EX_03

January 31, 2023

1 Assignment 3

1.0.1 Patrick Pfenning

[1]:

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

[2]:

```
# pull data
kpi = pd.read_csv('../data/kpisetting.csv')
kpi['date'] = pd.to_datetime(kpi['date'])
kpi = kpi.set_index('date', drop=True)
kpi.head()
```

[2]:

```
            visitors  downloads  installations  28dactive
date
2015-01-14     16489       1826            570        270
2015-01-15     16362        936            266        104
2015-01-16     16463        188             61         67
2015-01-17     15972        474            112         40
2015-01-18     16659        186            109         32
```

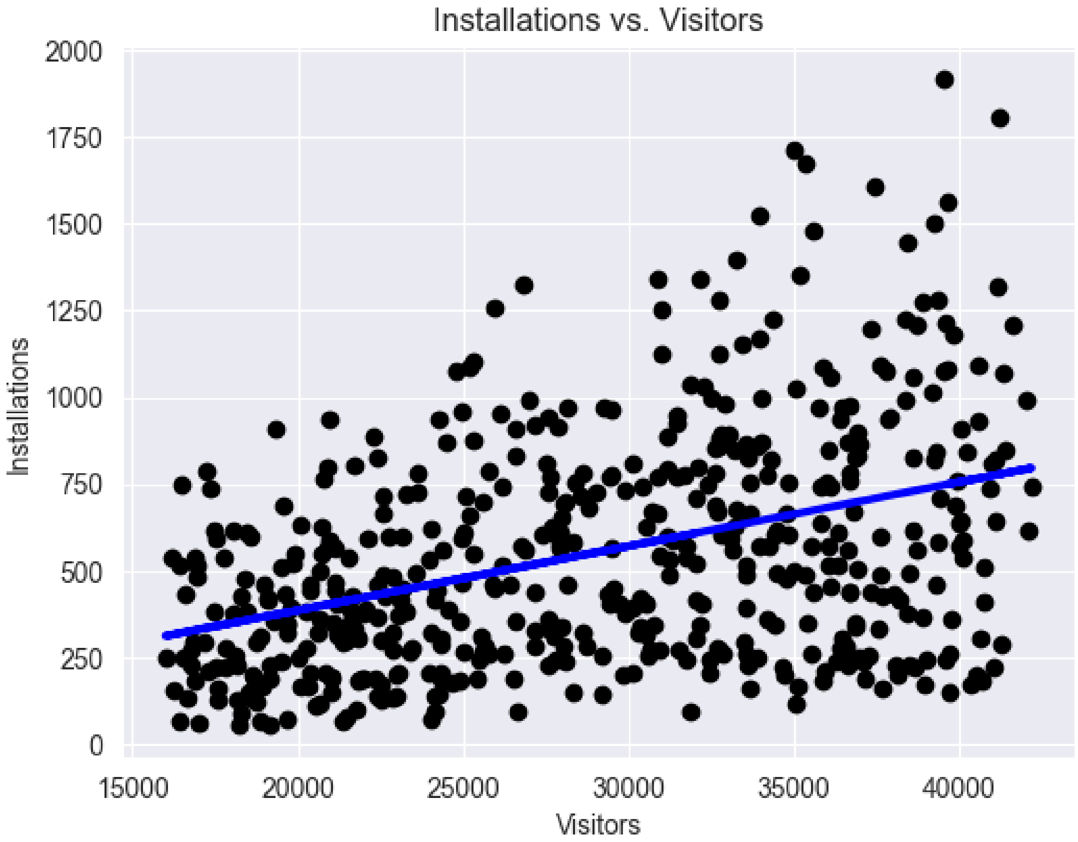
[3]:

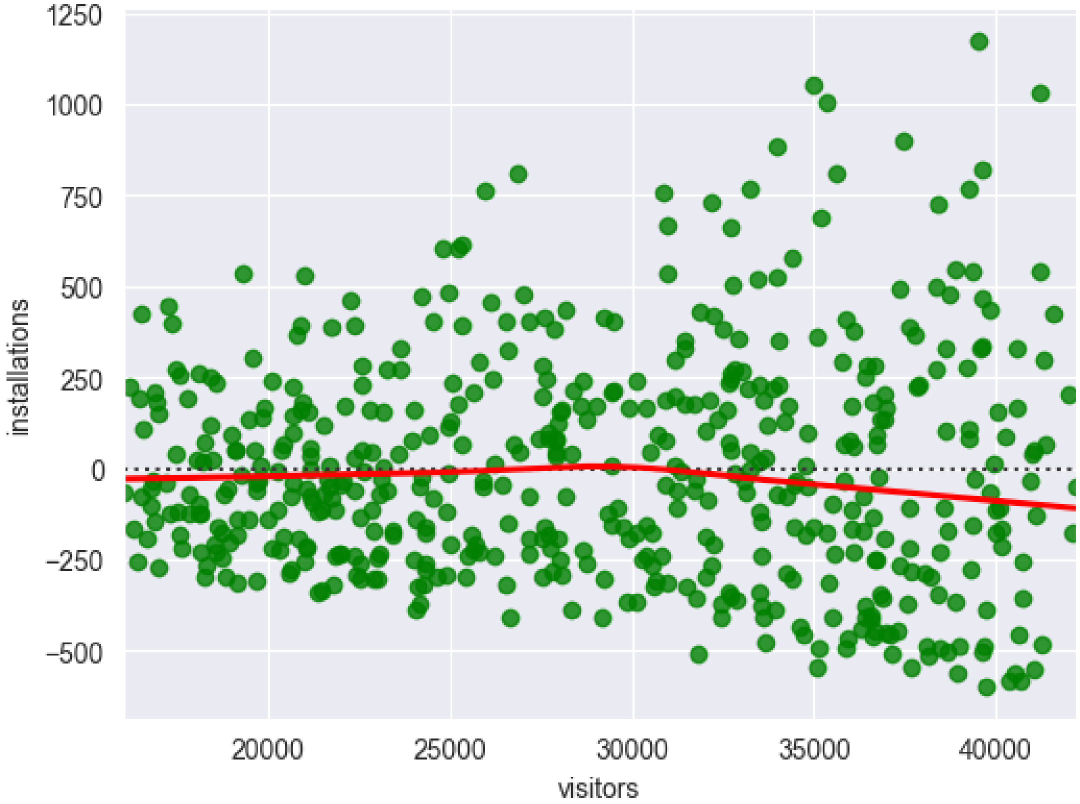
```
# get X, y
y = kpi.installations
X = kpi[['visitors', 'downloads', '28dactive']]
```

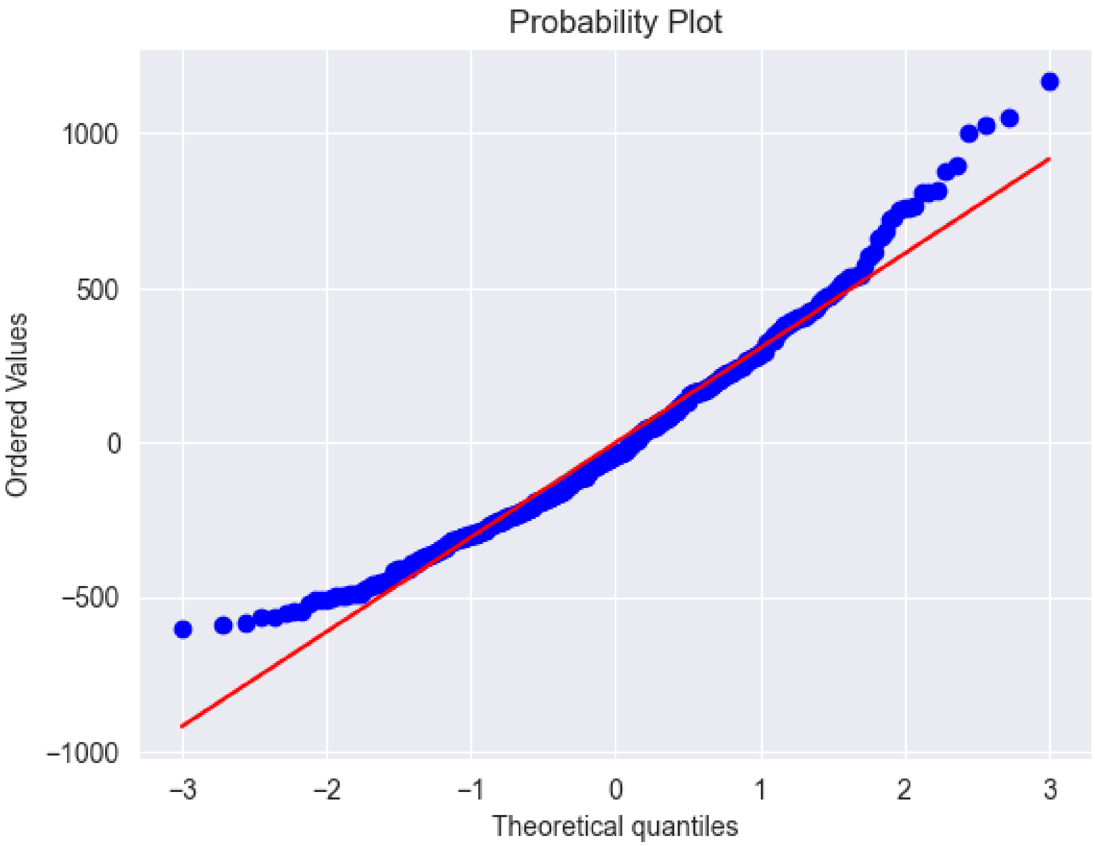
[4]:

```
from dataclasses import dataclass
import seaborn as sns
import scipy.stats as stats

@dataclass
class LM:

    X: pd.DataFrame
    y: pd.Series
    test_size: float = 0.2
    random_state: int = 42
    def __post_init__(self):
        self.X_train, self.X_test, self.y_train, self.y_test = train_test_split(
            self.X,
            self.y,
            test_size=self.test_size,
            random_state=self.random_state
        )

        self.regr = LinearRegression()
        self.regr.fit(self.X_train, self.y_train)
        self.pred = self.regr.predict(self.X_test)

    @staticmethod
    def __capitalize(string):
        return string[0].upper() + string[1:]


    def __lin_plot(self):
        plt.scatter(self.X_test, self.y_test, color="black")
        plt.plot(self.X_test, self.pred, color="blue", linewidth=3)

        xlab = self.__capitalize(self.X.columns[0])
        ylab = self.__capitalize(self.y.name)

        plt.xlabel(xlab)
        plt.ylabel(ylab)
        plt.title(f"{ylab} vs. {xlab}")

        plt.show()

    def __resid_plot(self):
        sns.residplot(x=self.X_test, y=self.y_test, lowess=True,␣
 ↪line_kws={'color': 'red'}, color='green')
        plt.show()

    def __prod_plot(self):
        stats.probplot(self.y_test - self.pred, plot=plt)
        plt.show()

    def plotter(self):
        self.__lin_plot()
        self.__resid_plot()
        self.__prod_plot()
```

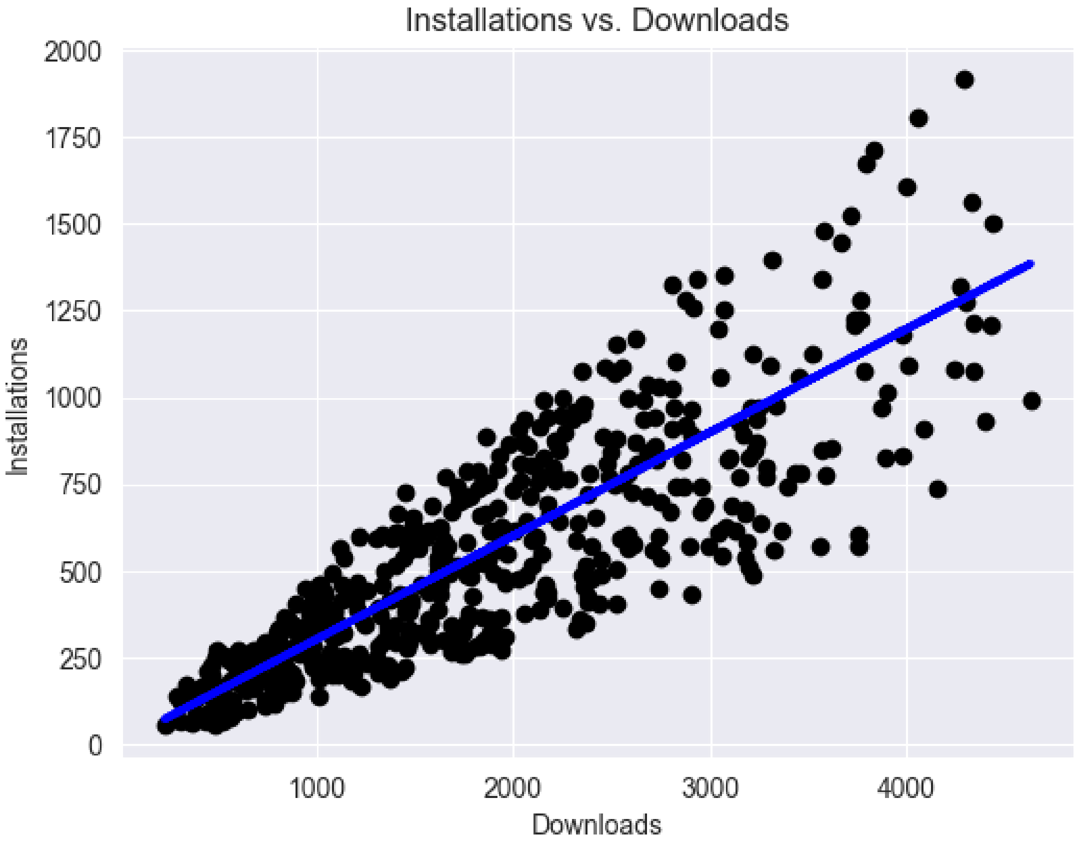
[5]:
```
visitors = LM(X[['visitors']], y)
visitors.plotter()
```

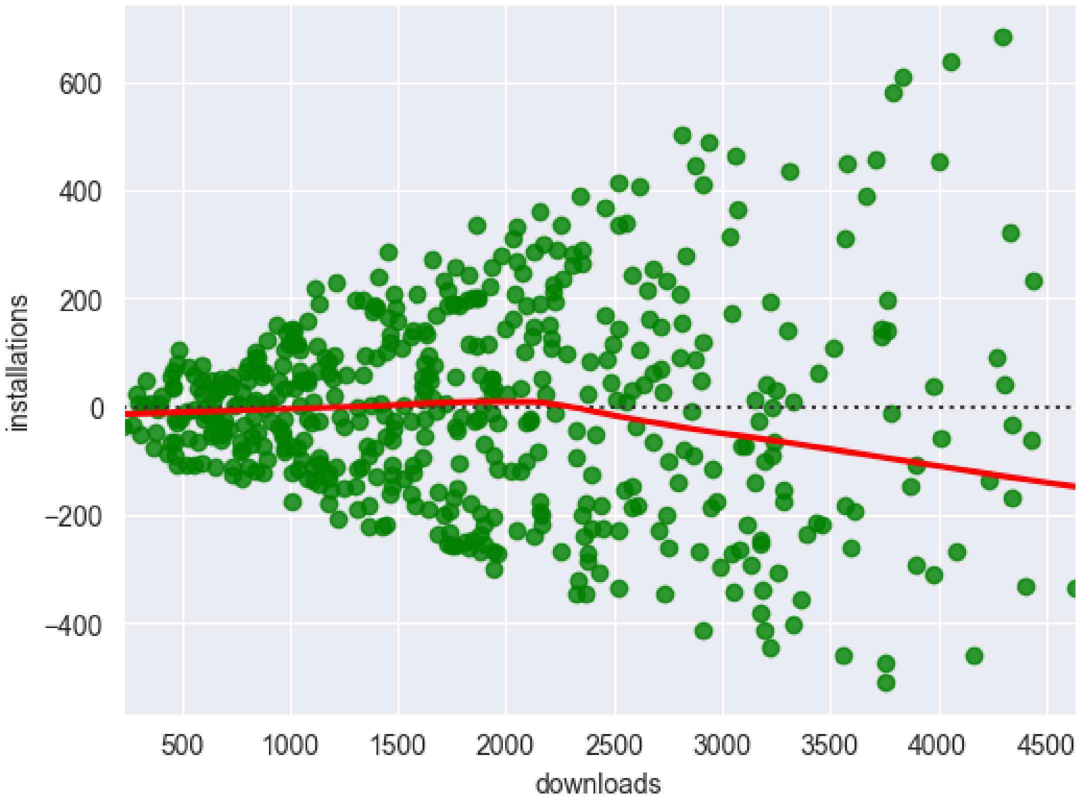

Installations vs. Visitors
Installations
2000
1750
1500
1250
1000
750
500
250
0
15000
20000
25000
30000
35000
40000
Visitors

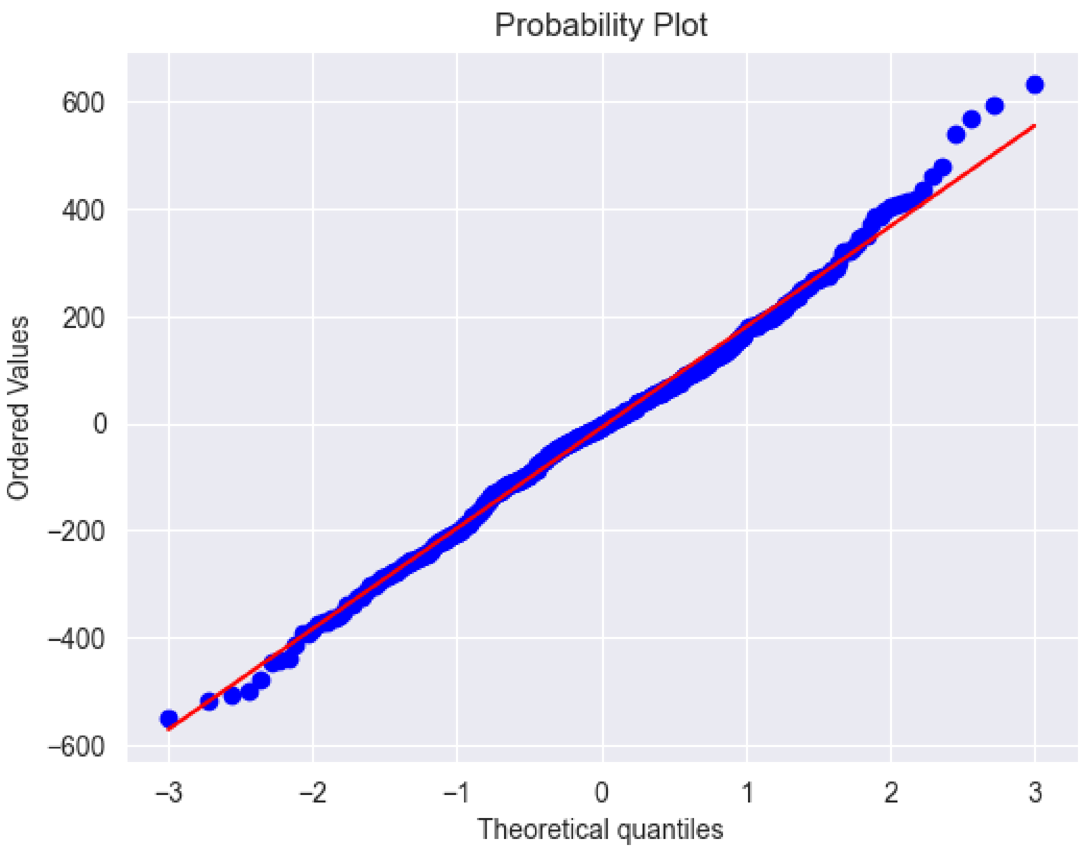

installations
1250
1000
750
500
250
0
−250
−500
20000
25000
30000
35000
40000
visitors



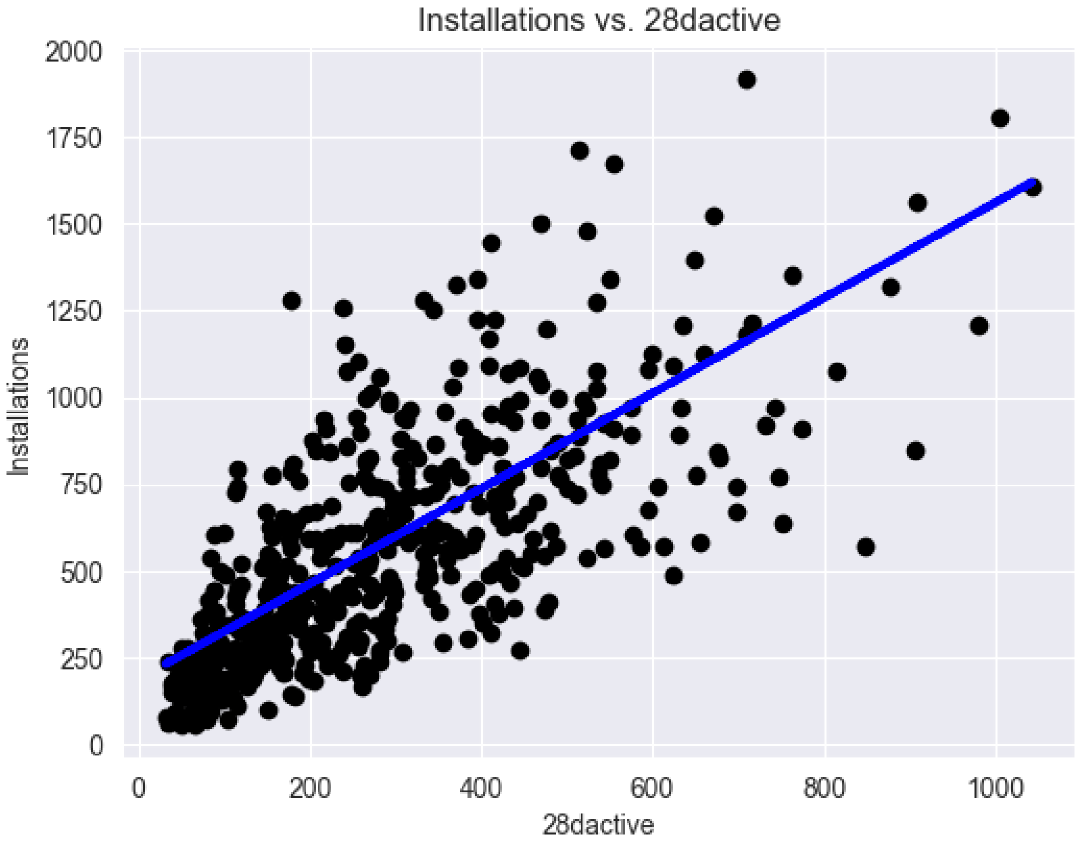
[6]:
```
downloads = LM(X[['downloads']], y)
downloads.plotter()
```

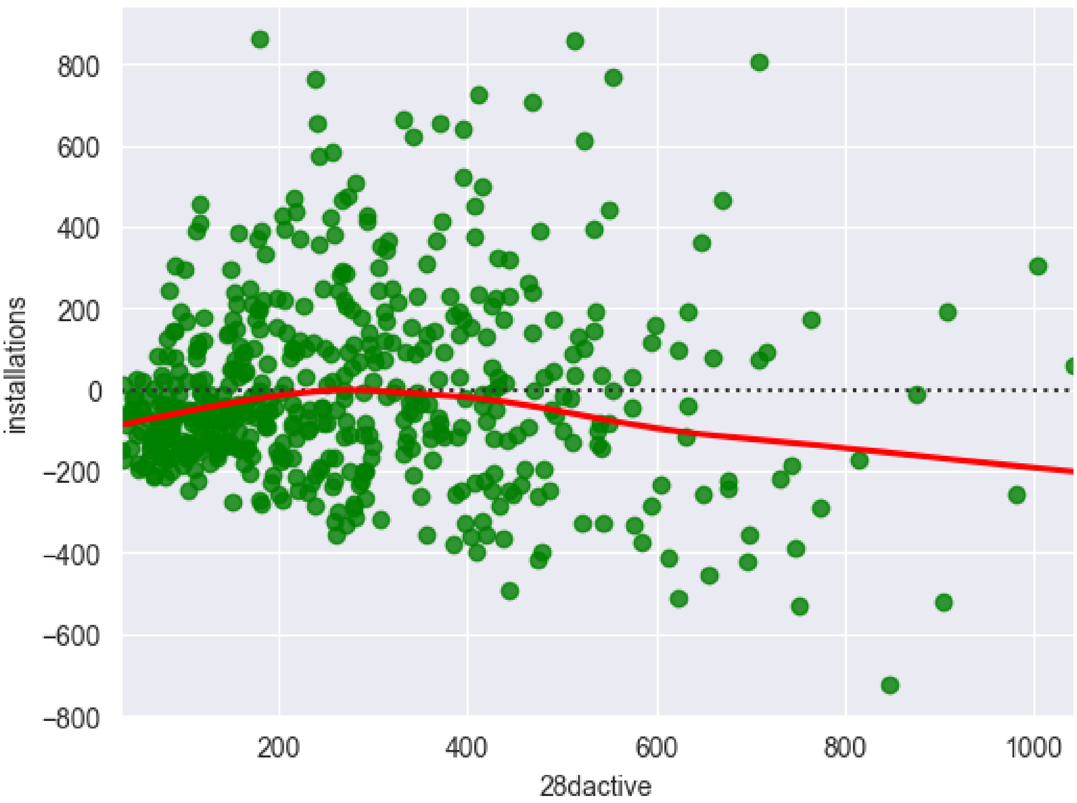

Installations vs. Downloads
2000
1750
1500
1250
1000
750
500
250
0
Installations
1000
2000
3000
4000
Downloads

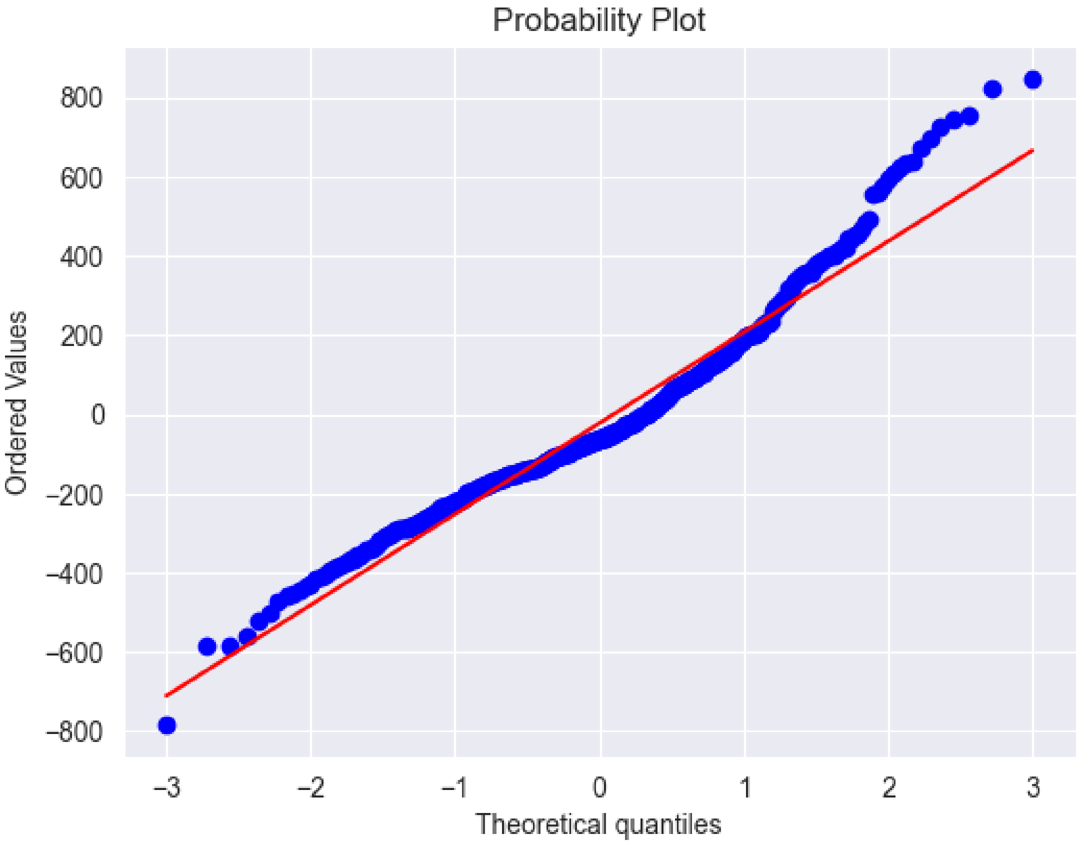

600
400
200
0
−200
−400
installations
500
1000
1500
2000
2500
3000
3500
4000
4500
downloads



```
[7]: d28 = LM(X[['28dactive']], y)
     d28.plotter()
```


Installations vs. 28dactive
Installations
2000
1750
1500
1250
1000
750
500
250
0
0
200
400
600
800
1000
28dactive


800
600
400
200
0
−200
−400
−600
−800
installations
200
400
600
800
1000
28dactive


Probability Plot
Ordered Values
800
600
400
200
0
−200
−400
−600
−800
−3
−2
−1
0
1
2
3
Theoretical quantiles