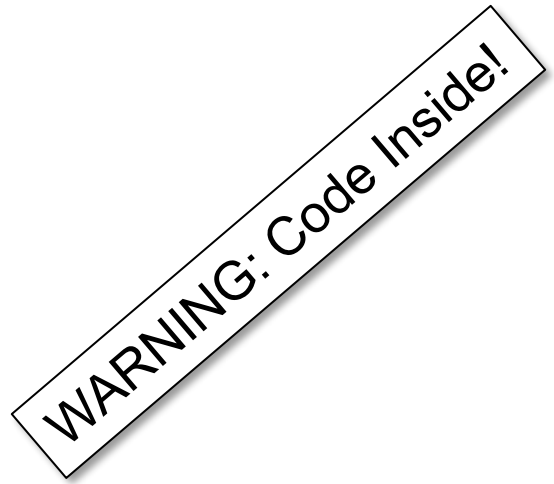
Enabling machine learning for weather and climate data in the cloud

CliMetLab

A Python package to support AI/ML activities in climate and meteorology

Baudouin Raoult



ECMWF

Looking at some Jupyter Notebooks

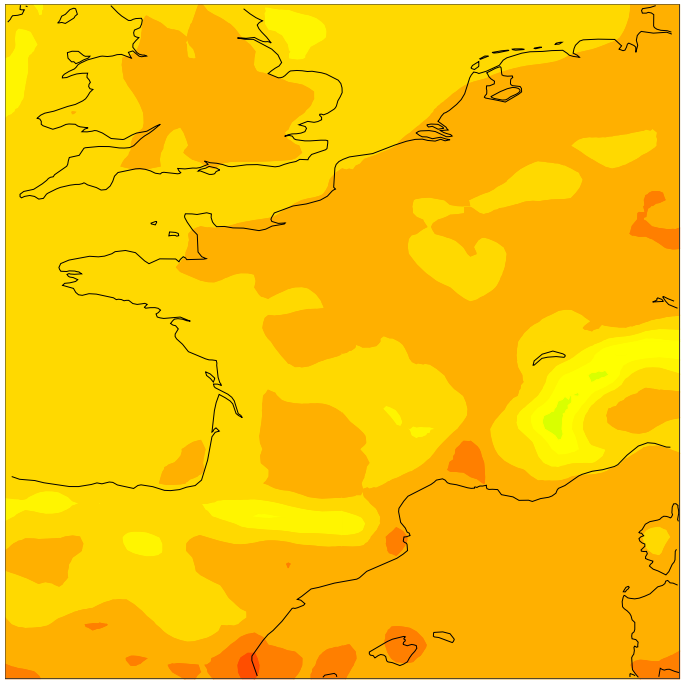
- A lot of code is dedicated to accessing the data…
  - wget, curl
  - unzip, untar
- …decoding them…
  - GRIB, NetCDF, CSV, etc
- … and tweaking plotting attribute to get something nice
- The number of lines for AI/ML is small

Why Python?

- Community software
  - Numpy (linear algebra), Pandas (statistics), matplotlib (graph plotting)
- Machine learning
  - Keras, Tensorflow, PyTorch, Scikit-learn
- Community software
  - NetCDF, xarray, cfgrib
- ECMWF software:
  - eccodes,odc, pybufr (GRIB, BUFR, ODB)
  - Magics (plotting)
  - webapi, cdsapi (data access)

How about that?

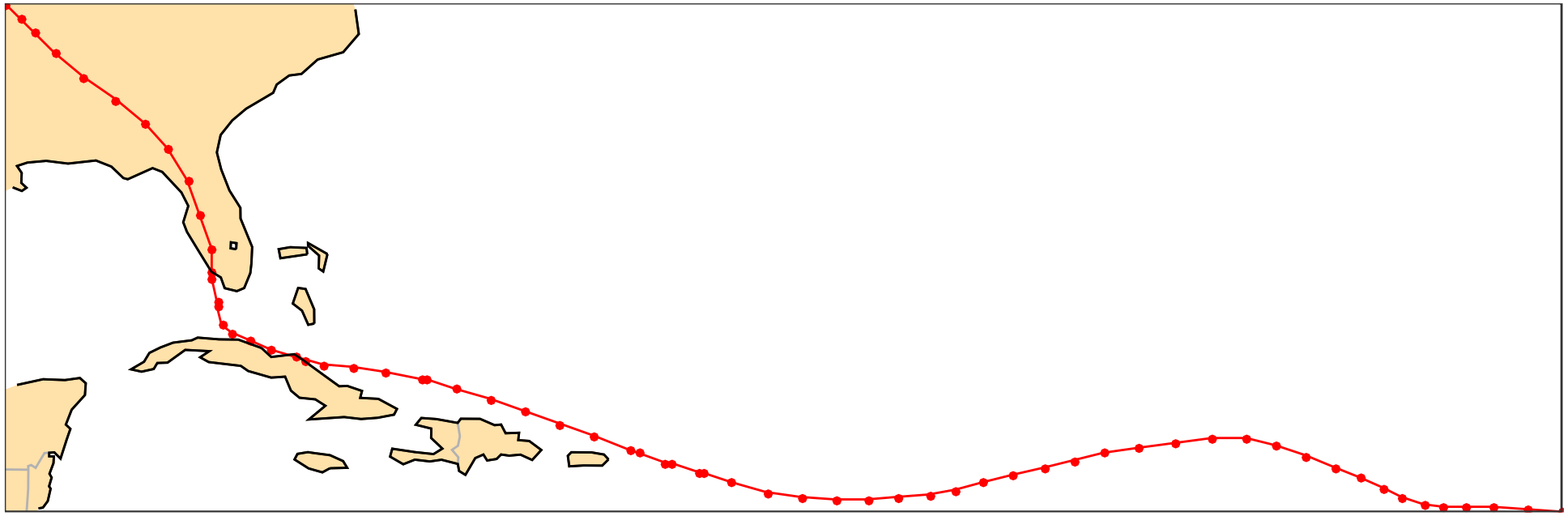
```
import climetlab as cml

data = cml.load_dataset("era5-temperature",
                        domain="France",
                        period=1980)

cml.plot_map(data["1980-06-09 18:00"])
```



ECMWF

And how about this?

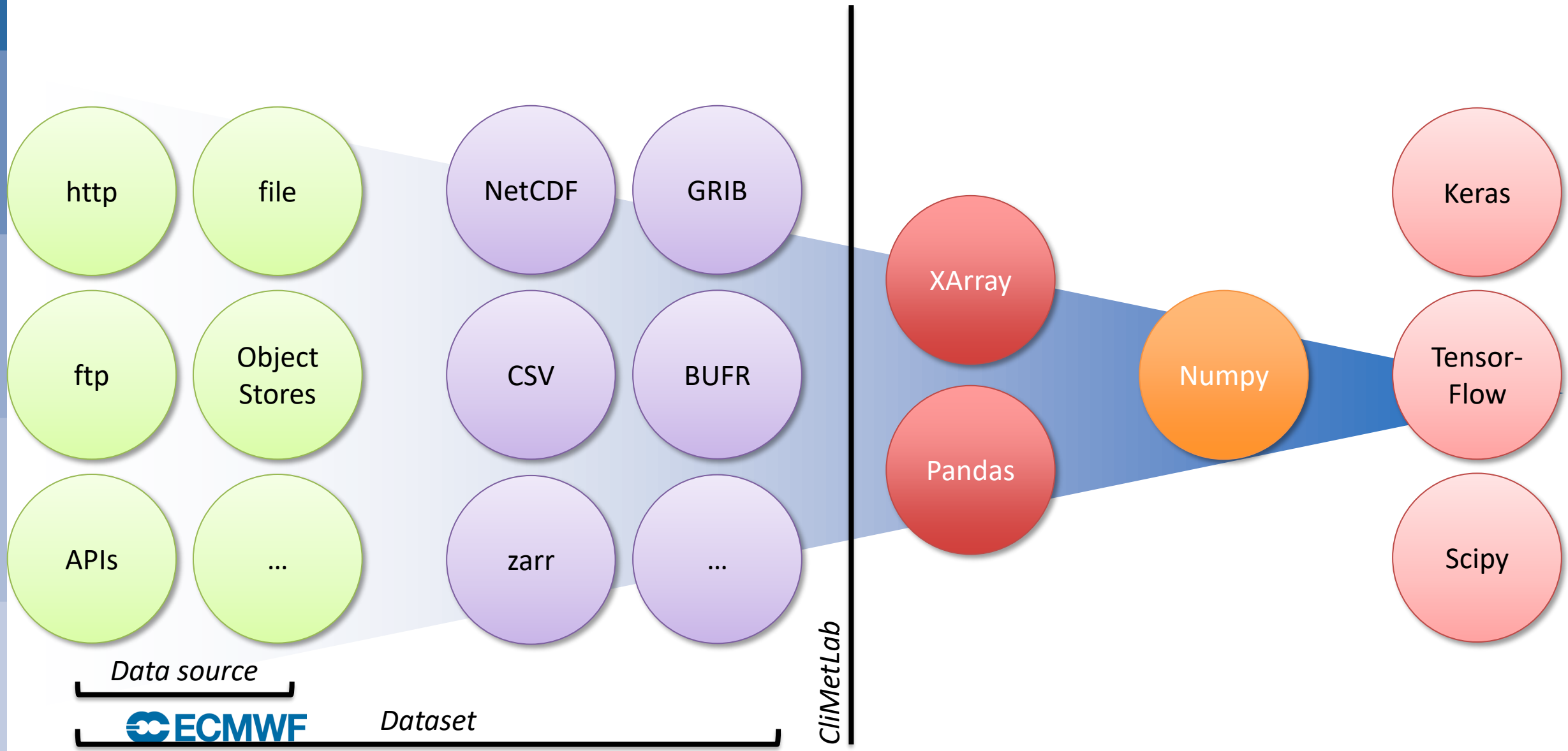
```
import climetlab as cml

data = cml.load_dataset("hurricane-database", "atlantic")
irma = data.to_pandas(name="irma", year=2017)

cml.plot_map(irma)
```



CliMetLab: Goals

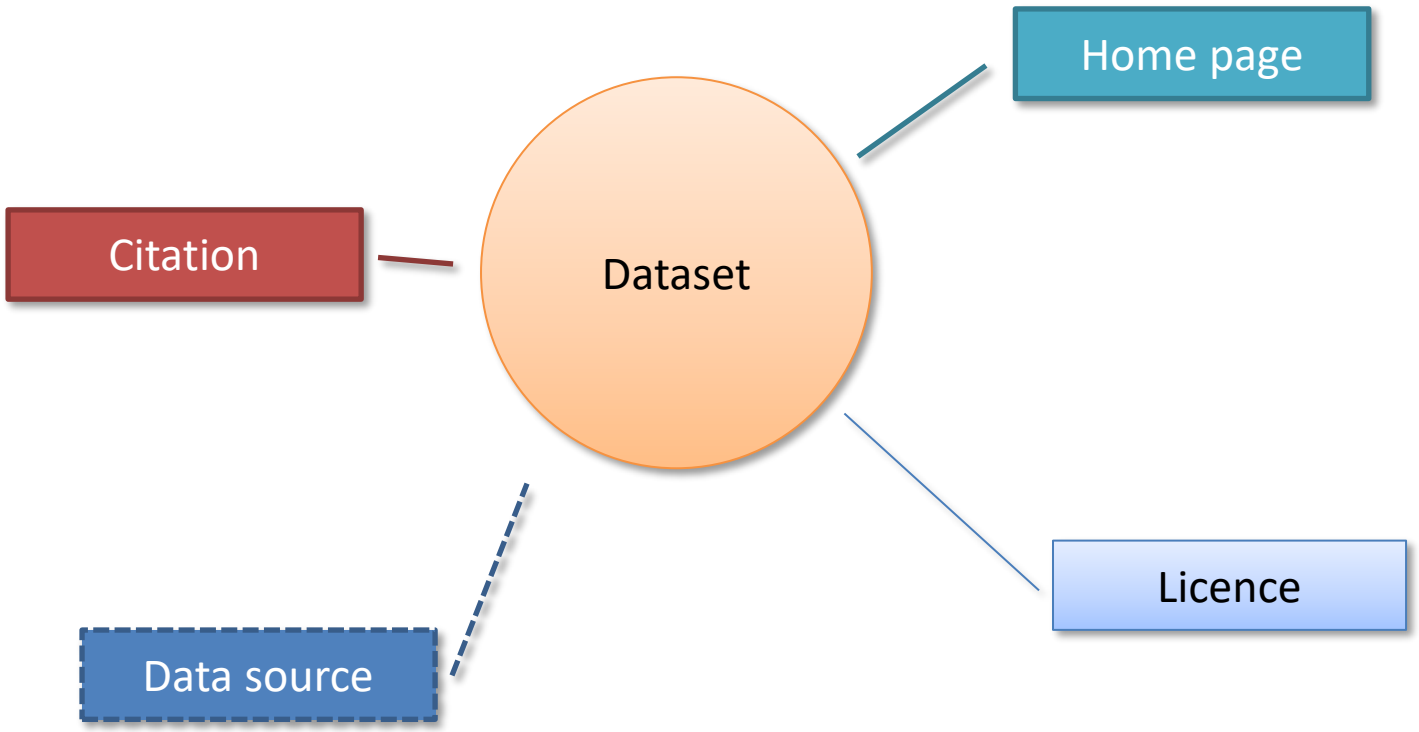
- Targets Jupyter notebooks (anywhere)
- Aims at minimising boilerplate code
- Provides seamless data access and plotting
- Full interoperability with major scientific Python packages
- Allows users to focus on sciences

Data ecosystem



ECMWF

Dataset metadata

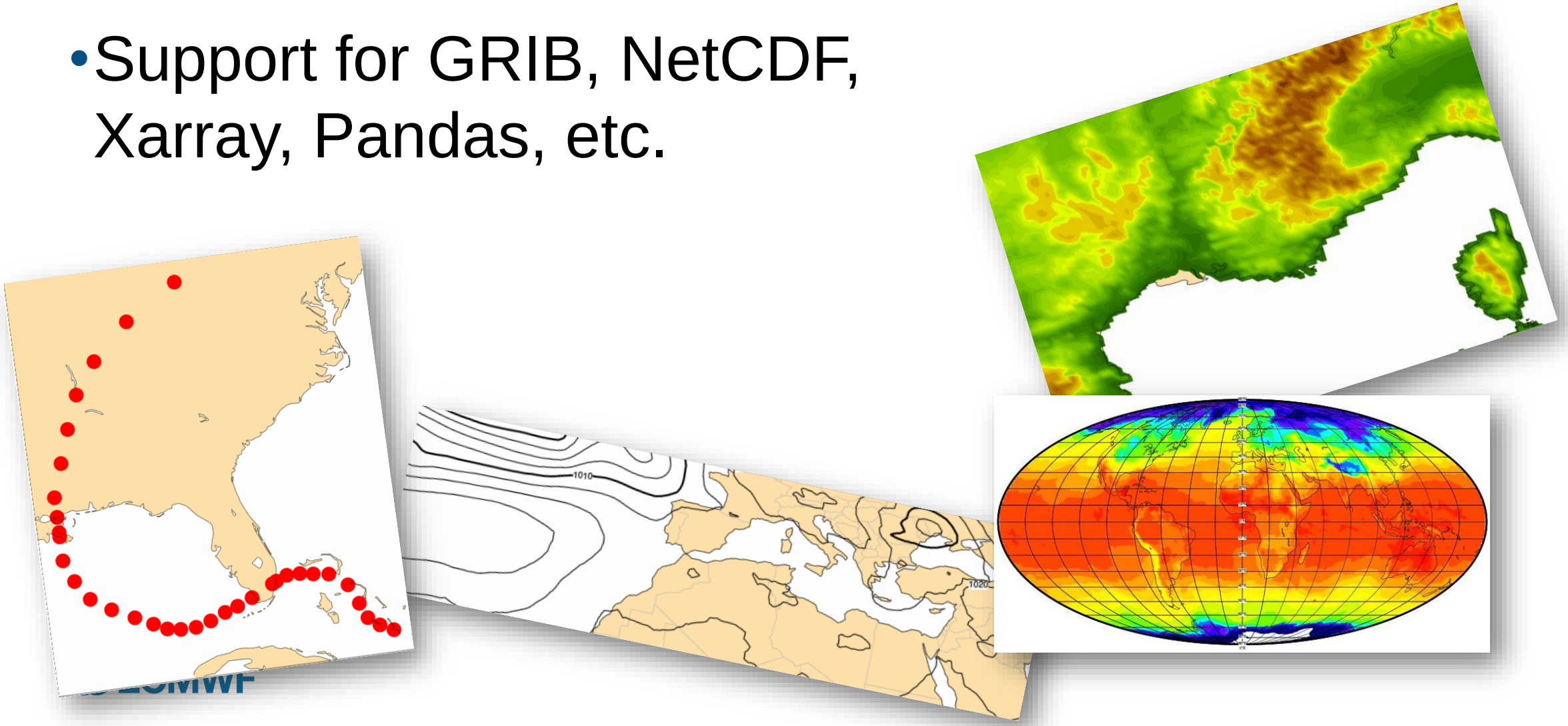


Interoperability

```
ds = climetlab.load_dataset("some-dataset")

xa = ds.to_xarray()

df = ds.to_pandas()

ar = ds.to_numpy()

(x_train, y_train), (x_test, y_test) = ds.load_data()
```

Automatic plotting of 2D maps

- Datasets "know" how to be best plotted
- Support for GRIB, NetCDF, Xarray, Pandas, etc.



Use well-know method names…

- …borrowed from Numpy, Pandas, Keras, etc. so users are in familiar territories

```
from keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

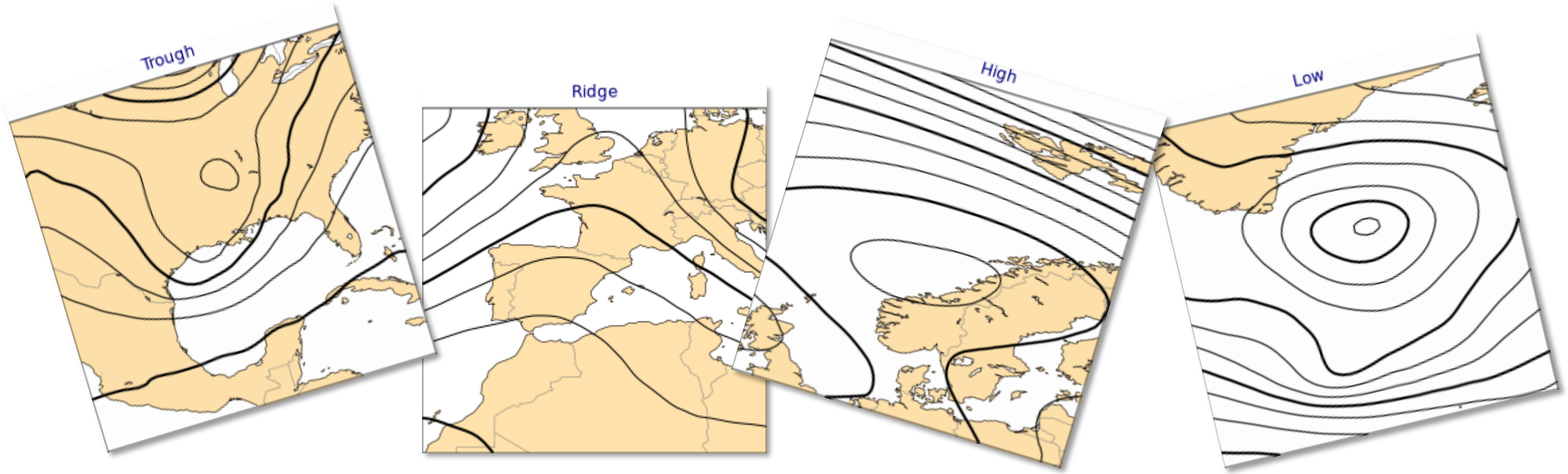
```
ds = climetlab.load_dataset("high-low")
(x_train, y_train), (x_test, y_test) = ds.load_data()
```

Example – importing Python packages

```
from tensorflow.keras.layers import Input, Dense, Flatten
from tensorflow.keras.models import Sequential

import climetlab as cml
```

Example – Loading a labeled dataset

```
highlow = cml.load_dataset("high-low")

for field, label in highlow.fields():
    cml.plot_map(field, width=256, title=highlow.title(label))
```



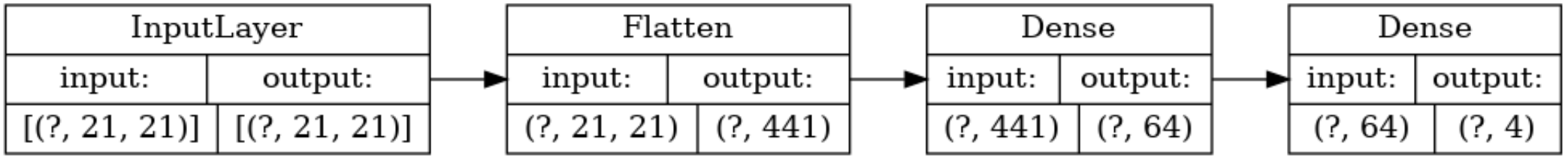
ECMWF

Example – Getting the training and test sets

```
(x_train, y_train, f_train),
(x_test, y_test, f_test) = highlow.load_data(test_size=0.3,
                                             fields=True)
```

Example – Building the model

```
model = Sequential()
model.add(Input(shape=x_train[0].shape))
model.add(Flatten())
model.add(Dense(64, activation="sigmoid"))
model.add(Dense(4, activation="softmax"))
```



Example – Compiling the model

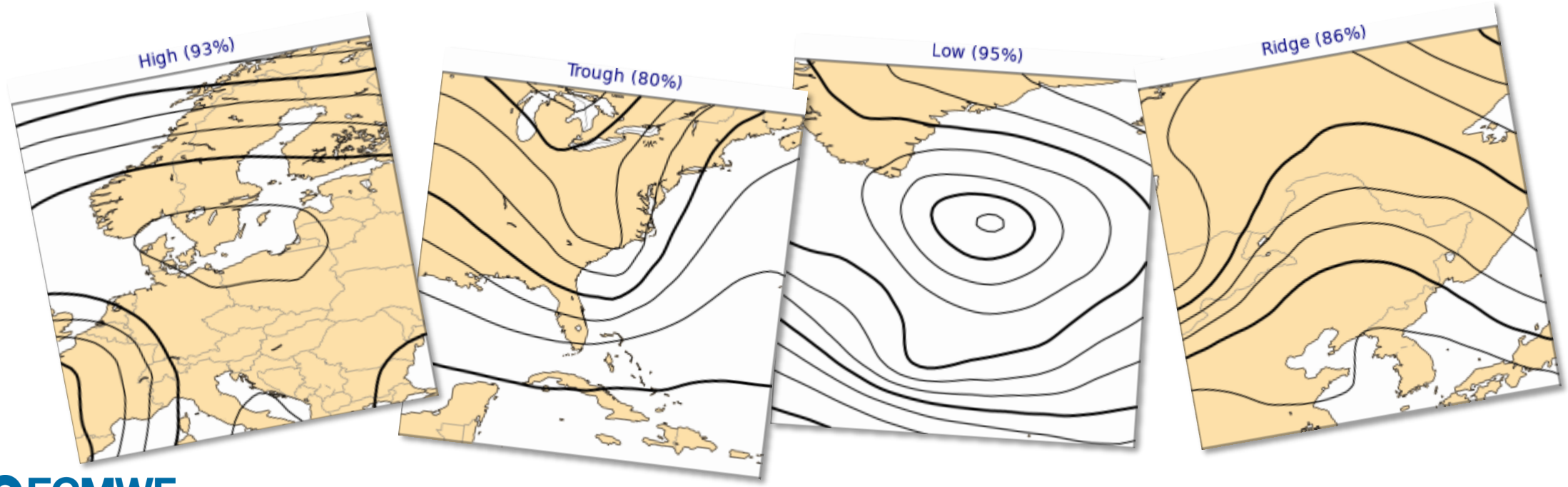
```
model.compile(optimizer="adam",
              loss="categorical_crossentropy",
              metrics=["accuracy"])
```

Example – Training the model

```
model.fit(x_train, y_train, epochs=100, verbose=0)

model.evaluate(x_test, y_test)
```

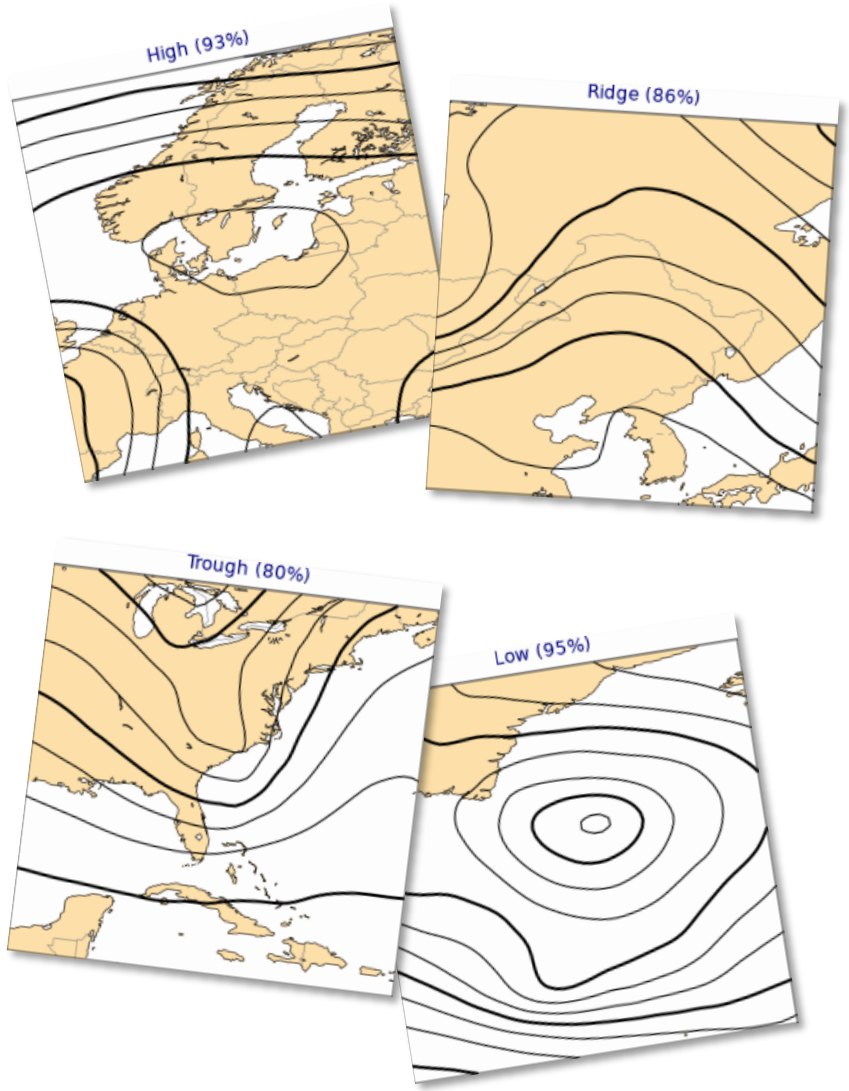
Example – Using the model

```
predicted = model.predict(x_test)

for p, f in zip(predicted, f_test):
    cml.plot_map(f, width=256,
                 title=highlow.title(p))
```



Example – Putting it all together…

```
from tensorflow.keras.layers import Input, Dense, Flatten
from tensorflow.keras.models import Sequential

import climetlab as cml

highlow = cml.load_dataset("high-low")
(x_train, y_train, f_train),
(x_test, y_test, f_test) = highlow.load_data(test_size=0.3,
                                             fields=True)

model = Sequential()
model.add(Input(shape=x_train[0].shape))
model.add(Flatten())
model.add(Dense(64, activation="sigmoid"))
model.add(Dense(4, activation="softmax"))

model.compile(optimizer="adam",
              loss="categorical_crossentropy",
              metrics=["accuracy"])
model.fit(x_train, y_train, epochs=100, verbose=0)

model.evaluate(x_test, y_test)
```



ECMWF

Example - Conclusion

- No MARS retrieval
- No GRIB decoding
- No conversion to Numpy

- Just focussing on the AI/ML model

Data Sources

- Local files
- URLs
- ecmwf-webapi
  - Access to the Centre's MARS archive)
- cdsapi
  - Access to Copernicus Climate Data Store)
- …
  - zarr/s3fs (In development)

Datasets

- Mostly proof of concepts for now

- WeatherBench (arXiv:2002.00469)
- Meteonet (Météo France)
- S2S AI/ML Competition (Upcoming)
- NOAA's hurricane database
- Your dataset?
- …
- Aim: all AI/ML reference datasets for meteorology/climate

Plugin architecture

- Easy addition of data sources
- Easy addition of datasets
- Easy addition of plotting styles
- Easy addition of data formats

Open development

- GitHub
- Read the Docs
- CI/CD
- …

What now?

```
pip3 install climetlab
```

- On Mac/Linux/Windows (Python3 only)

  http://climetlab.readthedocs.io

- It is beta!
  - APIs will change
  - Feedback welcome
  - Any idea of useful datasets?

Thank You!