

Data Analysis Lab

Big Data Analysis
Standardization
Normalization
Linear
transformation
Orthogonalization

Fátima Leal
2022/2023



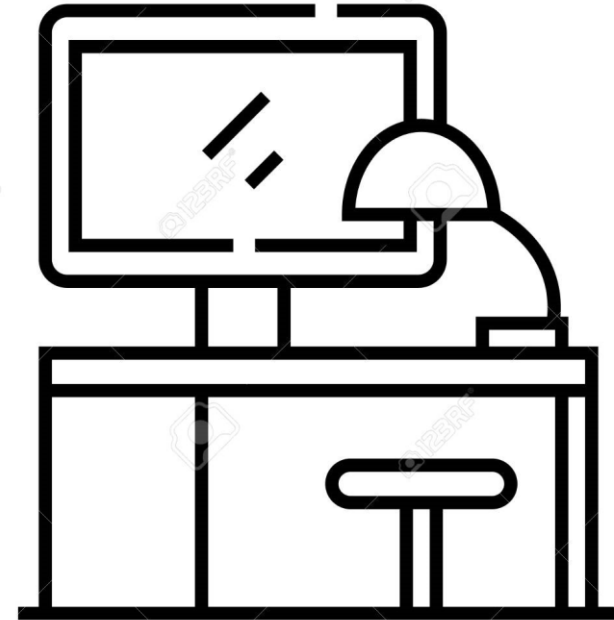
DEPARTAMENTO DE CIÊNCIA
E TECNOLOGIA



UNIVERSIDADE PORTUGALENSE

Previous Lesson

- Introduction to data analysis
- Statistical measures
- Graph Analysis

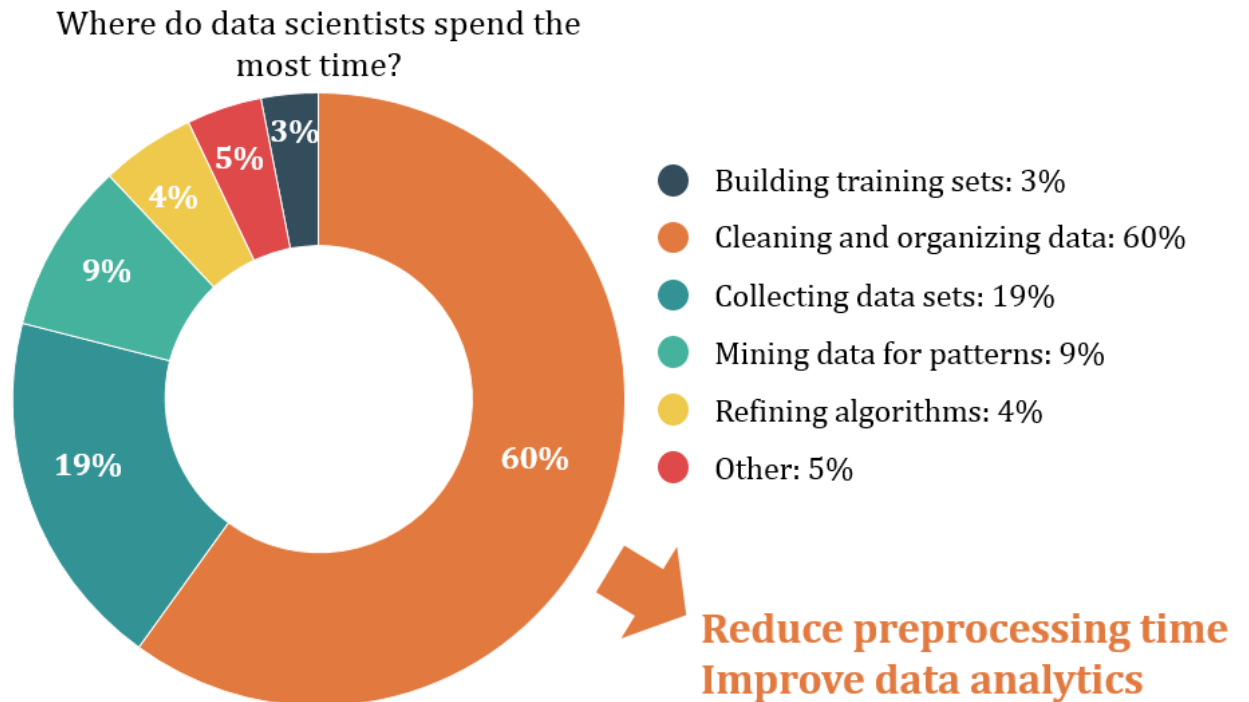


Outline

- Big Data Analysis
- Standardization
- Normalization
- Linear transformation
- Orthogonalization

Steps to data analysis

- Data collection
- Organization
- Pre-processing
- Transformation
- Modelling
- Interpretation



Looking Data: Relationships

- When examining the relationship between two or more variables, we should first think about the following questions:
- What features do the data describe?
- What variables are present? How are they measured?
- Which variables are quantitative and which are categorical?
 - **Quantitative** – age, weight, *i.e.*, Numerical values with magnitudes
 - **Categorical** – groups or categories, e.g., gender (F or M), Payment method (credit or cash)
- Is the purpose of the study to explore the nature of the relationship, or to show that one variable can explain other variable?

Data Preparation

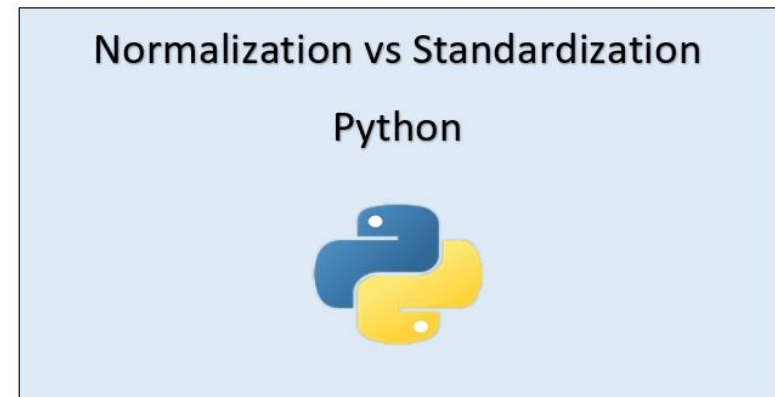
- If we want to employ a learning algorithm, we need to prepare our data.
- It is not uncommon for datasets to have some dirty data entries (i.e., samples, rows in CSV file, ...)
- Common Problems
 - Bad Character Encodings (Funny Characters)
 - Misaligned Data (e.g., row has too few/many columns)
 - Data in wrong format.

Data Preparation

- Blank/NaN entries do not work for Machine Learning!
- Need to replace the blank/NaN entry with something meaningful
- Delete the rows (generally not desirable)
- Replace with a Single Value
- Mean Average

Data Preparation: Feature Scaling

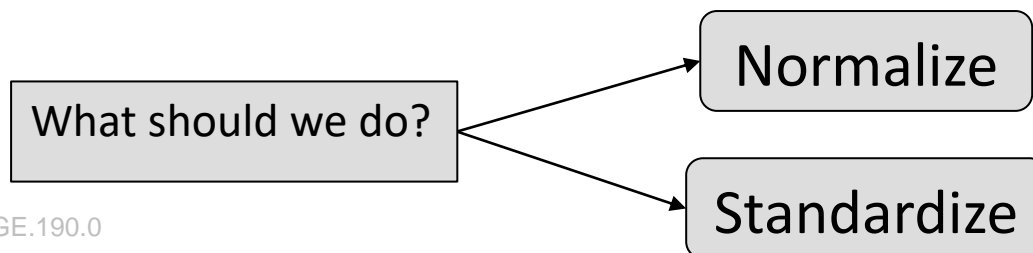
- If features do not have the same numerical scale will cause issues in models
- Sometimes, it is important to obtain the results within a specific range
- Feature scaling normalises a range of variables or features of data.
- In data processing, it is also known as data normalization and is generally performed during the data pre-processing step.
- To eliminate those problems, we use
 - Normalization
 - Standardization



Data Preparation: Feature Scaling

Hotel	TripAdvisor	Expedia	Hotel.com	XPTO
Scale	1-5	0-100	1-10	1-20
Porto Hotel	4	40	5	15
Lisbon Hotel	3	60	6	6
Coimbra Hotel	2	30	3	10
Faro Hotel	5	90	9	13
Lagos Hotel	4	67	10	15

Notice that in this example we have different scales for the same type of variable: rating



Data Preparation: Normalization

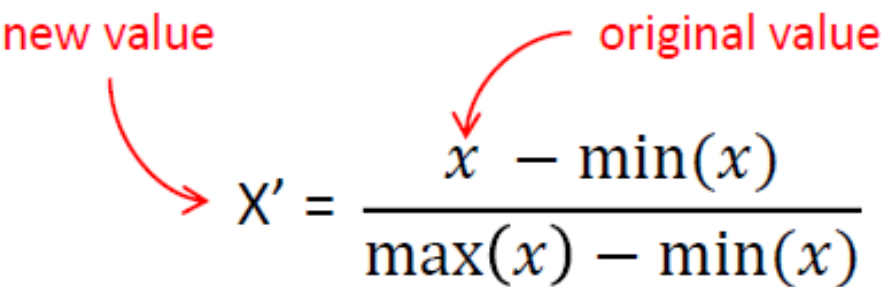
- Feature Scaling means scaling features to the same scale.
- Normalization scales features between 0 and 1, retaining their proportional range to each other.

Normalization

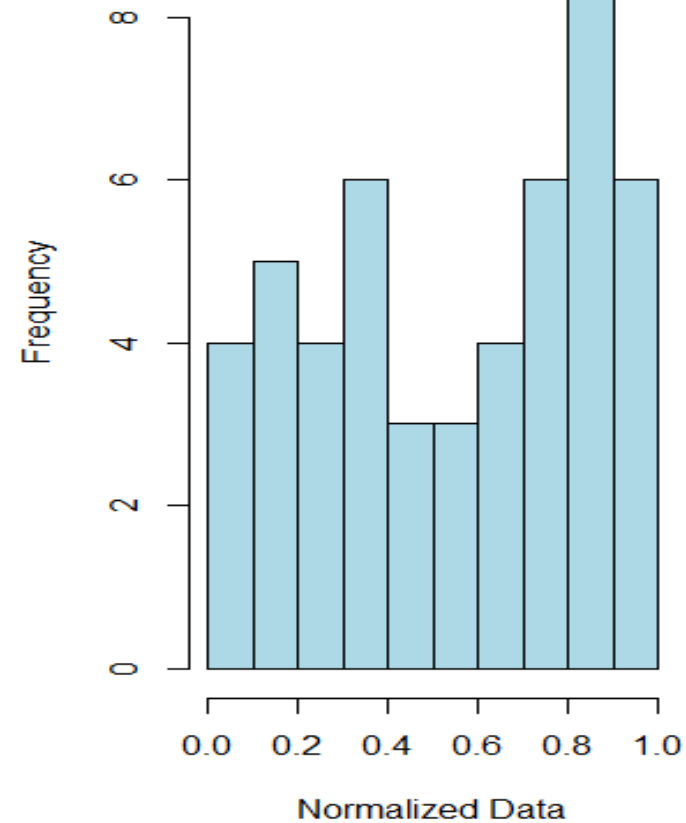
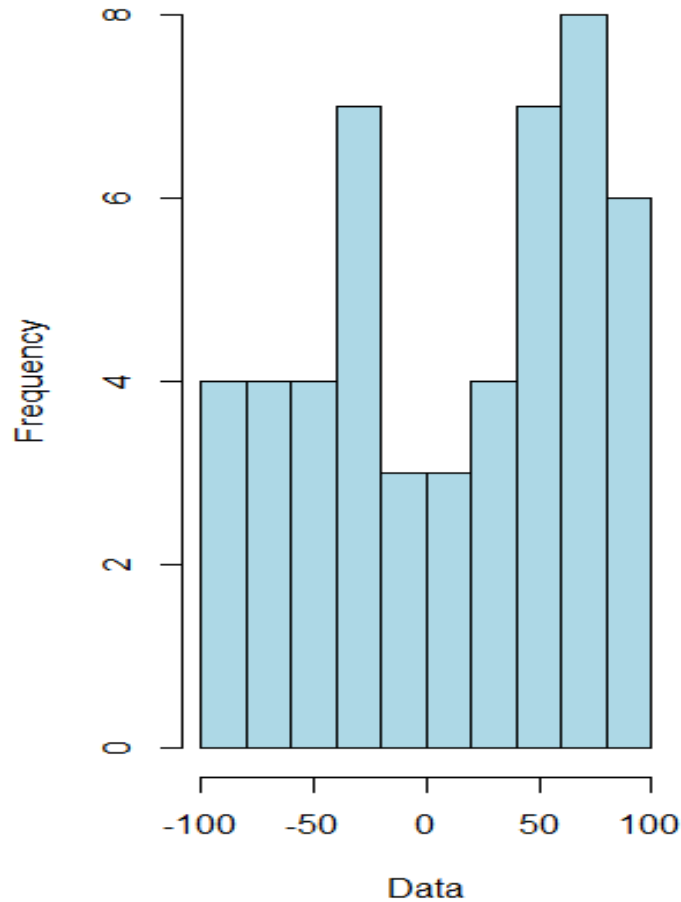
$$X' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

new value

original value



Data Preparation: Normalization



Data Preparation: Standardization

- Standardization scales features to have a mean (μ) of 0 and standard deviation (σ) of 1.

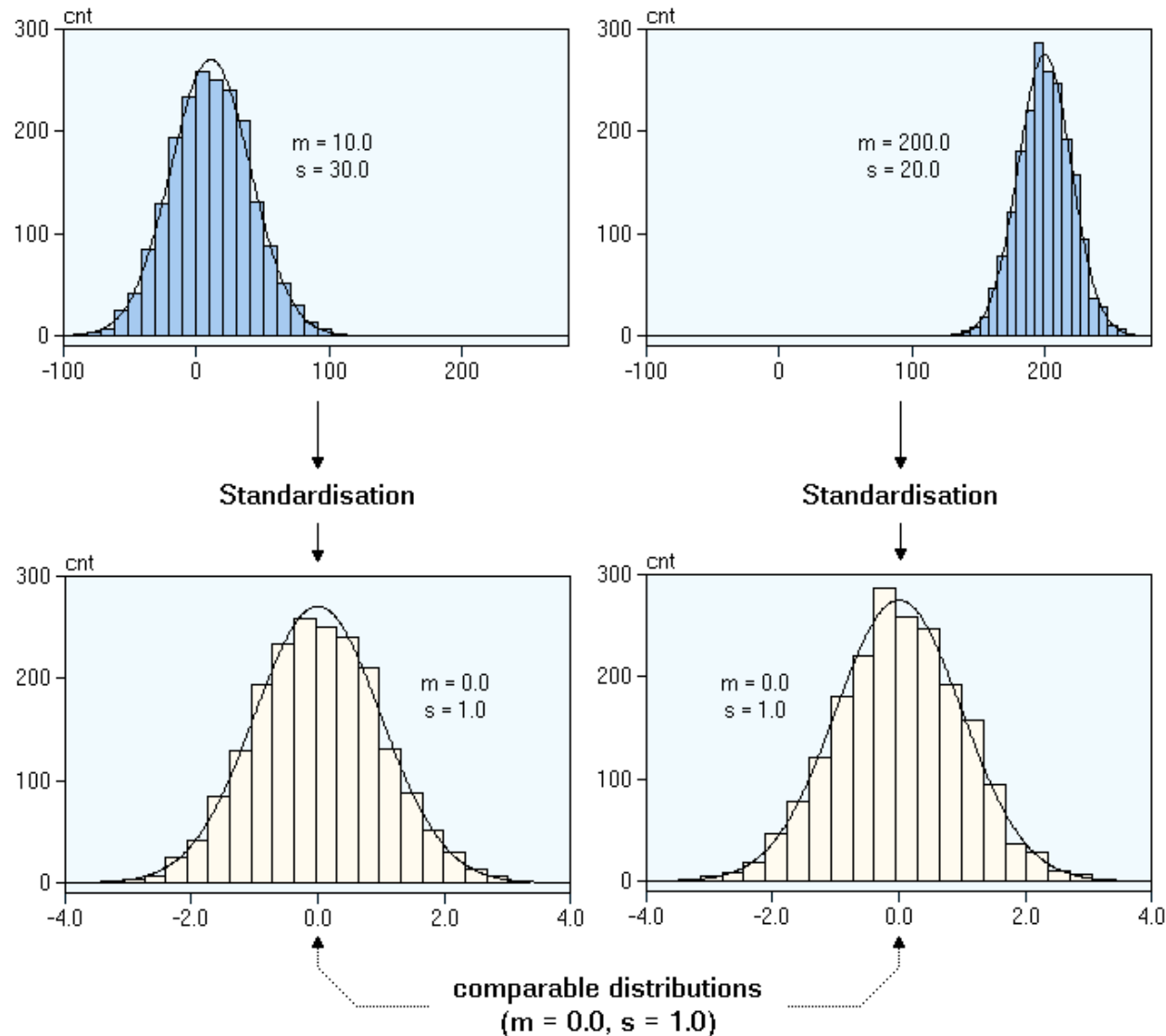
Standardization

$$X' = \frac{x - \mu}{\sigma}$$

Diagram illustrating the standardization formula with annotations:

- X' : new value
- x : original value
- μ : mean
- σ : standard deviation

Data Preparation: Standardization

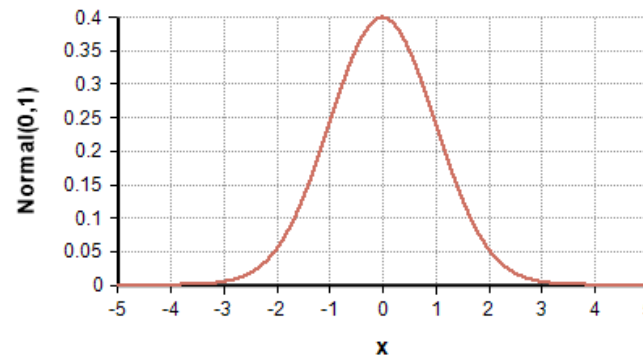


Normalization and Standardization

- Normalization makes the features more consistent with each other, which allows the model to predict outputs more accurately.
- Why does machine learning need normalization or standardization?
 - Machine learning models are math-based models with multiple variables. Having the variables in the same scale the performance will be better
- **Big questions: When to normalise or standardize?**
- **How it works in python?**

Normalization and Standardization

- When to normalise or standardize?
- **Normalization** is good to use when you know that the distribution of your data does not follow a Gaussian or normal distribution.
- **Standardization** can be helpful in cases where the data follows a Gaussian distribution.



- However, it is not always true. We should always analyse the best solution for our problem and the corresponding machine learning algorithm.
- It is not always required.

Normalization and Standardization

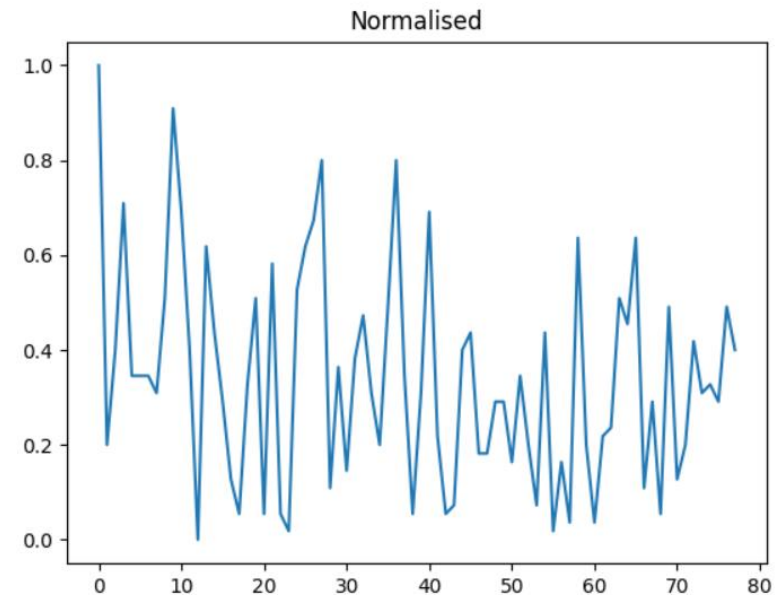
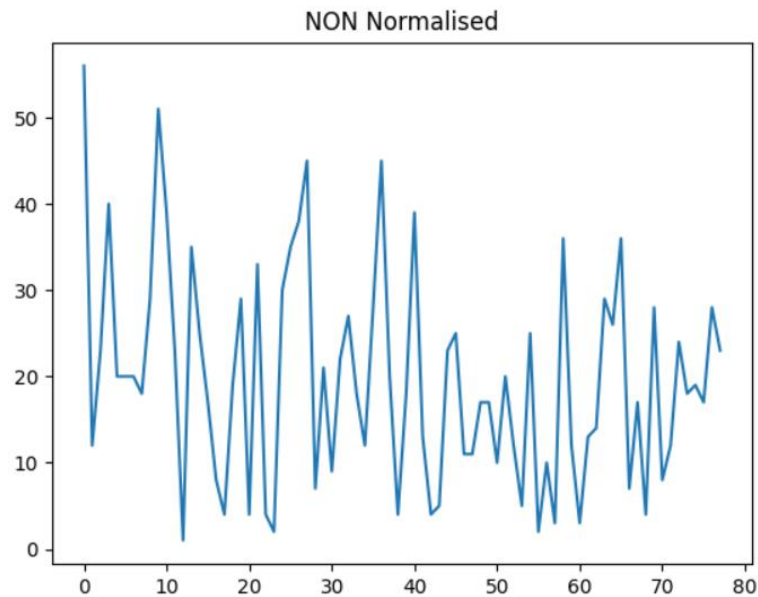
- **Normalize** using sklearn from Python
- To normalize your data, you need to import the `MinMaxScaler` from the `sklearn.preprocessing` library and apply it to our dataset. So, let's try that!

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt

df = pd.read_csv('Salaries.csv')
plt.plot(df['phd'])
plt.figure()

# fit scaler on data
normMinMax = MinMaxScaler()
# transform data
norm = normMinMax.fit_transform(df[['phd']].values)
print(norm)
plt.plot(norm)
plt.show()
```


Normalization and Standardization



Normalization and Standardization

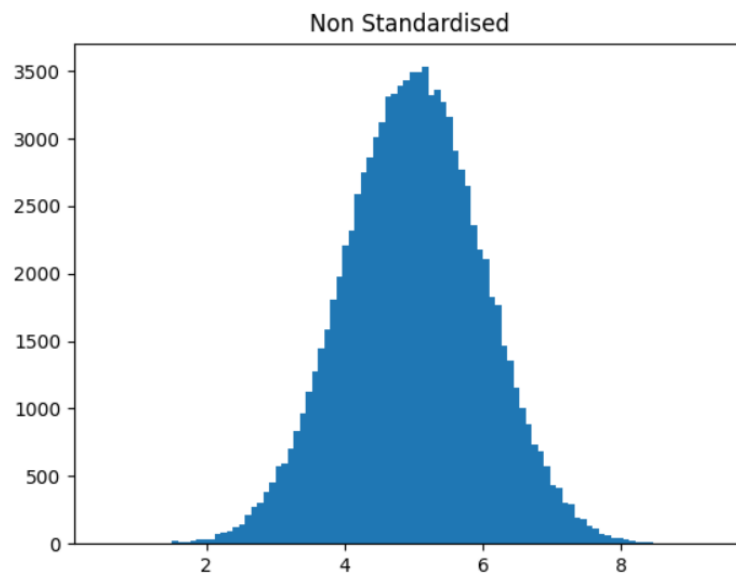
- **Standardizing** a dataset involves rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler

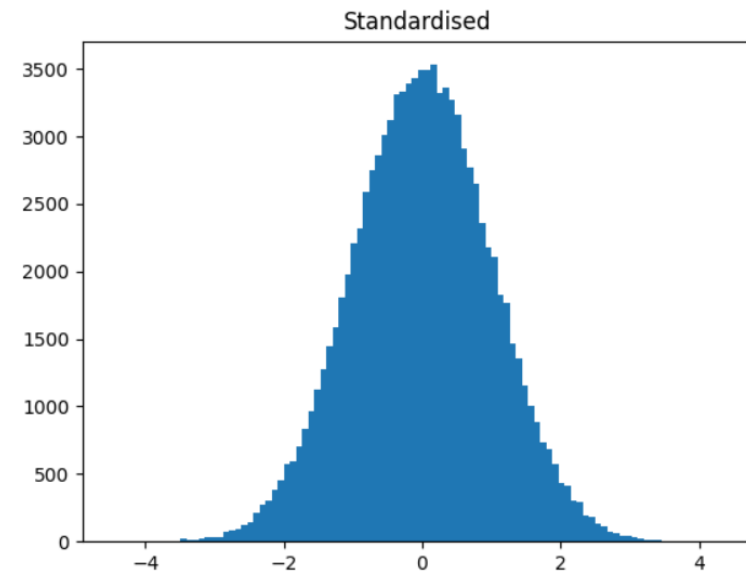
x = np.random.normal(5.0, 1.0, 100000)
x= np.array(x)
plt.hist(x, 100)
plt.figure()

scale= StandardScaler()
# standardization of dependent variables
scaled_data = scale.fit_transform(x.reshape(-1,1))
print(x.mean())
plt.hist(scaled_data,100)
plt.show()
```

Normalization and Standardization



m
a
el
•



Linear Transformation

- A linear transformation changes the original value x into a new variable x_{new}
- x_{new} is given by an equation of the form:

$$x_{new} = a + bx$$

- Example:
- A temperature x measured in degrees Fahrenheit can be converted to degrees Celsius by:

$$x_{new} = \frac{5}{9}(x - 32) = \frac{-160}{9} + \frac{5}{9}x$$

Linear Transformation

- Let us to come back to our ice cream shop and plot the Temperature

<i>Temperature</i>	<i>Number of Customers</i>
98	15
87	12
90	10
85	10
95	16
75	7

1. Add a simple linear transformation to the temperature like:
2. $\text{newT} = 2 + 3 * \text{df}[\text{'Temperature'}]$
3. Plot and analyse both graphs.

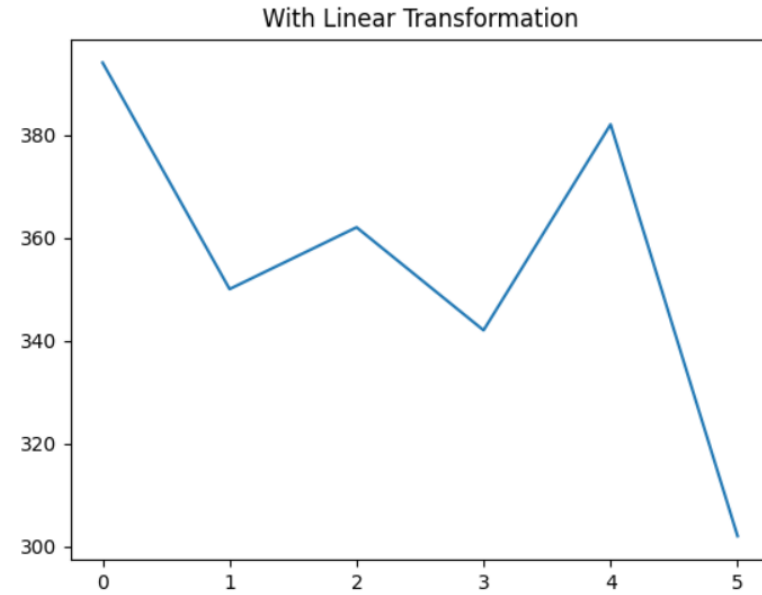
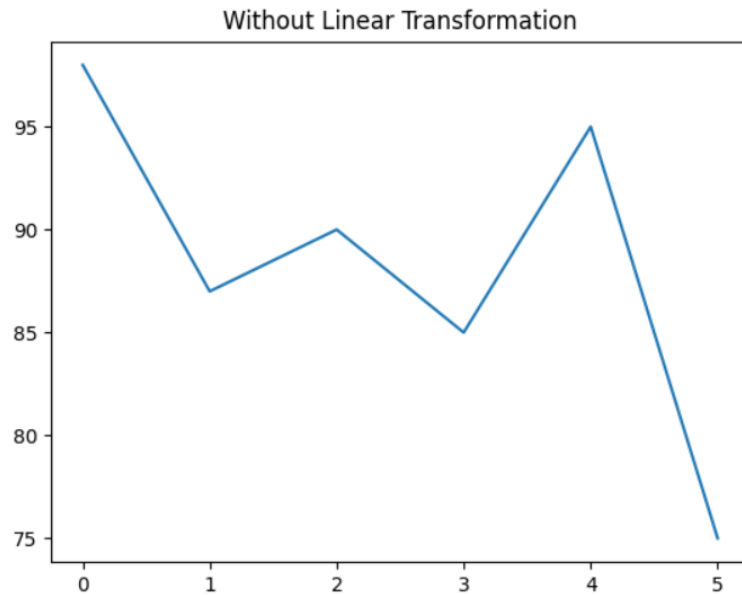
```
import pandas as pd
import matplotlib.pyplot as plt

data = {'Temperature':[98, 87, 90, 85, 95, 75],
        'N Customers':[15, 12, 10, 10, 16, 7]}
df = pd.DataFrame(data)

plt.plot(df["Temperature"])
plt.show()
```

What do we conclude?

Linear Transformation



Linear transformations do NOT change the overall shape of a distribution.

Linear Transformation

- A linear transformation preserves linear relationships between variables.
- Therefore, the correlation between x and y would be unchanged after a linear transformation.
- Let us continue with ice cream shop example:
- Calculate the correlation between temperature and the number of clients before and after the linear transformation of temperature
- What do you conclude?

```
print(df['Temperature'].corr(df['N Customers']))  
  
print(newT.corr(df['N Customers']))
```

Linear Transformation

- What is the transformation?
- A transformation applies a math operation to a variable

Original variable	Math operation	Transformed variable
X	Addition	$X_t = X + 5$
Y	Multiplication	$Y_t = 2 * Y$
A	Square root	$A_t = \sqrt{A}$
B	Reciprocal	$B_t = 1 / B$

Types of Transformation

Linear

- No effect on correlation

$$r_{YX} = r_{YX_t}$$

- Examples:

$$X_t = c * X$$

$$X_t = X / c$$

$$X_t = X + c$$

Non-linear

- Changes correlation

$$r_{YX} \neq r_{YX_t}$$

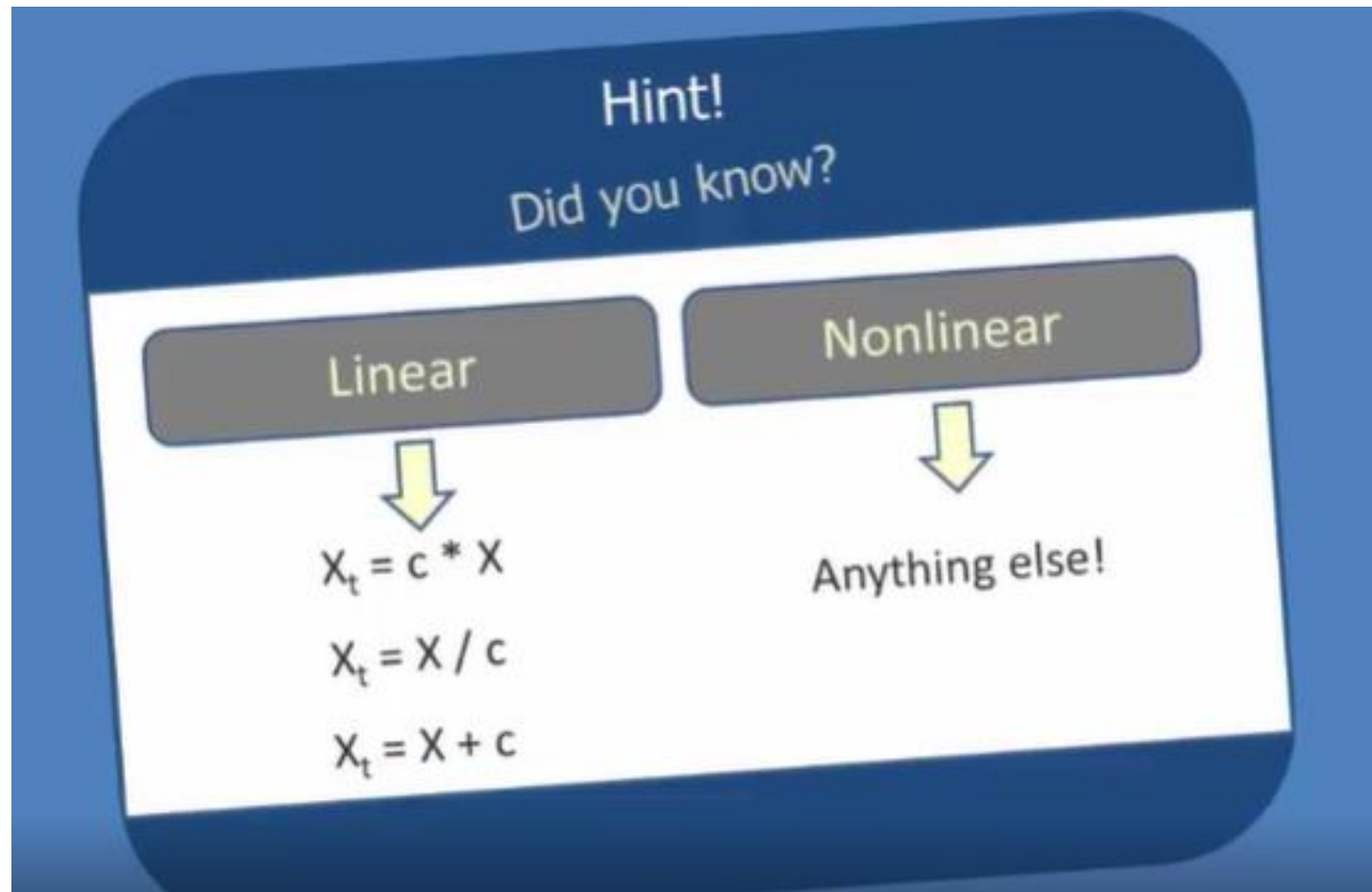
- Examples:

$$X_t = X^4$$

$$X_t = 1 / X$$

$$X_t = \log(X)$$

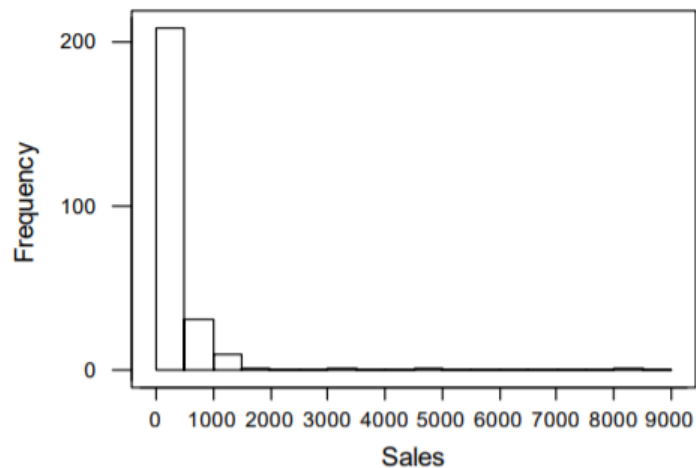
Types of Transformation



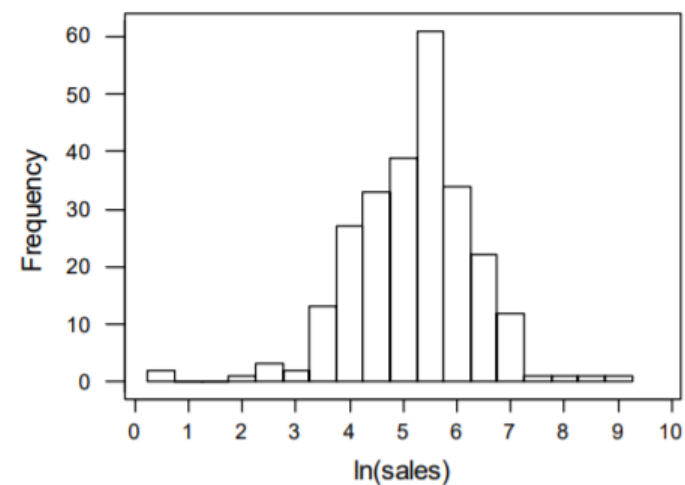
Non-Linear Transformation

- A very common nonlinear transformation in statistic is the logarithm transformation.

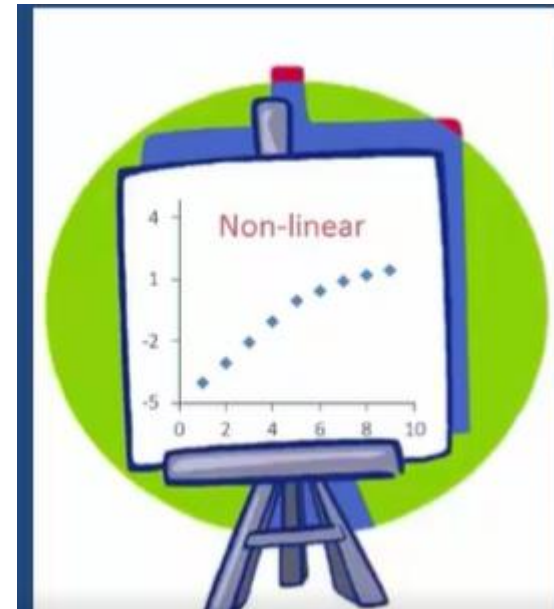
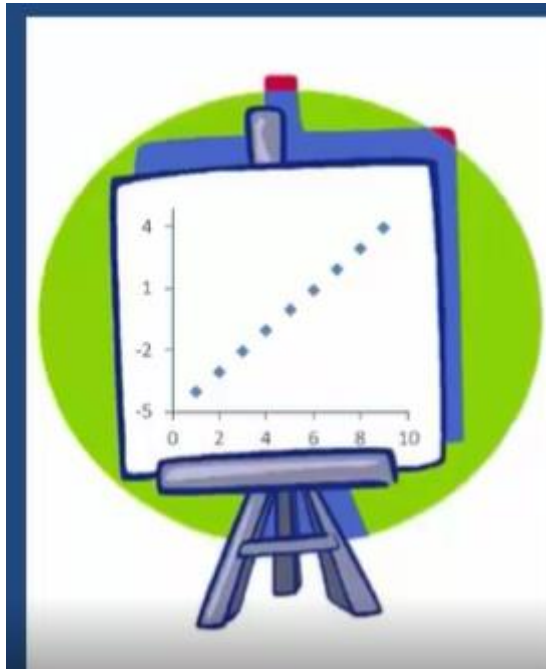
Histogram for sales data



Histogram for $\ln(\text{sales})$

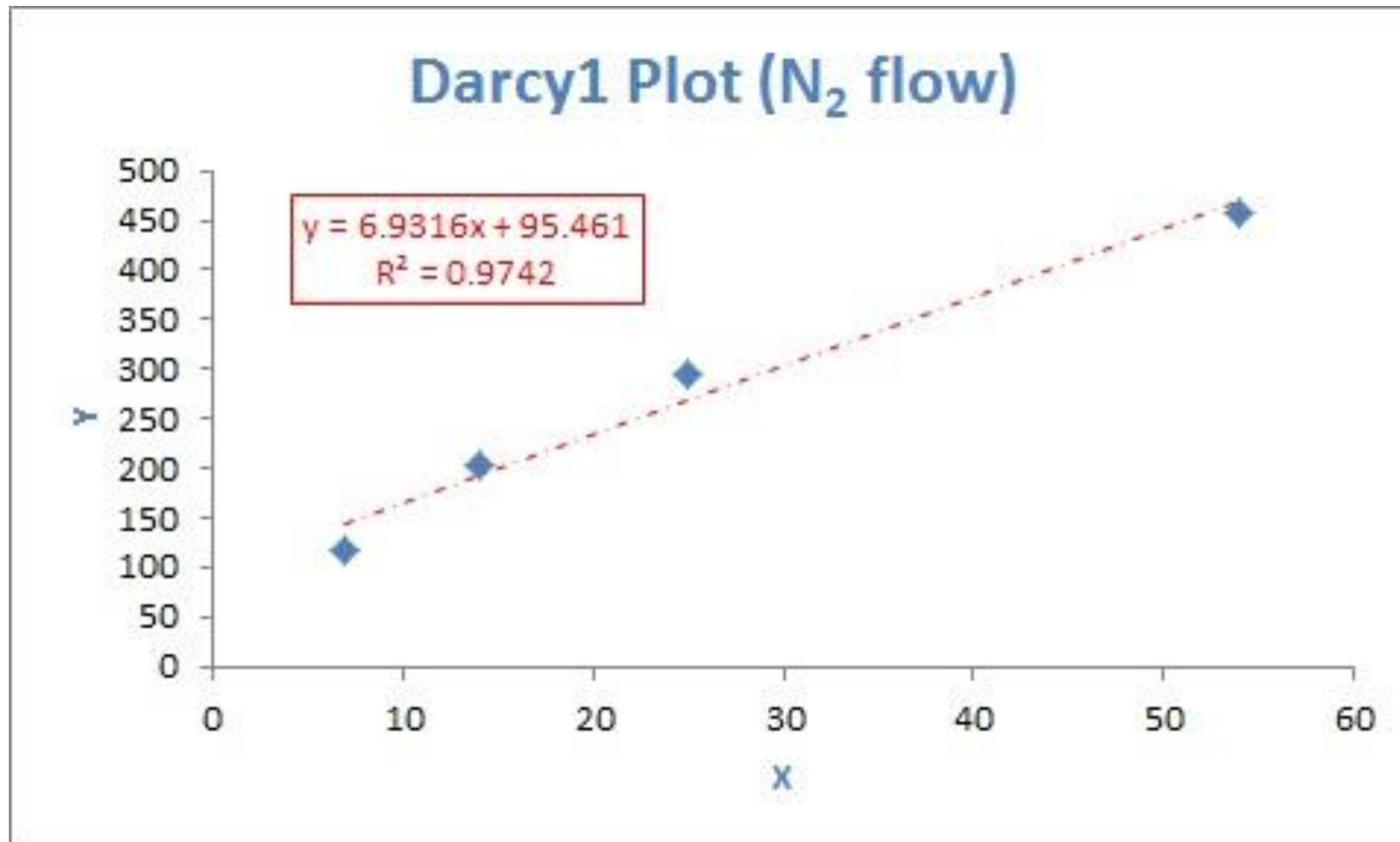


Linear Transformation



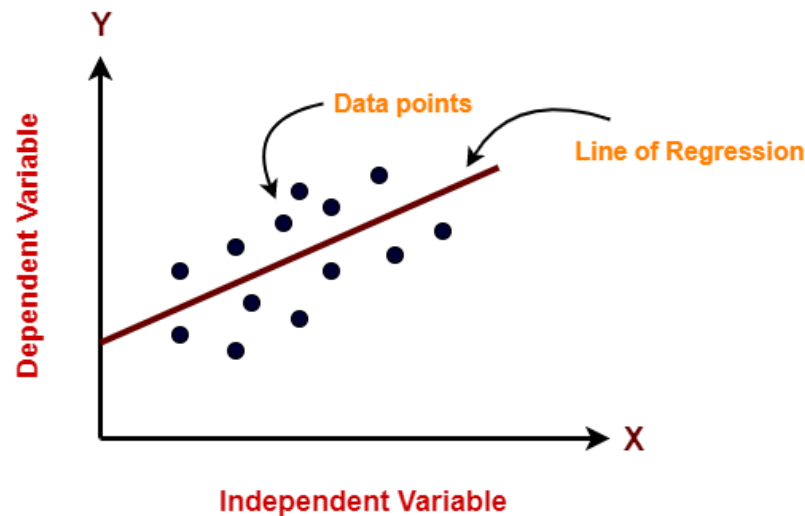
Transform raw data, *i.e.*, how to transform variables in regressions

Regressions

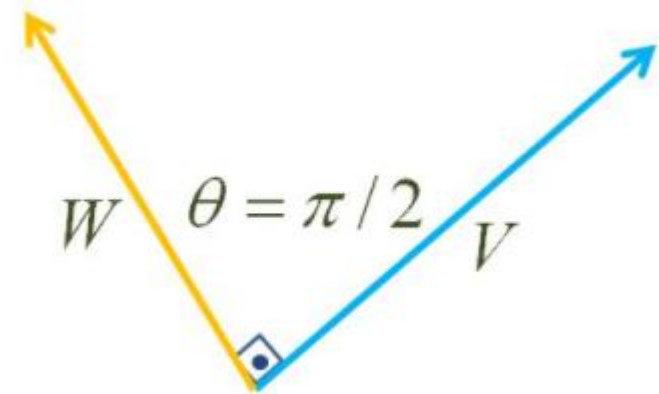
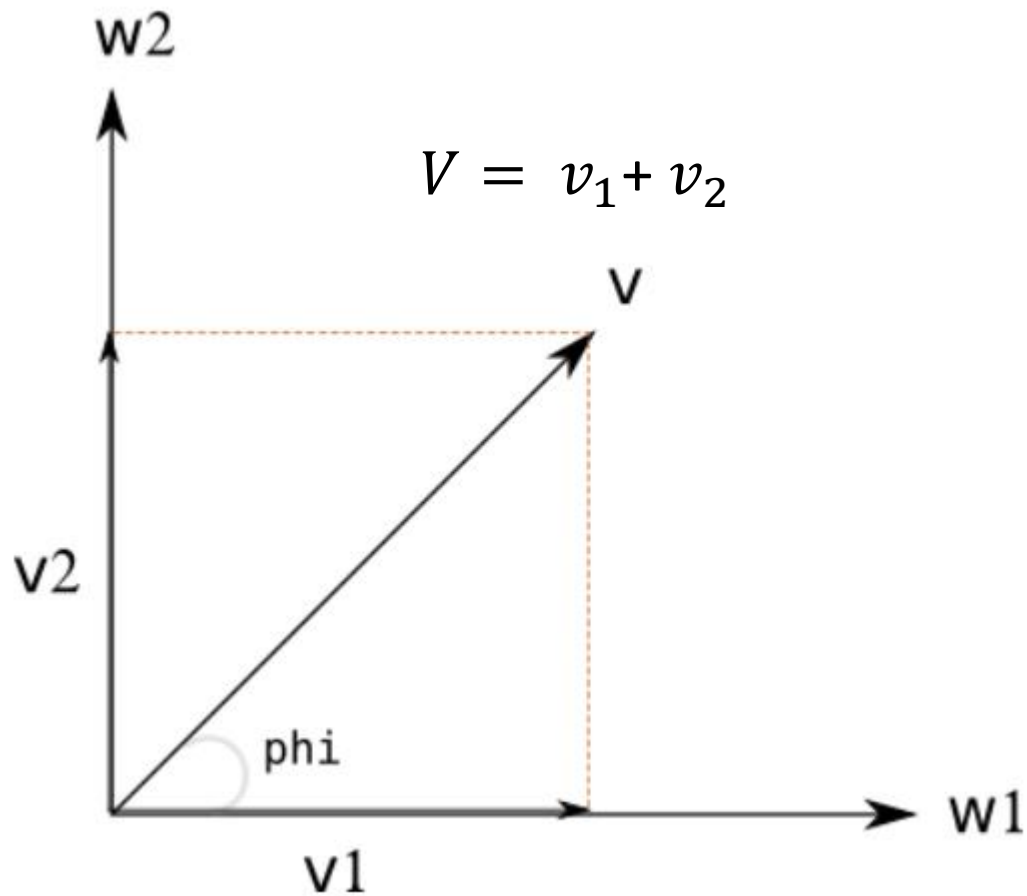


Response and explanatory variables

- A **response variable** measure an outcome of a study. An **explanatory variable** explains or causes changes in the response variables.
- Explanatory variables are often called **independent variables** and response variables are called **dependent variables**.
- The **dependent variables (y)** depend on **independent variables (x)**.



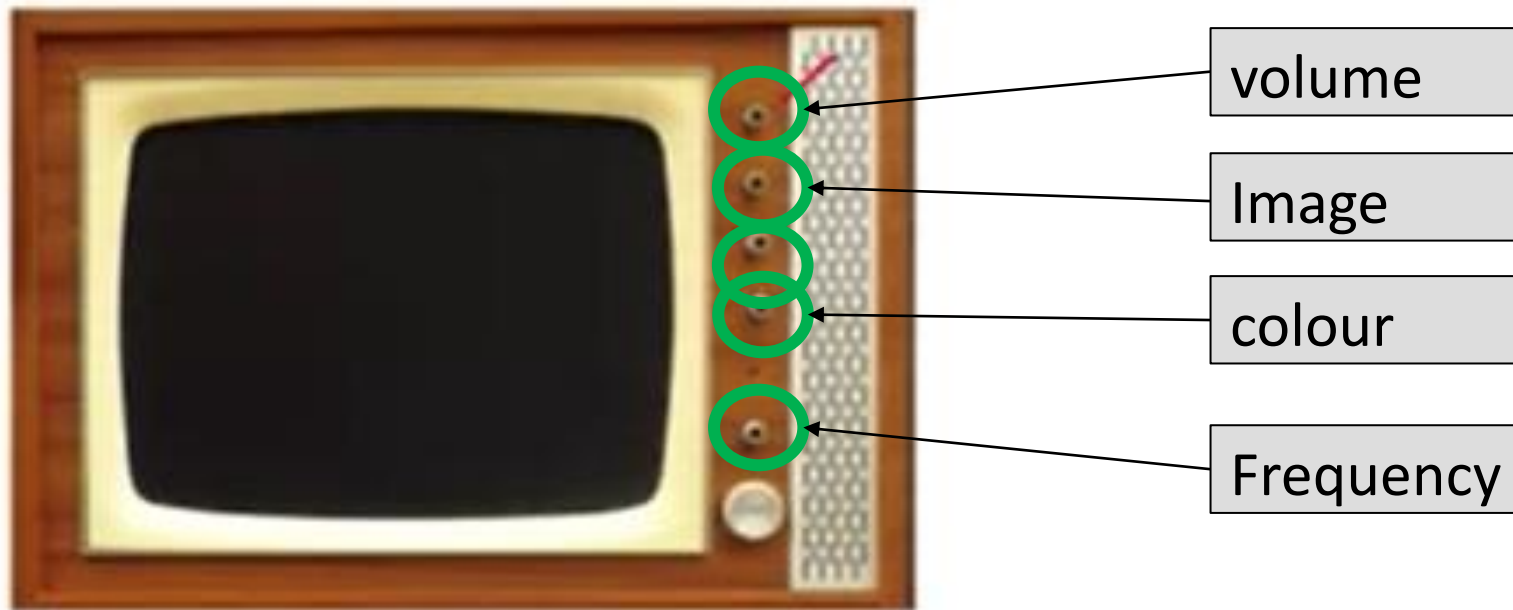
Orthogonalization



Orthogonalization

- One of the challenges in machine learning systems is that there's so many things you could try, so many things you could change.
- One of the things which allow to reduce the search space, i.e., obtain a clear-eyed about what to do in order to try to achieve one effect
- This is a process we call Orthogonalization.
- Let's take an example...

Orthogonalization



Imagine that you have a knob:



$1.7 * \text{volume} + 0.8 * \text{image} - 0.3 * \text{frequency} \dots$

Orthogonalization

- If you tune this knob, then the frequency, the fine tune, volume, equalizer etc, it all changes all at the same time.
- If you have a knob like that, it'd be almost impossible to tune the radio so that the radio gets tuned to the perfect noise-less TV channel to see a good image and sound.
- In this context, Orthogonalization can be associated to TV designers who designed the knobs where each knob kind does only one thing.
- Thus, it is much easier to tune the radio so that the radio gets tuned to the perfect noise-less radio channel.
- That's orthogonalization: a system design property that ensures that modification of an instruction or an algorithm component does not create or propagate side effects to other system components.

Orthogonalization

- In Machine Learning orthogonalization is almost the same.
- There are so many things we could try or change
- Parameter tuning is one of the area where we must pick right parameter to tune first from many possible ones.
- Orthogonalization is about what to tune to achieve one effect

Chain of assumptions in ML

Fit training set well on cost function



Fit dev set well on cost function



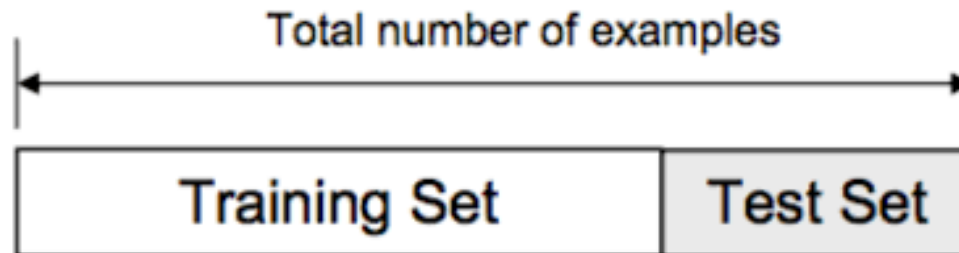
Fit test set well on cost function









Performs well in real world

Machine Learning

- **Training Dataset:** Examines the data and creates the model
- **Test Dataset:** Making a conclusion on how well the model performs.

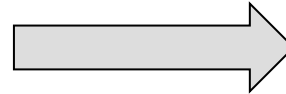


Machine Learning

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1 	8	1	?	2	7
User 2 	2	?	5	7	5
User 3 	5	4	7	4	7
User 4 	7	1	7	3	8
User 5 	1	7	4	6	?
User 6 	8	3	8	3	7

Machine Learning

x		y	Training set
1	2	0	
3	4	1	
5	6	1	
7	8	0	
9	10	1	
11	12	0	
13	14	0	
15	16	1	
17	18	1	
19	20	0	
21	22	1	
23	24	0	
x		y	Test set
17	18	1	
5	6	1	
23	24	0	
1	2	0	
3	4	1	
11	12	0	
15	16	1	
21	22	1	
7	8	0	
9	10	1	
13	14	0	
19	20	0	



Machine Learning Model

ML Results		
x		y
7	8	1
9	10	1
13	14	1
19	20	0

ML Results		
x		y
7	8	1
9	10	1
13	14	1
19	20	0

Machine Learning: Example

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

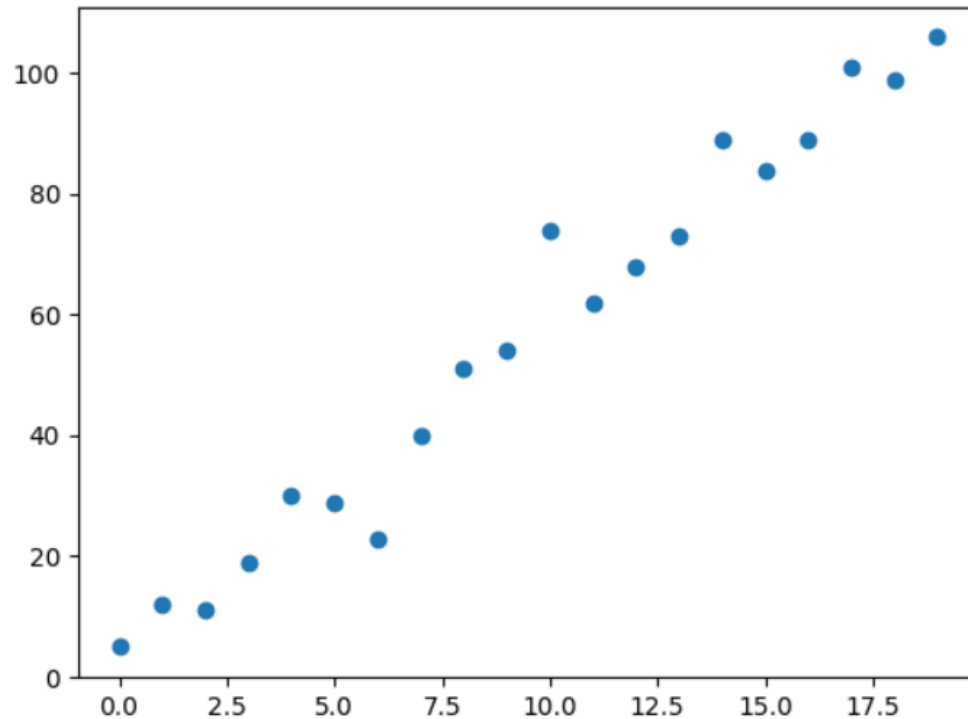
x = np.arange(20).reshape(-1, 1)
y = np.array([5, 12, 11, 19, 30, 29, 23, 40, 51, 54, 74,
              62, 68, 73, 89, 84, 89, 101, 99, 106])

plt.scatter(x,y)
plt.show()
plt.title("Original")
```

What do you conclude?

Analyse the correlations?

Machine Learning: Example



It is linear!

Let us apply a
regression Model.

Machine Learning: Example

```
x_train, x_test, y_train, y_test = train_test_split(x, y)
model = LinearRegression().fit(x_train, y_train)
print(model.score(x_test, y_test))
```

Apply the model

Split the data set

R^2

Machine Learning: Example

- Let us use the model

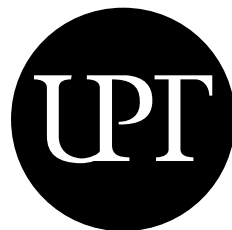
```
X_test = np.array([11, 16, 3, 9, 5]).reshape(-1,1) # put the
values which you want to predict
y_predict = model.predict(X_test)

print("Predicted values ", y_predict)
```

Machine Learning: Example

X	Real Y	Predicted Y
11	62	64
16	89	91
3	19	20
9	54	53
5	29	31





UNIVERSIDADE
PORTUCALENSE

Do conhecimento à prática.