# **Linear regression**

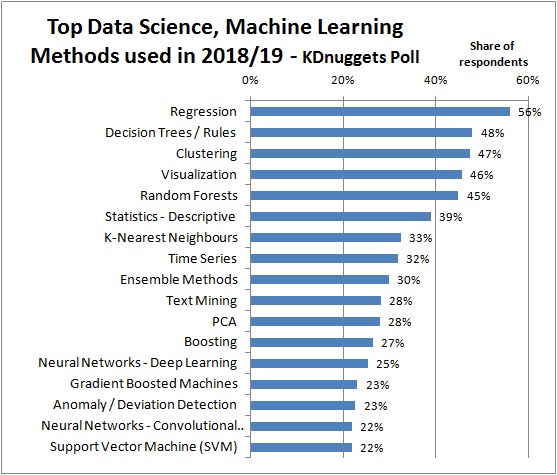
## **Plan**

1. Recap: Statistical Inference
2. Motivations about Linear Regression
3. Linear regression (visual approach with seaborn)
4. Linear regression (with statsmodels)
5. Conditions for Inference
6. Multivariate Linear Regression

## **Recap: Statistical Inference**

* Sampling distribution of the *mean*
* μ
  + ^
  + μ
  + = plausible distribution of values for
  + μ
  + [95% confidence intervals]
* Hypothesis testing
  + p-value : "probability that what you observed is just due to pure chance"
  + significance level
  + α
  + =
  + 0.05
* Central Limit Theorem extended
  + z-test for normal
  + N
  + distributions
  + t-tests for student
  + T
  + ν
  + distribution
* Bayesian inference
  + p
  + o
  + s
  + t
  + e
  + r
  + i
  + o
  + r
  + ∼
  + p
  + r
  + i
  + o
  + r
  + ∗
  + l
  + i
  + k
  + e
  + l
  + i
  + h
  + o
  + o
  + d
  + Maximum Likelihood Estimate (MLE)
  + Maximim A Posteriori Estimate (MAP)

## **1. Motivation**

****

* Most important applied statistical tool
* Interpretable
* Standard practice for **causal inference**

[Source KDnuggets](https://www.kdnuggets.com/2019/04/top-data-science-machine-learning-methods-2018-2019.html) (800 participants)

**Recall our business problem** 👇

*How to increase customer satisfaction while maintaining a healthy order volume?*

**Linear Regression** will help us analyse:

1. What features impact review\_score the most?
2. How to control the **confounding factors**?

## **2. Simple Linear Regression (visual approach with seaborn)**

### **The mpg (miles per gallon) dataset**

🥋 Let's take an example!

🚗 The **mpg** dataset

👉 Contains ~400 models of car statistics from 1970 to 1982

**import** **pandas** **as** **pd**

**import** **matplotlib.pyplot** **as** **plt**

**import** **seaborn** **as** **sns**

mpg = sns.load\_dataset("mpg").dropna()

mpg.head()

|  | **mpg** | **cylinders** | **displacement** | **horsepower** | **weight** | **acceleration** | **model\_year** | **origin** | **name** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 18.0 | 8 | 307.0 | 130.0 | 3504 | 12.0 | 70 | usa | chevrolet chevelle malibu |
| **1** | 15.0 | 8 | 350.0 | 165.0 | 3693 | 11.5 | 70 | usa | buick skylark 320 |
| **2** | 18.0 | 8 | 318.0 | 150.0 | 3436 | 11.0 | 70 | usa | plymouth satellite |
| **3** | 16.0 | 8 | 304.0 | 150.0 | 3433 | 12.0 | 70 | usa | amc rebel sst |
| **4** | 17.0 | 8 | 302.0 | 140.0 | 3449 | 10.5 | 70 | usa | ford torino |

Full description of the dataset is [here](https://data.world/dataman-udit/cars-data) 🏎️

The columns we will focus on are:

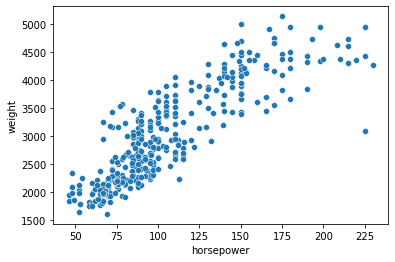
* mpg: miles per gallon
* cylinders
* displacement: volume of all the pistons (in cc)
* horsepower
* weight: pounds (lbs)
* acceleration: zero to sixty miles per hour (in seconds)

mpg.describe().apply(**lambda** x: round(x))

|  | **mpg** | **cylinders** | **displacement** | **horsepower** | **weight** | **acceleration** | **model\_year** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 392 | 392 | 392 | 392 | 392 | 392 | 392 |
| **mean** | 23 | 5 | 194 | 104 | 2978 | 16 | 76 |
| **std** | 8 | 2 | 105 | 38 | 849 | 3 | 4 |
| **min** | 9 | 3 | 68 | 46 | 1613 | 8 | 70 |
| **25%** | 17 | 4 | 105 | 75 | 2225 | 14 | 73 |
| **50%** | 23 | 4 | 151 | 94 | 2804 | 16 | 76 |
| **75%** | 29 | 8 | 276 | 126 | 3615 | 17 | 79 |
| **max** | 47 | 8 | 455 | 230 | 5140 | 25 | 82 |

### **Regress weight on horsepower ?**

sns.scatterplot(x='horsepower', y='weight', data=mpg);



❓ Find a regression line

^

y

=

(

β

0

+

β

1

h

o

r

s

e

p

o

w

e

r

)

that is the **closest** to the *weights*

🤓 Find

β

=

(

β

0

,

β

1

)

that minimizes the **norm**

∥

w

e

i

g

h

t

s

−

(

β

0

+

β

1

h

o

r

s

e

p

o

w

e

r

s

)

∥

**Ordinary Least Square (OLS) regression**

* uses the "natural"
* L
* 2
* **Euclidian Distance**
* solves the
* β
* that minimizes **Sum of Squared Residuals (SSR)**

a

r

g

m

i

n

β

392

∑

i

=

1

(

w

e

i

g

h

t

i

−

(

β

0

+

β

1

h

o

r

s

e

p

o

w

e

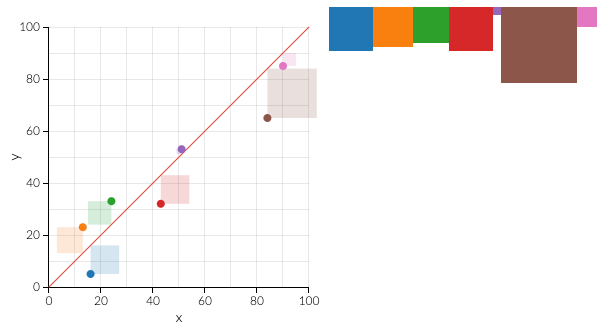
r

s

)

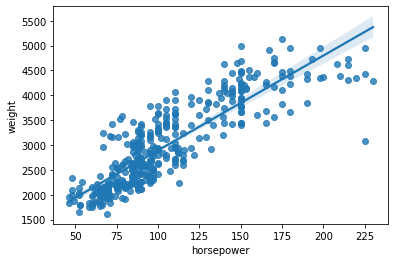
)

2

👉 [Dynamic vizualisation](http://setosa.io/ev/ordinary-least-squares-regression/)

⚠️ ***OLS is very sensitive to outliers!***

sns.regplot(x='horsepower', y='weight', data=mpg);



### **Interpretation**

❌ "Higher horsepower causes higher weight"

✅ "Powerful cars seem heavier"

* By how much? Measured by the **slope** of the line =
* β
* 1

✅ "Horsepower seems to explain a good deal of the weights' variations"

* How much? Measured by the **correlation coefficient**
* ρ
* ∈
* [
* −
* 1
* ,
* 1
* ]

round(mpg.corr(numeric\_only=**True**), 2)

|  | **mpg** | **cylinders** | **displacement** | **horsepower** | **weight** | **acceleration** | **model\_year** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **mpg** | 1.00 | -0.78 | -0.81 | -0.78 | -0.83 | 0.42 | 0.58 |
| **cylinders** | -0.78 | 1.00 | 0.95 | 0.84 | 0.90 | -0.50 | -0.35 |
| **displacement** | -0.81 | 0.95 | 1.00 | 0.90 | 0.93 | -0.54 | -0.37 |
| **horsepower** | -0.78 | 0.84 | 0.90 | 1.00 | 0.86 | -0.69 | -0.42 |
| **weight** | -0.83 | 0.90 | 0.93 | 0.86 | 1.00 | -0.42 | -0.31 |
| **acceleration** | 0.42 | -0.50 | -0.54 | -0.69 | -0.42 | 1.00 | 0.29 |
| **model\_year** | 0.58 | -0.35 | -0.37 | -0.42 | -0.31 | 0.29 | 1.00 |

*## R-squared (r2) is often preferred, from [0 to 1]*

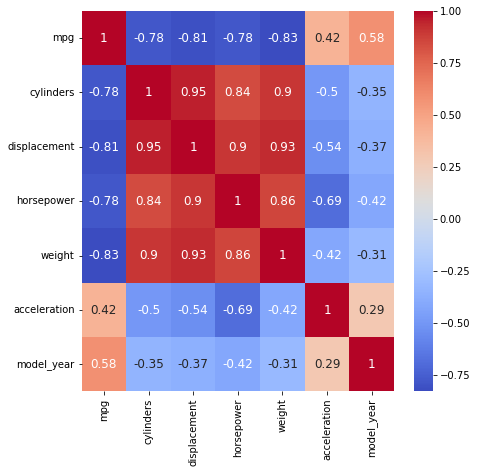
print('R-Squared = ', (mpg.corr(numeric\_only=**True**)['weight']['horsepower'])\*\*2)

R-Squared = 0.7474254996898221

plt.figure(figsize = (7,7))

sns.heatmap(round(mpg.corr(numeric\_only=**True**), 2),

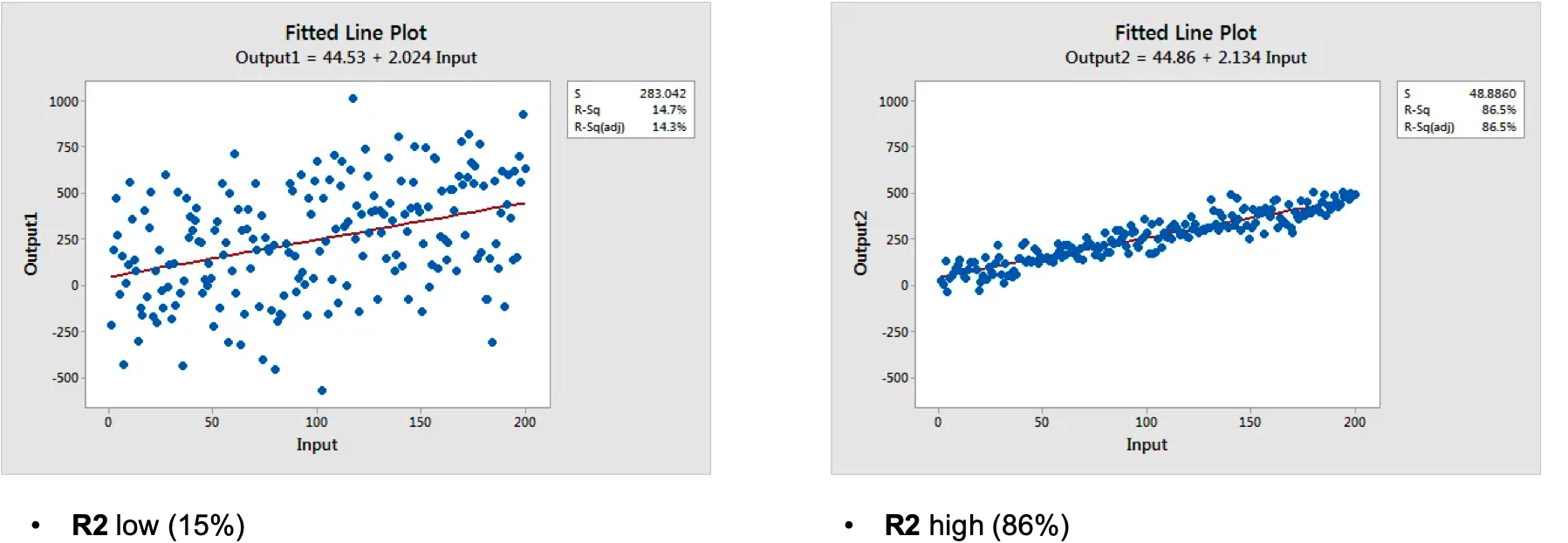
cmap="coolwarm", annot=**True**, annot\_kws={"size":12});



R

2

**(explanation of the variance)**

****

* % of the variance of weights that is explainable by the variance of horsepower
* Ranges from 0 (explains nothing) to 1 (perfect relationship)

📚 [Interpret R-squared - Statistics by Jim](https://statisticsbyjim.com/regression/interpret-r-squared-regression/)

❓*Am I confident that this relationship* ***generalizes*** *well to all car models in the world❓*

Measured by 👇

* confidence intervals
* p-values associated with hypothesis testing

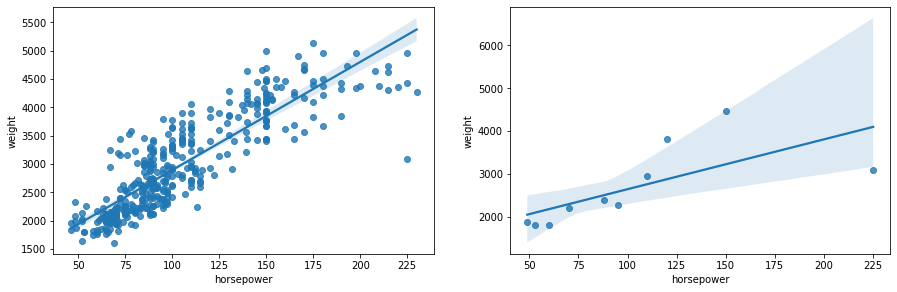
plt.figure(figsize=(15,10))

plt.subplot(2,2,1)

sns.regplot(x='horsepower', y='weight', data=mpg, ci=95)

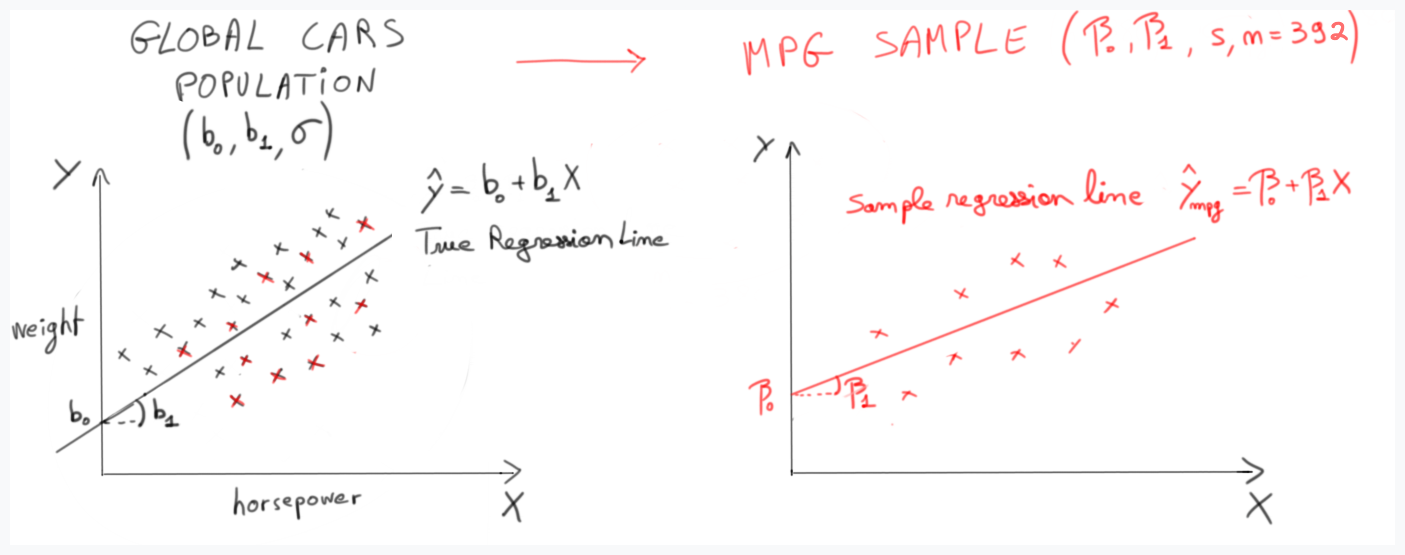
plt.subplot(2,2,2)

sns.regplot(x='horsepower', y='weight', data=mpg.sample(10, random\_state=6), ci=95);



Imagine for a moment the mpg dataset was made of only 10 cars, as per sampled above 👆

* **What if you had found a negative correlation** with horsepower❗️❓
* How much would you trust the regression coefficients ❗️❓



## **3. Simple Linear Regression (with statsmodels)**

### **Statsmodels**

👉 [statsmodels.org](http://www.statsmodels.org/stable/index.html)

pip install statsmodels

* Simple Linear ML models + Statistical Inference
* Very easy to use
* ~ Replace [**R**](https://www.r-project.org/) in Python

**Two ways to use statsmodels**

**"Standard" API**:

**import** **statsmodels.api** **as** **sm**

Y = mpg['weight']

X = mpg['horsepower']

model = sm.OLS(Y, X).fit() *# Finds the best beta*

model.predict(X) *# The Y\_pred (regression-line)*

**"Formula" API***(more intuitive)*:

**import** **statsmodels.formula.api** **as** **smf**

model = smf.ols(formula = 'weight ~ horsepower', data=data).fit()

Formula uses the [patsy](https://patsy.readthedocs.io/en/latest/formulas.html) syntax derived from R

*# Instantiate a model*

model = smf.ols(formula='weight ~ horsepower', data=mpg)

*# Train the model to find the best line*

model = model.fit()

model

<statsmodels.regression.linear\_model.RegressionResultsWrapper at 0x177bc4760>

### **Interpretation**

print(model.params)

Intercept 984.500327

horsepower 19.078162

dtype: float64

👉 Horsepower

β

1

: "For each increase of 1 horsepower, a car's weight increases on average by 19 lbs (pounds)"

👉 Intercept

β

0

: "a car with 0 horsepower would weigh 984 lbs"

model.rsquared

0.7474254996898198

👉 74% of the variance of weight is explained by the variance of horsepower

model.summary()

| **Dep. Variable:** | weight | **R-squared:** | 0.747 |
| --- | --- | --- | --- |
| **Model:** | OLS | **Adj. R-squared:** | 0.747 |
| **Method:** | Least Squares | **F-statistic:** | 1154. |
| **Date:** | Wed, 14 Sep 2022 | **Prob (F-statistic):** | 1.36e-118 |
| **Time:** | 16:17:41 | **Log-Likelihood:** | -2929.9 |
| **No. Observations:** | 392 | **AIC:** | 5864. |
| **Df Residuals:** | 390 | **BIC:** | 5872. |
| **Df Model:** | 1 |  |  |
| **Covariance Type:** | nonrobust |  |  |

|  | **coef** | **std err** | **t** | **P>|t|** | **[0.025** | **0.975]** |
| --- | --- | --- | --- | --- | --- | --- |
| **Intercept** | 984.5003 | 62.514 | 15.748 | 0.000 | 861.593 | 1107.408 |
| **horsepower** | 19.0782 | 0.562 | 33.972 | 0.000 | 17.974 | 20.182 |

| **Omnibus:** | 11.785 | **Durbin-Watson:** | 0.933 |
| --- | --- | --- | --- |
| **Prob(Omnibus):** | 0.003 | **Jarque-Bera (JB):** | 21.895 |
| **Skew:** | 0.109 | **Prob(JB):** | 1.76e-05 |
| **Kurtosis:** | 4.137 | **Cond. No.** | 322. |

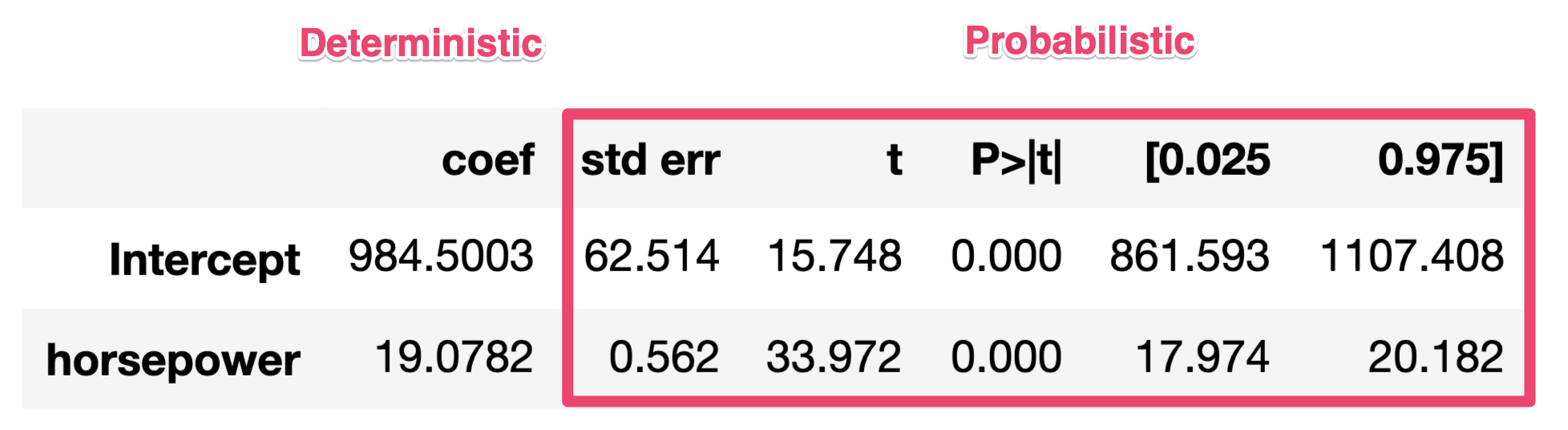
Notes:

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

### **Inferential Analysis: can I trust my coefficients**

### β

### **?**

****

📖 Under certain conditions (randomness of sampling, etc.):

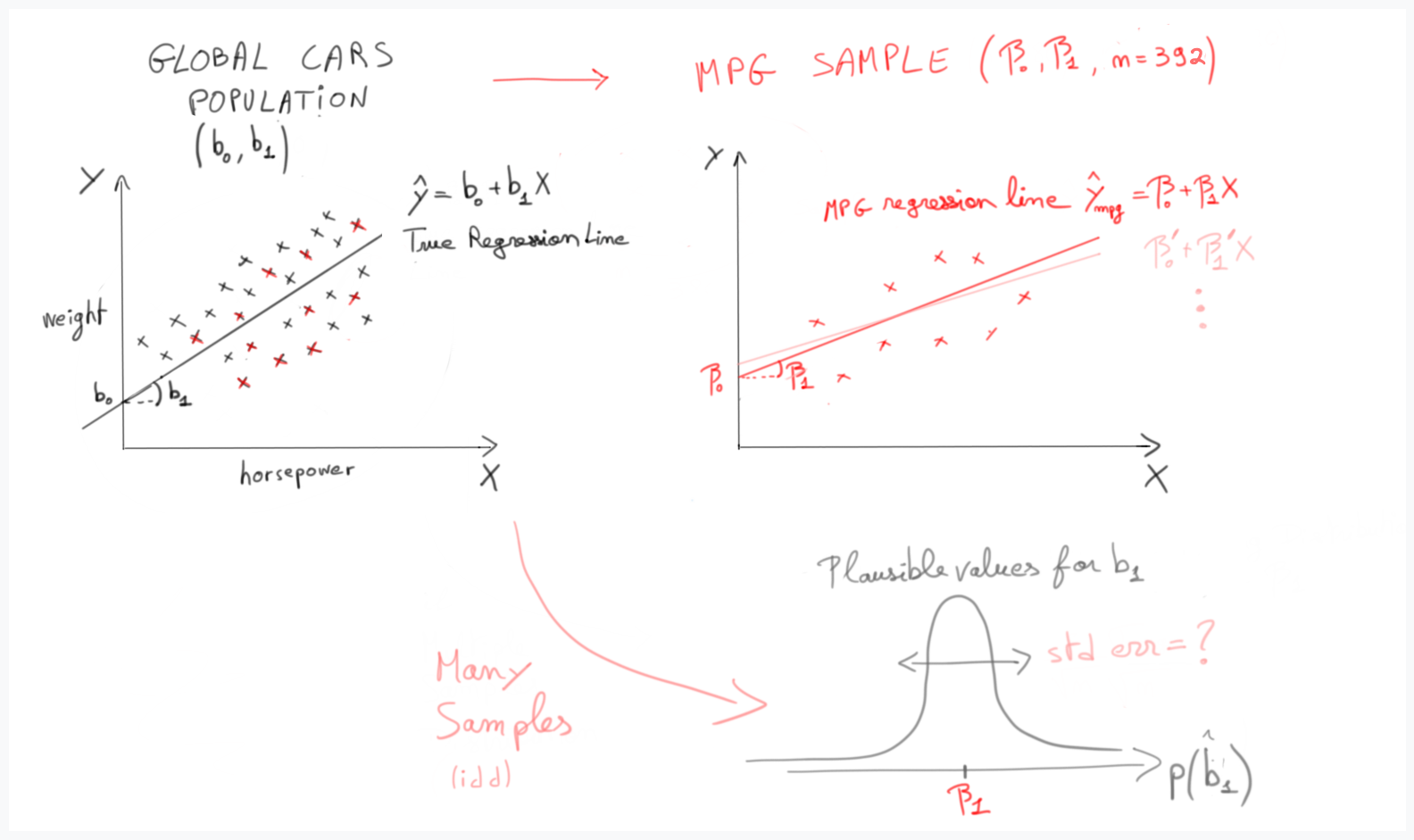
* The **Distribution of plausible values** for the real
* b
* 1
* can be **estimated** via the sample mpg
* ^
* b
* 1
* = 19.07 [17.9 - 20.1] with 95% confidence interval
* p
* (
* ^
* b
* 1
* )
* ∼
* T
* n
* =
* 390
* ∼
* N
* (
* c
* o
* e
* f
* =
* 19.07
* ,
* s
* t
* d
* e
* r
* r
* =
* 0.562
* )

**std err on the slope**

b

1

**?**

****

std err

(

b

1

)

=

std err

(

w

e

i

g

h

t

s

h

o

r

s

e

p

o

w

e

r

)

=

1

√

n

−

2

s

(

r

e

s

i

d

u

a

l

s

)

s

(

h

o

r

s

e

p

o

w

e

r

)

=

0.562

std err

(

b

1

)

=

1

√

n

−

2

s

(

r

e

s

i

d

u

a

l

s

)

s

(

h

o

r

s

e

p

o

w

e

r

)

=

√

1

n

−

2

∑

(

y

i

−

^

y

i

)

2

∑

(

x

i

−

¯

x

)

2

=

0.562

*# Let's check the formula ourself!*

n = 392

residuals = model.predict(mpg['horsepower']) - mpg['weight']

residuals.std() / mpg.horsepower.std() / (n-2)\*\*0.5

0.5615843732511717

p

(

^

b

1

)

∼

N

(

19

,

0.56

)

💡 Why divide by n-2? We divide by the number of degrees of freedom, which is n-2 in this case. Want to know more about it? Check out this [video](https://www.youtube.com/watch?v=4otEcA3gjLk)

➡️ The **t-statistic**, **p-value** and **95% confidence interval** correspond to the Null Hypothesis:

H

0

: In reality, horsepower is **not** correlated with weights (

b

1

=

0

)

If

H

0

were true, the observed

β

1

would have a t-score of:

t

=

β

1

−

b

1

std err

(

b

1

)

=

β

1

−

0

std err

(

b

1

)

=

19.0782

0.562

=

33.972

std deviations above mean!

p-value = "probability that what you observed is just due to pure chance"

= Proba of observing a sample slope of

β

1

=

19.0728

or bigger...

* **assuming that H0 is true**
* i.e. *assuming that the real slope*
* β
* 1
* *was actually 0*
* Since n = 392, we can use a Gaussian Distribution

= Proba of observing

β

1

>

19.0728

if it was sampled from a distribution

N

(

0

,

0.562

)

= Proba of observing t > 33.0927 from a standard distribution

N

(

0

,

1

)

≈

0

p-value

≈

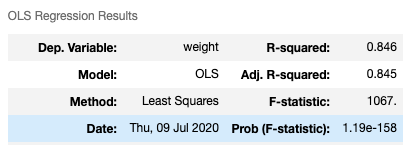
0

≤

< 0.05

* It is almost **impossible that the feature wouldn't be correlated** with the target variable
* The relationship between weight and horsepower is **statistically significant**

**F-statistic = overall statistical significance of the regression**

****

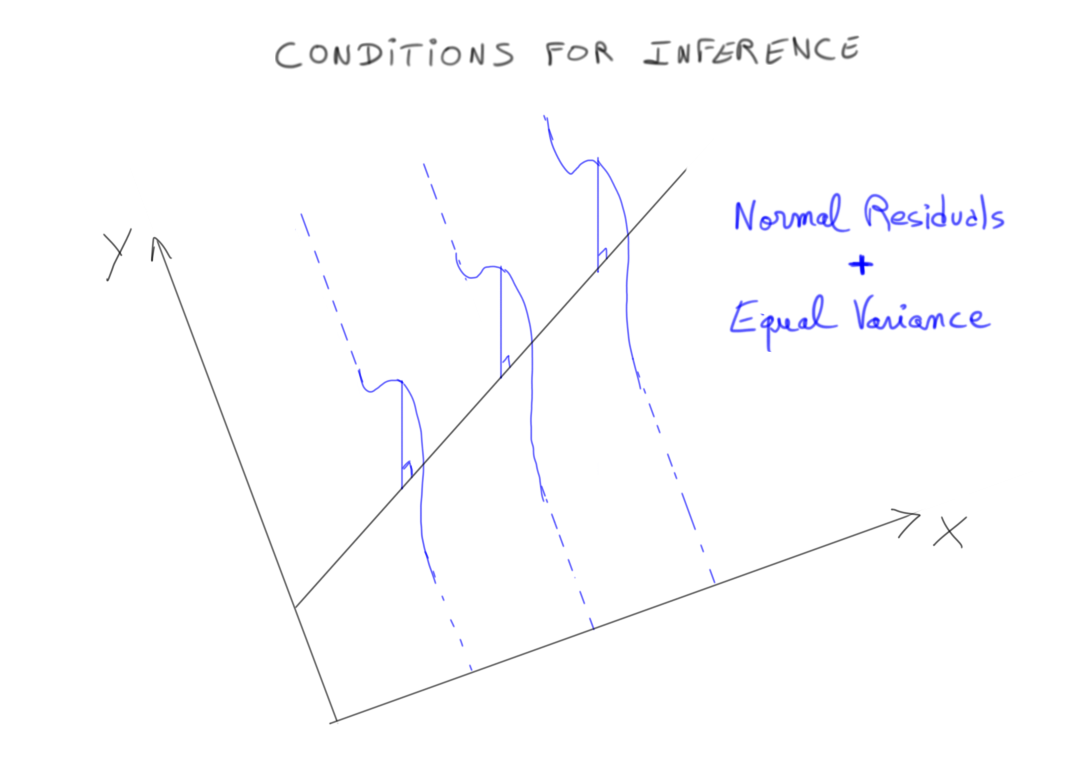
* The F-Statistic represents the combined p-value **of all your coefficients**
* It measures the null hypothesis
* H
* 0
* **: all coefs are null**
* F
* ⊂
* [
* 1
* ,
* ∞
* ]
* F
* ∼
* 1
* ⟹
* H
* 0
* cannot be ruled out
* F
* >>
* 1
* ⟹
* at least one coef p-value < 0.05
* F
* >>
* 1
* ⟹
* the regression is statistically significant

## **4. Checking the assumptions for inferential analysis**

✔︎ Random sampling

✔︎ Independent sampling (sample with replacement, or n < 10% global pop.)

⚠️ **Residuals normally distributed and of equal variance**

****

### **Are residuals normally distributed ?**

predicted\_weights = model.predict(mpg['horsepower'])

predicted\_weights

0 3464.661329

1 4132.396983

2 3846.224560

3 3846.224560

4 3655.442944

...

393 2625.222220

394 1976.564728

395 2587.065897

396 2491.675089

397 2548.909574

Length: 392, dtype: float64

residuals = predicted\_weights - mpg['weight']

residuals

*# also avaiable via model.resid*

0 -39.338671

1 439.396983

2 410.224560

3 413.224560

4 206.442944

...

393 -164.777780

394 -153.435272

395 292.065897

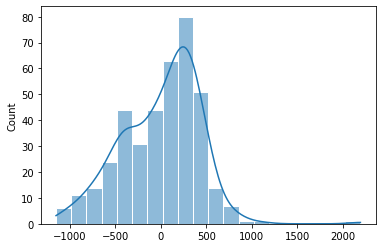
396 -133.324911

397 -171.090426

Length: 392, dtype: float64

*# visual check*

sns.histplot(residuals, kde=**True**, edgecolor='w');



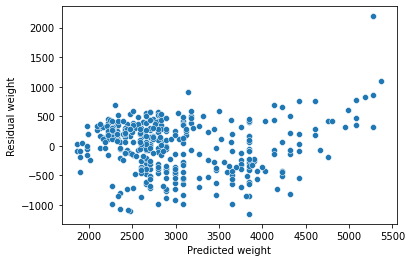
### **Are residuals of equal variance?**

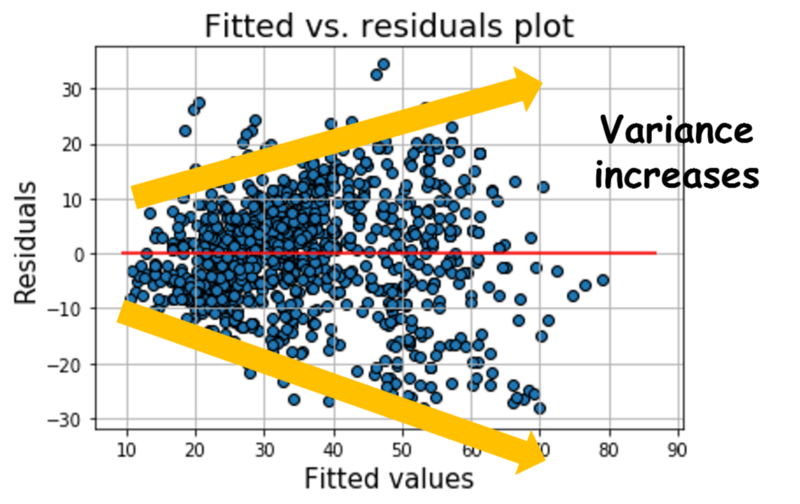
*# Check with Residuals vs. Fitted scatterplot*

sns.scatterplot(x=predicted\_weights, y=residuals)

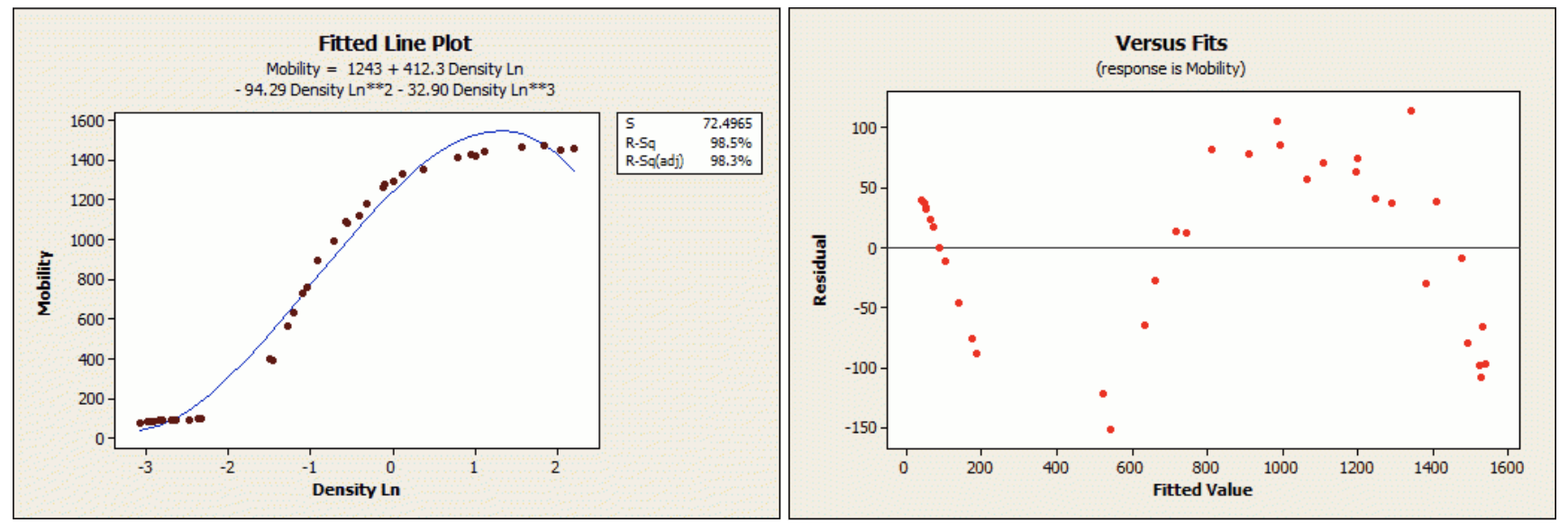
plt.xlabel('Predicted weight')

plt.ylabel('Residual weight');



👀 Beware of **heteroscedasticity**

👀 Beware also of **autoregressive residuals**

****

👉 If a pattern is seen, a factor might be missing in your model!  
👉 Frequent issue in Time Series (ex: inflation, weekly patterns etc.)

### **What if my residuals are really not random?**

✅ R-squared remains perfecly valid (deterministic coef)

⚠️ However, inferencial coefs cannot be trusted

* p-values and confidence intervals may be smaller than they should be
* Don't be too confident that your model generalizes well

💡 Fixes?

* Try to create/add new features that explain the residual patterns?
* Try to model a transformed version of Y instead (e.g. log(Y)...)?
* Try other statistical representations than linear ones (next module...)

## **5. Multivariate Linear Regressions**

✏️ Let's run a second OLS model where we regress weight on both horsepower and cylinders

w

e

i

g

h

t

∼

β

0

+

β

1

h

o

r

s

e

p

o

w

e

r

+

β

2

c

y

l

i

n

d

e

r

s

*# run OLS model*

model2 = smf.ols(formula='weight ~ horsepower + cylinders', data=mpg).fit()

model2.rsquared

0.8458154043882244

#### **R-squared**

84% of the variance in the cars' weights can be explained by the combined variations in horsepower and cylinders

⚠️ In order for the

R

2

to be meaningful, the regression must contain an "intercept" (i.e. the matrix X of features must contain a column vector of ones)

⚠️ Contrary to simple linear regression,

R

2

≠

C

o

r

r

(

Y

,

X

i

)

2

R

2

=

1

−

∑

(

y

i

−

^

y

i

)

2

∑

(

y

i

−

¯¯¯

y

)

2

=

1

−

S

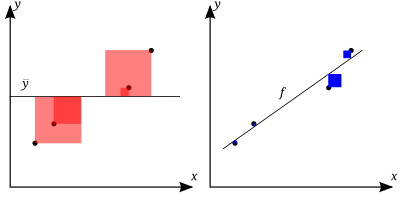
S

residuals

S

S

mean



❓

R

2

= by how much is my model better than a "simple mean" prediction ❓

🚀

R

2

=

1

best case scenario where the target is 100% explained by the features

🤨

R

2

=

0

simple mean

😱

R

2

<

0

can exist and in this worst case scenario, predicting the mean would be even better than running a Linear Model!

model2.params

Intercept 528.876711

horsepower 8.231070

cylinders 290.356425

dtype: float64

Each increase in horsepower increases the weight by 8, holding cylindersnumber constant.

Controlling for the cylinders number, each increase in horsepower increases the weight by 8 lbs

### **Partial regression plots**

Visualize your multivariate regression coefficients!

**import** **statsmodels.api** **as** **sm**

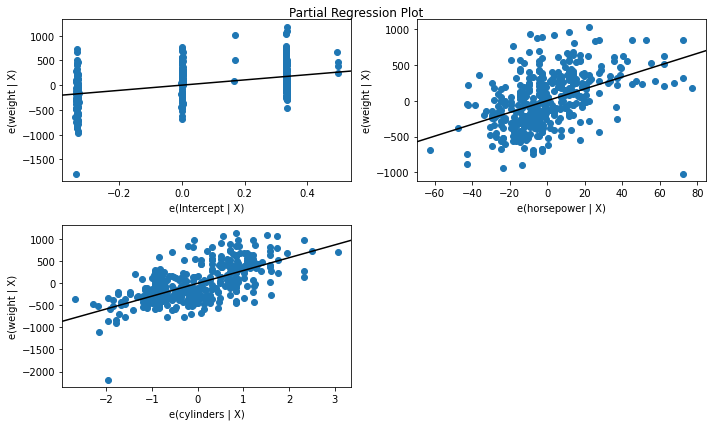
fig = plt.figure(figsize=(10,6))

fig = sm.graphics.plot\_partregress\_grid(model2, fig=fig)

eval\_env: 1

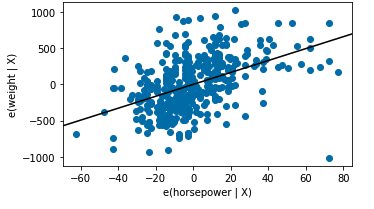
eval\_env: 1

eval\_env: 1



* Visualize the effect of a particular explaining variable on the dependent variable *while holding all other explaining variable constant.*
* You can see which variables have the strongest influence on **mpg**, after accounting for all other variables, by evaluating the slope of each chart in the grid.

**FYI How to construct partial regression plots ?**🤯



* Each point is a car in our dataset
  + Y values are the residuals of the predicted weights by using all features except horsepower (i.e. using cylinders)
  + These residuals contain the remaining information about weight that couldn't be explained without horsepower
* X value is the residual of predicting horsepower by using all other features (i.e. using cylinders)
  + These residuals contain the new information that horsepower brings to the table, which is not already explained by the other features in the model.

📚 [A good example](https://www.youtube.com/watch?v=Xii1jVLnX60&ab_channel=MikkoR%C3%B6nkk%C3%B6)

### **Categorical features?**

mpg['origin'].unique()

array(['usa', 'japan', 'europe'], dtype=object)

*# Use C(variable) in the formula*

model3 = smf.ols(formula='weight ~ C(origin)', data=mpg).fit()

model3.params

Intercept 2433.470588

C(origin)[T.japan] -212.242740

C(origin)[T.usa] 939.019208

dtype: float64

A car made in Japan is on average 212 lbs lighter than a European one

* When passing a categorical variable, *statsmodels* uses the first variable as the reference.
* The intercept is equal to the mean of the reference (here origin==europe)
* Each coefficient corresponds to the difference with the mean of the reference

mpg.groupby('origin').agg({'weight':'mean'})

|  | **weight** |
| --- | --- |
| **origin** |  |
| **europe** | 2433.470588 |
| **japan** | 2221.227848 |
| **usa** | 3372.489796 |

*# Drop the intercept if you want to*

model3 = smf.ols(formula='weight ~ C(origin) -1', data=mpg).fit()

model3.params

C(origin)[europe] 2433.470588

C(origin)[japan] 2221.227848

C(origin)[usa] 3372.489796

dtype: float64

## **Regression Diagnostic Cheat Sheet**

| **Check** | **Description** | **Diagnosis** |
| --- | --- | --- |
| Goodness-of-fit | The model explains a good deal of the observed variance of the dependent variable | R-square |
| Statistical significance | Can we trust the regression coefficients of the model - do they generalize? | p-values and F-statistic |
| Inference conditions | Random Residuals: zero-mean, constant variance, not correlated | Residual plots |

## **6. (Appendix) Mathematical Solution for OLS**

We want to model

Y

(**dependent variable, or target**) by a linear combination of multiple

X

i

(**independent variables, or features**)

→

Y

=

β

0

+

β

1

X

1

+

β

2

X

2

+

…

+

β

k

X

k

+

r

e

s

i

d

u

a

l

s

→

Y

=

X

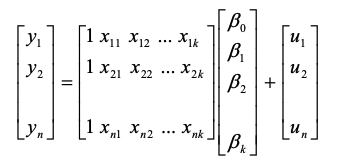
→

β

+

→

u



Ordinary Least Squares (OLS) finds the

β

that minimizes the Euclidian norm of the residuals

|

|

u

|

|

2

|

|

u

|

|

2

=

|

|

→

Y

−

X

→

β

|

|

2

Given the definition of the Euclidian norm

|

|

u

|

|

2

=

u

.

u

=

u

T

u

with

u

′

= transposed(

u

)

u

′

u

=

(

Y

′

−

β

′

X

′

)

(

Y

−

X

β

)

=

Y

′

Y

−

2

β

′

X

′

Y

+

X

′

X

β

2

Minimized when derivative equals 0

∂

∂

β

(

Y

′

Y

−

2

β

′

X

′

Y

+

X

′

X

β

2

)

=

−

2

X

′

Y

+

2

X

′

X

β

💡 Intuitively similar to 1-D derivative formula

−

2

X

′

Y

+

2

X

′

X

β

=

0

X

′

X

β

=

X

′

Y

β

=

(

X

′

X

)

−

1

X

′

Y

* (
* X
* ′
* X
* )
* −
* 1
* X
* '
* is called the "pseudo-inverse" of X
  + Exists only if
  + (
  + X
  + ′
  + X
  + )
  + is inversible
* Requires all features of
* X
* to be **independent** i.e **non-multicollinear**!
* i.e. X is full-rank matrix (
* r
* a
* n
* k
* (
* X
* )
* = number of features)

The **rank** of a matrix is the **dimension of the vector space** formed by its columns

X = np.array([

[1., 0., 1.],

[0., 1., 1.],

[0., 0., 0.]

])

np.linalg.matrix\_rank(X)

2

☹️

⟹

No single solution to OLS if two features are multicollinear

☹️

⟹

**Can't trust regression coefficients if the features are multicollinear**

**Computational Complexity?**

* Inverting
* X
* ′
* X
* with a basic techinque is of
* O
* (
* k
* 3
* )
* complexity
* Inverting
* X
* ′
* X
* with an advanced technique is of
* O
* (
* k
* 2.4
* )
* complexity
* Computing the pseudo-inverse
* (
* X
* ′
* X
* )
* −
* 1
* X
* '
* directly using SVD decomposition reduces the complexity to
* O
* (
* k
* 2
* )

👉 Not great for numerous features

k

👉 Great for numerous observation

n

as it scales proportionally with

O

(

n

)

## **Bibliography 📚**

* OLS Inference Assumption by [KDNuggets](https://www.kdnuggets.com/2019/07/check-quality-regression-model-python.html) and [Stats By Jim](https://statisticsbyjim.com/regression/ols-linear-regression-assumptions/)
* [StatsQuest - Linear Regression](https://www.youtube.com/playlist?list=PLblh5JKOoLUIzaEkCLIUxQFjPIlapw8nU) (1h Youtube intuitive video)
* [Statistics by Jim - Regression Analysis](https://statisticsbyjim.com/) (concise blog/book, intuitive)
* [Coursera, Regression Models](https://www.coursera.org/learn/regression-models) (4-week free video classroom)
* [G.James / D. Witten / T. Hastie / R. Tibshirani - An Introduction to Statistical Learning - Section 3 regression](https://www.statlearning.com/) (BSc level maths)

## **🚀 Your turn!**

* Challenge 01: Model review\_score of orders as a linear function of multiple explaining variables
* Challenge 02: Analyze which sellers (and products) are repetitively under-performers