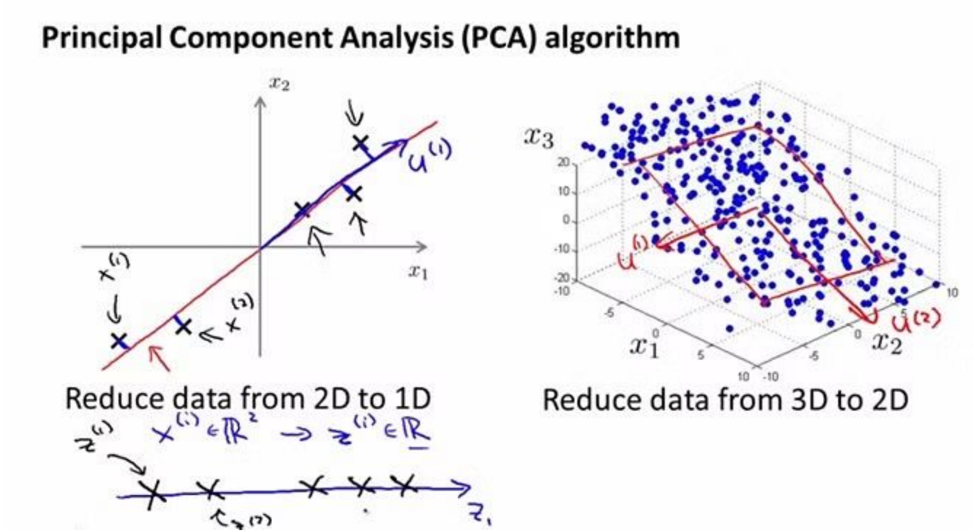
Università degli
Studi di Milano-Bicocca

PCA & Dashboards

Prof. Flavio Piccoli - Dr. Mirko Paolo Barbato

Principal Component Analysis (PCA)

- Mathematical procedure to **reduce the dimensionality** of a dataset (e.g. from 4200 variables to 5)
- dataset contains many variables correlated with each other
- PCA retains the variation present in the dataset, up to the maximum extent



- Transforms the variables to a new set of variables, which are known as the **principal components** (PCs)
- PCs are orthogonal (uncorrelated) to each other
- PC ordered such that the retention of variation present in the original variables decreases as we move down in the order

Image Source: Machine Learning Lectures by Prof. Andrew NG at Stanford University

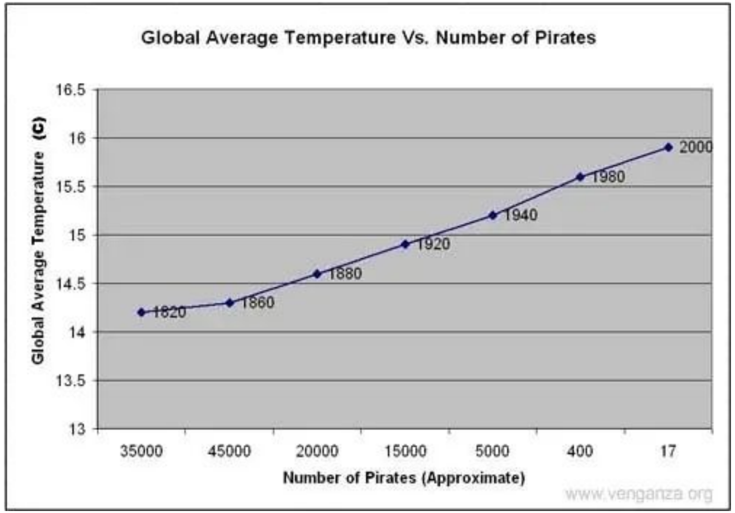
Terms

- **Dimensionality**
  - is the number of random variables (features) in a dataset

- **Correlation**
  - it shows how strongly two variables are related to each other
  - it is a value in the interval [-1, 1]
  - high positive correlation → variables are directly proportional
  - high negative correlation → variables are inversely proportional

  N.B.: Correlation does not imply causation!

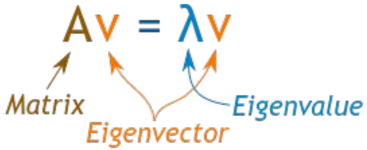
  - two variables can be highly correlated but have no relationship
  - https://www.tylervigen.com/spurious-correlations



- **Orthogonal variables**
  - uncorrelated to each other
  - correlation between any pair of variables is 0

Eigenvectors and eigenvalues

- **Eigenvectors**
  - Consider a non-zero vector v
  - v is an eigenvector of a square matrix A, if Av is a scalar multiple of v, i.e:



$$Av = \lambda v$$

  - where v is the eigenvector and λ is the eigenvalue associated to it.

Example:

For this matrix, $\begin{bmatrix} -6 & 3 \\ 4 & 5 \end{bmatrix}$, an eigenvector is $\begin{bmatrix} 1 \\ 4 \end{bmatrix}$ with a matching eigenvalue of 6. Let's check if it is true.

$$\boxed{\begin{bmatrix} -6 & 3 \\ 4 & 5 \end{bmatrix}}\boxed{\begin{bmatrix} 1 \\ 4 \end{bmatrix}} = \begin{bmatrix} -6 \times 1 + 3 \times 4 \\ 4 \times 1 + 5 \times 4 \end{bmatrix} = \begin{bmatrix} 6 \\ 24 \end{bmatrix} = \boxed{6}\boxed{\begin{bmatrix} 1 \\ 4 \end{bmatrix}}$$

matrix

eigenvector

eigenvalue

How do we find eigenvectors and eigenvalues?

- We start by finding the eigenvalue. Remember that:

$$Av = \lambda v$$

- We can put an identity matrix in the right part:

$$Av = \lambda I v$$

- Bring everything in the left side:

$$Av - \lambda I v = 0$$

- If v is hopefully non-zero, we can solve for lambda using the determinant:

$$|A - \lambda I| = 0$$

Finding the eivenvalues

- If v is hopefully non-zero, we can solve for lambda using the determinant:

$$|A - \lambda I| = 0$$

- Let's try with previous matrix:

$$\left| \begin{bmatrix} -6 & 3 \\ 4 & 5 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right| = 0$$

- Computing products and subtractions we obtain:

$$\begin{vmatrix} -6 - \lambda & 3 \\ 4 & 5 - \lambda \end{vmatrix} = 0$$

- Finally, we compute the determinant

$$(-6 - \lambda)(5 - \lambda) - 3 \times 4 = 0$$

Finding the eigenvalues

- Simplifies to:

$$\lambda^2 + \lambda - 42 = 0$$

- Solving, we obtain:

$$\lambda = -7 \; or \; 6$$

- These are two possible eigenvalues!

- Now, let's find the associated eigenvectors

Finding the eigenvectors

- Let's start by finding the eigenvector associated to the eigenvalue $\lambda = 6$
- We insert the eigenvector as unknown and solve the system to determine its values

$$\begin{bmatrix} -6 & 3 \\ 4 & 5 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 6 \begin{bmatrix} x \\ y \end{bmatrix}$$

- Solving, we obtain:

$$-6x + 3y = 6x$$
$$4x + 5y = 6y$$

- Taking everything on the left side:

$$-12x + 3y = 0$$
$$4x + 5y = 0$$

- Both equations show that:

$$y = 4x$$

Finding the eigenvectors

- So, for the eigenvalue $\lambda = 6$ there are many eigenvectors associated that respect the eq. $y = 4x$, e.g.:

$$\begin{bmatrix} 1 \\ 4 \end{bmatrix}, \begin{bmatrix} 2 \\ 8 \end{bmatrix} and \begin{bmatrix} 3 \\ 12 \end{bmatrix}$$

- Find the eigenvectors associated with the eigenvalue $\lambda = -7$

Finding the eigenvectors

- So, for the eigenvalue $\lambda = 6$ there are many eigenvectors associated that respect the eq. $y = 4x$, e.g.:

$$\begin{bmatrix} 1 \\ 4 \end{bmatrix}, \begin{bmatrix} 2 \\ 8 \end{bmatrix} and \begin{bmatrix} 3 \\ 12 \end{bmatrix}$$

- Find the eigenvectors associated with the eigenvalue $\lambda = -7$

$$x = -3y$$

$$\begin{bmatrix} -3 \\ 1 \end{bmatrix}, \begin{bmatrix} -6 \\ 2 \end{bmatrix}, \begin{bmatrix} -9 \\ 3 \end{bmatrix}, \dots$$

PCA

1. Normalize the data (standardization)
2. Calculate the covariance matrix (suppose only two variables x_1 and x_2)

$$Matrix(Cov) = \begin{bmatrix} Var[X_1] & Cov[X_1, X_2] \\ Cov[X_2, X_1] & Var[X_2] \end{bmatrix}$$

3. Calculate eigenvalues and eigenvectors of the covariance matrix
4. Order eigenvalues from largest to smallest (so that it gives the components in order of significance).
   a. dataset with n variables → n eigenvalues, n eigenvectors
   b. **We can reduce the number of variables by keeping only the most important**
5. Create a matrix composed by the corresponding eigenvectors

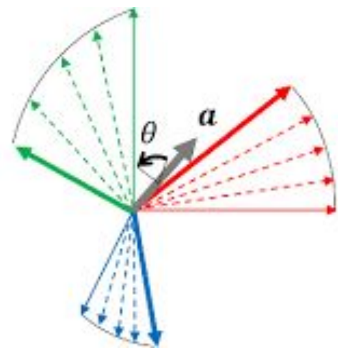
$$FeatureVector = \begin{bmatrix} eig_1 \\ eig_2 \end{bmatrix}$$

PCA

6. Get principal components of data

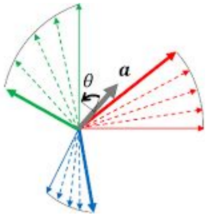
$$NewData = FeatureVector^T \times ScaledData^T$$

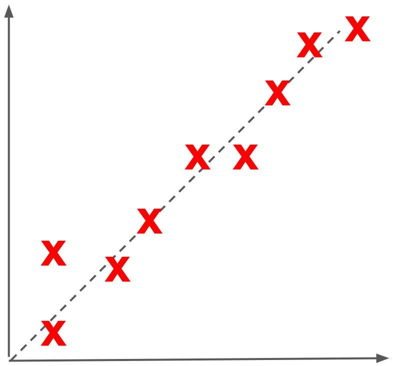
FeatureVector is also called **rotation matrix** as it changes the axis:

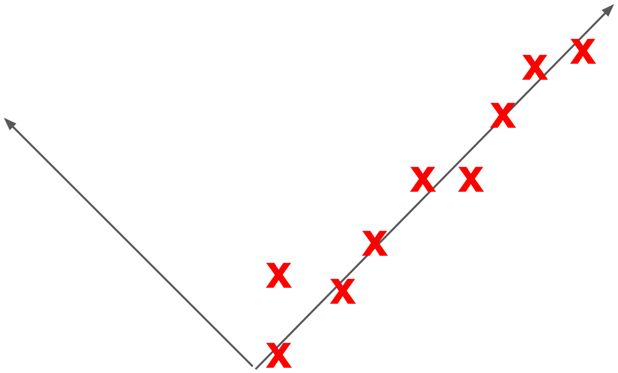


PCA

FeatureVector is also called **rotation matrix** as it changes the axis:

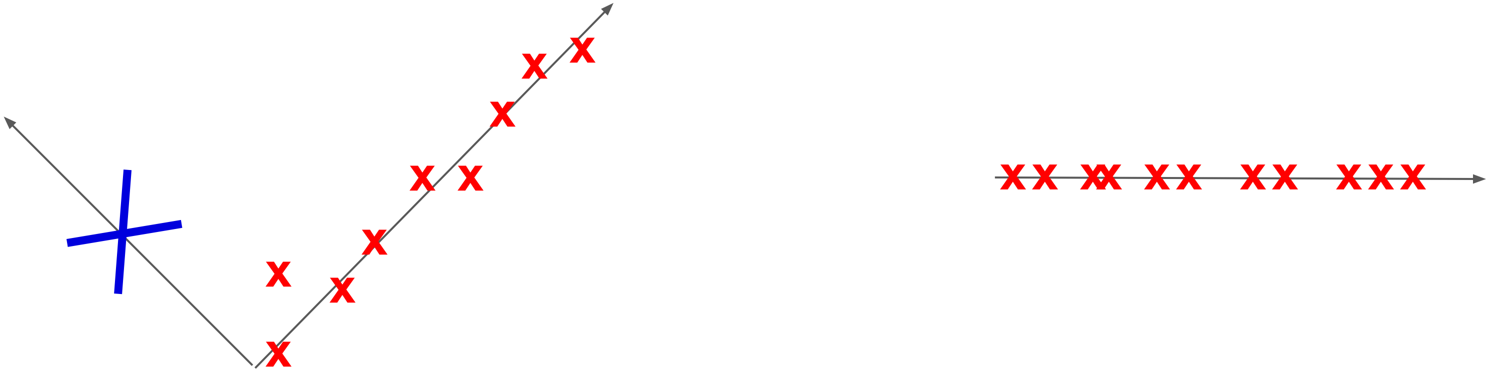






PCA for feature reduction

It is possible to use PCA to get rid of unuseful features



PCA in Python

- Sklearn package offer convenient function to compute PCA

sklearn.decomposition.PCA(n_components = ...)

- **n_components** can be:
  - **a float**: specifies the retained variance to keep
  - **an integer**: specifies the number of principal components to keep

- It follows the schema of all skearn objects
  - fit: train the object on the specified data
  - transform: use the fitted object on new data
  - fit_transform: performs both the operations together

Exercise 1 - Mastering PCA

1. Download the validation set of the Lucas dataset (file lucas_dataset_val.csv)

2. Compute the PCA of this set
   - Use a retained variance of 0.9.
   - Print the number of principal components survived.

3. Invert the PCA transformation (*pca.inverse_transform*)

4. Plot the first original sample together with the same sample inverted (*with Matplotlib*)

5. Repeat the process with a retained variance of 0.99, then with 0.9999. Finally, try with only one component.

Exercise 1 - Mastering PCA

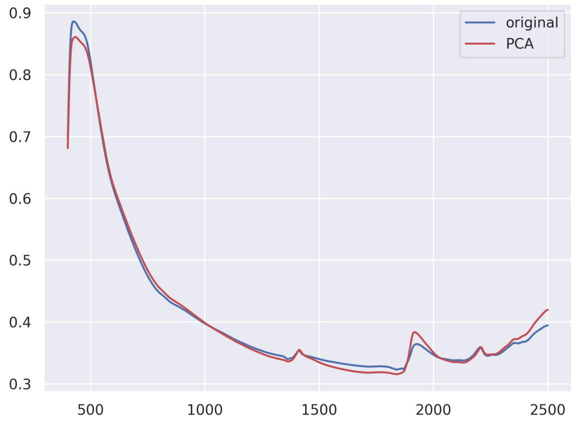
1. Download the validation set of the Lucas dataset (file lucas_dataset_val.csv)

2. Compute the PCA of this set
   - Use a retained variance of 0.9.
   - Print the number of principal components survived.

3. Invert the PCA transformation

4. Plot the first original sample together with the same sample inverted

5. Repeat the process with a retained variance of 0.99, then with 0.9999. Finally, try with only one component.



Dashboarding

R&D process

Analysis of the state of art

↓

Data collection / analysis

↓

Architectural Design

↓

Parameter Search

↓

Benchmarking

↓

Deployment

Deployment

How to provide the service?

Three strategies:

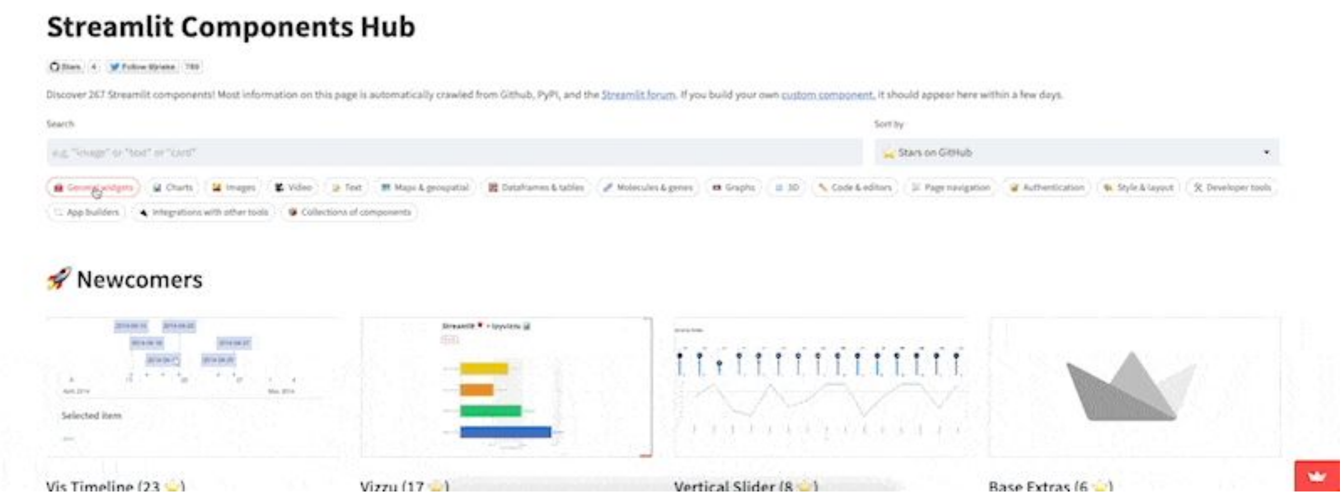
- **As-a-Service**: your forcasting model will be provided as a remote service
- **Product Integration**: your model will be integrated inside a product
- **Standalone product**: your method will be a product itself dispensed through a dashboard

Streamlit

- It offers a powerful set of layouts and widgets
- useful for creating a highly-interactive GUI
- To install streamlit:

```
pip install streamlit
```

- You can search streamlit components on Streamlit itself! *inception*



https://components.streamlit.app/

Layouts

Sidebar


```python
import streamlit as st

with st.sidebar:
    add_radio = st.radio(
        "Choose a shipping method",
        ("Standard", "Express")
    )
```

Columns


```python
import streamlit as st

col1, col2 = st.columns(2)

with col1:
   st.header("A cat")
   st.image("https://static.streamlit.io/cat.jpg")

with col2:
   st.header("A dog")
   st.image("https://static.streamlit.io/dog.jpg")
```

Tabs


```python
import streamlit as st

tab1, tab2 = st.tabs(["Cat", "Dog"])

with tab1:
   st.header("A cat")
   st.image("./cat.jpg", width=200)

with tab2:
   st.header("A dog")
   st.image("./dog.jpg", width=200)
```

Layouts

Expander

Container


```python
import streamlit as st

with st.expander("See explanation"):
    st.write("Here it is")
```

```python
import streamlit as st

with st.container():
   st.write("This is inside the container")
```

Prints

In streamlit it is very easy to print everything, especially Pandas dataframes

```
import streamlit as st


# print a number or a string
st.write(1234)


# print a Pandas dataframe
st.write(pd.DataFrame({
    'first column': [1, 2, 3, 4],
    'second column': [10, 20, 30, 40],
}))
# you can also use st.dataframe


# print latex equation
st.latex(r'\mu = \frac{1}{N}\sum_{n=1}^{N} e_n')
```

1234

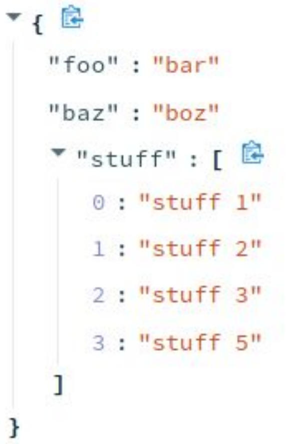
	first column	second column
0	1	10
1	2	20
2	3	30
3	4	40

$$\mu = \frac{1}{N}\sum_{n=1}^{N} e_n$$

Prints

It’s also possible to print json files

```python
import streamlit as st

st.json({
    'foo': 'bar',
    'baz': 'boz',
    'stuff': [
        'stuff 1',
        'stuff 2',
        'stuff 3',
        'stuff 5',
    ],
})
```



metrics

```python
import streamlit as st

st.metric( label="Temperature",
           value="70 °F",
           delta="1.2 °F")
```

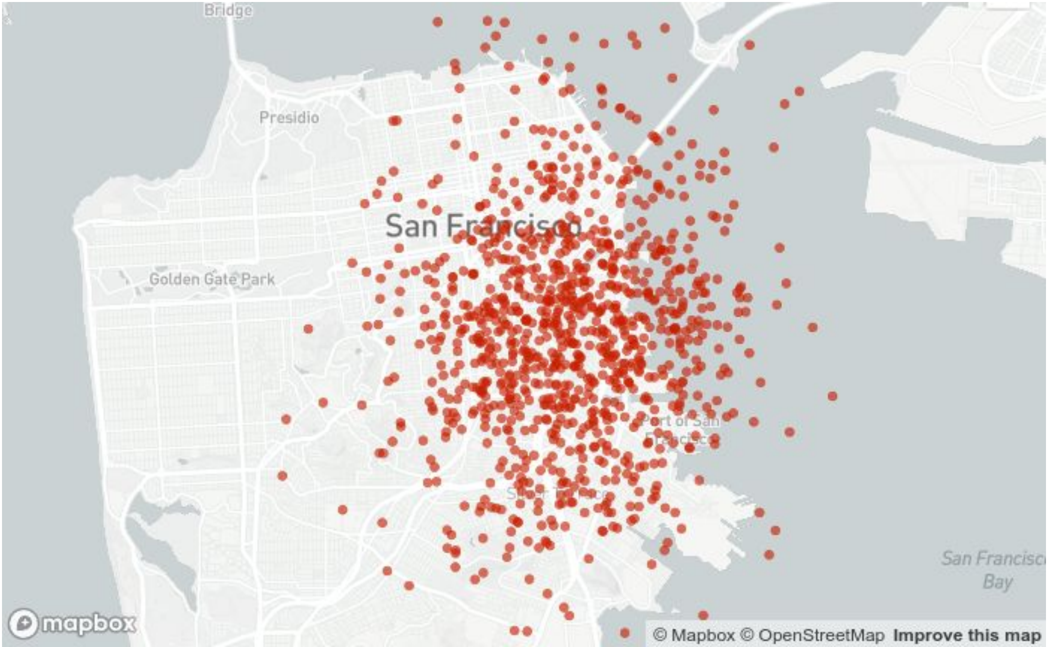
Plotting

Plotting is extremely easy as well

```python
import streamlit as st
import pandas as pd
import numpy as np

chart_data = pd.DataFrame(
    np.random.randn(20, 3),
    columns=['a', 'b', 'c'])

st.line_chart(chart_data)
```

Plotting geographical data

```
import streamlit as st
import pandas as pd
import numpy as np

df = pd.DataFrame(
    np.random.randn(1000, 2) / [50, 50] + [37.76, -122.4],
    columns=['lat', 'lon'])

st.map(df)
```



Inputs

Button
Display a button widget.

```
clicked = st.button("Click me'
```

Download button
Display a download button widget.

```
st.download_button("Download f
```

Checkbox
Display a checkbox widget.

```
selected = st.checkbox("I agre
```

Radio
Display a radio button widget.

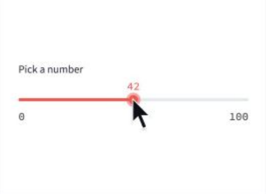
```
choice = st.radio("Pick one",
```

Selectbox
Display a select widget.

```
choice = st.selectbox("Pick or
```

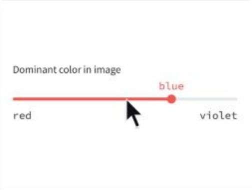
Multiselect
Display a multiselect widget. The multiselect widget starts as empty.

```
choices = st.multiselect("Buy'
```



Slider
Display a slider widget.

```
number = st.slider("Pick a num
```



Select-slider
Display a slider widget to select items from a list.

```
size = st.select_slider("Pick
```

Text input
Display a single-line text input widget.

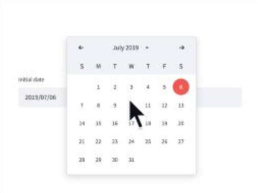
```
name = st.text_input("First na
```

Number input
Display a numeric input widget.

```
choice = st.number_input("Pick
```

Text-area
Display a multi-line text input widget.

```
text = st.text_area("Text to t
```



Date input
Display a date input widget.

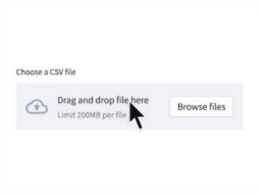
```
date = st.date_input("Your bir
```

Inputs

Time input

Display a time input widget.

```
time = st.time_input("Meeting
```



File Uploader

Display a file uploader widget.

```
data = st.file_uploader("Uploa
```

Camera input

Display a widget that allows users to upload images directly from a camera.

```
image = st.camera_input("Take
```

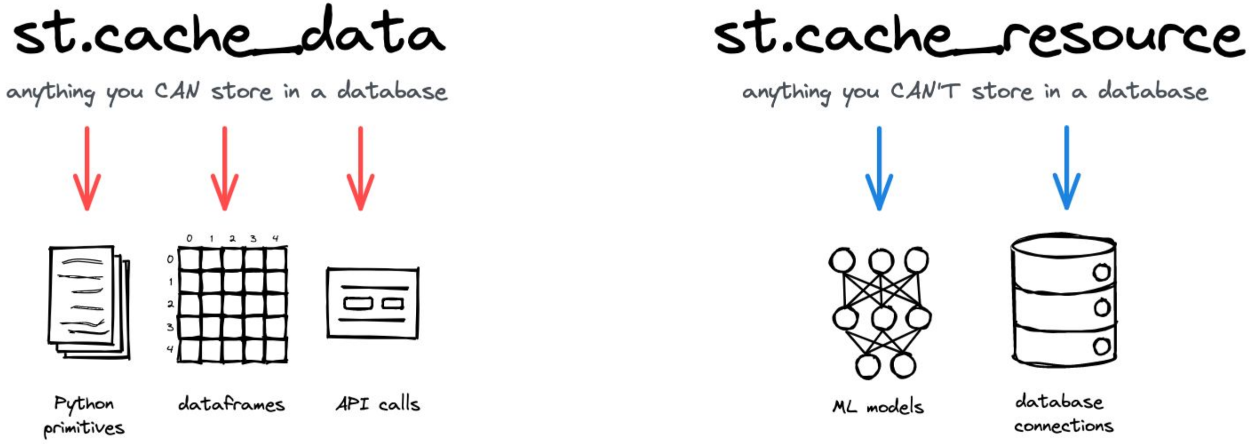
Color picker

Display a color picker widget.

```
color = st.color_picker("Pick
```

Caching

- Data and objects that do not need to be updated can be loaded in a function with
  - **@st.cache_data** or
  - **@st.cache_resource** decorator



Example:

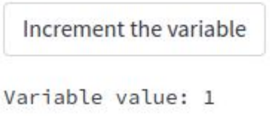
```
@st.cache_data
def load_data(fn):
    # read csv
    df = pd.read_csv(fn)
    # return
    return df
```

Maintaining the state

- Variables (except the ones associated to widgets) are reset at each interaction.
- This does not include dataframes and variables loaded from cache
- To make the system stateful
  - you can use the dictionary **st.session_state**

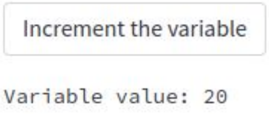
without state, it does not update the value up to 1

```python
import streamlit as st

# define variable
my_var = 0

# if button is clicked, increment variable
if st.button('Increment the variable'):
        my_var += 1

# display variable
st.text(f'Variable value: {my_var}')
```



with state, it works as expected

```python
import streamlit as st

# if variable is not in session state, initialize it
if 'my_var' not in st.session_state:
    st.session_state['my_var'] = 0

# if button is clicked, increment variable
if st.button('Increment the variable'):
    # increment variable
    st.session_state['my_var'] += 1

# display variable
st.text(f'Variable value: {st.session_state["my_var"]}')
```



First example - simple calculator

Let’s create a simple app that reads two numeric values, an operation and prints the output

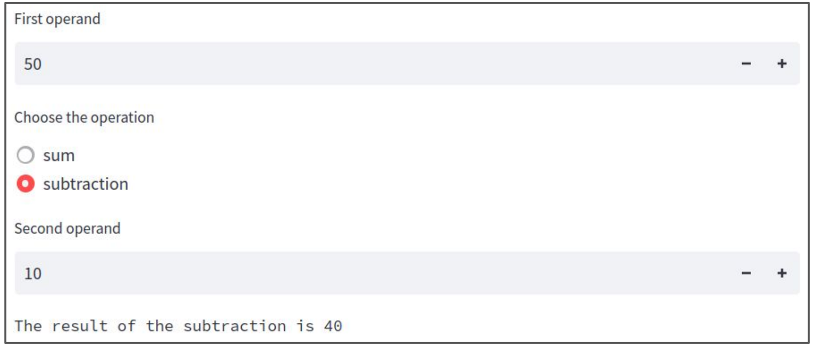
```
import streamlit as st

# define the first operand
first_number = st.number_input('First operand', value=50, step=10)

# define operation selector
operation = st.radio(
    "Choose the operation",
    ['sum', 'subtraction']
)

# define the second operand
second_number = st.number_input('Second operand', value=10, step=10)

# compute operation
if operation == 'sum':
        res = first_number + second_number
else:
        res = first_number - second_number

# print output
st.text(f'The result of the {operation} is {res}')
```

- Start it with streamlit run calculator.py

First example - simple calculator

Let’s create a simple app that reads two numeric values, an operation and prints the output

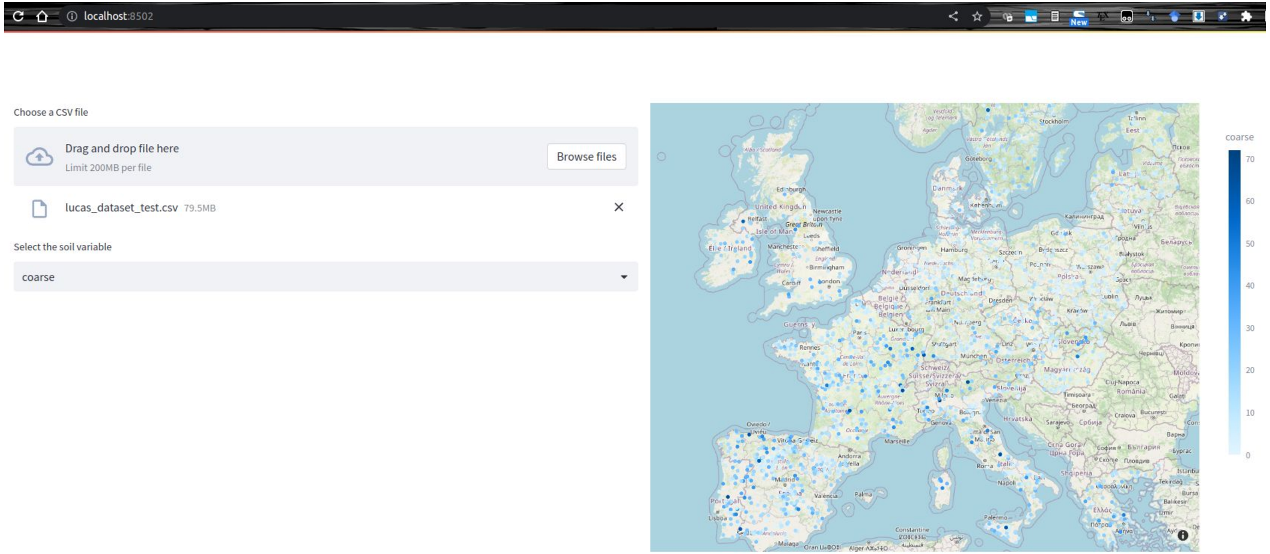
```python
import streamlit as st

# define the first operand
first_number = st.number_input('First operand', value=50, step=10)

# define operation selector
operation = st.radio(
    "Choose the operation",
    ['sum', 'subtraction']
)

# define the second operand
second_number = st.number_input('Second operand', value=10, step=10)

# compute operation
if operation == 'sum':
            res = first_number + second_number
else:
            res = first_number - second_number

# print output
st.text(f'The result of the {operation} is {res}')
```



- Start it with streamlit run calculator.py

Exercise 2 - data visualization

- For this example, you will need to use UNIMIB's virtual machine
- Goal: create a dashboard for visualizing the LUCAS dataset
- Layout: two columns, as shown in picture
- On left column:
  - a file uploader where the user will drop a CSV file containing part of the LUCAS dataset
  - a selectbox where the user will choose the soil variable that will be displayed on the map
- On the right column:
  - the map (use the function inside the file on eLearning)



Exercise 3 - PCA manipulation

- Goal: create a dashboard for visualizing the principal components of the LUCAS dataset
- Download the partial file and follow the instructions (#todo: lines)