ClassifAI:
A tool for coding & classification

https://github.com/datasciencecampus/classifAI

Luke Roantree – Senior Data Scientist,
Central Data Science & AI,
Office for National Statistics (UK)

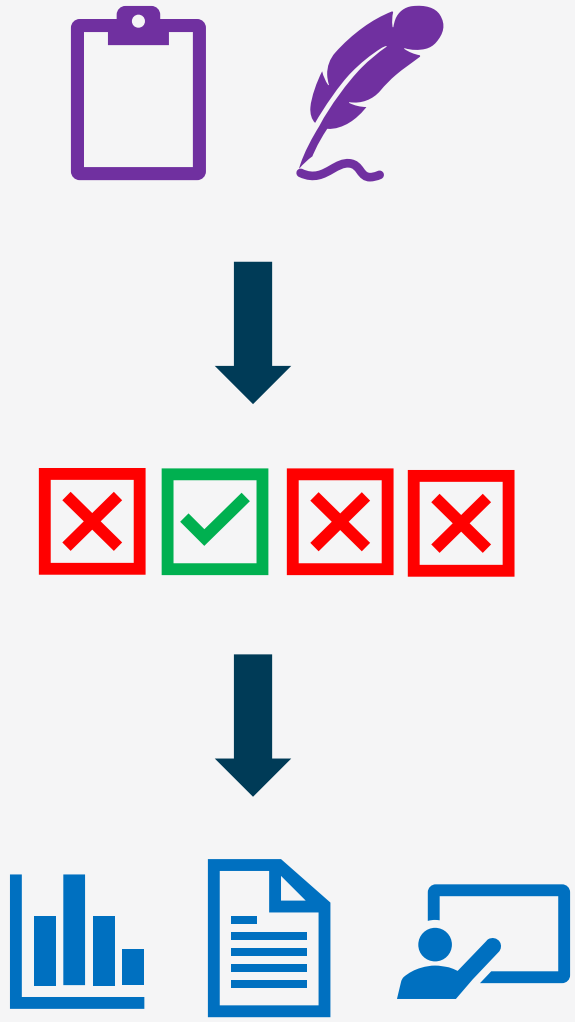
09 March 2026

Data Science Campus

UK ONS ClassifAI - UNESCAP

Problem Context

- Survey responses need to be mapped to distinct classification codes
- Often a hierarchical structure, comprising 50-1000 individual codes at most granular level
- Substantial human effort required to assign these classifications
- Automation/assistance with ‘traditional’ NLP & ML approaches often struggles – classification with 100+ categories is typically **very** difficult & requires large (good quality) training data
- They usually don’t deal well with semantic similarities / differences (e.g. for occupational coding, a job of “professor” may not be classified if only “lecturer” is seen in training data)



Our answer to this - ClassifAI

- ClassifAI uses LLM encoder models to project text into a '**semantic vector space**' – this addresses "professor" - "lecturer" type issues
- This flexibility on **semantic relationships** also reduces the amount of training data required
- It embeds the **historical examples** as vectors, and saves them (along with the original text & any metadata) to a '**VectorStore**'
- A query (e.g. a new response) can be '**searched**' against the VectorStore

ClassifAI is an open-source python package available on GitHub

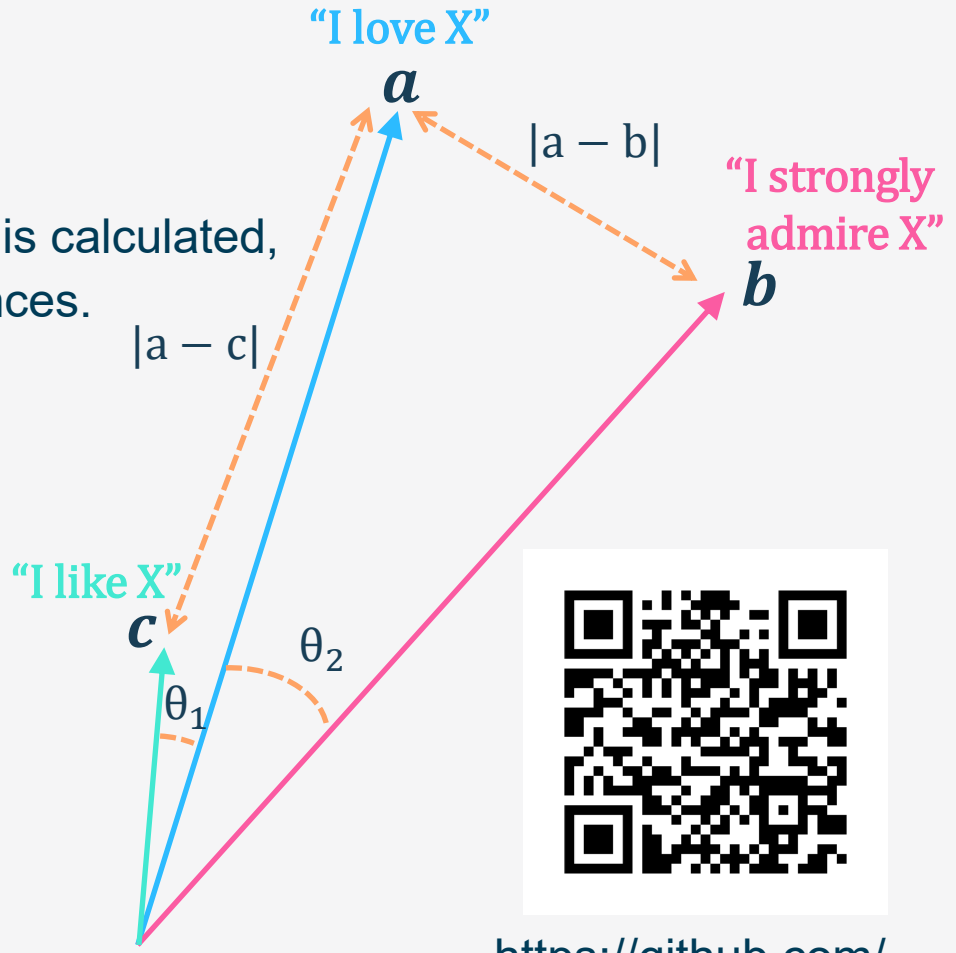
https://github.com/datasciencecampus/classifAI

ClassifAI Approach

- **Searching**:
  > A query (e.g. new response) is embedded as a vector,
  > The 'distance' of that vector from every vector in the VectorStore is calculated,
  > The 'nearest' N historical examples are returned, with their distances.

- **Distance**:
  There are several ways of measuring 'distance' between vectors, appropriate in different contexts. E.g.:
  > L_2 distance – good if 'strength' of a text is meaningful
  > cosine similarity – good if 'strength' is not meaningful

- **Two key advantages**:
  > No 'training' step required,
  > Classification decisions can be explainable.
  (e.g. "response X was very similar to historical record Y, which was confidently coded as 1.2.3.4")

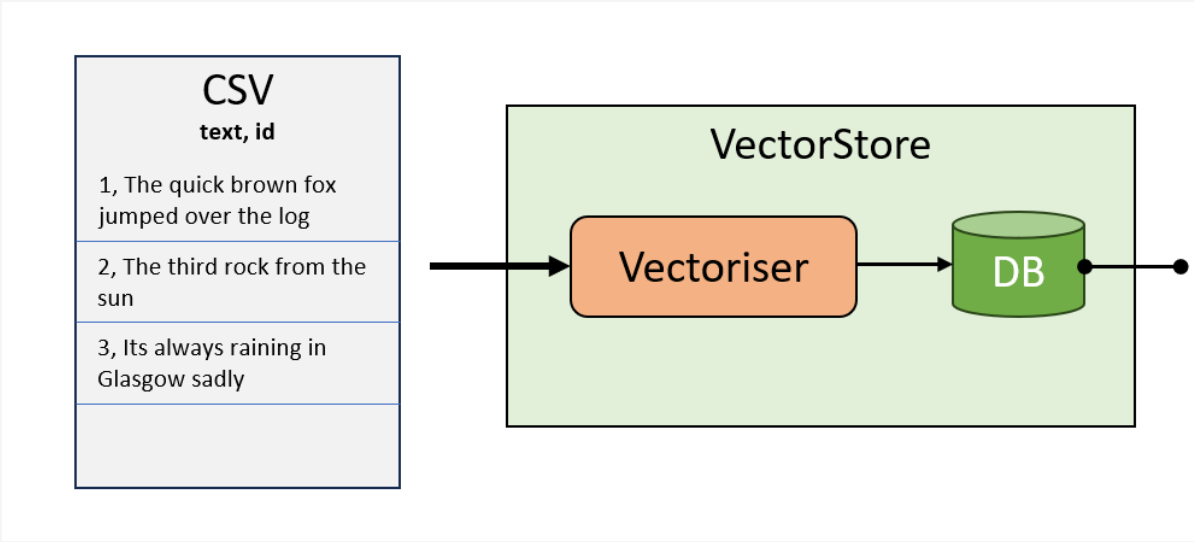


https://github.com/datasciencecampus/classifAI

ClassifAI – What's in the box?



HuggingFaceVectoriser
GcpVectoriser
OllamaVectoriser
<VectoriserBase>



.embed()

.search()

.reverse_search()

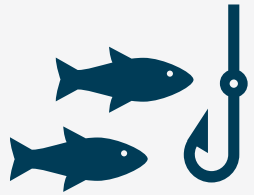
ClassifAI

	A	B	C
1	**id**	**text**	**source**
2	101	Fruit farmer: Grows and harvests fruits such as apples, oranges, and berries.	survey 22
3	101	Vegetable farmer: Cultivates and harvests vegetables like carrots, potatoes, and lettuce.	survey 55
4	102	Dairy farmer: Manages cows for milk production and processes dairy products.	survey 34
5	102	Sheep farmer: Raises sheep for wool, meat, and other products.	survey 11
6	103	Construction laborer: Performs physical tasks on construction sites, such as digging and carrying materials.	survey 10
7	103	Bricklayer: Lays bricks, concrete blocks, and stones to build walls and structures.	survey 1

`vectorstore.search()` ← "Landlord (pub)"

	query_id	query_text	doc_id	doc_text	rank	score	source
0	1	Landlord (pub)	128	Property manager: Oversees the maintenance and...	1	16.245012	survey_1
1	1	Landlord (pub)	138	Bartender: Mixes and serves drinks to customers.	2	12.781824	survey other
2	1	Landlord (pub)	112	Bartender: Prepares and serves drinks in bars ...	3	11.883011	survey_1
3	1	Landlord (pub)	138	Barback: Assists bartenders by restocking supp...	4	11.759285	survey other
4	1	Landlord (pub)	128	Home inspector: Evaluates the condition of pro...	5	10.751433	survey_1

ClassifAI – New Feature: Hooks



- → Hooks are pre- / post-processing tasks you can add to be applied to any of the VectorStore's inputs / outputs
- → E.g. data cleaning, linking external datasets, result aggregation
- → Define your own and / or use classifAI's provided ones
- → **This will soon include 'Retrieval Augmented Generation' (RAG) functionality**

	query_id	query_text	doc_id	doc_text	rank	score	RAG_response
0	1	APPLE MERCHANT	101	Fruit farmer: Grows and harvests fruits such a...	0	0.652508	1
1	1	APPLE MERCHANT	101	Fruit farmer: Grows and harvests fruits such a...	1	0.652508	0
2	1	APPLE MERCHANT	101	Orchard worker: Maintains and harvests fruit t...	2	0.635190	0
3	1	APPLE MERCHANT	107	Mobile app developer: Creates applications for...	3	0.620049	0
4	1	APPLE MERCHANT	128	Real estate agent: Assists clients in buying, ...	4	0.618706	0

	query_id	query_text	doc_id	doc_text	rank	score	RAG_response
0	1	APPLE MERCHANT	101	Fruit farmer: Grows and harvests fruits such a...	0	0.652508	1
1	1	APPLE MERCHANT	101	Fruit farmer: Grows and harvests fruits such a...	1	0.652508	2
2	1	APPLE MERCHANT	101	Orchard worker: Maintains and harvests fruit t...	2	0.635190	3
3	1	APPLE MERCHANT	107	Mobile app developer: Creates applications for...	3	0.620049	4
4	1	APPLE MERCHANT	128	Real estate agent: Assists clients in buying, ...	4	0.618706	5

	query_id	query_text	doc_id	doc_text	rank	score	RAG_response
0	1	APPLE MERCHANT	101	Fruit farmer: Grows and harvests fruits such a...	0	0.652508	apples
1	1	APPLE MERCHANT	101	Fruit farmer: Grows and harvests fruits such a...	1	0.652508	apples
2	1	APPLE MERCHANT	101	Orchard worker: Maintains and harvests fruit t...	2	0.635190	fruit trees
3	1	APPLE MERCHANT	107	Mobile app developer: Creates applications for...	3	0.620049	app
4	1	APPLE MERCHANT	128	Real estate agent: Assists clients in buying, ...	4	0.618706	selling

ClassifAI – deploying as a service

```python
from classifai.indexers import VectorStore
from classifai.vectorisers import HuggingFaceVectoriser
from classifai.servers import run_server

demo_vectoriser = HuggingFaceVectoriser(model_name="sentence-transformers/all-MiniLM-L6-v2")
demo_vectorstore = VectorStore(
    file_name="./data/fake_soc_dataset.csv",
    data_type="csv",
    vectoriser=demo_vectoriser,
    output_dir="./demo_vdb",
    overwrite=True,
    hooks={"search_preprocess": lambda x: x.lower() if isinstance(x, str) else x},
)

run_server([demo_vectorstore], ["example"])
```

REQUEST

```
curl -X 'POST' \
  'http://127.0.0.1:8000/example/search?n_results=2' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{"entries": [{ "id":"1","description":"Landlord (pub)}]}'
```

RESPONSE


```
{"data": [{"input_id":"1","response":[
    { "label": "128",
      "description": "Property manager:
                      Oversees the
                      maintenance and
                      leasing of rental
                     properties.",
      "score": 16.245054092407227,
      "rank": 1
    },
    { "label": "138",
      "description": "Bartender: Mixes
                      and serves drinks
                      to customers.",
      "score": 12.781867111206055,
      "rank": 2
    }
]}]}
```

Why Should You Use It?

- **Free** (& open-source) – Easily accessible & installable on (almost) any system with a python environment. Detailed documentation website coming soon.

- **Flexible** – Comes ‘batteries included’ but facilitates customisation. Suitable for use-cases anywhere from a standalone script on a laptop, to the backend of a cloud application with 100s (or more) of users.

- **Fast** – Uses optimised Polars & Numpy for internal processing. Building a VectorStore is a one-off cost (seconds-minutes), and can be re-loaded, then searching against it takes microseconds (seconds for very large batches).

- **Feedback** - Our users’ experiences and use-cases shape the direction of development. Want a feature? Raise an issue / open a PR.

Questions?

From early question submissions:

Q1: "***Can AI classification work in offline or low-bandwidth situations?***"
Yes, but restricted to HF/Ollama vectorisers, and limitations on hook pre-/post-processing.

Q2: "***How can these tools be adopted by other organizations using their own taxonomies and data?***"
Key requirement: Provide input file of (good quality) labelled historical data with any desired metadata.
More adventurous: Use hooks to link additional taxonomy info to search results, make use of this info in RAG re-ranking / classification / other.

Q3: "***How much training data is needed to get reliable results?***"
At minimum, ~10 examples per classification code. More is better, but **quality is more important.**
The historical examples you provide should be such that anything semantically (very) close to them should be(unambiguously) classified with the same code.

Q4: "***Could we have access to the presenters for further follow-up?***"
Yes – email cdsai@ons.gov.uk mentioning "classifAI" in the subject.