TUTORIAL 14: RECAP ON VALIDATION

Creating Business Value with Generative AI
Fall 2025

AARHUS BSS
DEPARTMENT OF MANAGEMENT
AARHUS UNIVERSITY

5. December 2025

Magnus Bender
Assistant Professor

AGENDA FOR TODAY

A. Validation in general
B. An exemplary mini project
    a. Data generation
    b. An app
    c. Validation
    d. Archive export for WISEflow
C. Teaching evaluation

VALIDATION IN GENERAL

- Overview
- Metrics

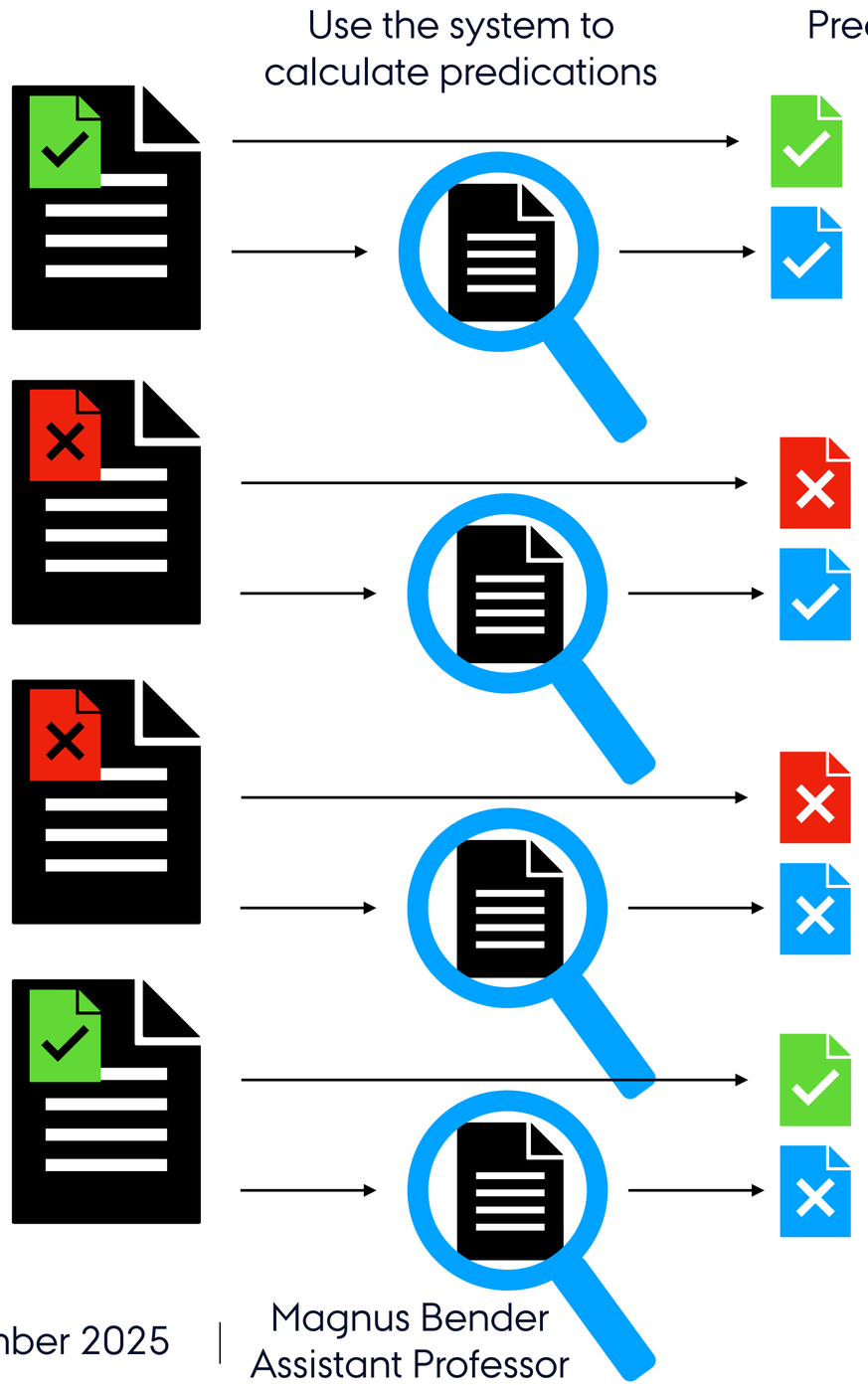
AARHUS BSS
DEPARTMENT OF MANAGEMENT
AARHUS UNIVERSITY

5. December 2025

Magnus Bender
Assistant Professor

OVERVIEW

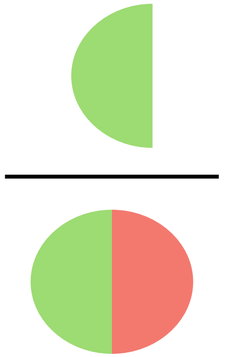
- A system (LLM, etc.) makes predictions based on inputs
- For a validation we need data:
  - Inputs to the system
  - The desired outputs for the inputs
  - A way to compare of prediction matches desired output
- Do all the predictions and count matches and missmatches



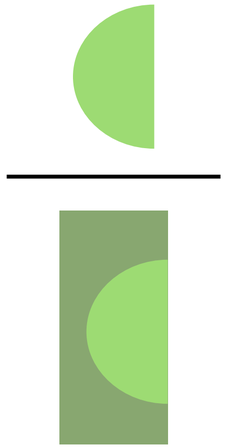
Accuracy: 50%

METRICS

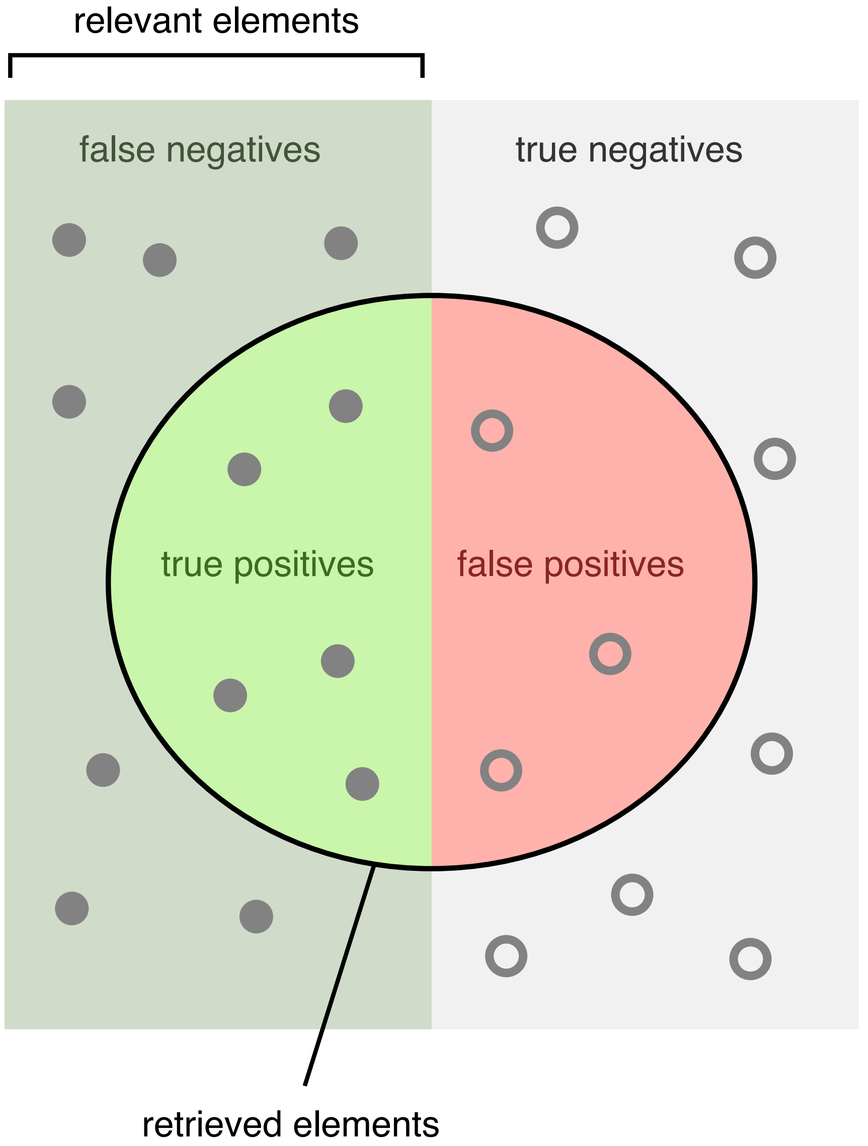
How many retrieved items are relevant?

Precision = 

How many relevant items are retrieved?

Recall = 

```
sklearn.metrics.precision_score()
sklearn.metrics.recall_score()
sklearn.metrics.precision_recall_fscore_support()
```



https://commons.wikimedia.org/wiki/File:Precisionrecall.svg
„Precision and recall“ by Walber is licensed under CC BY-SA 4.0

AN EXEMPLARY MINI PROJECT

- Data generation, app, and validation
- Archive export for WISEflow

AARHUS BSS
DEPARTMENT OF MANAGEMENT
AARHUS UNIVERSITY

5. December 2025

Magnus Bender
Assistant Professor

THE USE-CASE: SALUTATION ANALYSIS

- Input is a salutation: „Hey“, „Hi“, „Hello“, „God dag!“, „Kære“, etc.
- Output for each salutation:
  - Language (English, Danish, ...)
  - Level of formality (Formal, Informal, ...)
  - Time of day (Morning, Evening, ...)

- Three steps
  1. Generate synthetic data using OpenAI API: `Magnus_Recap/data_generation/generate.ipynb`
  2. Create a simple app: `Magnus_Recap/final_project/app.py`
  3. Provide an evaluation (validation): `Magnus_Recap/final_project/validate.ipynb`

DATA GENERATION

```
language = ["Danish", "English", "German", "Norwegian", "Swiss German", "French"]
formality = ["very formal", "formal", "informal", "partner"]
time_of_day = ["morning", "mid day", "evening", "night"]

def generate(language:str, formality:str, time_of_day:str) -> str:
  prompt = "Please generate a short salutation  ..."
  prompt += "Language: {language}, Formality: {formality}, Time of day: {time_of_day}"

  completion = client.chat.completions.create(
    model="gpt-5-nano",
    messages=[{"role":"user", "content" : prompt}]
  )
  return completion.choices[0].message.content

for i in range(100):
  salutation = generate(random.choice(language),
    random.choice(formality), random.choice(time_of_day))
  # write to json file -> the 'salutation' and the values for:
  #   language, formality, time_of_day
```

Generate 100 examples based on random selections of language, formality, and time of day.

RESULT: SYNTHETIC DATA

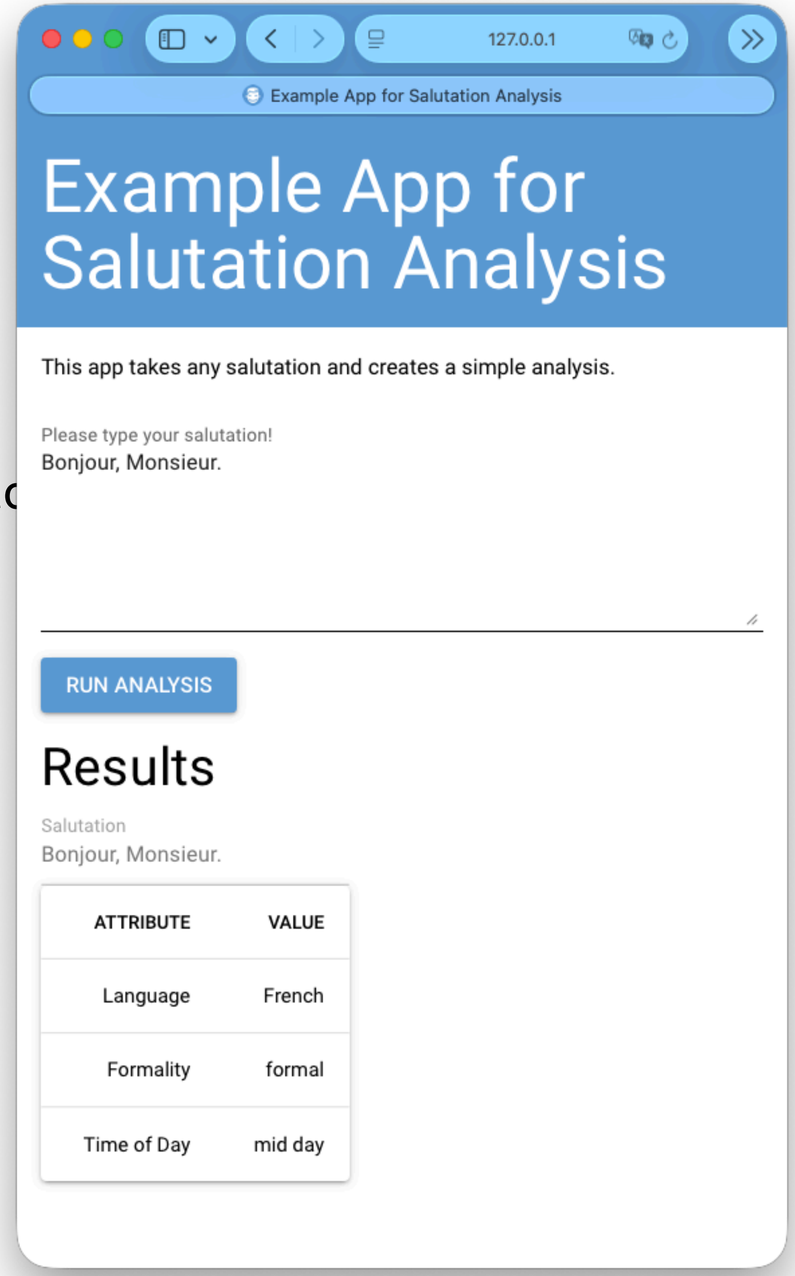
- We know for each item:
  - The used parameters
    - Won't be visible to the app
    - Are the ground truth to compare to later
  - The generated salutation
    - Inputs for our app as test cases
    - Are used as basis to compare with the estimates/ predictions of our app

```
[
  {
    "parameters": {
      "language": "English",
      "formality": "very formal",
      "time_of_day": "mid day"
    },
    "salutation": "Good afternoon, Sir."
  },
  {
    "parameters": {
      "language": "French",
      "formality": "very formal",
      "time_of_day": "morning"
    },
    "salutation": "Bonjour, Monsieur."
  },
  ...
]
```

AN APP

- Two files:
  - `salutation_analysis.py`
    - Contains the actual analysis process:
    - ```
      async def analyze_salutation(salutation:str) -> Salutatio
          response = await client.responses.parse(
            model="gpt-5-nano",
            ...
            text_format=SalutationAnalysis
          )
          return response.output_parsed
      ```
  - `app.py`
    - Contains the NiceGUI interfaces and functions
    - ```
      from salutation_analysis import analyze_salutation

      analysis = await analyze_salutation(salutation)
      ```



VALIDATION

Use the same actual analysis process as the app (`analyze_salutation` function).

```
from salutation_analysis import analyze_salutation

async def single_validation(data_item:dict[str,dict[str,str]|str]) -> tuple[str, str]:
  sal, par = data_item['salutation'], data_item['parameters']
  analysis = await analyze_salutation(sal)

  ground_truth = par['language'], par['formality'], par['time_of_day']
  estimates = analysis.language, analysis.formality, analysis.time_of_day

  return ground_truth, estimates

await single_validation({
  "parameters": {"language": "English","formality": "informal","time_of_day": "evening" },
  "salutation": "Hey, good evening!"
})
# (('English', 'informal', 'evening'), ('English', 'informal', 'evening'))
```

SOME SCORES

```
import json
from sklearn.metrics import classification_report

t_e = []
for d_i in json.load(open(
    "../data_generation/synthetic-data.json"
  )):
  gt, es = await single_validation(d_i)
  t_e.append((ft, es))

classification_report(
  [value for v in t_e for value in v[0]], # ground
  [value for v in t_e for value in v[1]], # estima
)
```

	precision	recall	f1-score	support
Danish	0.79	0.92	0.85	12
English	1.00	1.00	1.00	17
French	1.00	1.00	1.00	18
German	1.00	1.00	1.00	18
Norwegian	0.91	0.77	0.83	13
Swiss German	1.00	1.00	1.00	22
evening	0.70	1.00	0.82	21
formal	0.26	0.73	0.39	15
informal	0.96	0.96	0.96	28
mid day	0.88	1.00	0.93	28
morning	1.00	0.86	0.93	29
night	1.00	0.59	0.74	22
partner	1.00	0.39	0.56	28
very formal	0.68	0.45	0.54	29
accuracy			0.82	300
macro avg	0.87	0.83	0.83	300
weighted avg	0.88	0.82	0.82	300

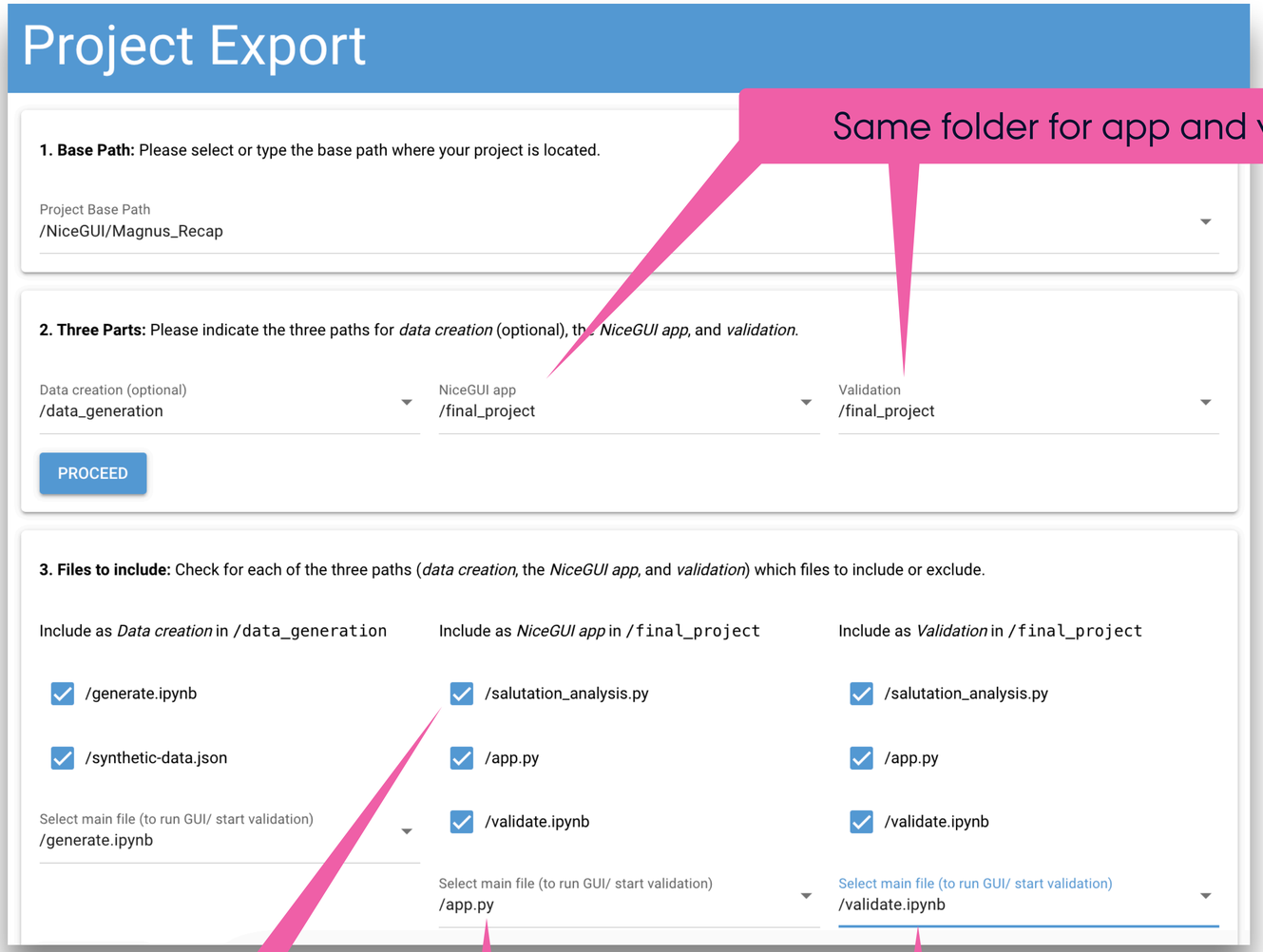
FREQUENT ERRORS

- Language
  - *truth → estimate (count)*
  - Norwegian → Danish (3)
  - Danish → Norwegian (1)
- Time of Day
  - night → evening (9)
  - morning → mid day (4)

- Formality
  - very formal → formal (16)
  - partner → formal (14)
  - formal → very formal (3)
  - partner → very formal (3)
  - informal → formal (1)
  - formal → informal (1)

EXPORT APP

- Exemplary export of salutation app on uCloud.
- Salutation app has only two folders:
  - `data_generation`
    - Data and generation notebook
  - `final_project`
    - App and validation code



Same folder for app and validation

Python script containing analysis code is used by app and validation

Different main files for app and validation

CLOSING REMARKS

- This is a small example, there are many many ways to do things like this!
- If you have real data, you will still need labels/ desired outputs, but …
  - … may create them manually.
  - … may use a free data set from the internet.
  - … may ask a different LLM for labels (slightly weak, but totally fine for project). …
- You do not need to have your „core functionality“ in a separate file (as in example project for importing from `app.py` and `valiadate.ipynb`)
  - You may export results from your app.
  - You may copy code. …
- Each validation will be different!
  - Depending on the use-case, you need to evaluate in different settings and need different approaches to be able to calculate scores.
  - Don’t worry about the actual values of your scores, we do not grade based on them!

TEACHING EVALUATION

Thank you for your feedback and comments!

AARHUS BSS
DEPARTMENT OF MANAGEMENT
AARHUS UNIVERSITY

5. December 2025

Magnus Bender
Assistant Professor

REOCCURRING TOPICS

- Course description: Discrepancies between expectations and reality
  - Will update for next year
- Structure and order of lectures
  - Plan to rearrange some of them
  - Start earlier with practical requirements for project
- Feedback system
  - Did not scale with 150 students in class → will revise to new system next year
  - Nice ideas like 1-1 consultations with us
- Coding and technical skills
  - Difficult to change: Right now the aim is that you have to program something
  - Consider while revising course description
  - More classic/ detailed „Introduction to programming“ difficulty to fully fit in

DEPARTMENT OF MANAGEMENT
AARHUS UNIVERSITY