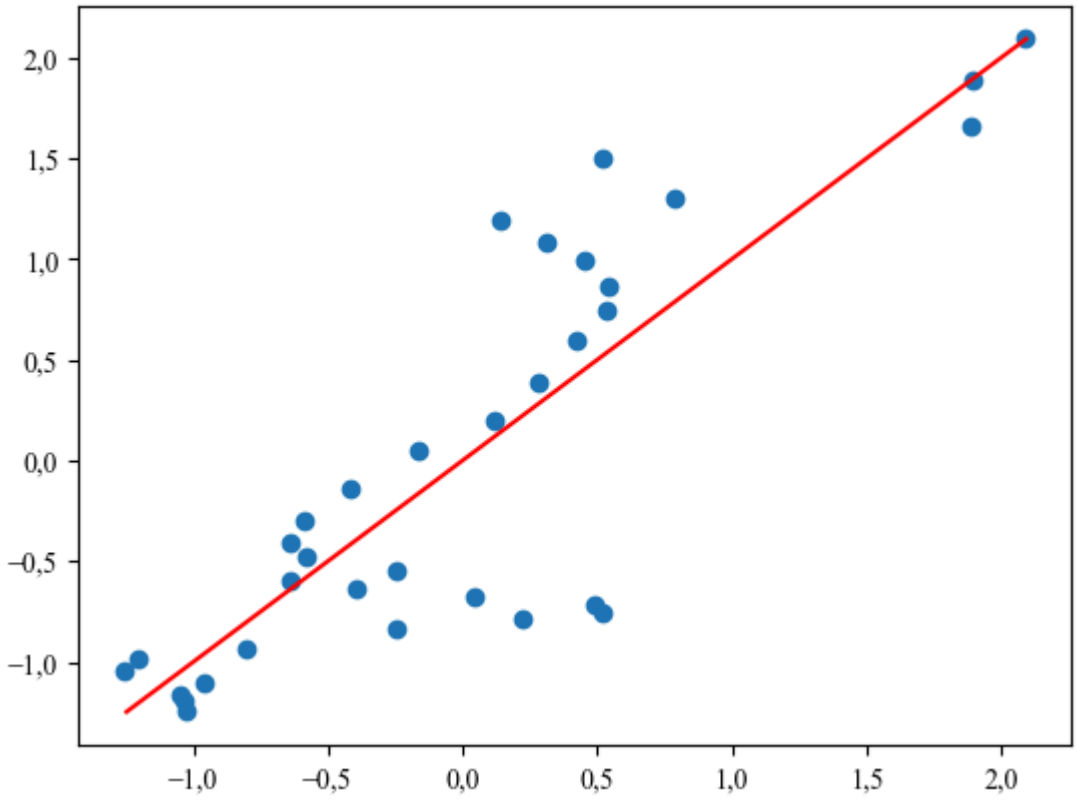
(4, 0.387745316128422)



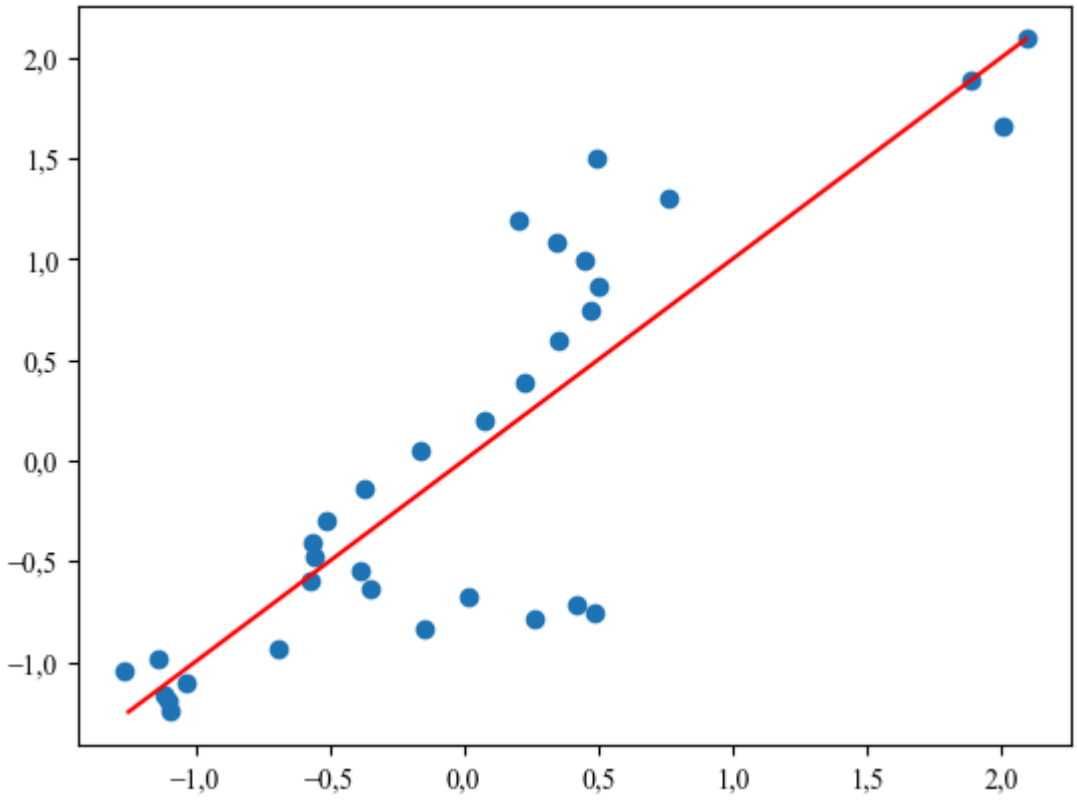
(5, 0.458711075212763)



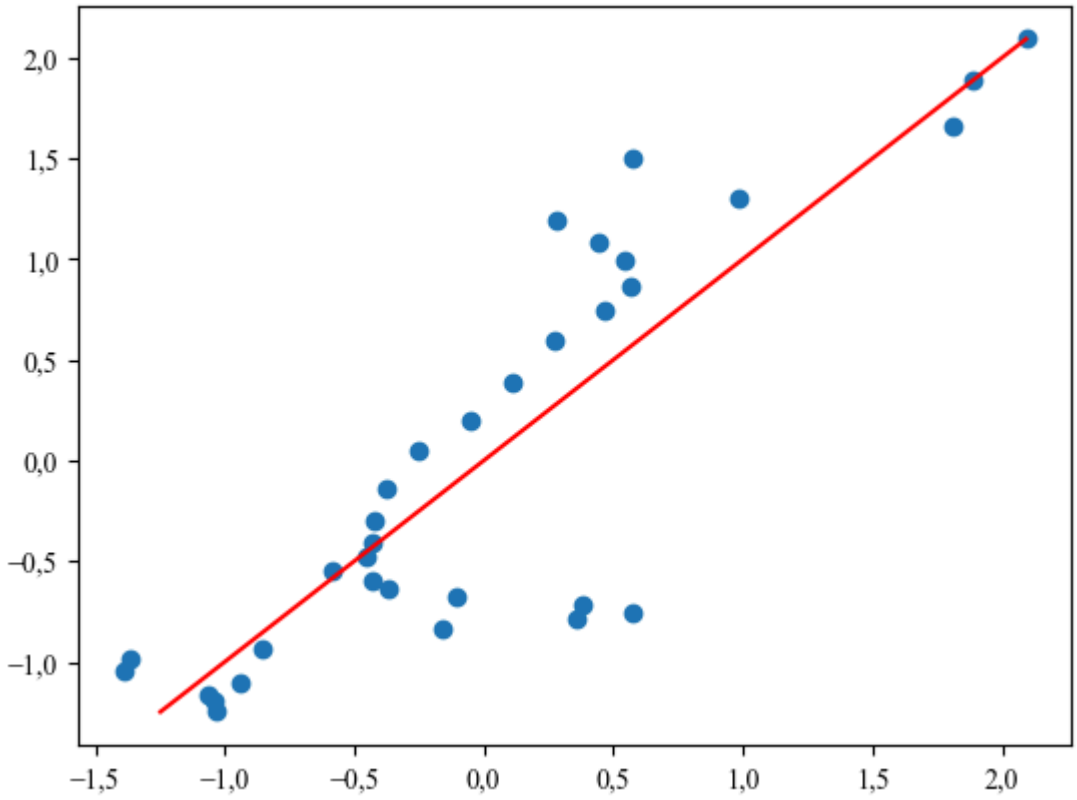
(6, 0.725723192334403)



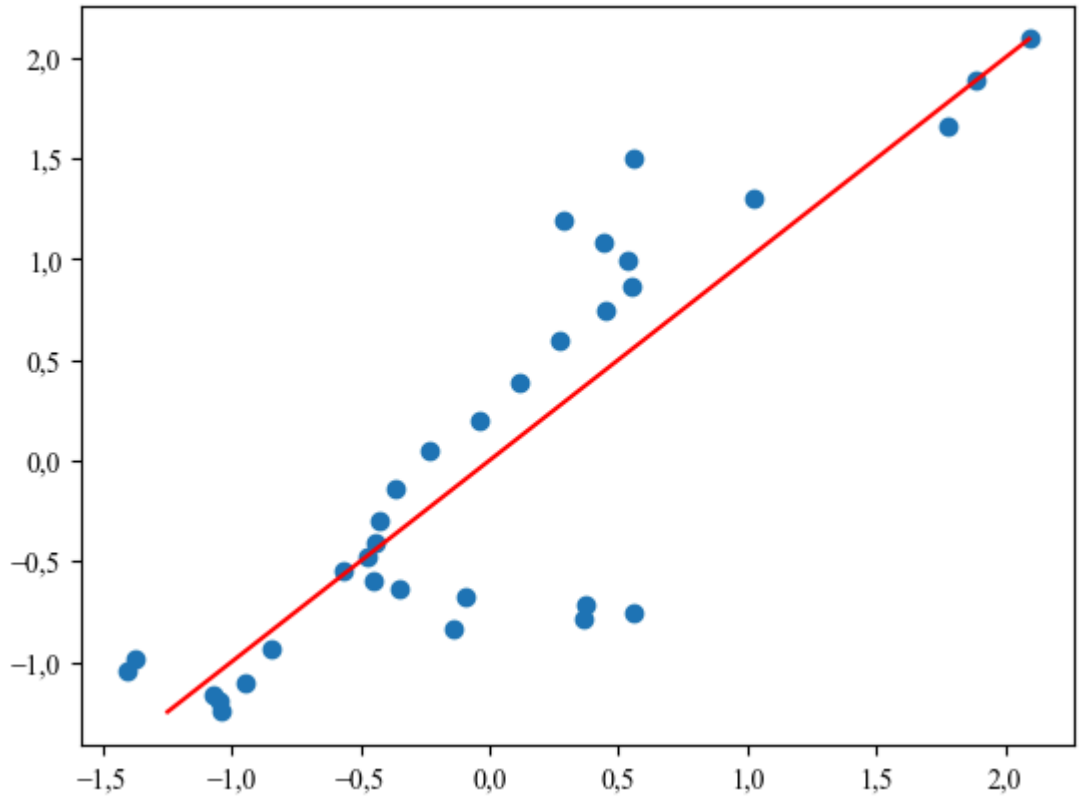
(7, 0.729671039607428)



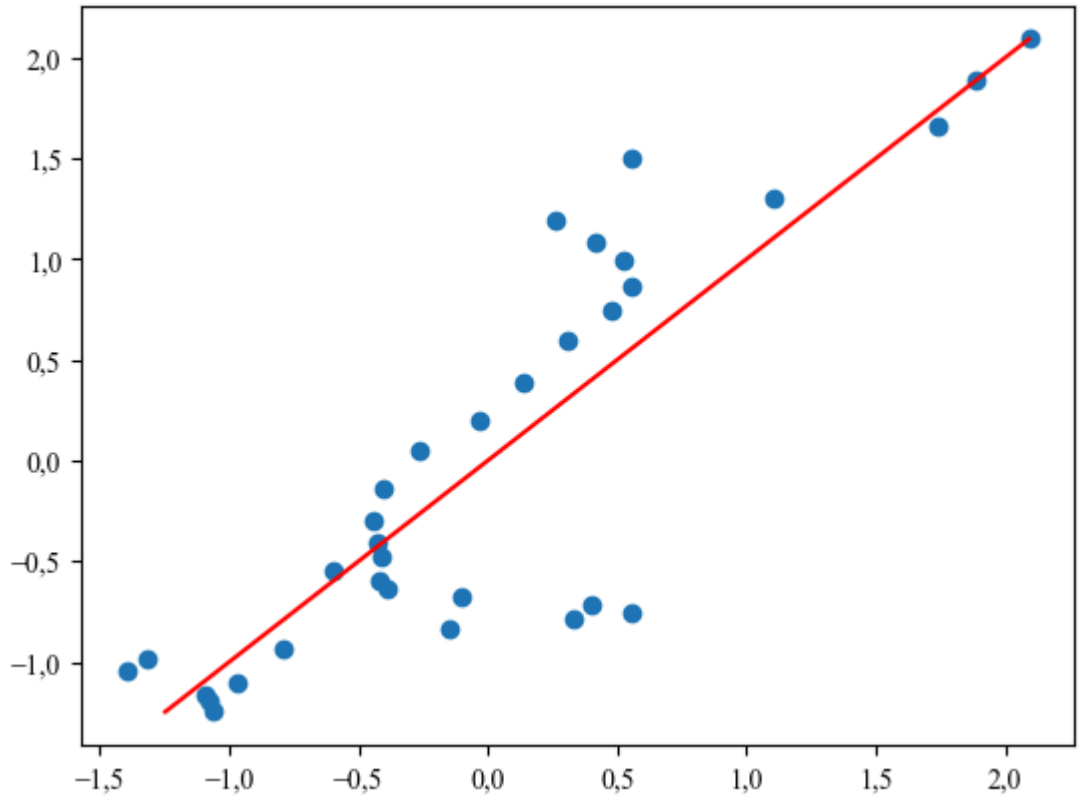
(8, 0.742017700524081)



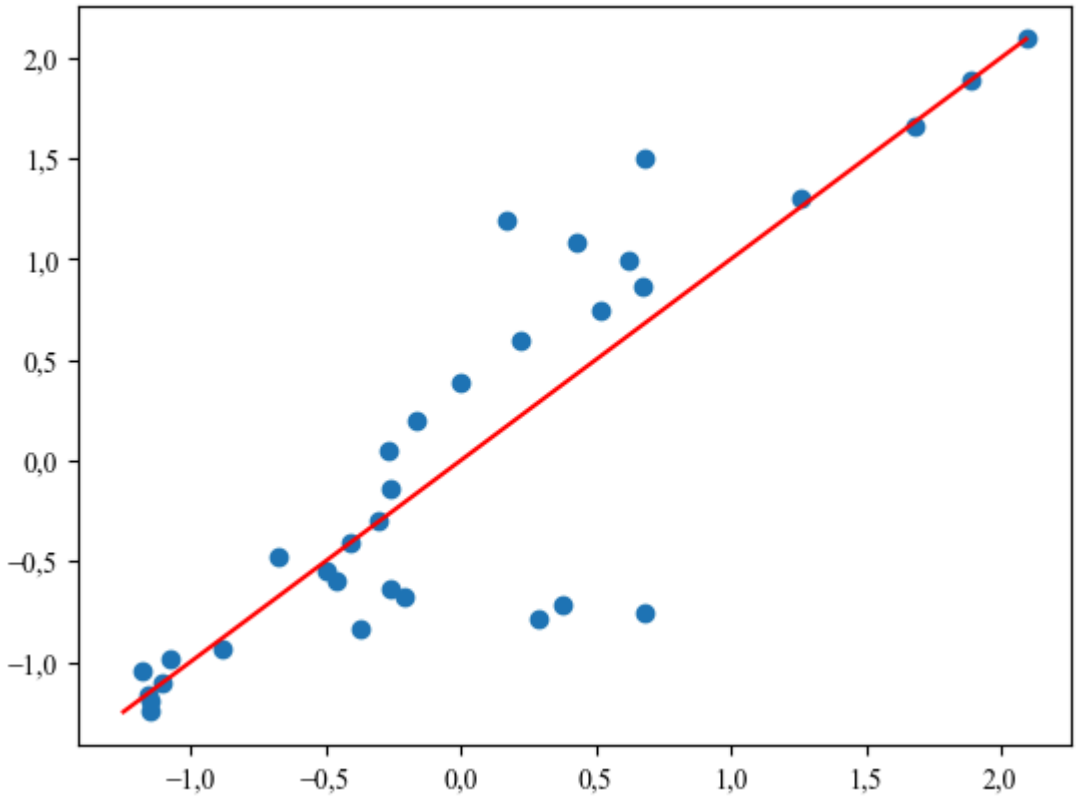
(9, 0.742219503378833)



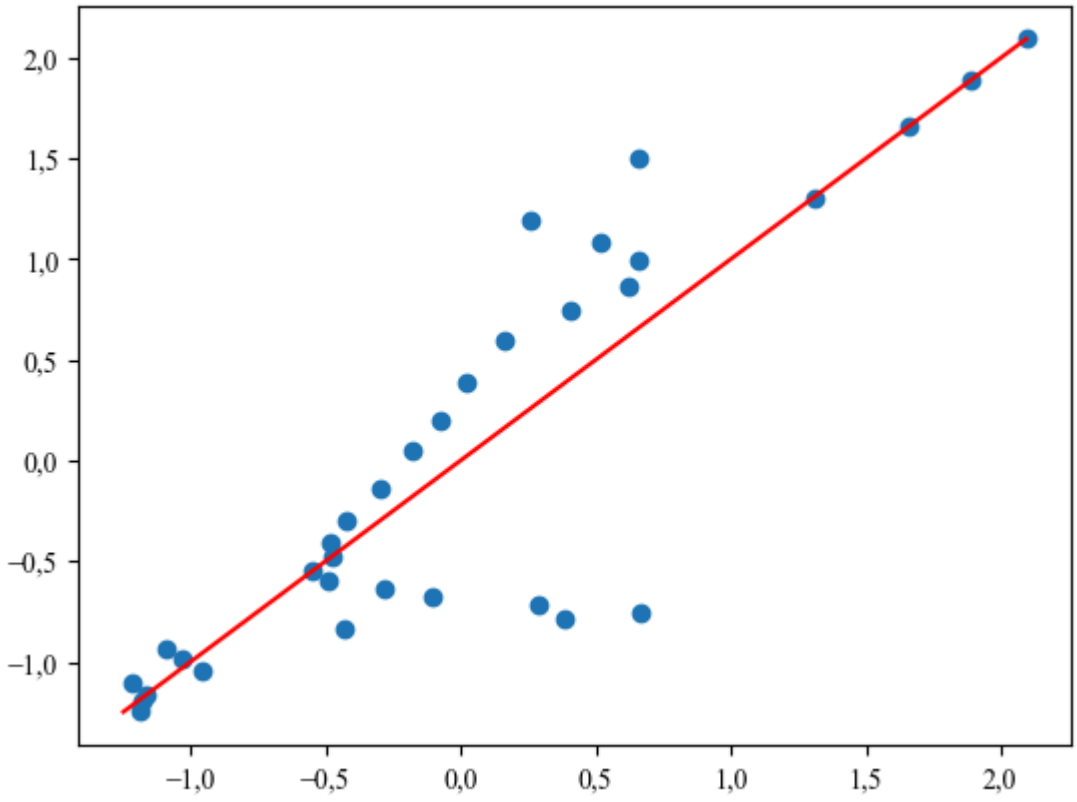
$(10,\ 0.7431827859313)$



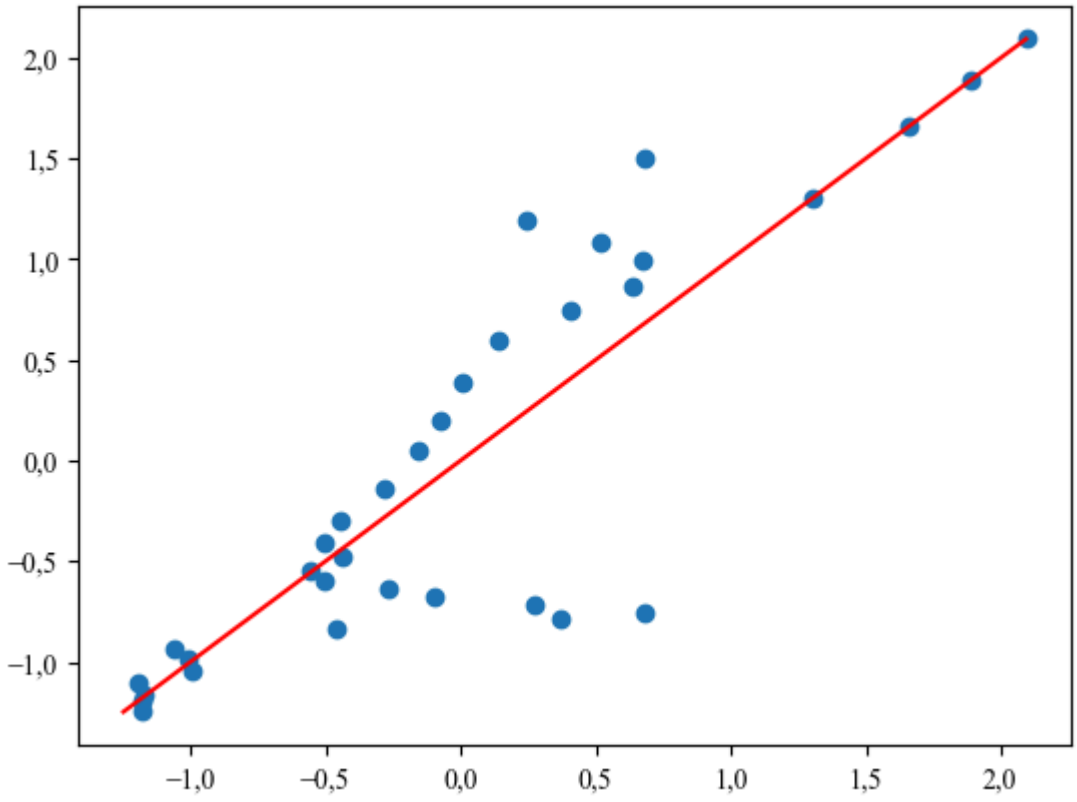
$(11,\ 0.757813850502449)$



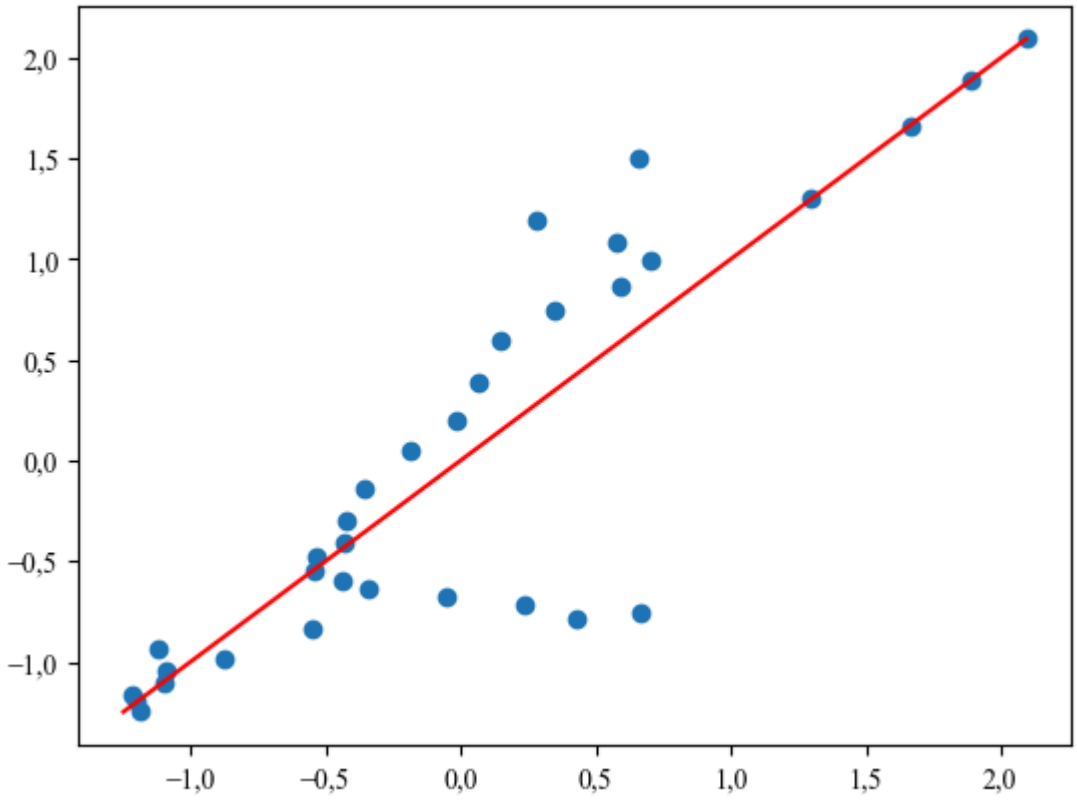
(12, 0.765680687726364)



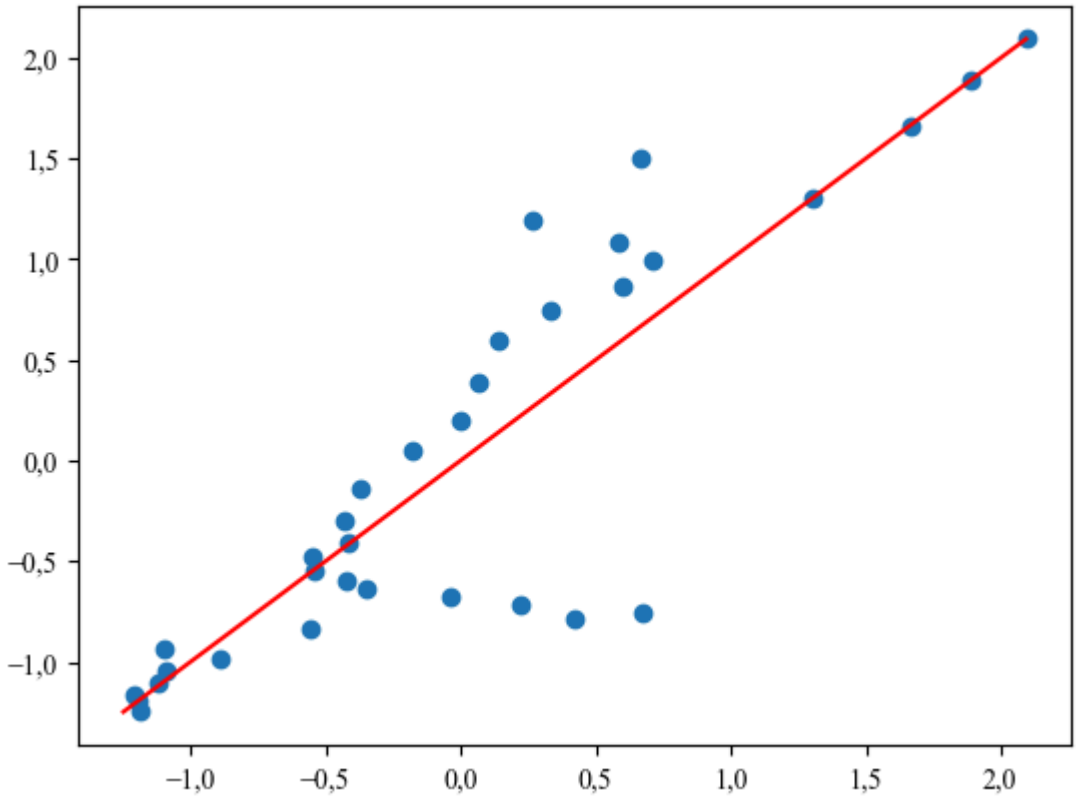
(13, 0.765989254619546)



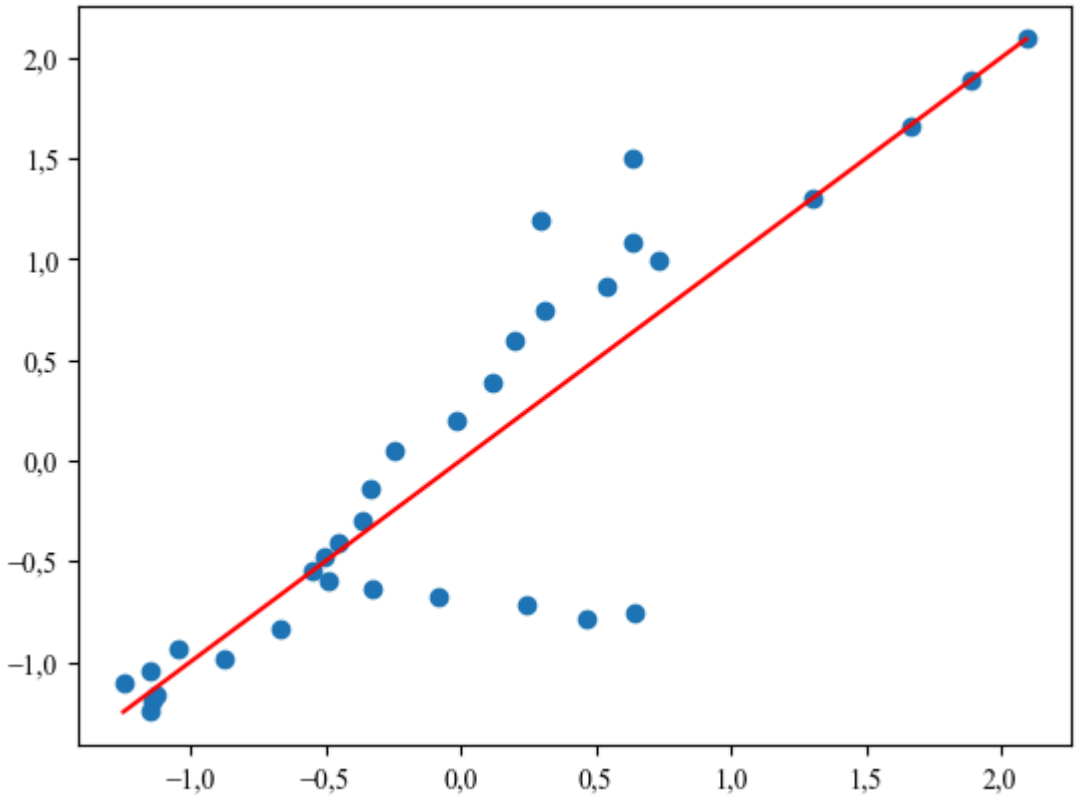
(14, 0.769235046558645)



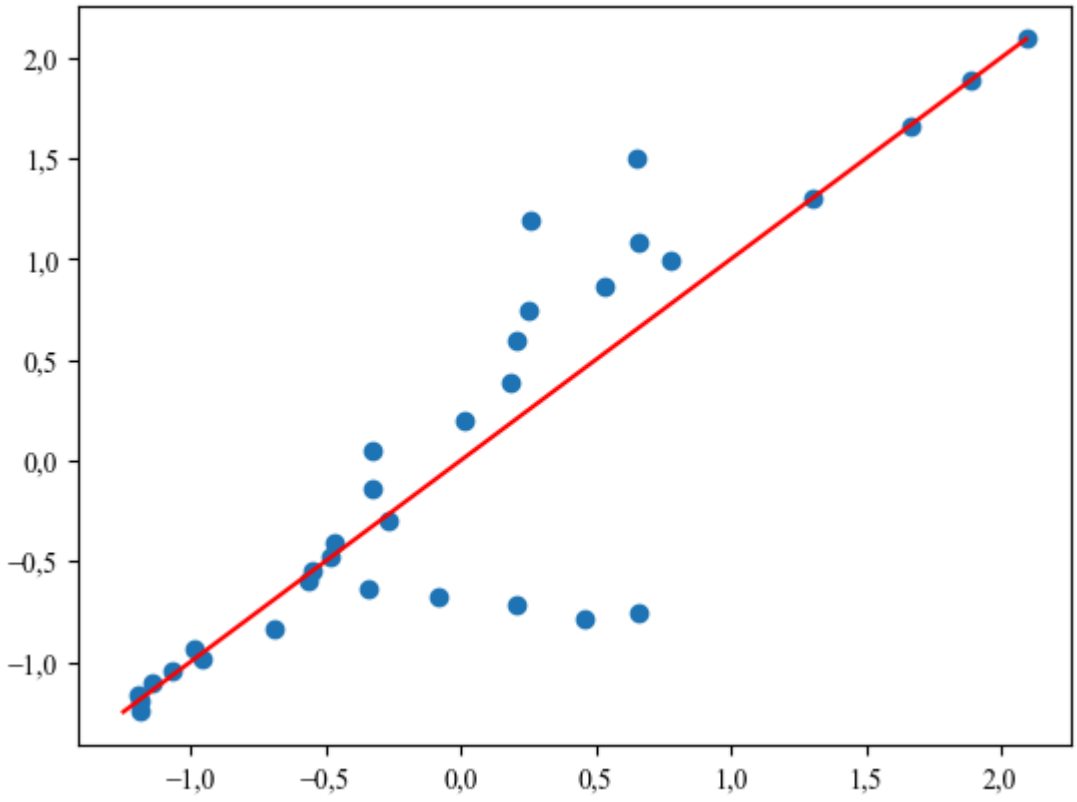
(15, 0.769335392444522)



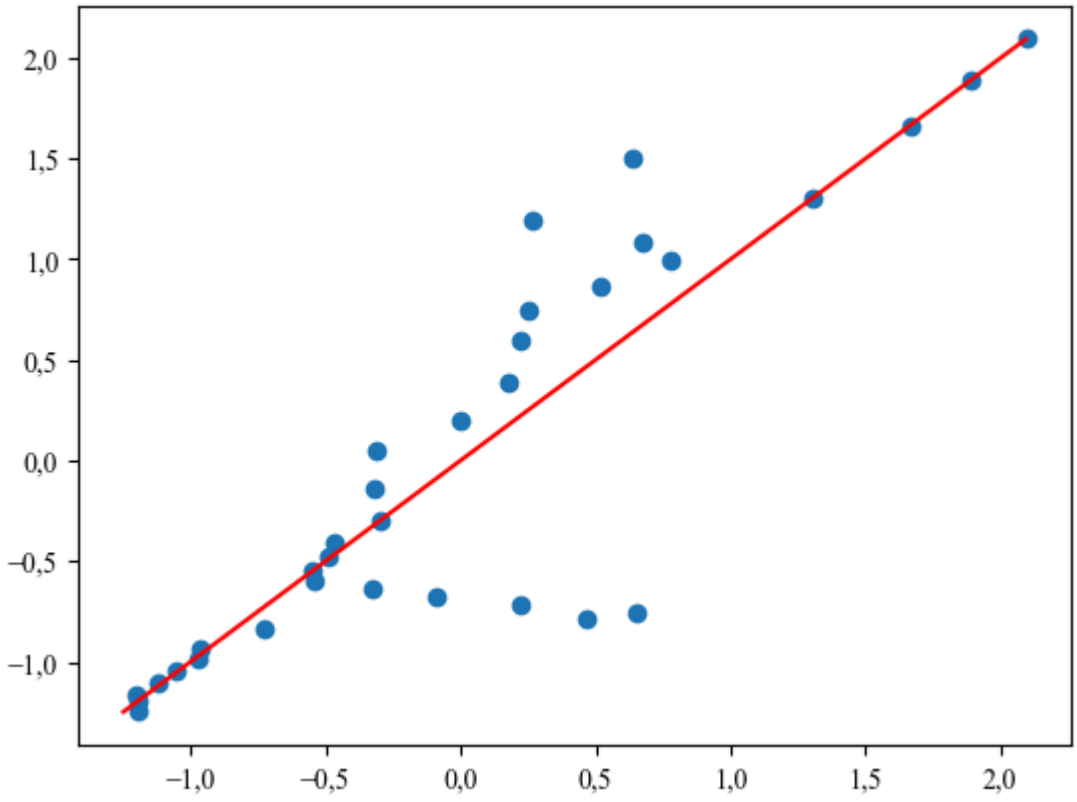
(16, 0.771915294388348)



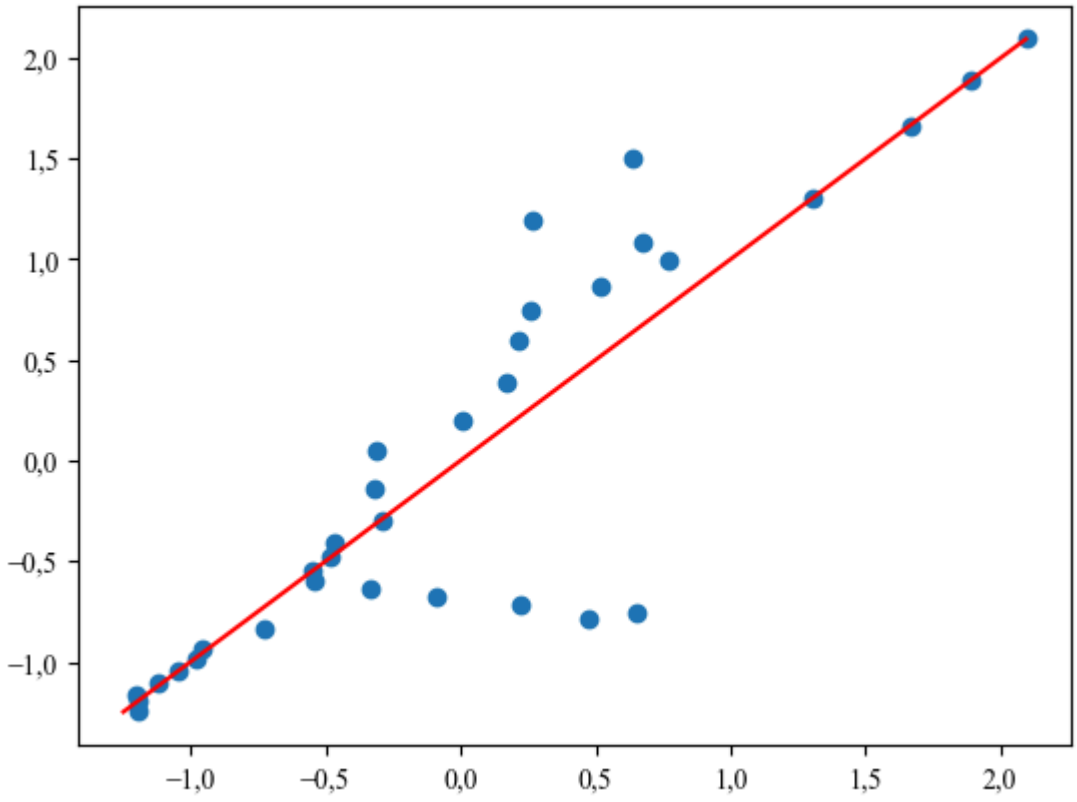
(17, 0.774083176630271)



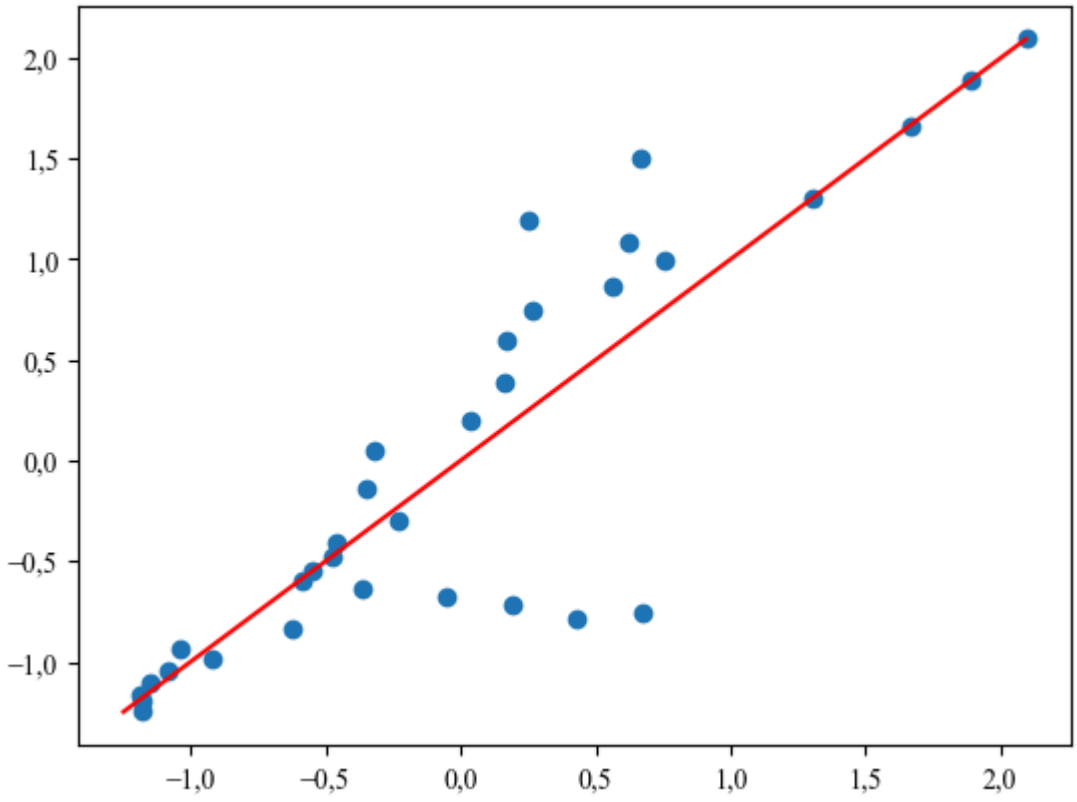
(18, 0.774269818010829)



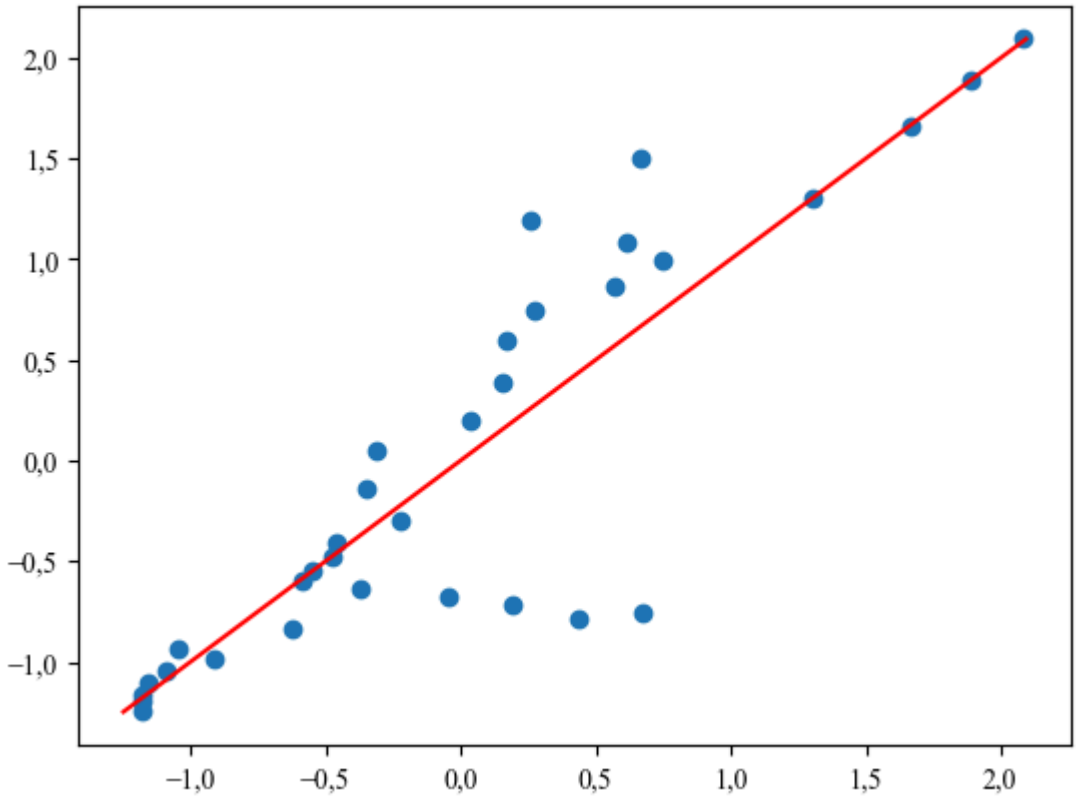
(19, 0.774321718824306)



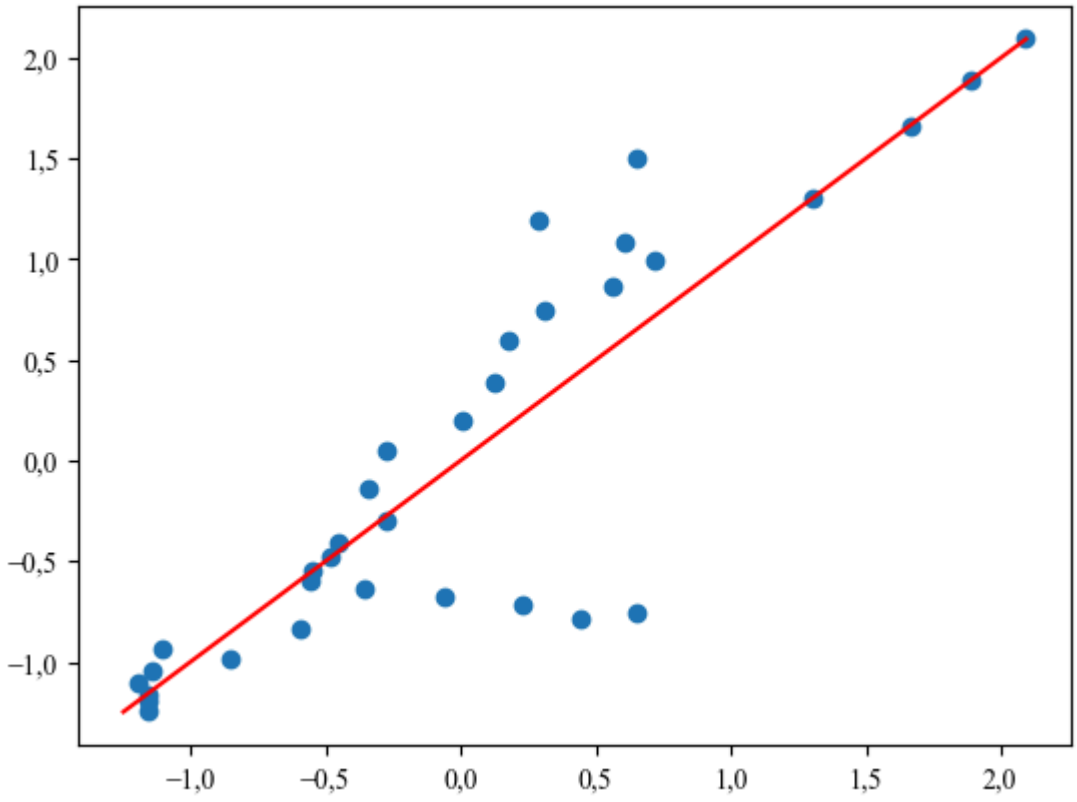
(20, 0.772967899307067)



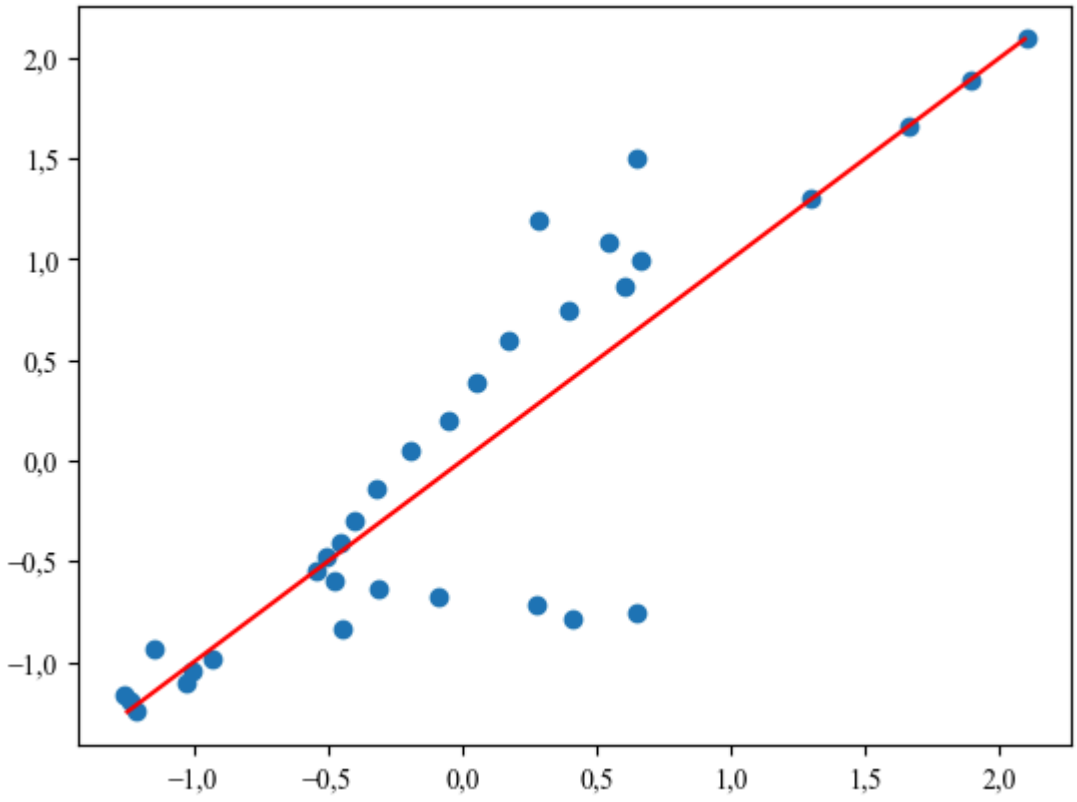
(21, 0.77281532752959)



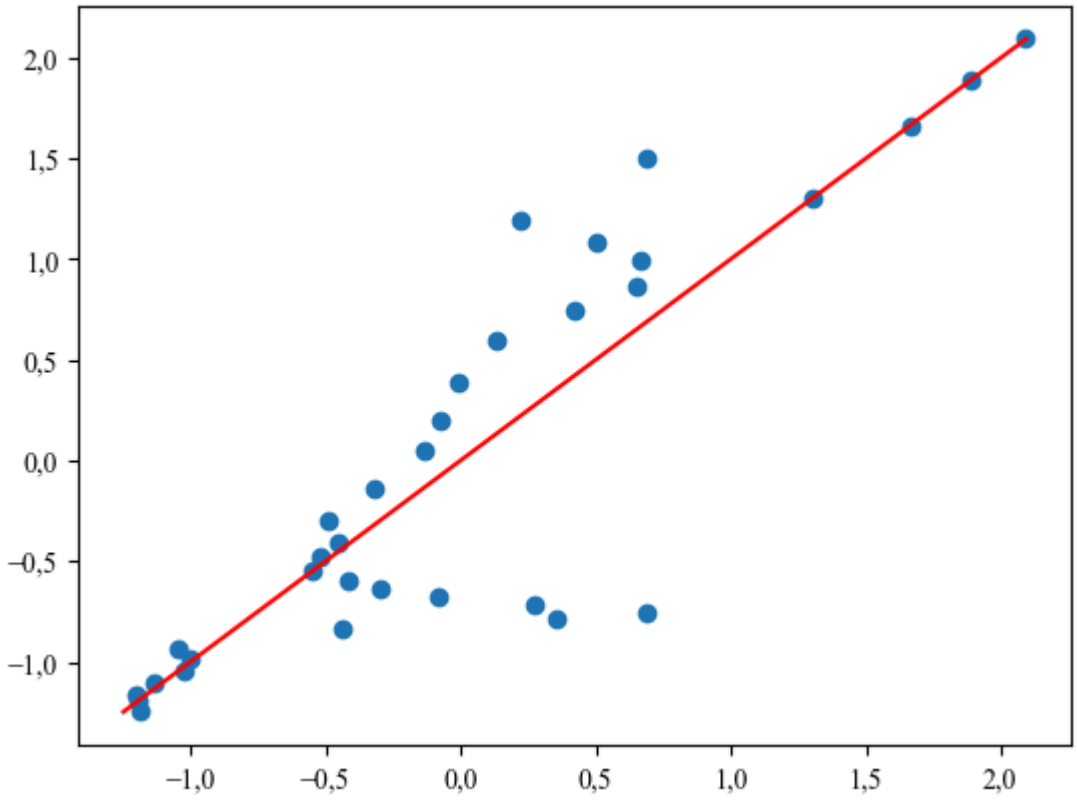
(22, 0.771540206078329)



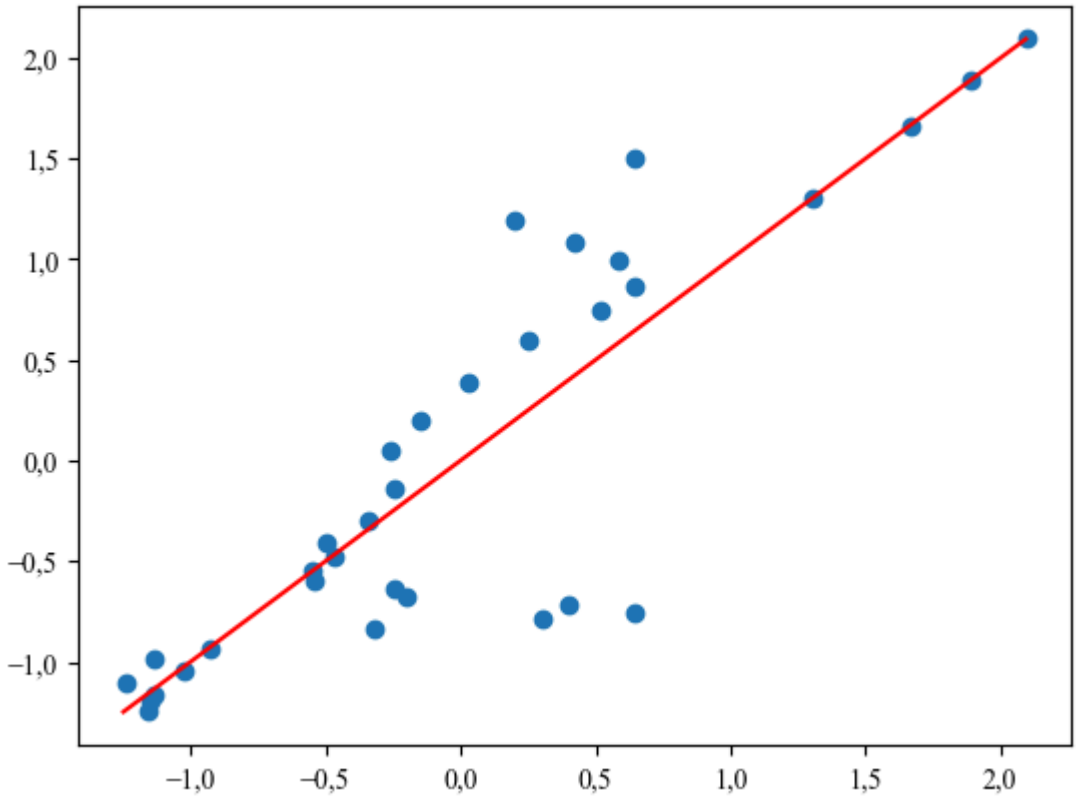
(23, 0.766909411403903)



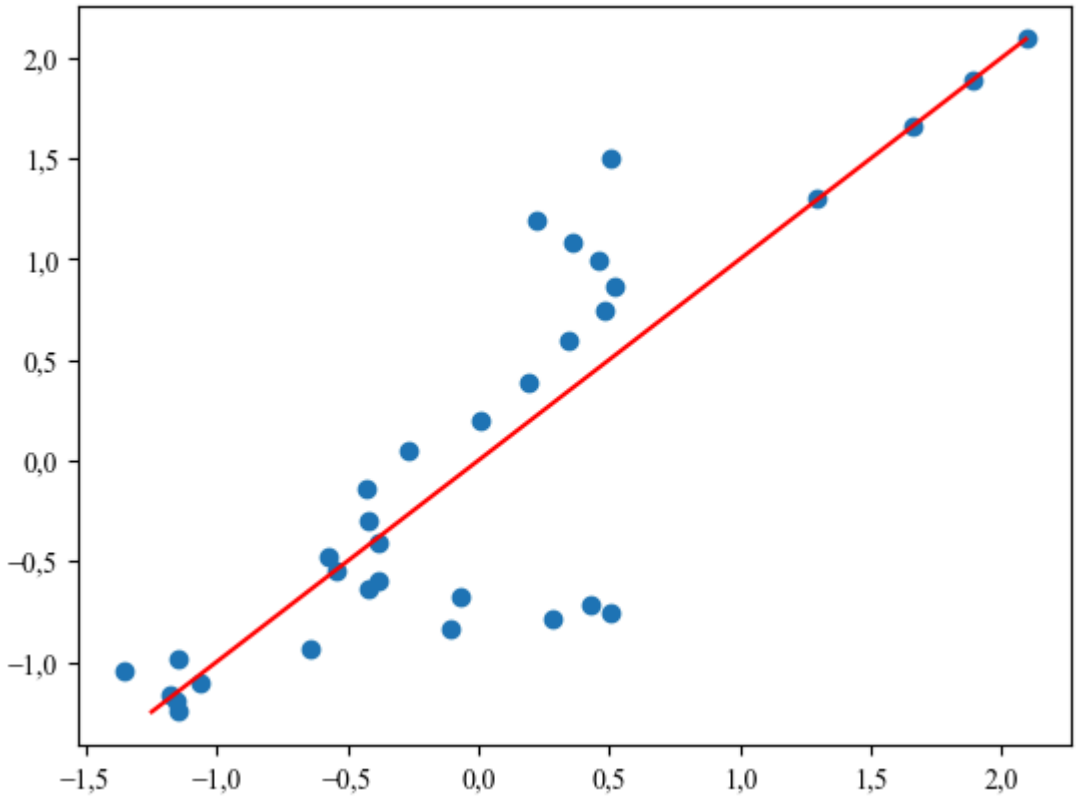
(24, 0.764305377060083)



(25, 0.758087151500497)



$(26,\ 0.741578308142467)$



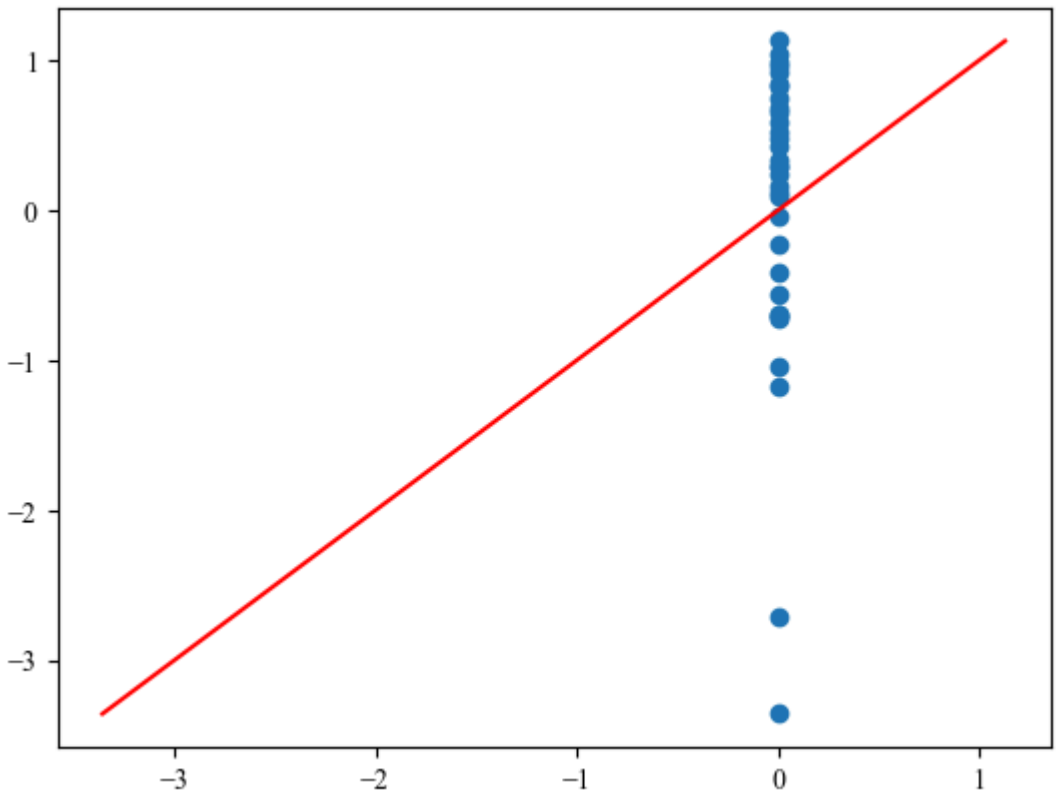
Все равно R2 мал.

Если смотреть на график распределения под углом в 90

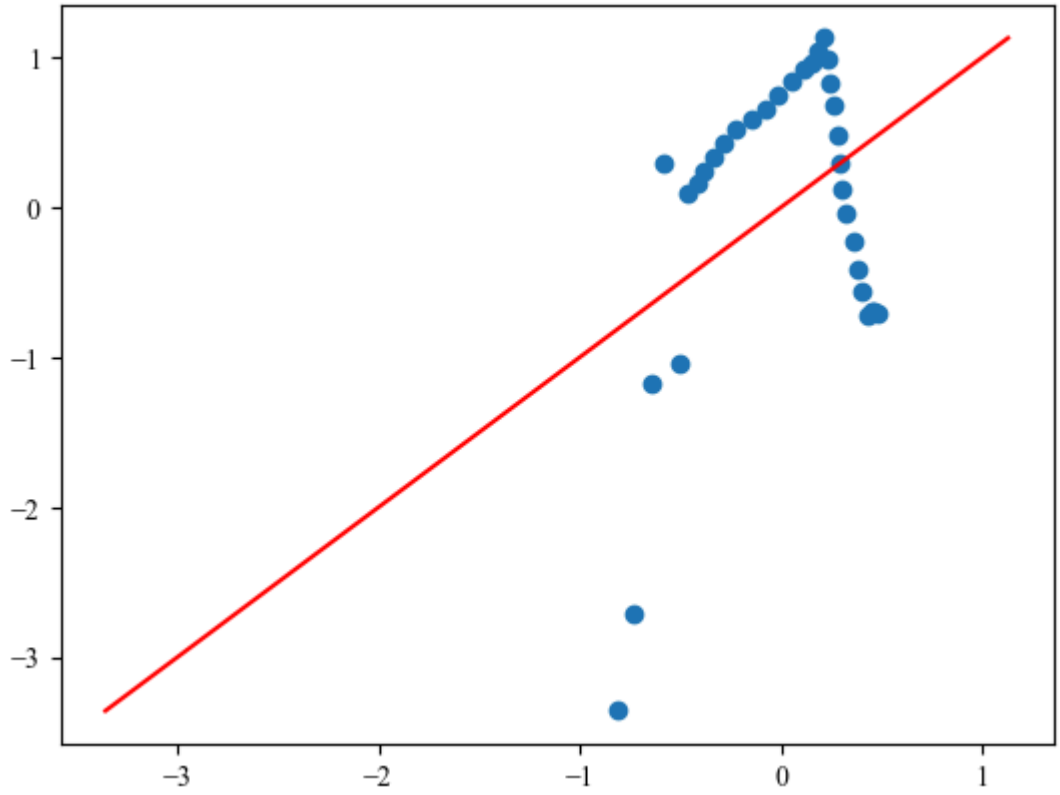
градусов, то можно предположить, что это полином степени N

In [10]:
```
degrees=range(27)


add_text(f'{country}.docx',f'Все равно R2 счет мал.\n\n Если смотреть на график

for deg in degrees:
        yp=y
        pf = PolynomialFeatures(degree=deg)
        yp = pf.fit_transform(yp)

        model = OLS(x,yp).fit()

        display((deg, model.rsquared))
        plt.scatter(model.predict(yp), x)
        plt.plot([np.min(x),np.max(x)],[np.min(x),np.max(x)],color='red')

        if deg%6.5==0:

                try:
                        plt.savefig(f'{country}_y_deg_{deg}.png')
                        add_text(f'{country}.docx',f'Степень полинома = {deg},\n
                        add_image_to_docx(f'{country}.docx',f'{country}_y_deg_{d
                except:
                        pass



        plt.show()
```

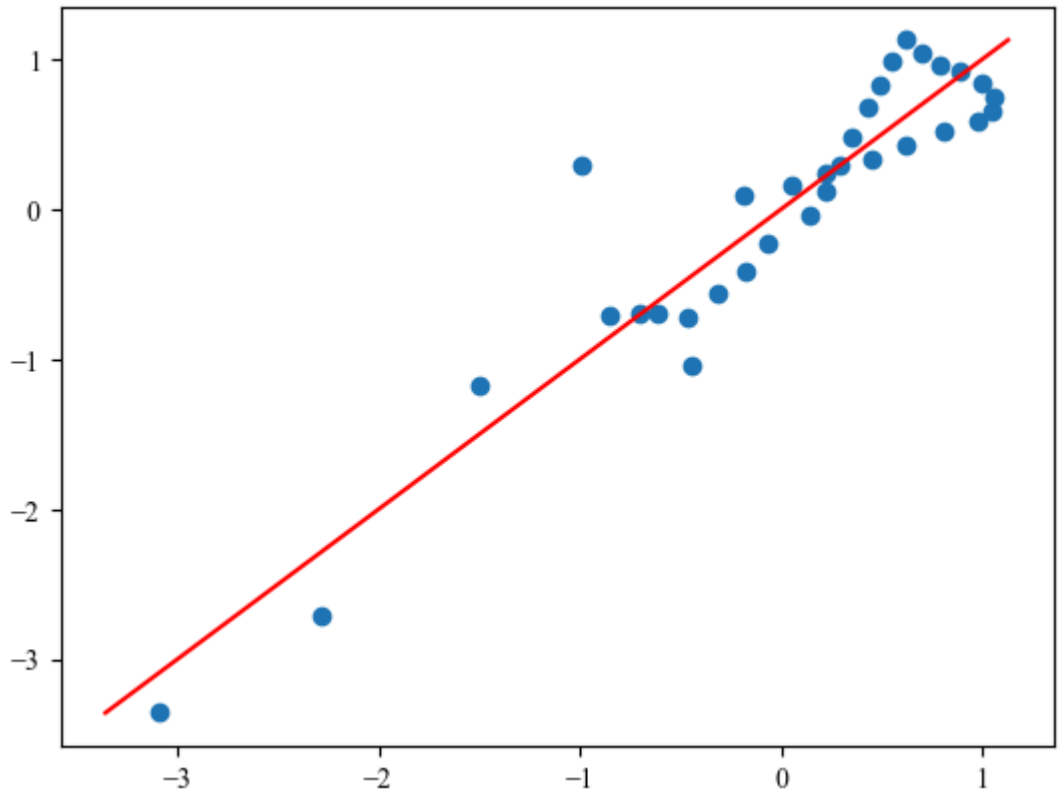
$(0,\ 0.0)$



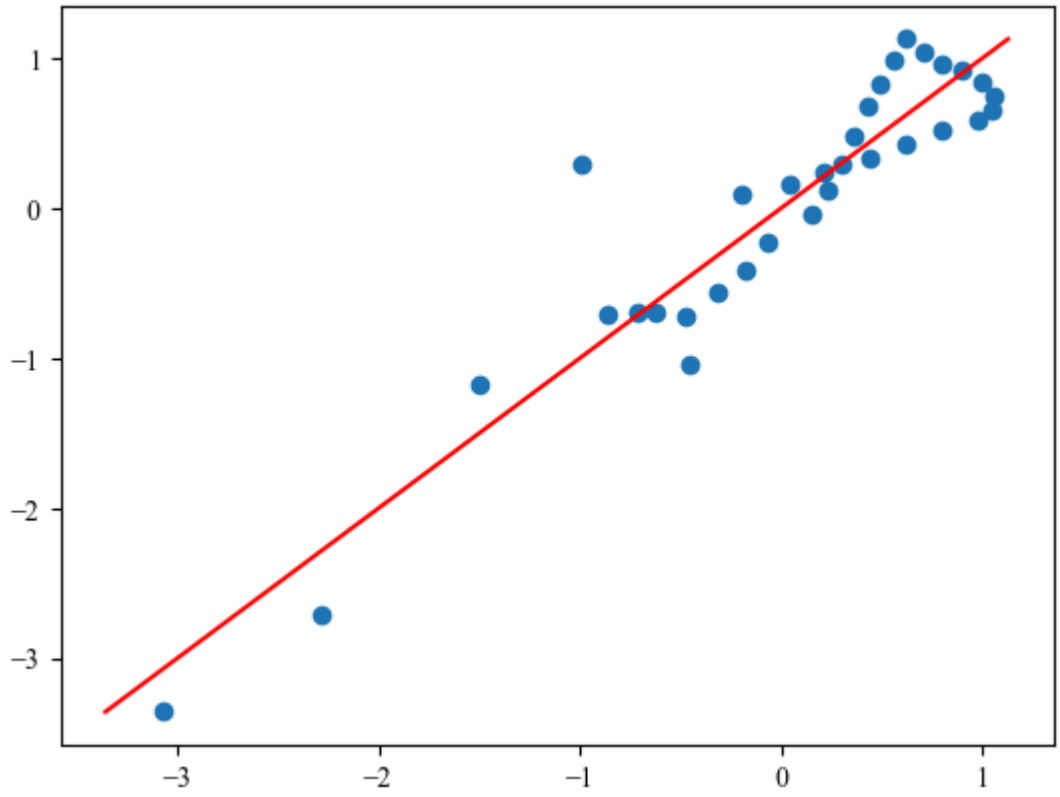
(1, 0.149425381139715)



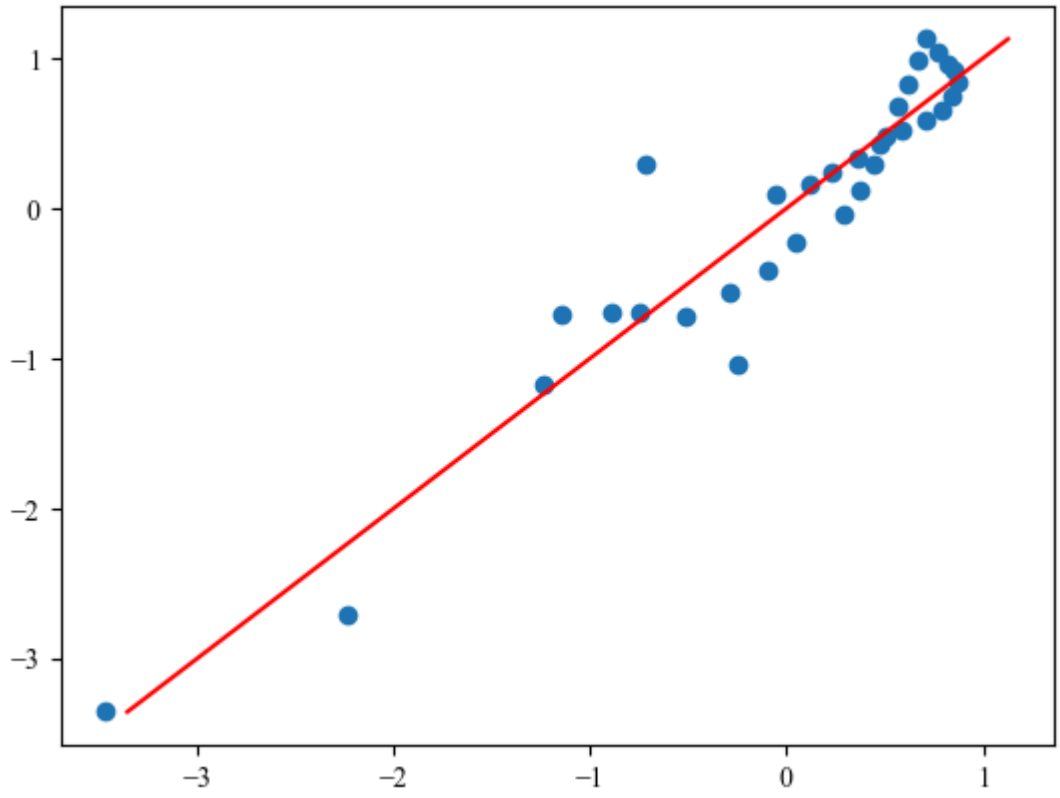
(2, 0.876476472708803)



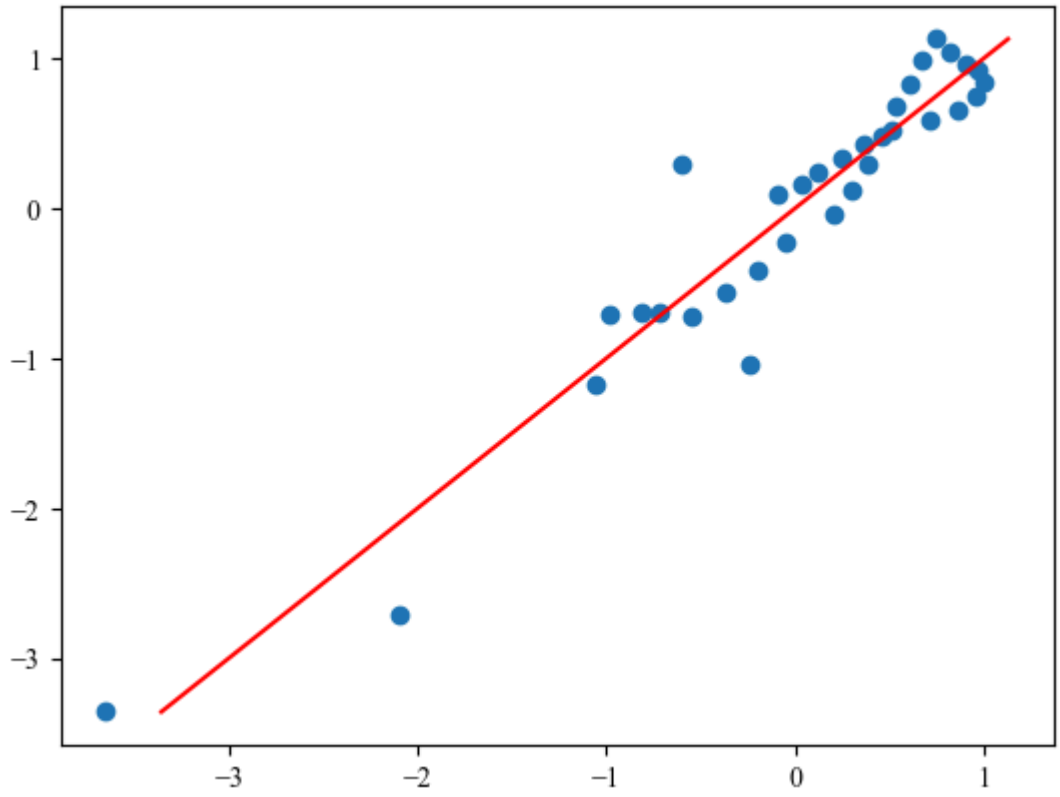
(3, 0.876533296626221)



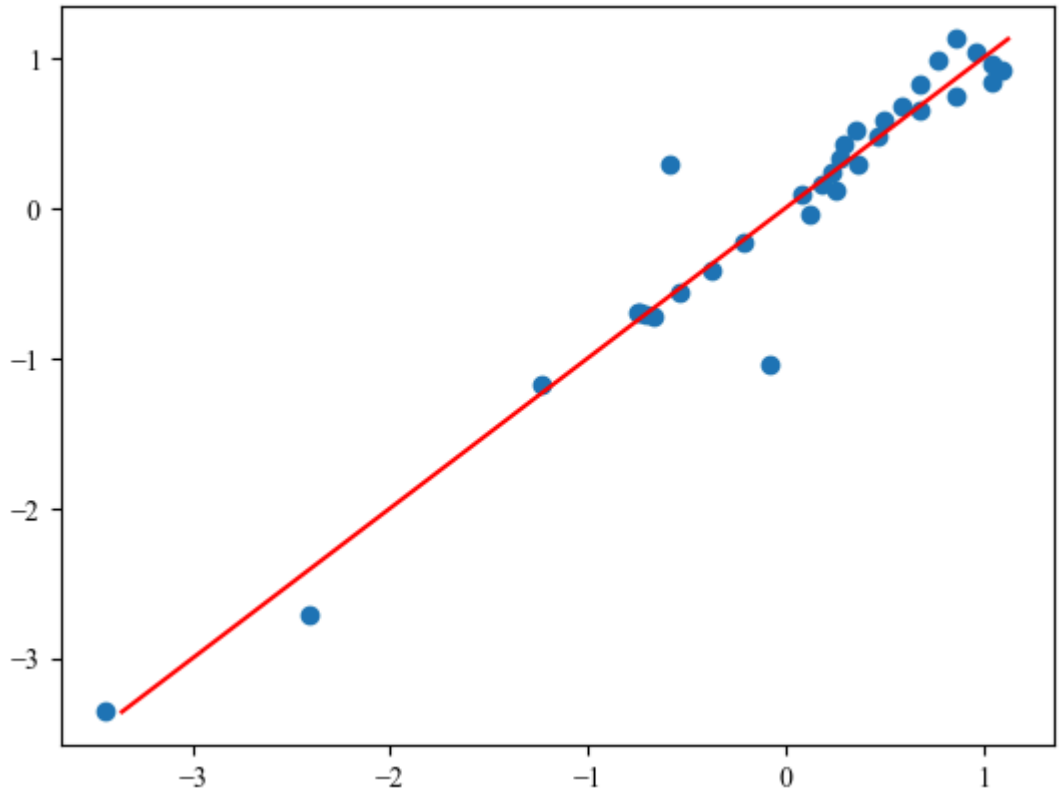
$(4,\ 0.905113104949166)$



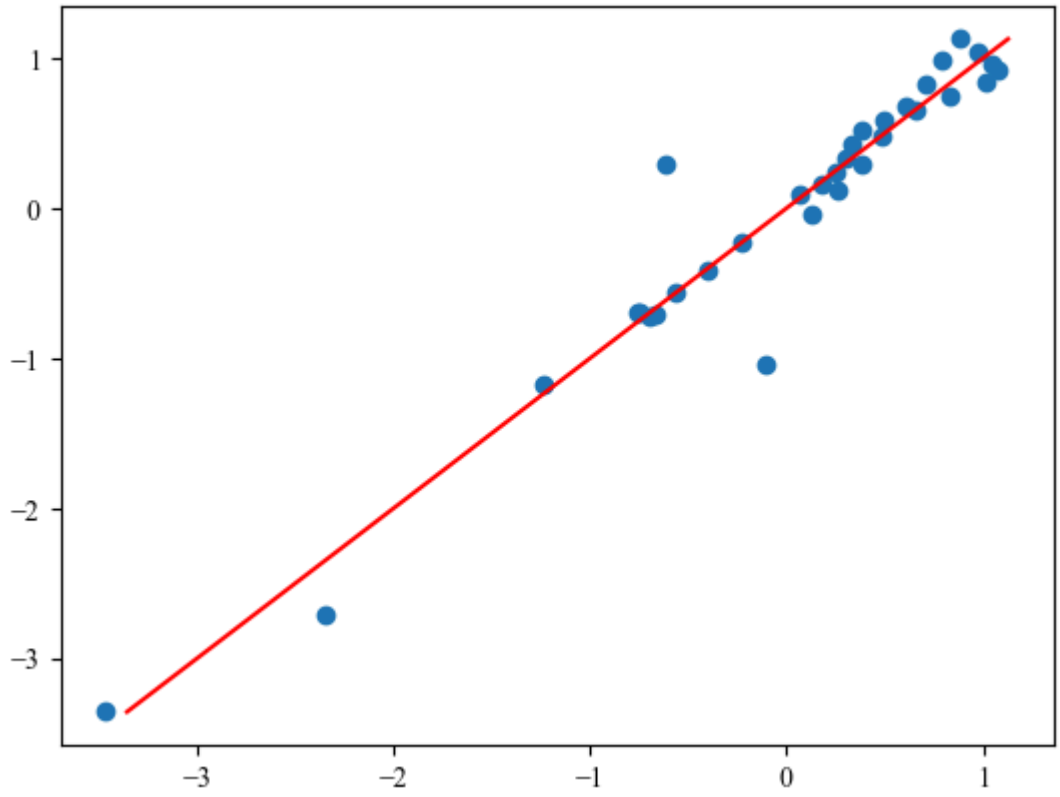
$(5,\ 0.913770989288803)$



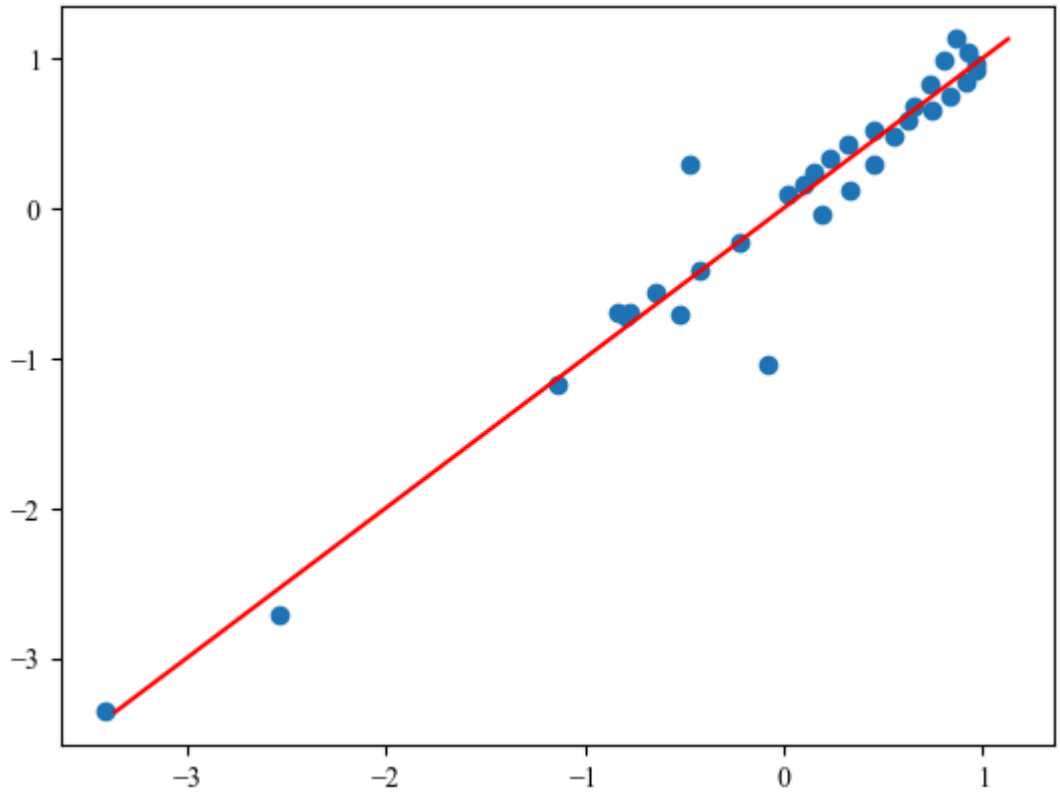
$(6,\ 0.934234879808364)$



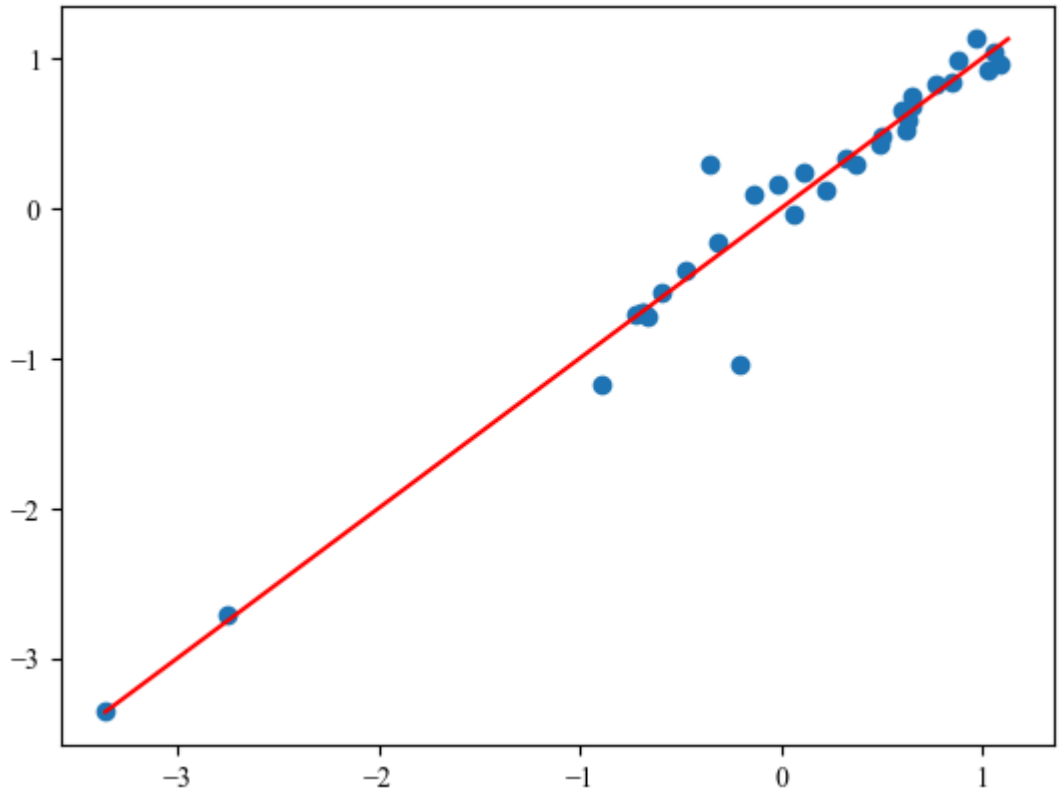
$(7,\ 0.934802372264512)$



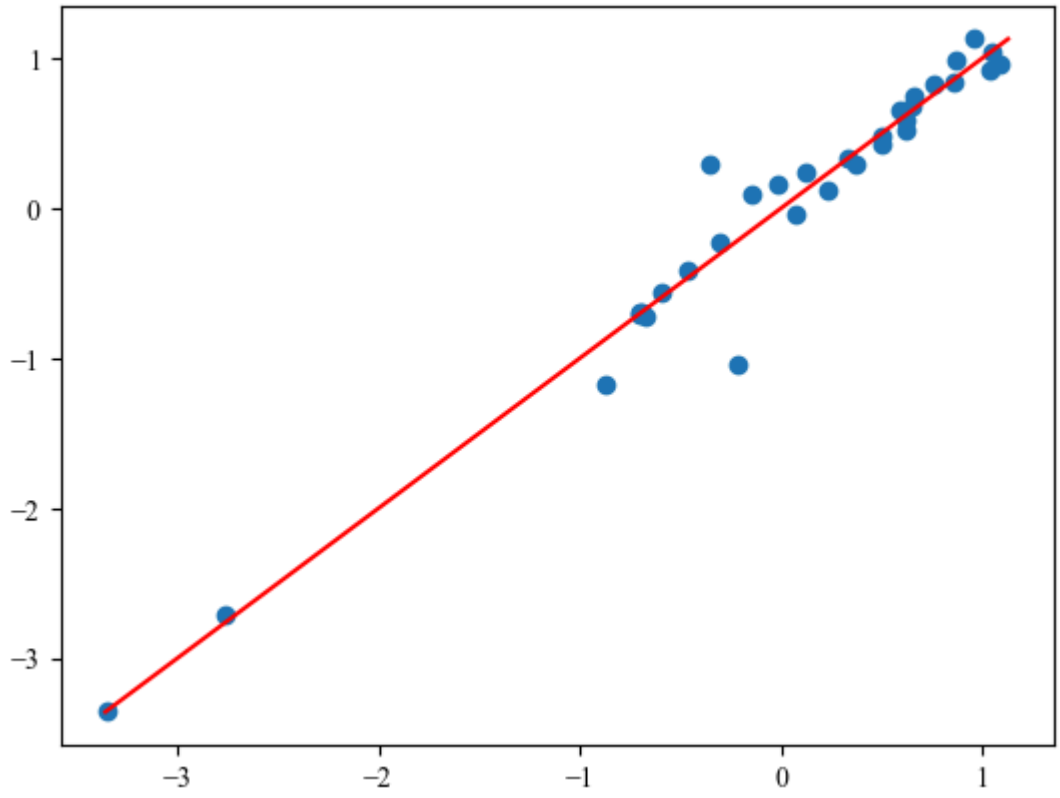
$(8,\ 0.941420372309881)$



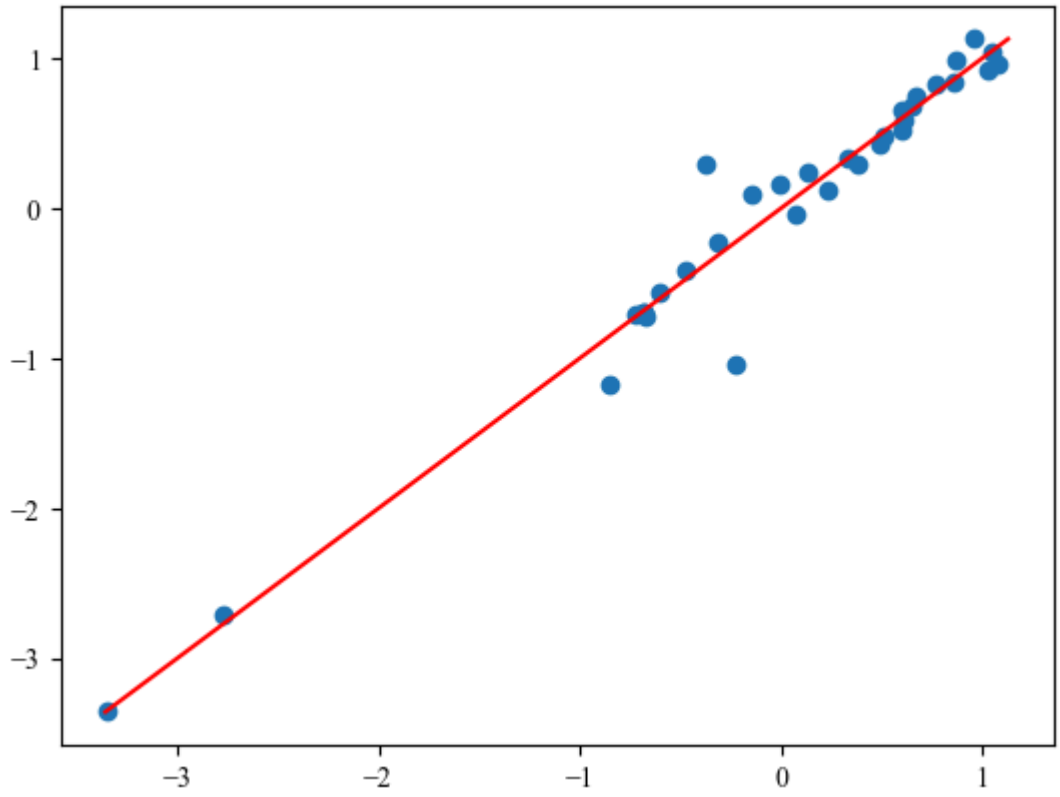
$(9,\ 0.956105752402458)$



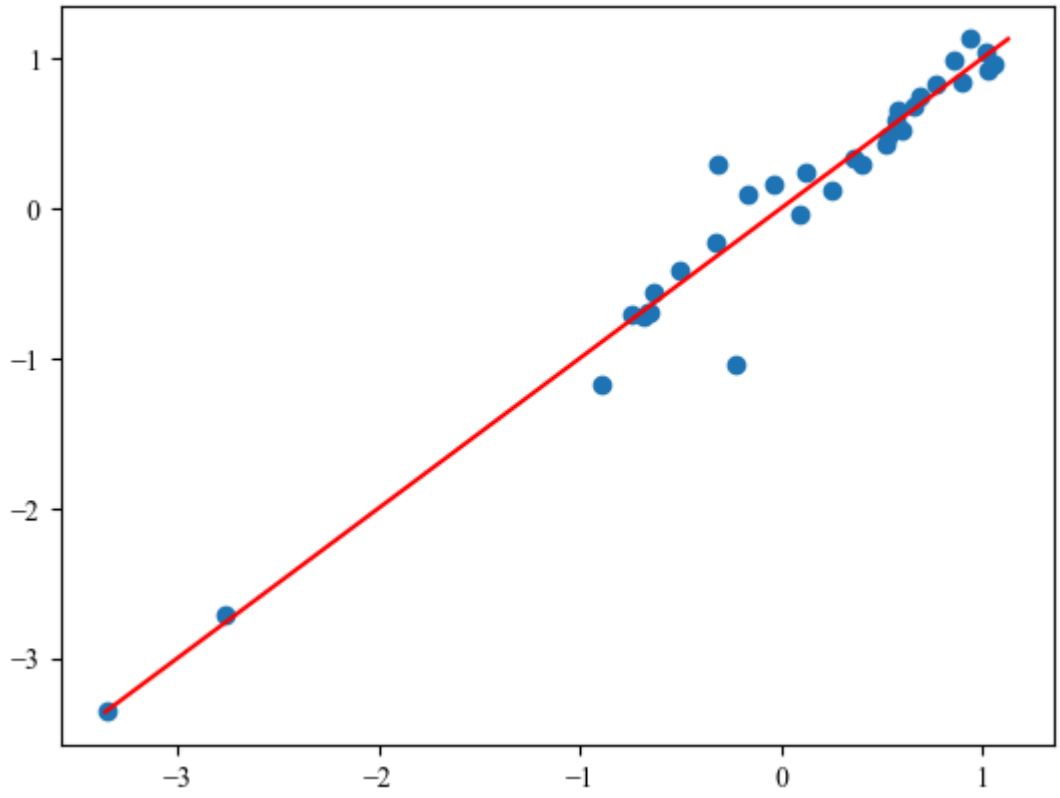
$(10,\ 0.956178078377025)$



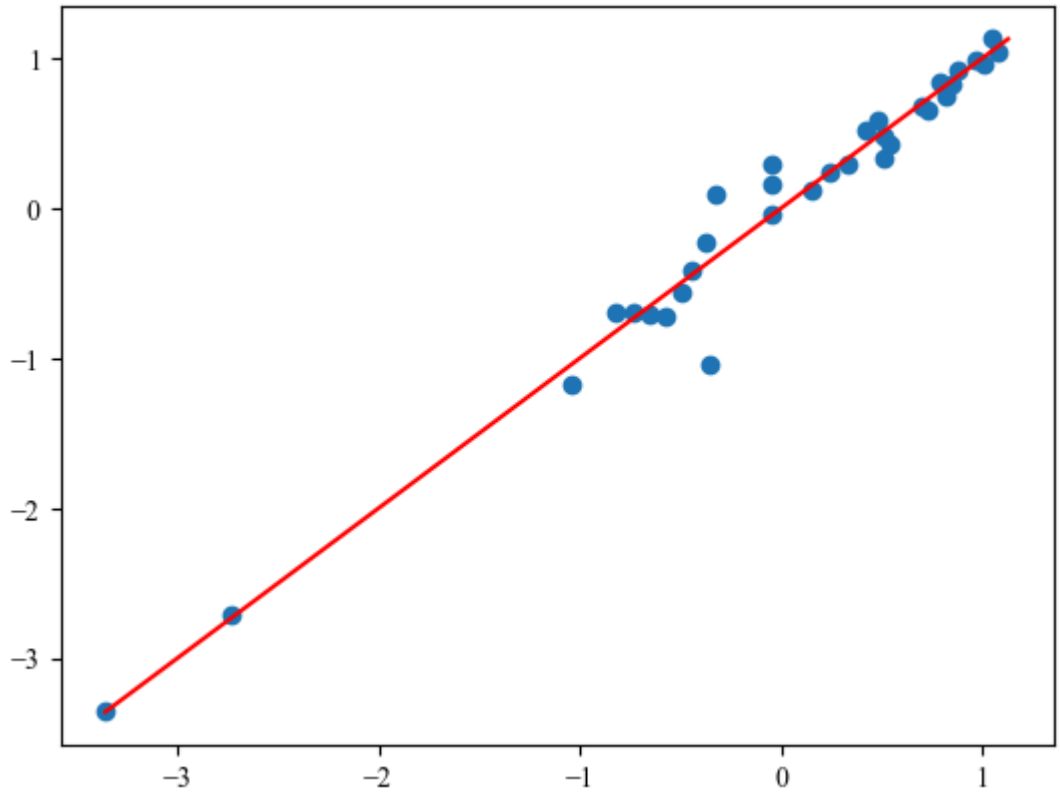
$(11,\ 0.956267777129556)$



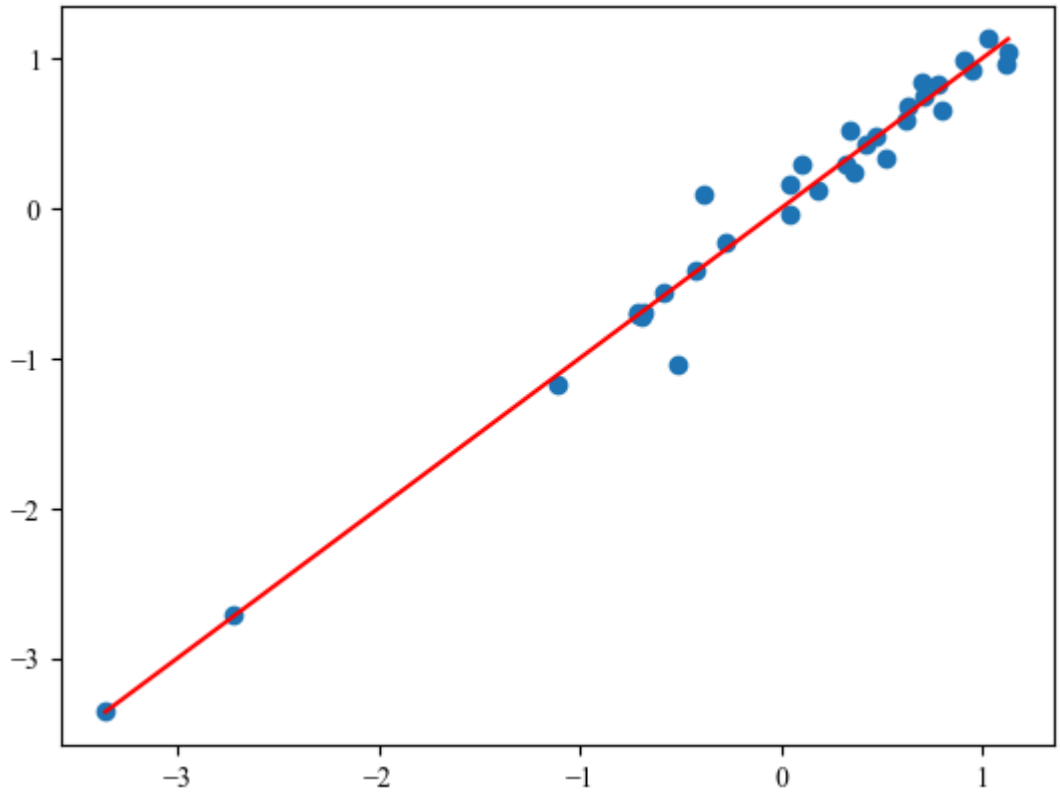
(12, 0.956897423044805)



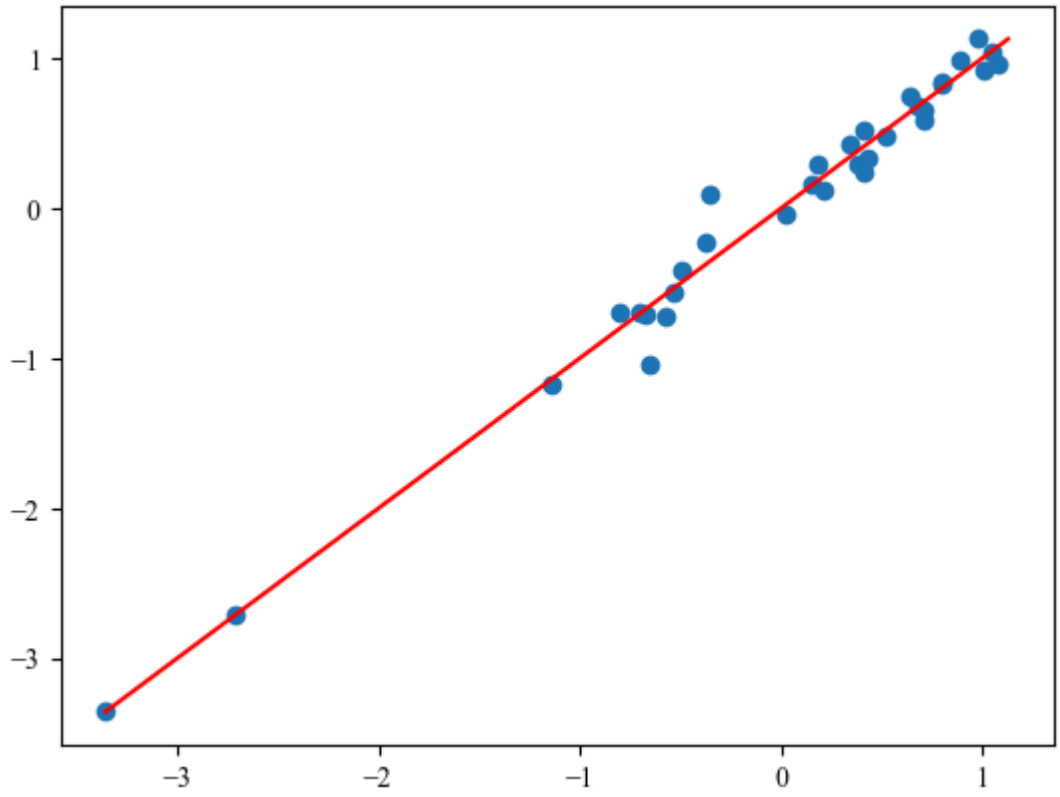
(13, 0.969899049060189)



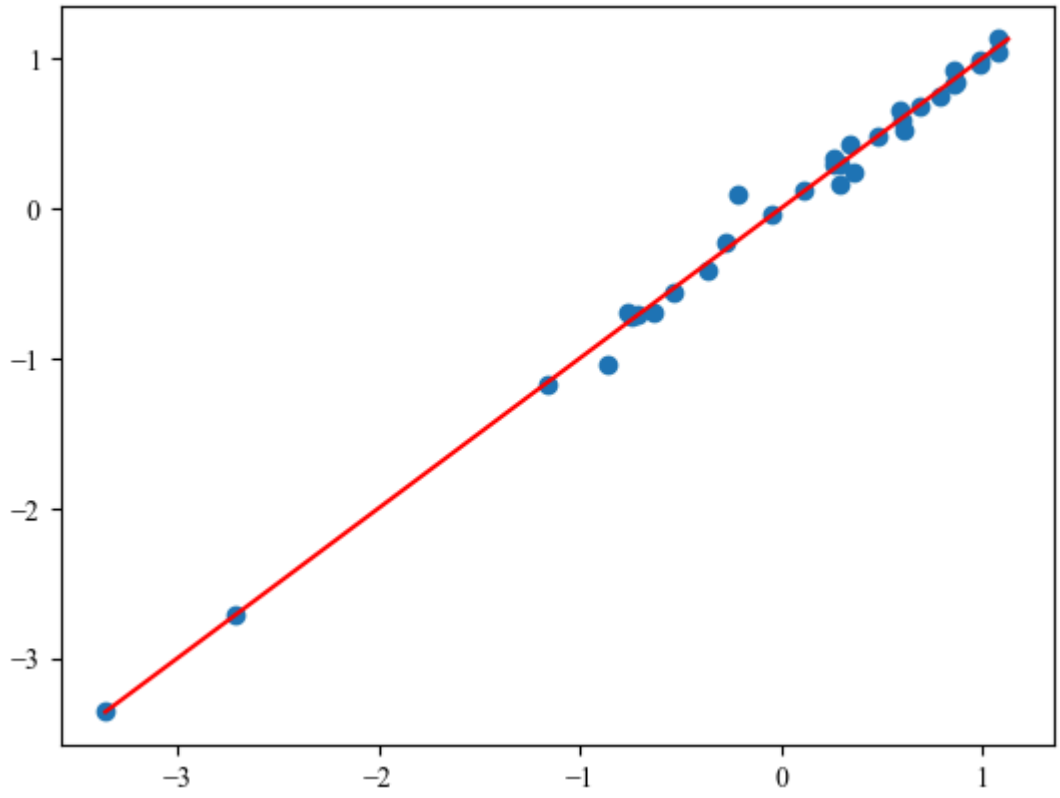
$(14,\ 0.977225579585309)$



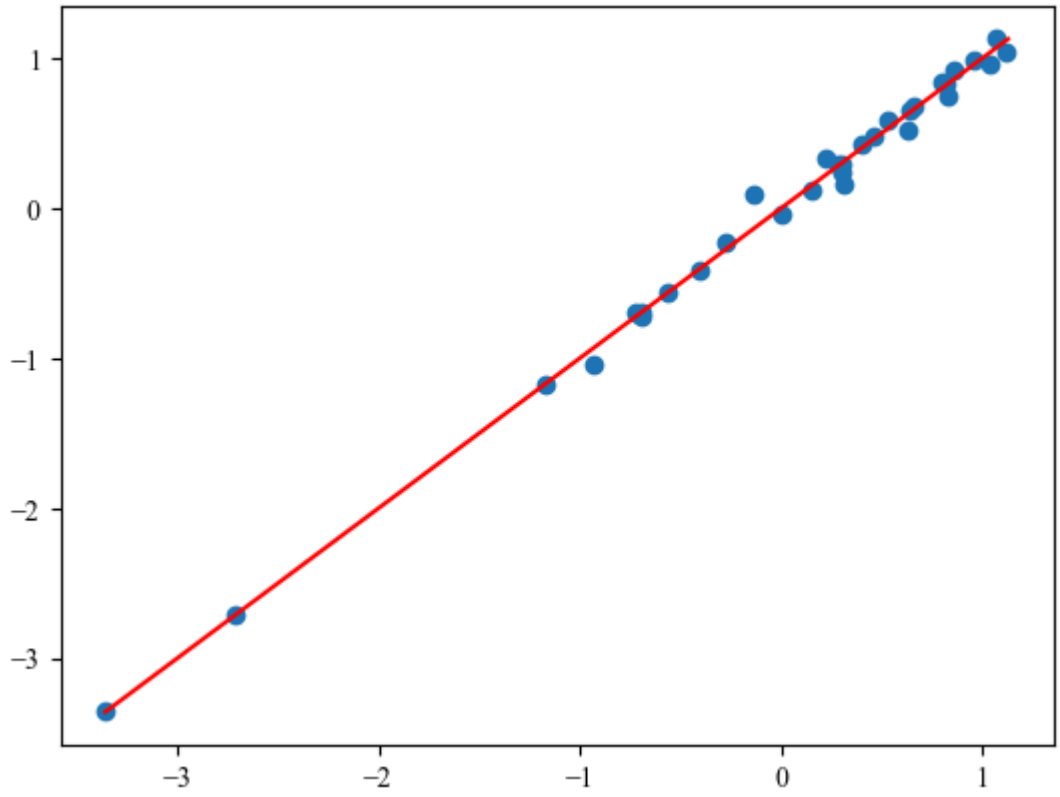
$(15,\ 0.982104278775743)$



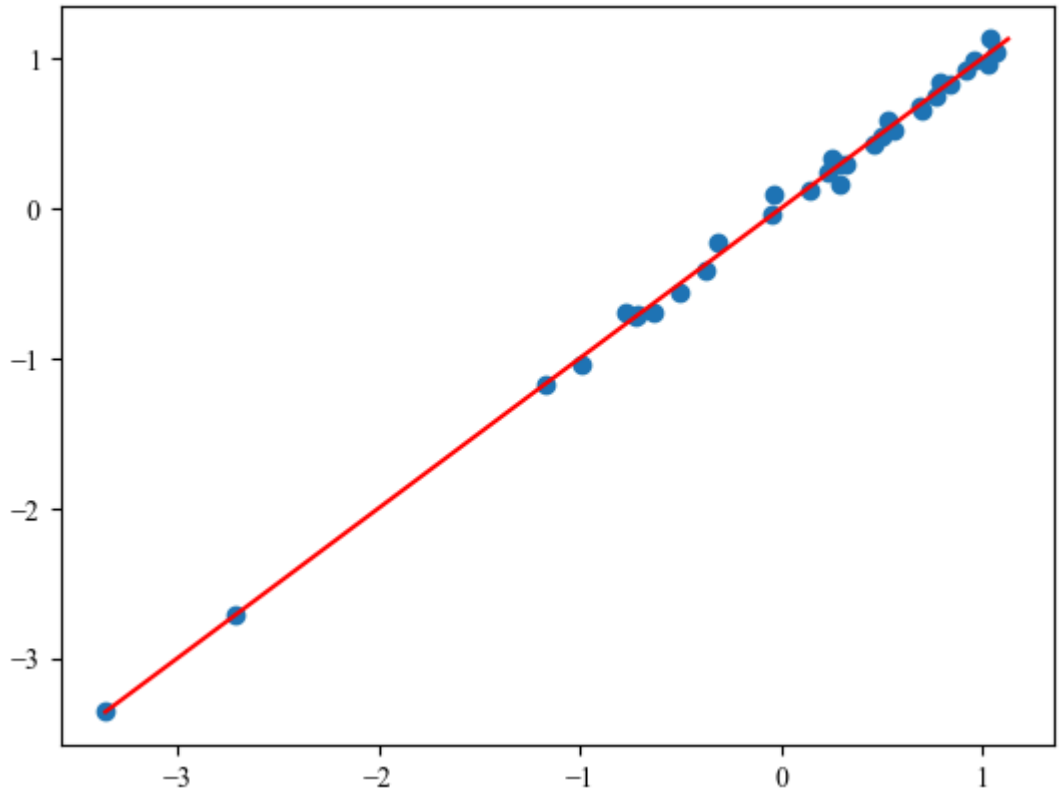
$(16,\ 0.99364633885899)$



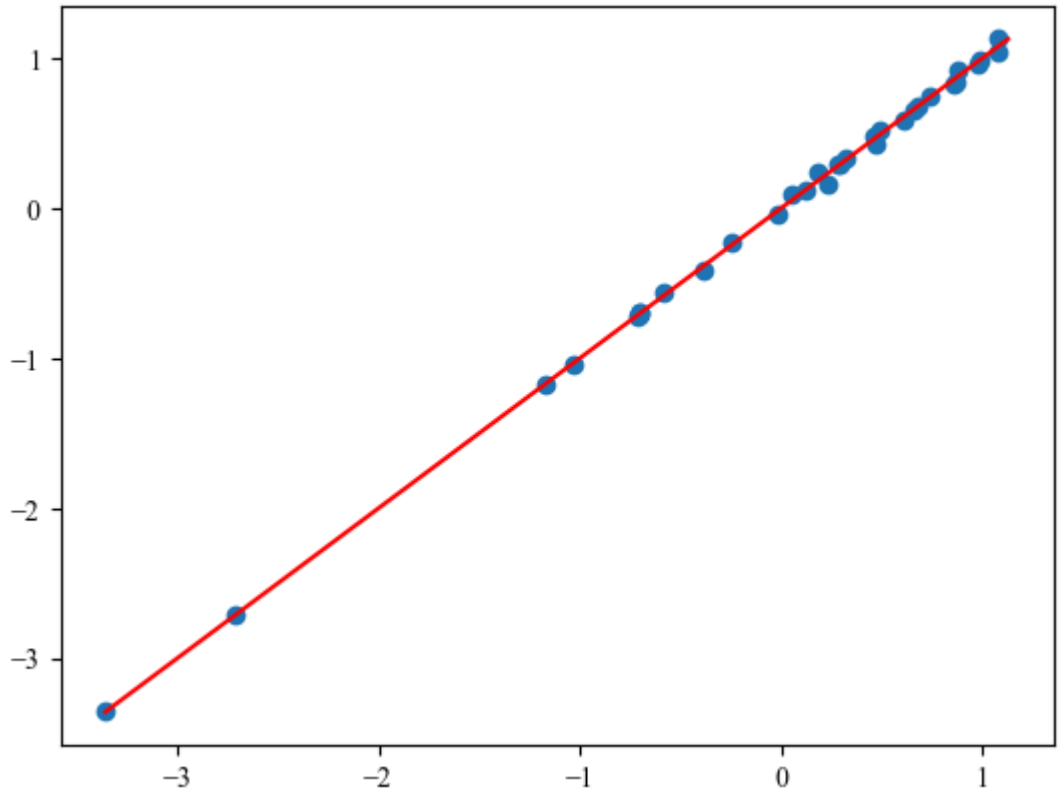
$(17,\ 0.99542351404418)$



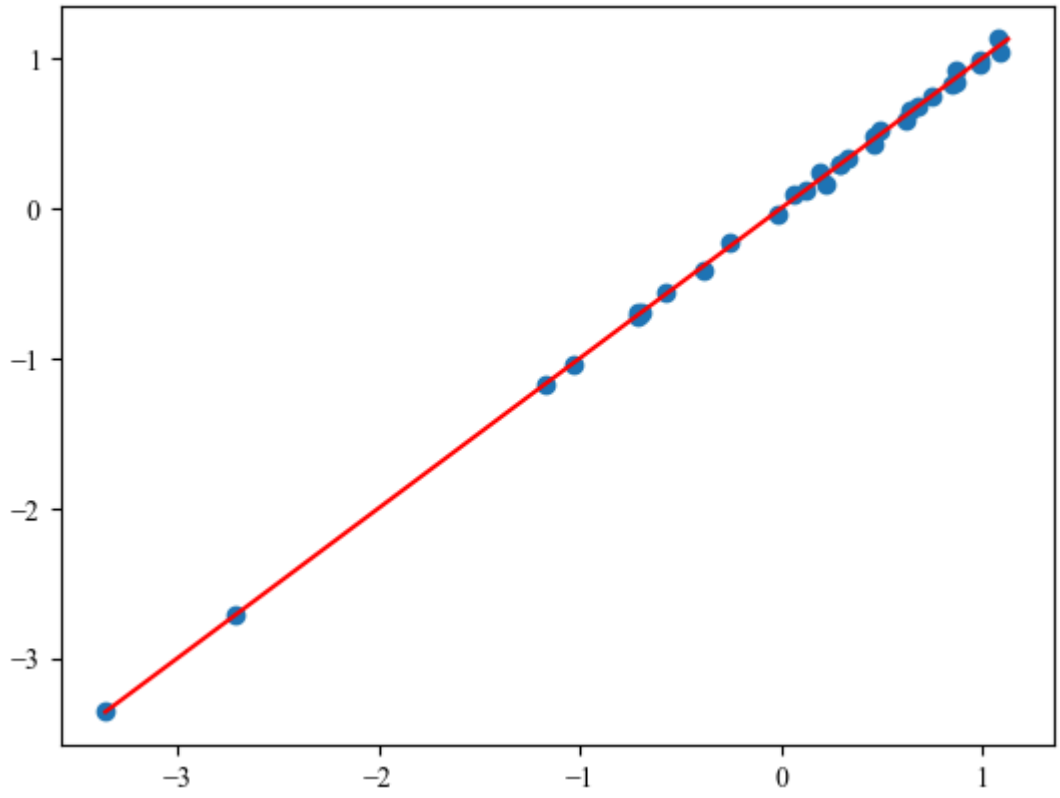
(18, 0.99721409426392)



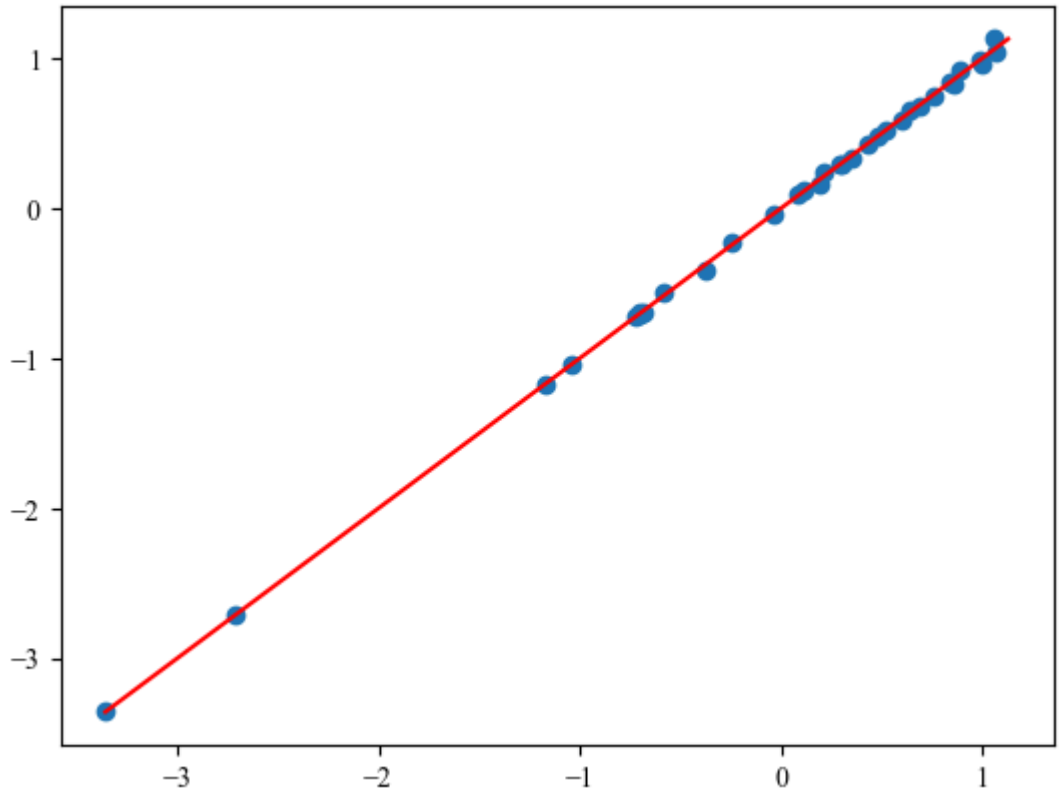
(19, 0.999332621602501)



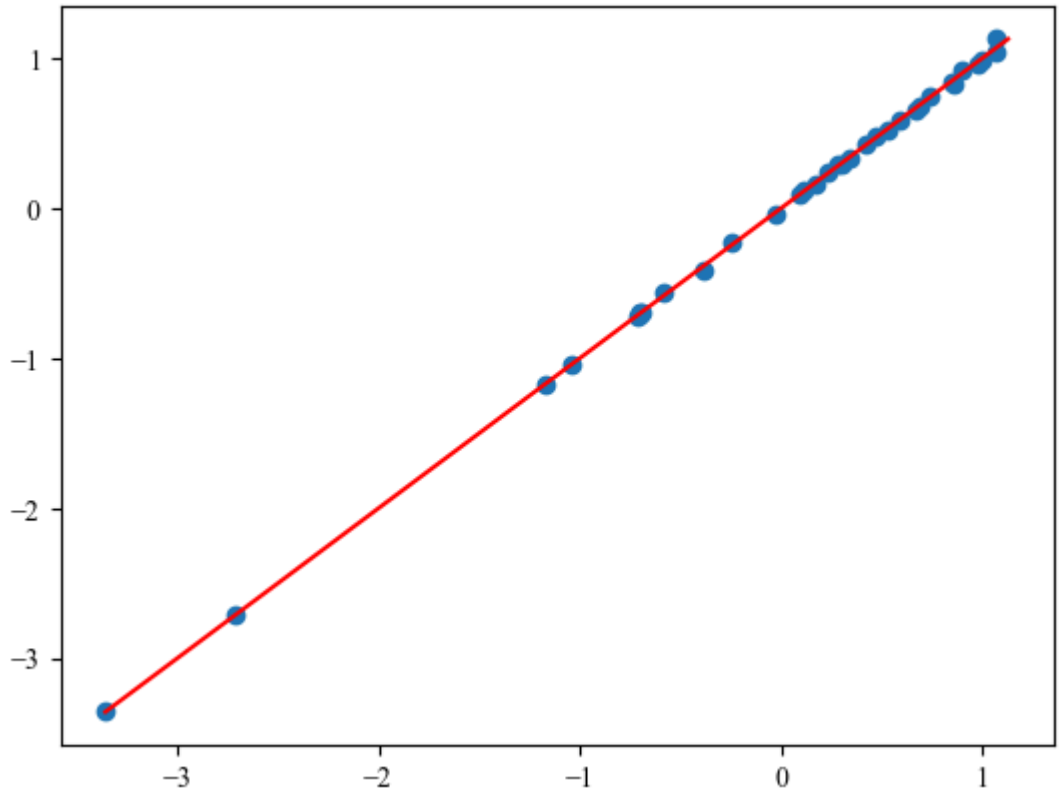
$(20,\ 0.999377864687939)$



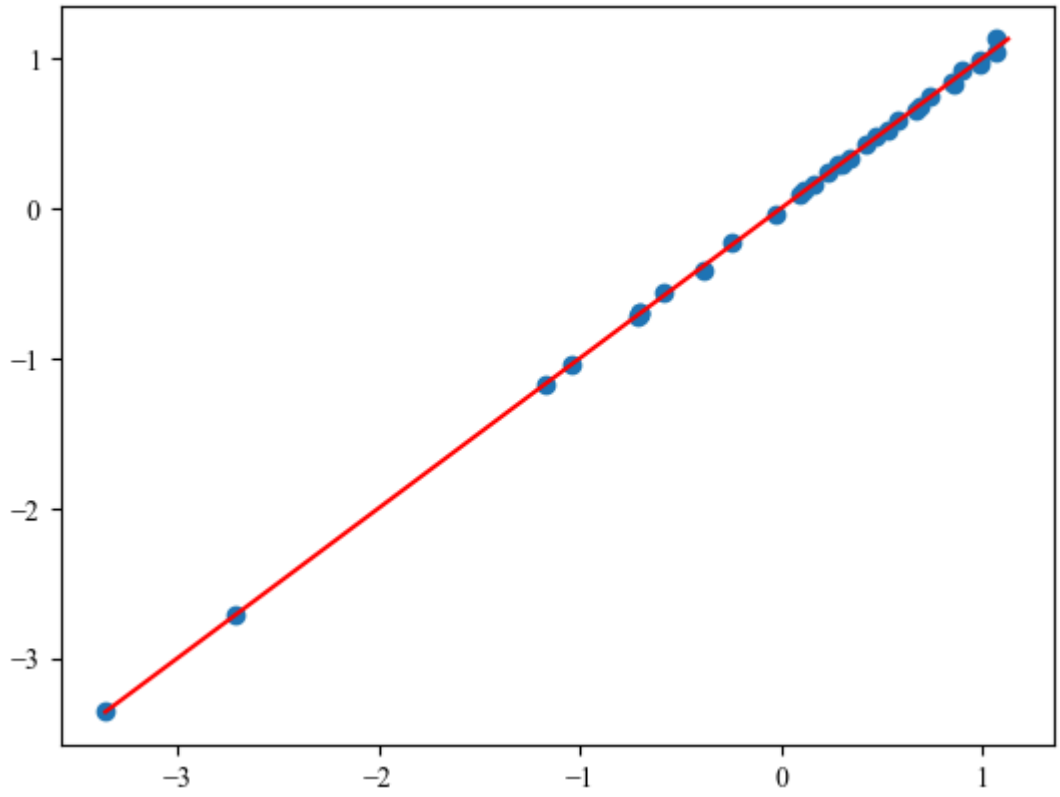
$(21,\ 0.999620790163873)$



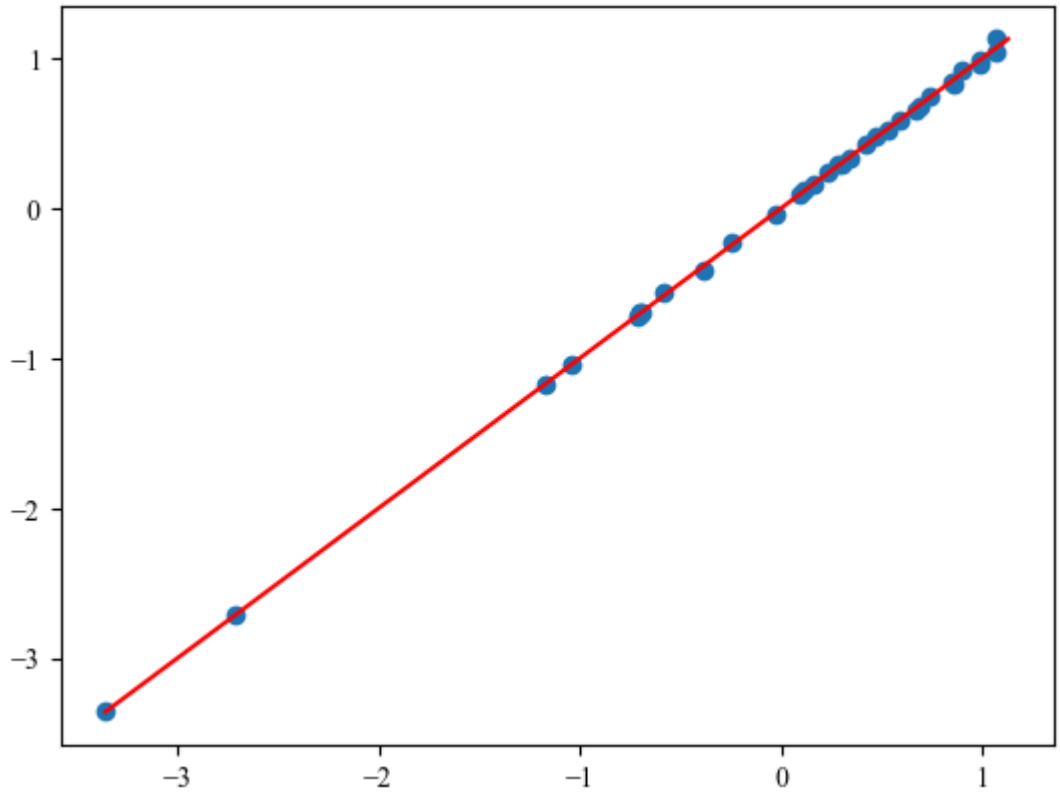
$(22,\ 0.999740505545667)$



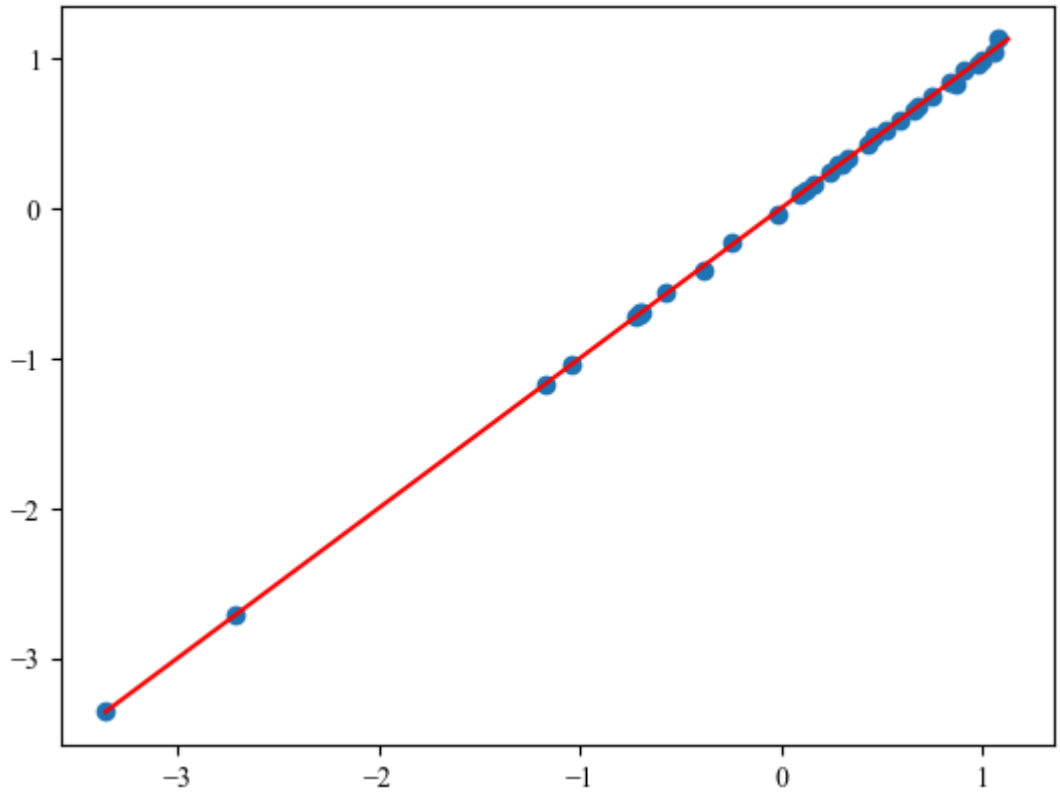
$(23,\ 0.999742652261439)$



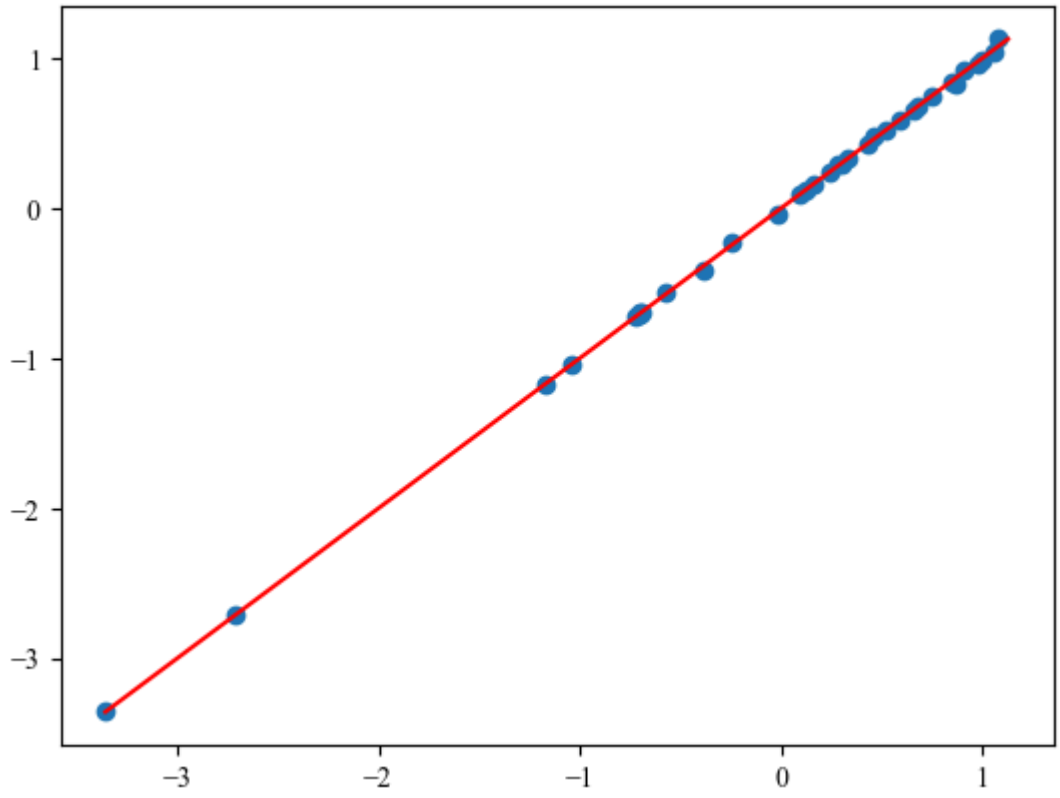
$(24,\ 0.999743909971418)$



$(25,\ 0.999785028662359)$



$(26,\ 0.999780516437845)$



Мы нашли модель со `R2 = 0.999780516437845`

In [11]:
```
add_text(f'{country}.docx', f'Мы нашли модель с R2 = {model.rsquared}')
model.summary()
```

Out[11]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	1.000
Model:	OLS	**Adj. R-squared:**	0.999
Method:	Least Squares	**F-statistic:**	1518.
Date:	Sun, 03 Nov 2024	**Prob (F-statistic):**	3.16e-12
Time:	20:46:23	**Log-Likelihood:**	92.175
No. Observations:	33	**AIC:**	-134.3
Df Residuals:	8	**BIC:**	-96.94
Df Model:	24		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.7705	0.031	25.120	0.000	0.700	0.841
x1	-0.4420	0.329	-1.342	0.216	-1.201	0.317
x2	0.1699	1.905	0.089	0.931	-4.224	4.563
x3	-5.9797	13.087	-0.457	0.660	-36.157	24.198
x4	-9.8588	32.131	-0.307	0.767	-83.952	64.235
x5	123.7688	178.368	0.694	0.507	-287.548	535.086
x6	89.4388	242.660	0.369	0.722	-470.135	649.013
x7	-1057.9965	1181.311	-0.896	0.397	-3782.104	1666.111
x8	-107.1445	1054.086	-0.102	0.922	-2537.871	2323.582
x9	4651.3676	4471.359	1.040	0.329	-5659.606	1.5e+04
x10	-1309.6939	3149.427	-0.416	0.688	-8572.285	5952.897
x11	-1.17e+04	1.05e+04	-1.118	0.296	-3.59e+04	1.24e+04
x12	6216.7514	7119.985	0.873	0.408	-1.02e+04	2.26e+04
x13	1.795e+04	1.57e+04	1.144	0.286	-1.82e+04	5.41e+04
x14	-1.309e+04	1.18e+04	-1.110	0.299	-4.03e+04	1.41e+04
x15	-1.702e+04	1.5e+04	-1.133	0.290	-5.16e+04	1.76e+04
x16	1.599e+04	1.34e+04	1.191	0.268	-1.5e+04	4.69e+04
x17	9409.5899	8727.918	1.078	0.312	-1.07e+04	2.95e+04
x18	-1.201e+04	1e+04	-1.201	0.264	-3.51e+04	1.11e+04
x19	-2253.4417	2726.070	-0.827	0.432	-8539.771	4032.888
x20	5450.1625	4599.656	1.185	0.270	-5156.664	1.61e+04
x21	-407.4663	815.200	-0.500	0.631	-2287.321	1472.388
x22	-1337.6353	1149.198	-1.164	0.278	-3987.691	1312.421

x23	372.5588	379.195	0.982	0.355	-501.867	1246.984
x24	115.5401	103.502	1.116	0.297	-123.137	354.217
x25	-64.5171	62.367	-1.034	0.331	-208.337	79.302
x26	7.7881	9.257	0.841	0.425	-13.559	29.135

Omnibus:	12.495	**Durbin-Watson:**	2.662
Prob(Omnibus):	0.002	**Jarque-Bera (JB):**	19.576
Skew:	0.804	**Prob(JB):**	5.61e-05
Kurtosis:	6.413	**Cond. No.**	1.72e+15

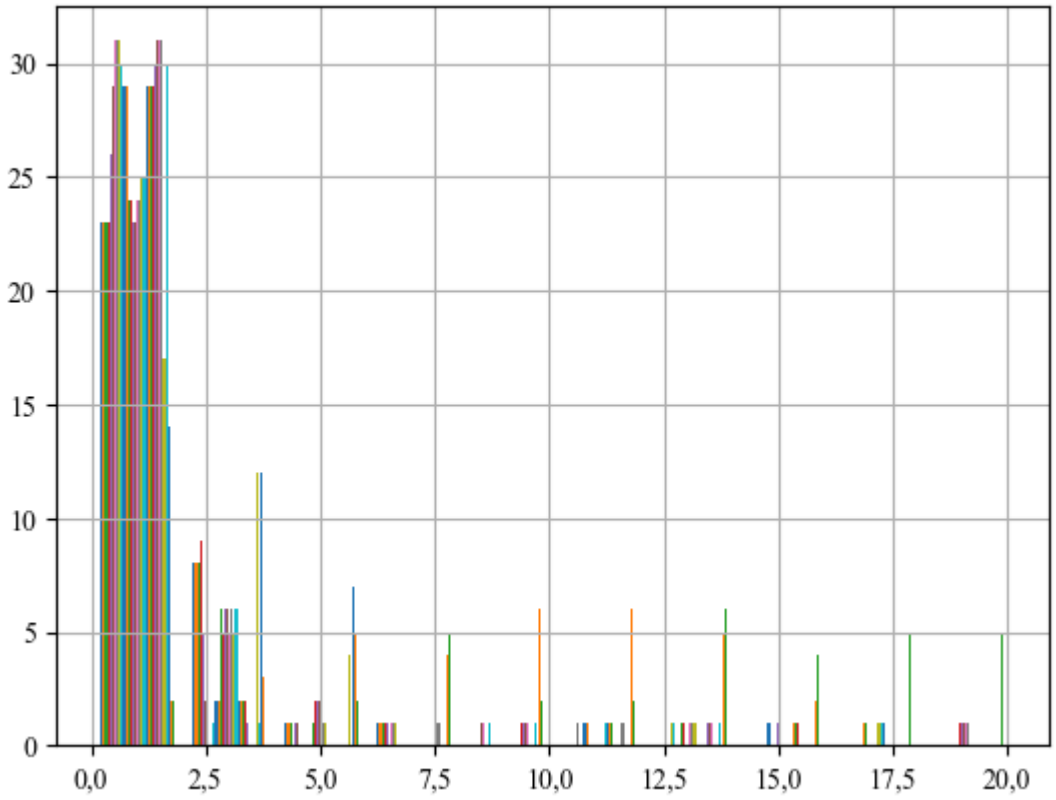
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 2.11e-14. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

Тестируем

Проверим остатки на нормальность визуально

In [12]:
```
add_text(f'{country}.docx', f'\n\n Проведем тестирование модели. \n\n Проверим о
```

In [13]:
```
ost = (x-model.predict(yp))**2
plt.hist(ost)
plt.grid()

try:
    plt.savefig(f'{country}_5.png')
except:
    pass

add_image_to_docx(f'{country}.docx',f'{country}_5.png')
plt.show()
```



Расскажем про Тест Колмогорова - Смирнова

In [14]:
```
ks_teor = """
Тест Колмогорова-Смирнова (или K-S тест) — это непараметрический статистический

Основные этапы алгоритма теста Колмогорова-Смирнова:
Сбор данных. Получаем выборку, для которой нужно проверить соответствие распреде

Определение теоретического распределения. Выбираем теоретическое распределение,

Построение эмпирической функции распределения (ЭФР):

Вычисляем кумулятивные частоты значений в выборке, чтобы построить эмпирическую

Построение теоретической функции распределения (ТФР):

На основе выбранного теоретического распределения рассчитываем его кумулятивную

Вычисление статистики Колмогорова-Смирнова:

Определяем максимальное отклонение между эмпирической и теоретической функциями

Сравнение с критическим значением:

Полученное значение D сравнивается с критическим значением для заданного уровня
Если D превышает критическое значение, гипотеза о совпадении распределений откло

Интерпретация результатов:

Если D меньше критического значения: гипотеза о том, что данные следуют теоретич
Если D больше критического значения: гипотеза о соответствии распределению откло
```

```
Тест Колмогорова-Смирнова часто используется для проверки нормальности и других
"""
add_text(f'{country}.docx', f'\n\nK-S тест',font_size=15)
add_text(f'{country}.docx', ks_teor,font_size=12, par_allign='LEFT')
```

Jarque-Bera

In [15]:
```
jb_teor= '''

Тест Джарка-Бера (Jarque-Bera) — это статистический тест, используемый для прове

Основные этапы алгоритма теста Джарка-Бера:
Сбор данных. Получаем выборку, для которой нужно проверить нормальность.

Вычисление параметров:

n: объем выборки.
Среднее значение выборки.
Стандартное отклонение выборки.


Рассчитываем асимметрию и эксцесс:

Асимметрия (skewness). Измеряет, насколько данные симметричны относительно средн
Эксцесс (kurtosis). Показывает, насколько распределение «пикообразно» или «плоск

Расчет статистики теста Джарка-Бера: JB = (n/6) * (S^2 + (K^2)/4). Чем больше зн

Сравнение с критическим значением:

Полученное значение статистики JB сравнивается с критическим значением из распре

Если JB превышает критическое значение, то гипотеза нормальности отклоняется.

Интерпретация результатов:
Если JB меньше критического значения: гипотеза о нормальности не отклоняется, и
Если JB больше критического значения: гипотеза о нормальности отклоняется, что г

Этот тест полезен для предварительного анализа данных и проверки предположения о
```

In [16]:
```
add_text(f'{country}.docx', f'\n\nJarque-Bera',font_size=15)
add_text(f'{country}.docx', jb_teor,font_size=12, par_allign='LEFT')
jb_stat, jb_p_value = jarque_bera(model.resid)
print("Статистика Jarque-Bera:", jb_stat)
add_text(f'{country}.docx', f'Статистика Jarque-Bera: {jb_stat}')

print("p-значение:", jb_p_value)
add_text(f'{country}.docx', f'p-значение: {jb_p_value}')

if jb_p_value < 0.05:
    print("Данные не распределены нормально")
    add_text(f'{country}.docx', "Данные не распределены нормально")
else:
    print('Данные распределены нормально')
    add_text(f'{country}.docx', "Данные распределены нормально")
```

```
Статистика Jarque-Bera: 19.57625069164637
p-значение: 5.611399306311759e-05
Данные не распределены нормально
```

Shapiro-Wilk

In [17]:
```
add_text(f'{country}.docx', f'\n\nShapiro-Wilk',font_size=15)

stat, p_value = st.shapiro(model.resid)
print("Статистика Shapiro-Wilk:", stat)
add_text(f'{country}.docx', f'Статистика Shapiro-Wilk: {stat}')

print("p-значение:", p_value)
add_text(f'{country}.docx', f'p-значение: {p_value}')

if p_value > 0.05:
    print("Распределение данных похоже на нормальное")
    add_text(f'{country}.docx', "Распределение данных похоже на нормальное")
else:
    print('Распределение данных отличается от нормального')
    add_text(f'{country}.docx', "Распределение данных отличается от нормального"
```

```
Статистика Shapiro-Wilk: 0.8510308430092555
p-значение: 0.0003587078429321765
Распределение данных отличается от нормального
```

Helwig

In [18]:
```
add_text(f'{country}.docx', f'\n\nHelwig',font_size=15)

def helwig_test(data):
    # Шаг 1: Сортируем данные и определяем размер выборки
    add_text(f'{country}.docx', f'Шаг 1: Сортируем данные и определяем размер вы
    data_sorted = np.sort(data)
    n = len(data)

    # Шаг 2: Оценка среднего и стандартного отклонения
    add_text(f'{country}.docx', f'Шаг 2: Оценка среднего и стандартного отклонен
    mean, std = np.mean(data), np.std(data, ddof=1)

    # Шаг 3: Вычисляем эмпирическую функцию распределения (ЭФР)
    add_text(f'{country}.docx', f'Шаг 3: Вычисляем эмпирическую функцию распреде
    ecdf = np.arange(1, n + 1) / n

    # Шаг 4: Строим теоретическую нормальную функцию распределения (НФР)
    theoretical_cdf = st.norm.cdf(data_sorted, mean, std)
    add_text(f'{country}.docx', 'Шаг 4: Строим теоретическую нормальную функцию

    # Шаг 5: Вычисляем максимальное отклонение между ЭФР и НФР
    add_text(f'{country}.docx','Шаг 5: Вычисляем максимальное отклонение между Э
    max_deviation = np.max(np.abs(ecdf - theoretical_cdf))

    # Вывод результата
    add_text(f'{country}.docx','Вывод результата')

    print("Максимальное отклонение (D):", max_deviation)
```

```
    add_text(f'{country}.docx',f'Максимальное отклонение (D): {max_deviation}')
    return max_deviation

n = len(model.resid)

alpha = 0.05
critical_value = kstwobign.ppf(1 - alpha) / np.sqrt(n)
if helwig_test(model.resid) > critical_value:
    print(f"Гипотеза о нормальности отвергается на уровне значимости {alpha}.")
    add_text(f'{country}.docx',f"Гипотеза о нормальности отвергается на уровне з
else:
    print(f"Нет оснований отвергнуть гипотезу о нормальности на уровне значимост
    add_text(f'{country}.docx',f"Нет оснований отвергнуть гипотезу о нормальност
```

```
Максимальное отклонение (D): 0.2560889275648254
Гипотеза о нормальности отвергается на уровне значимости 0.05.
```

Сравнение тестов

In [19]:
```
t = '''Сравнение методов согласия Хельвига, Шапиро-Вилька и Джарка-Бера (Jarque-

1. Тест Хельвига
Цель: Метод Хельвига основан на анализе корреляций и используется для оценки сог
Применение: Обычно применяется для оценки многомерного согласия признаков или пр
Преимущества:
Хорошо подходит для многомерных данных, поскольку анализирует согласие между нес
Позволяет оценить общую структуру корреляций между признаками, что важно для ана
Недостатки:
Не подходит для проверки нормальности распределения данных.
Может требовать больших выборок для корректного анализа многомерных данных.

2. Тест Шапиро-Вилька
Цель: Проверка нормальности распределения данных в выборке.
Применение: Часто используется для малых и средних выборок (до 2000 наблюдений),
Преимущества:
Очень чувствителен к отклонениям от нормальности, особенно в малых выборках.
Является одним из самых мощных тестов для проверки нормальности, так как учитыва
Недостатки:
Может давать ложные результаты для больших выборок (более 2000 наблюдений), так
Не подходит для многомерных данных, так как используется для одномерного распред

3. Тест Джарка-Бера (Jarque-Bera)
Цель: Проверка нормальности распределения путем оценки асимметрии (skewness) и э
Применение: Часто применяется для данных больших объемов, особенно в эконометрич
Преимущества:
Хорошо подходит для больших выборок, так как рассчитывается на основе асимметрии
Удобен для случаев, когда нужны простые показатели нормальности (асимметрия и эк
Недостатки:
Менее чувствителен для малых выборок, так как асимметрия и эксцесс могут быть не
Не учитывает порядок значений в выборке, что делает его менее точным для малых в

Вывод:
Для малых выборок (до 2000 наблюдений) тест Шапиро-Вилька наиболее подходит для
Для больших выборок (более 2000 наблюдений) тест Джарка-Бера предпочтителен, так
Тест Хельвига лучше использовать, когда требуется оценить согласие нескольких пе
Таким образом, выбор метода зависит от цели исследования, объема выборки и харак

print(t)
```

```
add_text(f'{country}.docx', f'\n\nСравнение тестов',font_size=15)
add_text(f'{country}.docx', t,font_size=12, par_allign='LEFT')
```

Сравнение методов согласия Хельвига, Шапиро-Вилька и Джарка-Бера (Jarque-Bera) полезно для выбора подходящего теста для проверки нормальности распределения данных. Каждый из этих методов имеет свою область применения и особенности, которые могут быть полезны в разных контекстах.

1. Тест Хельвига
Цель: Метод Хельвига основан на анализе корреляций и используется для оценки согласия признаков, особенно в социально-экономических и психометрических исследованиях.
Применение: Обычно применяется для оценки многомерного согласия признаков или при проведении факторного анализа.
Преимущества:
Хорошо подходит для многомерных данных, поскольку анализирует согласие между несколькими переменными.
Позволяет оценить общую структуру корреляций между признаками, что важно для анализа взаимозависимости.
Недостатки:
Не подходит для проверки нормальности распределения данных.
Может требовать больших выборок для корректного анализа многомерных данных.

2. Тест Шапиро-Вилька
Цель: Проверка нормальности распределения данных в выборке.
Применение: Часто используется для малых и средних выборок (до 2000 наблюдений), чтобы оценить, насколько распределение данных близко к нормальному.
Преимущества:
Очень чувствителен к отклонениям от нормальности, особенно в малых выборках.
Является одним из самых мощных тестов для проверки нормальности, так как учитывает порядок значений в выборке.
Недостатки:
Может давать ложные результаты для больших выборок (более 2000 наблюдений), так как становится излишне чувствительным к малейшим отклонениям.
Не подходит для многомерных данных, так как используется для одномерного распределения.

3. Тест Джарка-Бера (Jarque-Bera)
Цель: Проверка нормальности распределения путем оценки асимметрии (skewness) и эксцесса (kurtosis).
Применение: Часто применяется для данных больших объемов, особенно в эконометрических и финансовых исследованиях.
Преимущества:
Хорошо подходит для больших выборок, так как рассчитывается на основе асимметрии и эксцесса, которые более устойчивы в больших объемах данных.
Удобен для случаев, когда нужны простые показатели нормальности (асимметрия и эксцесс).
Недостатки:
Менее чувствителен для малых выборок, так как асимметрия и эксцесс могут быть нестабильными.
Не учитывает порядок значений в выборке, что делает его менее точным для малых выборок.

Вывод:
Для малых выборок (до 2000 наблюдений) тест Шапиро-Вилька наиболее подходит для проверки нормальности, поскольку он высокочувствителен к отклонениям и учитывает порядок значений.
Для больших выборок (более 2000 наблюдений) тест Джарка-Бера предпочтителен, так как он основан на асимметрии и эксцессе, что стабильно в больших объемах данных.
Тест Хельвига лучше использовать, когда требуется оценить согласие нескольких переменных, а не нормальность, так как он лучше подходит для анализа многомерных зависимостей.

Таким образом, выбор метода зависит от цели исследования, объема выборки и характ
еристик данных.