Price Elasticity of Demand with a Simple Linear Regression

Part 1: It provides an easy explanation of Price elasticities by giving a detailed explanation of what price elasticity is and how we can interpret the data.

NOTE: "How sensitive is Sales Demand, if product Price changes?" "How much will my sales increase if I lower my price?"

What is price elasticity and how to interpret price elasticity?

Price elasticities help the marketer to know how the demand would be likely to react or change when the product price changes by increasing or decreasing the price.

It answers following questions:

1. **Does sales demand change, when the price changes?**

   When the price changes have no effect on the sales demand, this is called **inelastic price**. In other words, the demand is likely to not change in proportion to price changes, even when the price increases or decreases.

How is that happening?

Price inelasticity happens due that the product is likely to **not have competitors** and as a result it has the advantage to set their own prices without having the consumers juggling between price comparison of one similar product against the other.

Another of the common reasons is **brand loyalty**, no matter the price, the consumer would keep buying the product due to the brand reputation.

2. How likely are sales demands to change when price changes?

Price elasticities are usually negative, this means that when our price decreases our sales demand increases. It makes sense isn't it?

However, positive price elasticities are found in rare cases, where products do not conform to the law of demand.

In simple words, positive price elasticities have the following effect:

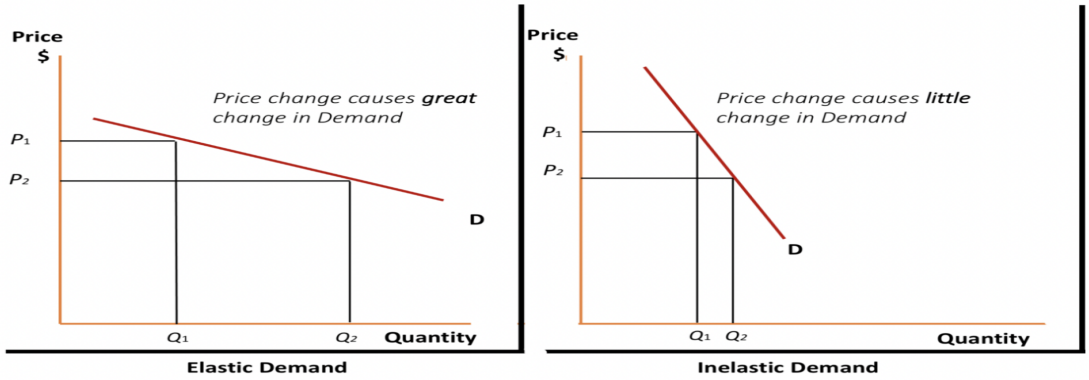
When product prices decrease, sales demand decreases. Or the opposite, when product prices increase, sales demand increases.

What are you saying, is that even possible?

Yes, these scenarios happen when we have **Veblen** goods. **Veblen** goods are typically high-quality **goods** , are exclusive, and are a status symbol, **perceived as luxury goods.**

In summary, what differs an elastic from an inelastic price is the following:

Demand is said to be elastic when demand has a higher proportionate response to a smaller change in price. On the other hand, demand is inelastic when there is little movement in demand with a significant difference in price.



Let's get to the point, how to measure price elasticity and how to interpret price elasticities?

Price Elasticity of Demand : % change in quantity / % change in Price |

(Percentage change in quantity sold divided by percentage change in price)

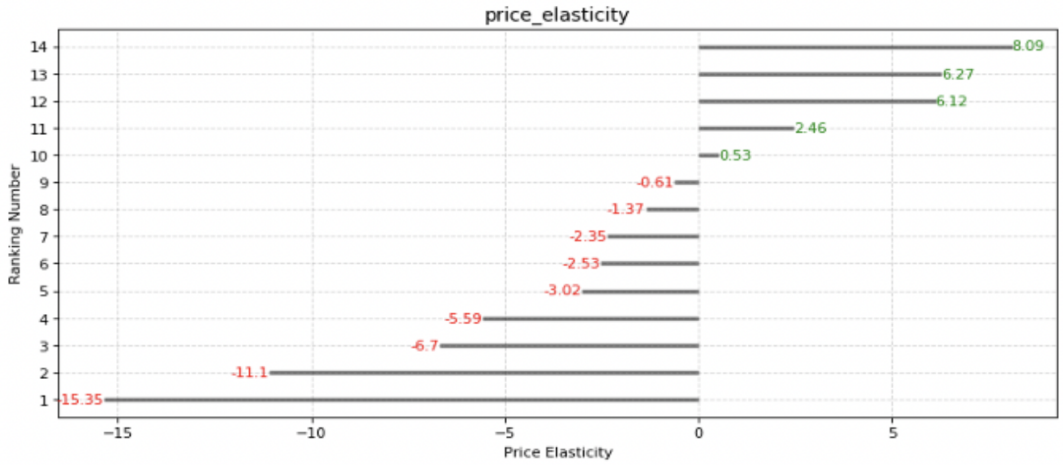
PED type	Description
Inelastic	Between -1 and 1
Negative Elasticity	Less than -1, this means -1.0 and below
Positive Elasticity	More than 1

How can we interpret price elasticities?

Price elasticities differ from product to product, some products might have a greater price elasticity than others.

One example it is the chart below:

If we analyze several laptops in an e-commerce platform, an Apple MacBook Pro laptop highlighted below has a greater negative price elasticity (-15.34) than a Lenovo laptop (-2.52).



	name	price_elasticity
ranking		
14	Apple - MacBook Pro¨ - 13 Display - Intel Core...	8.087901
13	Apple - MacBook Pro¨ - 13 Display - Intel Core...	6.272229
12	Samsung - Notebook 5 15.6 Touch-Screen Laptop ...	6.118304
11	Apple MacBook - 12 - Core m5 - 8 GB RAM - 512 ...	2.459541
10	Details About Asus Q304 13.3 Laptop I5 2.5 Ghz...	0.532044
9	Apple MacBook Pro MLUQ2LL/A 13.3 Notebook - In...	-0.613806
8	Ginsu BESTBUY5580020 Lenovo Ideapad 11.6 Laptop"	-1.371023
7	Details About Openbox Excellent: Asus Rog GI50...	-2.349560
6	Lenovo Flex 4 1470 80SA0000US 2-in-1 - 14 HD T...	-2.527233
5	Acer - 2-in-1 15.6 Refurbished Touch-Screen La...	-3.021049
4	Razer - Blade Pro 17.3 4K Ultra HD Touch-Scree...	-5.587323
3	Dell - Inspiron 15.6 Laptop - Intel Core i5 - ...	-6.704656
2	Details About Apple Macbook Air 13.3 Laptop (e...	-11.101426
1	Apple - MacBook Pro¨ - 13 Display - Intel Core...	-15.349549

Fig. Data used as an example, please do not make assumptions of real-life case scenarios.

Apple MacBook Pro with a negative price elasticity of -15.34, it is described as follows:

A 10% price decrease in Apple MacBook Pro-13, it increases sales demand by 153.5% or a 10% price increase in Apple MacBook Pro-13, it decreases sales demand by 153.5%

Lenovo Laptop with a negative price elasticity of -2.52, it is described as follows:

A 10% price decrease in Lenovo Laptop, it increases sales demand by 25.2% or a 10% price increase in Lenovo Laptop, it decreases sales demand by 25.2%

Part 2: In this article we are coding and explanation of how to build your own price elasticity model by using Linear Regression in Python.

Data Collection:

What data to collect?

This model would analyze the price elasticities of several electronic products in one sole category “Laptop, computer” for an e-commerce platform.

In order to analyze the price elasticities, we need to have products where they have historical price and quantity sold data (sales demand) , the price need at least to have more than one price variations of the product price with their respective quantity sold (sales demand) in order to be able to observe the sales demand reaction towards the product price changes.

Python libraries used are:

We would use the following python libraries: **pandas, numpy, matplotlib and statsmodels.**

Here, I imported pandas, numpy and matplotlib in the beginning but in the following steps you will see the most important from all, which it is **statsmodels** used mainly for the **Linear Regression model**.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv('price_clean1.csv', encoding = "ISO-8859-1")
df.head()
```

OUTPUT:

	Date_imp	Category_name	name	disc_price	merchant	condition	Sales_count
0	2017-10-10 05:00:00	speaker, portable, bluetooth	Boytone - 2500W 2	69.99	Bestjam.com	New	80
1	2017-08-28 07:00:00	speaker, portable, bluetooth	Boytone - 2500W 2	66.99	Bestjam.com	New	49
2	2017-08-12 09:00:00	speaker, portable, bluetooth	Boytone - 2500W 2	65.99	Bestjam.com	New	85
3	2017-08-01 03:00:00	speaker, portable, bluetooth	Boytone - 2500W 2	64.99	Bestjam.com	New	78
4	2017-07-26 15:00:00	speaker, portable, bluetooth	Boytone - 2500W 2	64.99	Bestjam.com	New	57

Table 1

In our data frame as you see above, we have what we need to make our model. **The main metrics** that you need are **product name, price and sales count,** which **reflects the quantity of sales.**

Data Frame Features explained:

- **Date_imp:** Date time impression when the data was captured
- **Category_name:** Category name is a feature included in order to analyze prices in sole groups such as Laptops, Speakers and others
- **name:** Name of the product
- **disc_price:** Price of the product. In case the product has a discount, the price value included is with the price already discounted
- **merchant:** merchant is the platform that the product is located. As a recommendation, analyze product prices from a sole platform due that consumer traffic might vary from platform to platform and the price elasticity analysis might not be very accurate
- **condition:** product condition. As a recommendation, analyze product prices from the same condition. Otherwise, this might affect your results due that product prices with conditions as “used” are usually cheaper than new product prices.

- **Sales_count:** sales count is the sales demand of the products with their respective prices

From the data, only one merchant was selected as mentioned above in order to not have any incongruences, it might be that other merchants have more user traffic (market share) than the others and as a result our price elasticities can be altered. We choose “Bestjam.com” e-commerce platform, and condition the product as “New”. Then we checked all category values that we have.

```
#Choose one merchant platform
df = df[df['merchant']=='Bestjam.com']
#Choose condition as New
df = df[df['condition']=='New']

#Check values in different category_name
category = df['Category_name'].value_counts()
print("Bestjam Category Percentage\n{}\n{}\n Bestjam Category values count\n{}\n{}".format(
      50*"-", (category / len(df.index))[:5] * 100, 50*"-", category[:5]))
```

OUTPUT:

```
Bestjam Category Percentage
--------------------------------------------------
laptop, computer                9.488792
speaker, portable, bluetooth    6.715691
car, speaker, subwoofer         4.899442
receiver, amplifier, home       4.429875
car, receiver, dash             3.641357
Name: Category_name, dtype: float64
 Bestjam Category values count
--------------------------------------------------
laptop, computer                1071
speaker, portable, bluetooth     758
car, speaker, subwoofer          553
receiver, amplifier, home        500
car, receiver, dash              411
Name: Category_name, dtype: int64
```

Figure 1

As per see, “Laptop, computer” category is the biggest category. For this reason, on this occasion, we would analyze the price elasticities of this category.

```
#Select Laptop
df_laptop = df[df['Category_name']=='laptop, computer']

#Make sure to not have any duplicates in the data
df_laptop = df_laptop.groupby(['Date_imp','name']).agg({'disc_price':'mean','Sales_count': 'mean' }).reset_index()

#laptop data used for price elasticities
df_laptop.head()
```

OUTPUT:

	Date_imp	name	disc_price	Sales_count
0	2017-01-24 00:00:00	Dell - XPS 2-in-1	1399.990000	26.000000
1	2017-03-03 15:00:00	12.3 32GB Multi-T	426.495000	75.000000
2	2017-03-03 15:00:00	15.4 MacBook Pro	2666.656667	62.333333
3	2017-03-03 15:00:00	Acer - 2-in-1 15.	899.990000	109.000000
4	2017-03-03 15:00:00	Acer 15.6 Chromeb	236.495000	78.500000

Table 2

Now that we filtered our data to “Laptop, computer” category and condition as “New” , we select the following columns in our data frame for analysis : **Date_imp, name, disc_price and Sales_count.**

Data Preparation:

For building the linear regression modeling, we set “Sales_count”(y value) as our dependent variable and as an independent variable “disc_price”(x value).

Due that the purpose of this model is to estimate the price elasticities of multiple products in one category. It captured the entire “x_values”, which are the product prices within its date impression and name, and the entire “y_values”, which are the quantity sold of the products within its date impression and name in two separated data frames, formatted as follows:

Independent variables (disc_price) as x_values:

```
#Format and build a dataframe with x_values for each product within the caregory
x_pivot = df_laptop.pivot(index= 'Date_imp' ,columns='name' ,values='disc_price')
x_values = pd.DataFrame(x_pivot.to_records())
print(x_values)
```

OUTPUT:

	Date_imp	12 MacBook (Mid 2	12.3 32GB Multi-T	13.3 MacBook Air	15.4 MacBook Pro	ASUS VivoBook Max	Acer - 2-in-1 15.	Acer 15.6 Chromeb	Alienware - R3 17	Apple - MacBook P	...	HP - ProBook 14 L	HP 15-AY103DX 15.	Lenovo - 100S-14I	Lenovo - Yoga 710
0	2017-01-24 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
1	2017-03-03 15:00:00	NaN	426.495	NaN	2666.656667	NaN	899.99	236.495	NaN	NaN	...	NaN	499.99	NaN	664.99
2	2017-03-04 04:00:00	NaN	426.495	NaN	2666.656667	NaN	899.99	236.495	NaN	NaN	...	NaN	499.99	NaN	664.99
3	2017-03-04 10:00:00	NaN	426.495	NaN	2666.656667	NaN	899.99	236.495	1599.99	NaN	...	799.99	499.99	NaN	664.99
4	2017-03-10 15:00:00	NaN	426.495	NaN	2666.656667	NaN	899.99	238.995	1599.99	NaN	...	799.99	429.99	NaN	664.99
5	2017-03-10 23:00:00	NaN	426.495	NaN	2599.990000	NaN	899.99	238.995	1899.99	NaN	...	819.99	429.99	229.99	664.99
6	2017-03-15 14:00:00	NaN	426.495	NaN	2560.656667	NaN	899.99	238.995	1899.99	NaN	...	819.99	429.99	218.49	664.99
7	2017-03-15 15:00:00	NaN	426.495	NaN	2560.656667	NaN	899.99	238.995	1899.99	NaN	...	819.99	429.99	218.49	664.99

Table 3

Dependent variable (Sales Count) as y_values:

```
#Format and build a dataframe with y_values for each product within the caregory
y_pivot = df_laptop.pivot( index = 'Date_imp',columns='name', values='Sales_count')
y_values = pd.DataFrame(y_pivot.to_records())
print(y_values)
```

OUTPUT:

	Date_imp	12 MacBook (Mid 2	12.3 32GB Multi-T	13.3 MacBook Air	15.4 MacBook Pro	ASUS VivoBook Max	Acer - 2-in-1 15.	Acer 15.6 Chromeb	Alienware - R3 17	Apple - MacBook P	...	HP - ProBook 14 L	HP 15-AY103DX 15.	Lenovo - 100S-14I	Lenovo - Yoga 710	8
0	2017-01-24 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	
1	2017-03-03 15:00:00	NaN	75.0	NaN	62.333333	NaN	109.0	78.5	NaN	NaN	...	NaN	34.0	NaN	83.5	
2	2017-03-04 04:00:00	NaN	98.0	NaN	82.000000	NaN	74.0	66.5	NaN	NaN	...	NaN	94.0	NaN	96.0	
3	2017-03-04 10:00:00	NaN	98.0	NaN	68.333333	NaN	74.0	66.5	141.0	NaN	...	82.0	94.0	NaN	96.0	
4	2017-03-10 15:00:00	NaN	62.0	NaN	45.000000	NaN	77.0	106.5	141.0	NaN	...	82.0	159.0	NaN	35.5	
5	2017-03-10 23:00:00	NaN	62.0	NaN	126.500000	NaN	77.0	106.5	99.0	NaN	...	18.0	159.0	81.0	35.5	
6	2017-03-15 14:00:00	NaN	62.0	NaN	56.000000	NaN	77.0	106.5	23.0	NaN	...	18.0	159.0	32.5	35.5	
7	2017-03-15 15:00:00	NaN	62.0	NaN	56.000000	NaN	77.0	106.5	23.0	NaN	...	135.0	159.0	32.5	35.5	

Table 4

Now, that the data is formatted in the correct manner for our Linear Regression. Let's build our Linear Regression model!

Price Elasticity Model using Linear Regression:

As we might know from the Part 1 of this article, price elasticity calculation is the following:

PED:

Percentage change in Quantity (Sales Demand) / percentage change in Price

which it is rewritten in the use of Linear Regression as follows:

PED in Linear Regression:

Coefficient (Slope) * Price mean / Quantity mean

As mentioned for the Linear Regression, **statsmodels library** is used. Below you see that in the for loop, data was transformed and appended into several data frames with x values and y values per product name. For later, fitting the Linear Regression with each product data frame and their respective x and y values.

```
points = []
results_values = {
    "name": [],
    "price_elasticity": [],
    "price_mean": [],
    "quantity_mean": [],
    "intercept": [],
    "t_score":[],
    "slope": [],
    "coefficient_pvalue" : [],
}
#Append x_values with y_values per same product name
for column in x_values.columns[1:]:
    column_points = []
    for i in range(len(x_values[column])):
        if not np.isnan(x_values[column][i]) and not np.isnan(y_values[column][i]):
            column_points.append((x_values[column][i], y_values[column][i]))
    df = pd.DataFrame(list(column_points), columns= ['x_value', 'y_value'])


    #Linear Regression Model
    import statsmodels.api as sm
    x_value = df['x_value']
    y_value = df['y_value']
    X = sm.add_constant(x_value)
    model = sm.OLS(y_value, X)
    result = model.fit()


    #(Null Hypothesis test) Coefficient with a p value less than 0.05
    if result.f_pvalue < 0.05:

        rsquared = result.rsquared
        coefficient_pvalue = result.f_pvalue
        intercept, slope = result.params
        mean_price = np.mean(x_value)
        mean_quantity = np.mean(y_value)
        tintercept, t_score = result.tvalues

        #Price elasticity Formula
        price_elasticity = (slope)*(mean_price/mean_quantity)

        #Append results into dictionary for dataframe
        results_values["name"].append(column)
        results_values["price_elasticity"].append(price_elasticity)
        results_values["price_mean"].append(mean_price)
        results_values["quantity_mean"].append(mean_quantity)
        results_values["intercept"].append(intercept)
        results_values['t_score'].append(t_score)
        results_values["slope"].append(slope)
        results_values["coefficient_pvalue"].append(coefficient_pvalue)

final_df = pd.DataFrame.from_dict(results_values)
df_elasticity =
final_df[['name','price_elasticity','t_score','coefficient_pvalue','slope','price_mean','quantity_mean','intercept','rsquared']]
```

OUTPUT:

	name	price_elasticity	t_score	coefficient_pvalue	slope	price_mean	quantity_mean	intercept
0	12.3 32GB Multi-T	-1.139091	-2.160724	3.429609e-02	-0.296467	404.073623	105.166667	224.961071
1	13.3 MacBook Air	-5.492504	-4.909908	2.581583e-05	-0.488629	944.048824	83.985294	545.274874
2	15.4 MacBook Pro	2.123778	4.507300	2.704239e-05	0.067899	2285.361981	73.065217	-82.109077
3	Acer - 2-in-1 15.	-1.845861	-8.610751	1.911017e-12	-0.260110	788.323333	111.086957	316.138058
4	Acer 15.6 Chromeb	2.137534	2.665846	9.615743e-03	0.791627	239.787681	88.804348	-101.017965
5	Alienware - R3 17	3.524163	3.778127	3.453336e-04	0.172286	1856.855672	90.776119	-229.133694
6	Apple - MacBook P	-0.938055	-2.453633	1.859401e-02	-0.050420	1541.532857	82.856746	160.580898
7	Apple 13.3 MacBoo	-1.392306	-3.853751	2.631234e-04	-0.080285	1108.512101	63.920290	152.916915
8	Apple MacBook - 1	-0.715617	-2.833421	6.079077e-03	-0.046088	1362.102319	87.723430	150.499815
9	Asus - 2-in-1 15.	3.393626	3.699011	5.473524e-04	0.177060	1231.166471	64.235294	-153.755275
10	Dell - Inspiron 1	-3.492304	-2.827894	6.222282e-03	-0.384907	780.288507	86.000000	386.338129
11	Dell XPS 15 15.6	0.968596	6.352889	2.371998e-08	0.061604	1864.340970	118.574627	3.723743
12	Details About Ali	1.211179	2.131532	3.671670e-02	0.061010	1697.932029	85.528986	-18.061952
13	Details About App	13.534203	4.803430	3.515649e-05	1.641183	785.004706	95.191176	-1193.145574
14	Details About Asu	-0.808131	-4.535730	2.440009e-05	-0.168563	503.359565	104.992754	189.840675
15	Ginsu BESTBUY5580	0.582514	2.231909	2.896955e-02	0.294840	166.931739	84.492754	35.274576
16	HP 15-AY103DX 15.	-3.657760	-8.264401	8.030442e-12	-0.896867	458.758116	112.485507	523.930496
17	Lenovo - 100S-14I	1.069219	2.743045	7.916960e-03	0.410058	196.443846	75.338462	-5.214859
18	Lenovo 80TX0007US	0.976715	3.806550	3.079022e-04	0.191042	341.801594	66.855072	1.556736
19	Lenovo Flex 4 147	-2.227008	-2.786971	6.915146e-03	-0.583676	350.692899	91.913043	296.604090
20	MSI - WS Series 1	-9.384572	-2.980876	4.170841e-03	-0.374286	2380.317869	94.934426	985.853400
21	New Asus Q524u 15	-0.972903	-13.601635	6.004450e-21	-0.158441	719.149420	117.115942	231.058416
22	Samsung - Noteboo	4.816791	3.173795	2.299214e-03	0.714910	599.295970	88.947761	-339.494974

Table 5

Further, there is an introduction to each metric calculated for you to take it into account when you analyze or build your own price analysis.

Note: I used to check the t-stats and p-values from your price elasticities.

Final Data Frame Metrics Explanation:

- **price_elasticity:** price elasticity per product
- **t_score:** equals to t-stats, which indicates the significance of the findings. The closer is to 0, the more likely there is no significance. The greater the t-stats, whether positive or negative, the greater the significance of the findings
- **coefficient_pvalue:** tests the null hypothesis that the coefficient is equal to zero (no effect). A low p-value (< 0.05) indicates that you can reject the null hypothesis
- **slope (coefficient):** is the ratio of the "price change" to the "quantity sold change" between (any) two distinct points on a line.

- **price_mean:** average price per product.
- **quantity_mean:** average quantity sold per product.
- **intercept:** equals the quantity of X when all other variables = 0

Data Visualization: Divergent Plot for Price Elasticities:

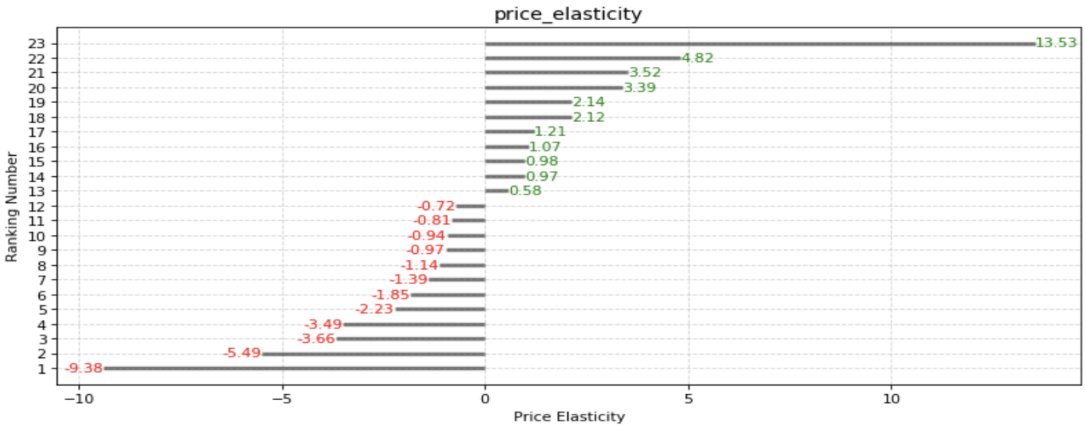
As a bonus, I am adding a data visualization function that would help you to present your price elasticities and how you can interpret the data from your visualization:

```
def divergent_plot(df, values_column, ylabel, xlabel):

    #Divergent plot
    df['ranking'] = df[values_column].rank( ascending = True).astype(int)
    df.sort_values(values_column, ascending =False, inplace = True)
    plt.figure(figsize = (12,5), dpi = 80)
    plt.hlines(y = df['ranking'] , xmin = 0, xmax = df[values_column], alpha = 0.5, linewidth = 3)

    #Add elasticity labels
    for x, y, tex in zip(df[values_column], df['ranking'] , df[values_column]):
        plt.text(x, y, round(tex, 2), horizontalalignment='right' if x < 0 else 'left',
                 verticalalignment='center', fontdict={'color':'red' if x < 0 else 'green', 'size':10})


    # Axis and title
    plt.gca().set(ylabel= ylabel, xlabel= xlabel)
    plt.yticks(df['ranking'])
    plt.title(values_column , fontdict={'size':13})
    plt.grid(linestyle='--', alpha=0.5)
    plt.show()


    #Adjust Ranking column and print dataframe
    pd.set_option('display.width', 4000)
    cols = list(df.columns)
    cols = [cols[-1]] + cols[:-1]
    df = df[cols]

    df = df.iloc[:,:3]
    df.set_index('ranking', inplace=True)
    display(df)
```

Above, this built-in function would help you for the visualization of price elasticities. Divergent plot is used because it gives the reader a more clear overview between **negative and positive price elasticities** for price analysis.

```
pe_plot = divergent_plot(df_elasticity, 'price_elasticity', 'Ranking Number', 'Price Elasticity')
```

OUTPUT:



ranking	name	price_elasticity
23	Details About App	13.534203
22	Samsung - Noteboo	4.816791
21	Alienware - R3 17	3.524163
20	Asus - 2-in-1 15.	3.393626
19	Acer 15.6 Chromeb	2.137534
18	15.4 MacBook Pro	2.123778
17	Details About Ali	1.211179
16	Lenovo - 100S-14I	1.069219
15	Lenovo 80TX0007US	0.976715
14	Dell XPS 15 15.6	0.968596
13	Ginsu BESTBUY5580	0.582514
12	Apple MacBook - 1	-0.715617
11	Details About Asu	-0.808131
10	Apple - MacBook P	-0.938055
9	New Asus Q524u 15	-0.972903
8	12.3 32GB Multi-T	-1.139091
7	Apple 13.3 MacBoo	-1.392306
6	Acer - 2-in-1 15.	-1.845861
5	Lenovo Flex 4 147	-2.227008
4	Dell - Inspiron 1	-3.492304
3	HP 15-AY103DX 15.	-3.657760
2	13.3 MacBook Air	-5.492504
1	MSI - WS Series 1	-9.384572

As per see from Part.1 of Price Elasticity of Demand using Linear Regression in Python:

Elasticities between 0- 1 are considered **inelastic**, as a result the products from rank 9 to 15 are **inelastic.**

Negative Price Elasticities are located from rank 1 to 8 and this would tell us the likelihood of when price decreases how likely sales demand would increase or vice versa.

For instance, let's analyze Lenovo Flex located in rank 5 with a negative price elasticity of -2.22:

> A 10% price decrease in Lenovo Flex, it increases sales demand by 22.2% or a 10% price increase in Lenovo Flex, it decreases sales demand by 22.2%