Reproducible Machine Learning in Climate Science

ESoWC 2019

We are a PhD Student and a Machine Learning Engineer 👀

Our aim was to build a Python toolbox for working with ML using weather and climate data ✌️

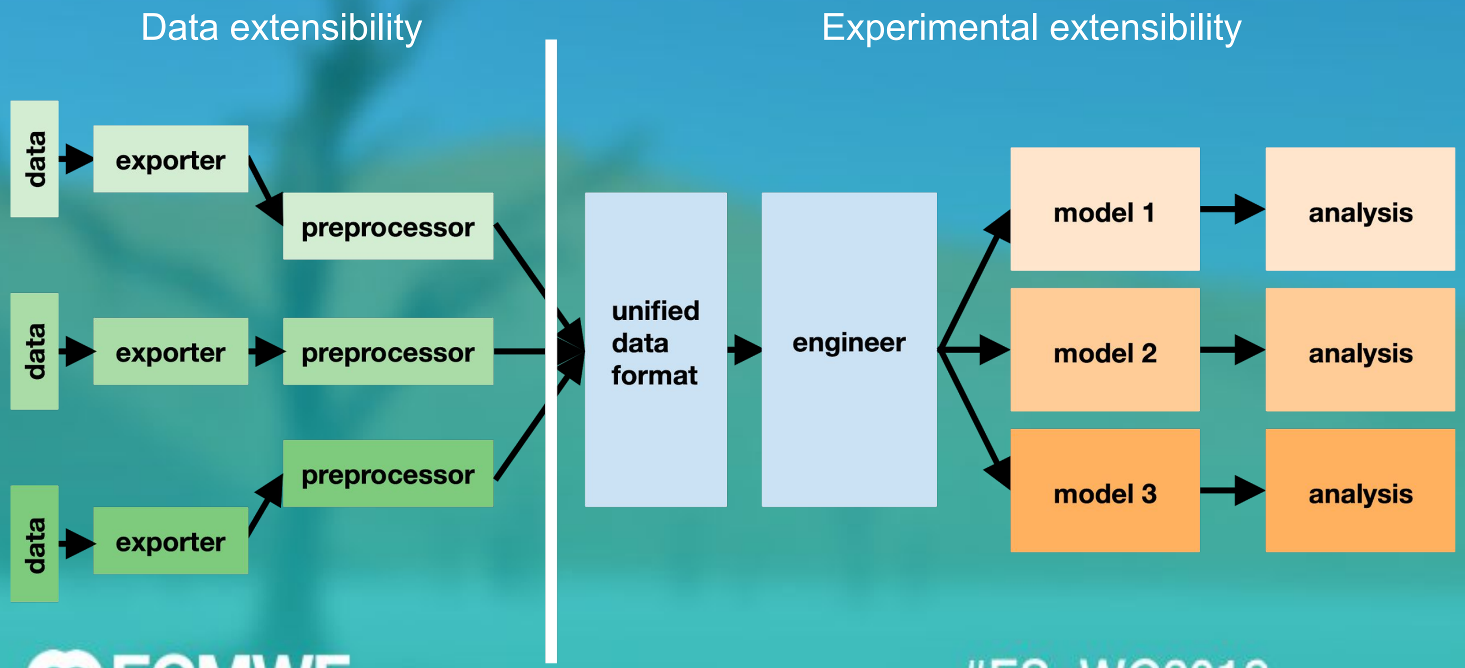
ML is hard for me and you,

Weather is tough for others too.

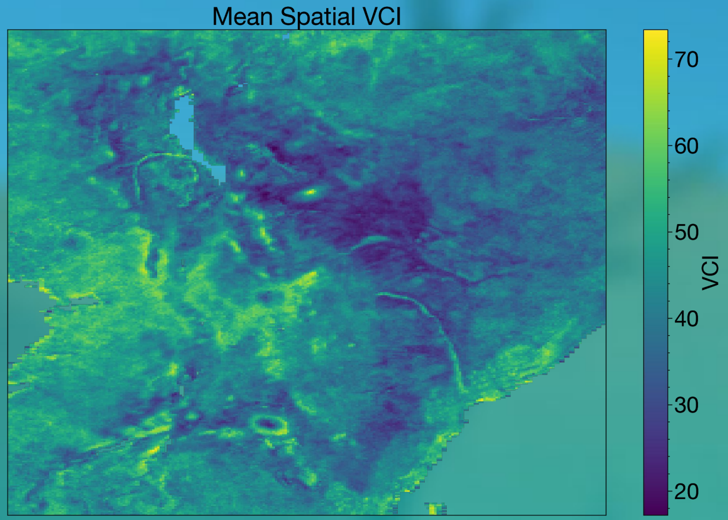
Interpretability makes us all fearful,

So we wrote some code to make it cheerful!

We developed a modular and extensible pipeline to apply machine learning to climate science



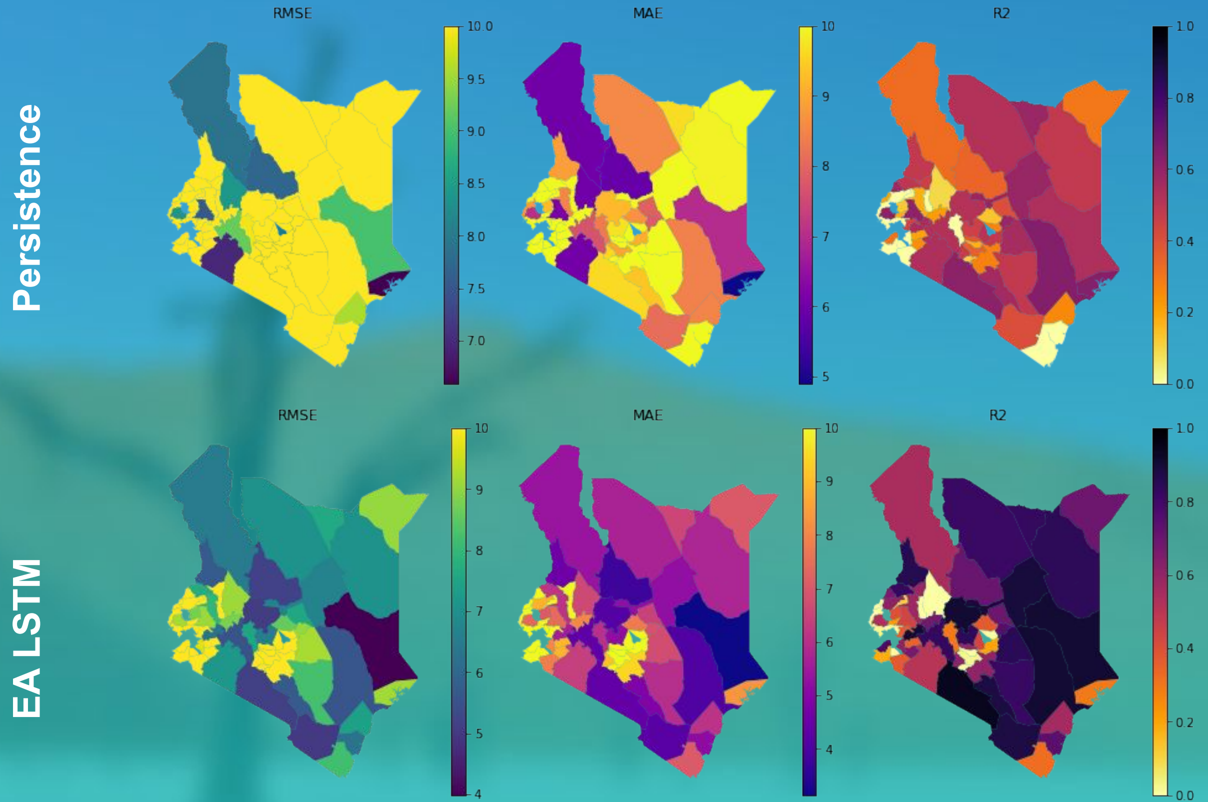
First results - using machine learning to predict vegetation health



	VCI3M			VCI	
District	**Adede et al. [2019]**	**EA-LSTM**	**Persistence**	**EA-LSTM**	**Persistence**
Mandera	0.71	0.91	0.57	0.80	0.34
Marsabit	0.77	0.75	0.50	0.77	0.54
Turkana	0.83	0.39	0.41	0.27	0.30
Wajir	0.71	0.82	0.52	0.83	0.51

Table 1: R^2 values reported by Adede et al. [2019], our EA-LSTM model and the baseline persistence model. For the Adede et al. [2019] model, VCI is aggregated over three months - results in both the aggregated and unaggregated case are presented. The performance of the model in Turkana compared to the other districts suggests there is still region-specific information the model is missing.

Our initial experiments showed some skill!

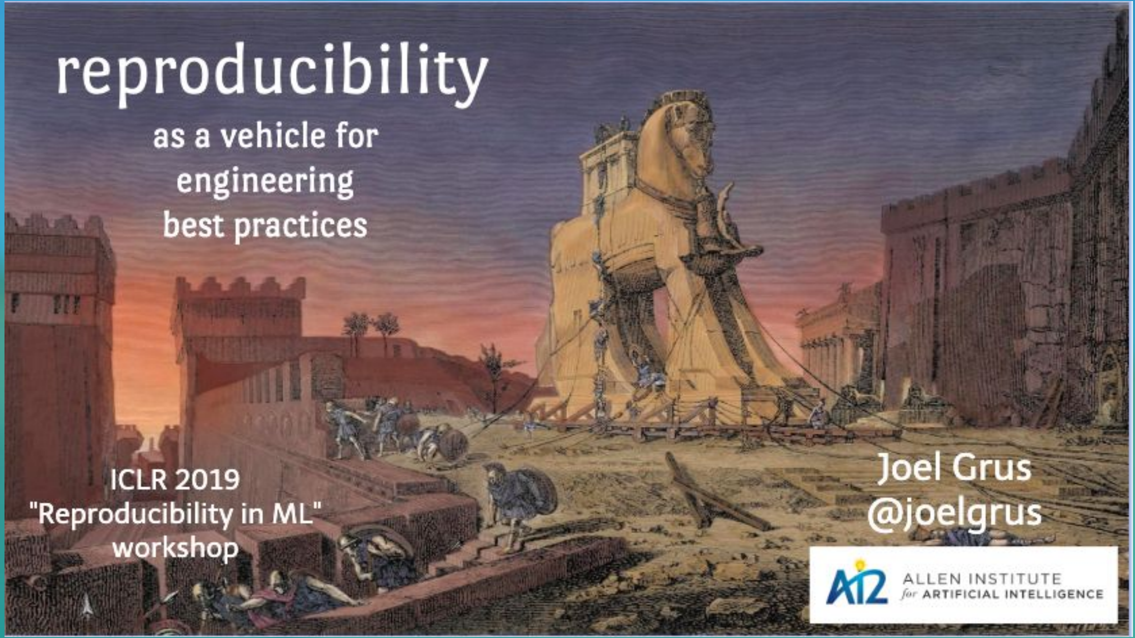


ECMWF

#ESoWC2019

G

We were strongly influenced by this talk 🧐



Reproducibility
@ ICLR 2019

Reproducibility is central to the ideals of scientific research

Openness

Replicability / Repeatability

Reproducibility

Extensibility

Collaboration

ECMWF

#ESoWC2019

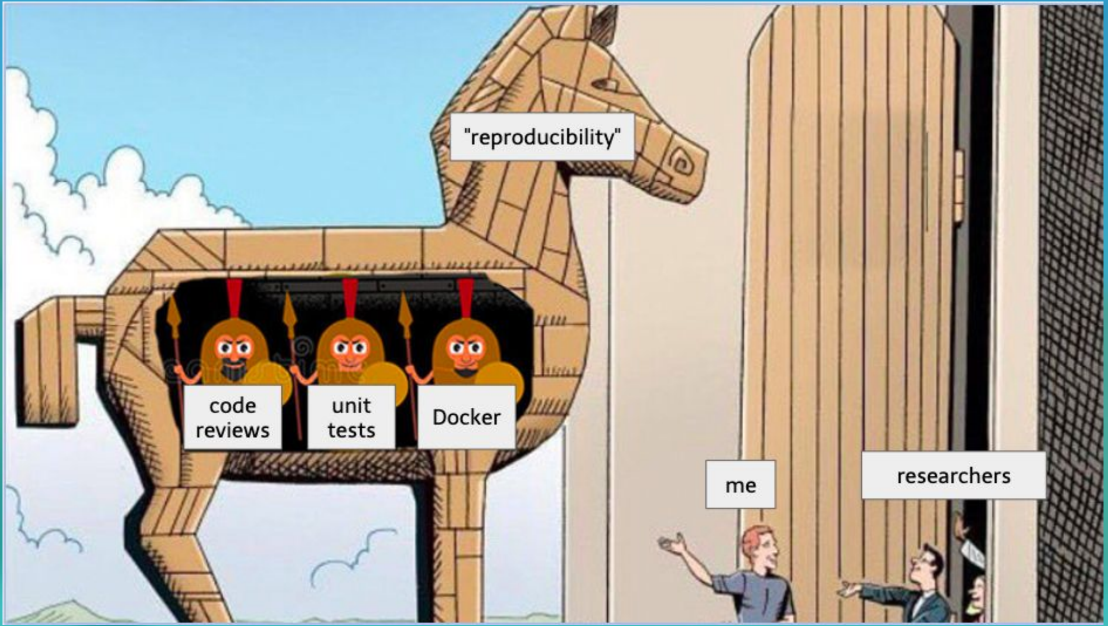
T

Open and Reproducible Science

"you can't stand on the shoulders of giants if they keep their shoulders private"

Grus 2019

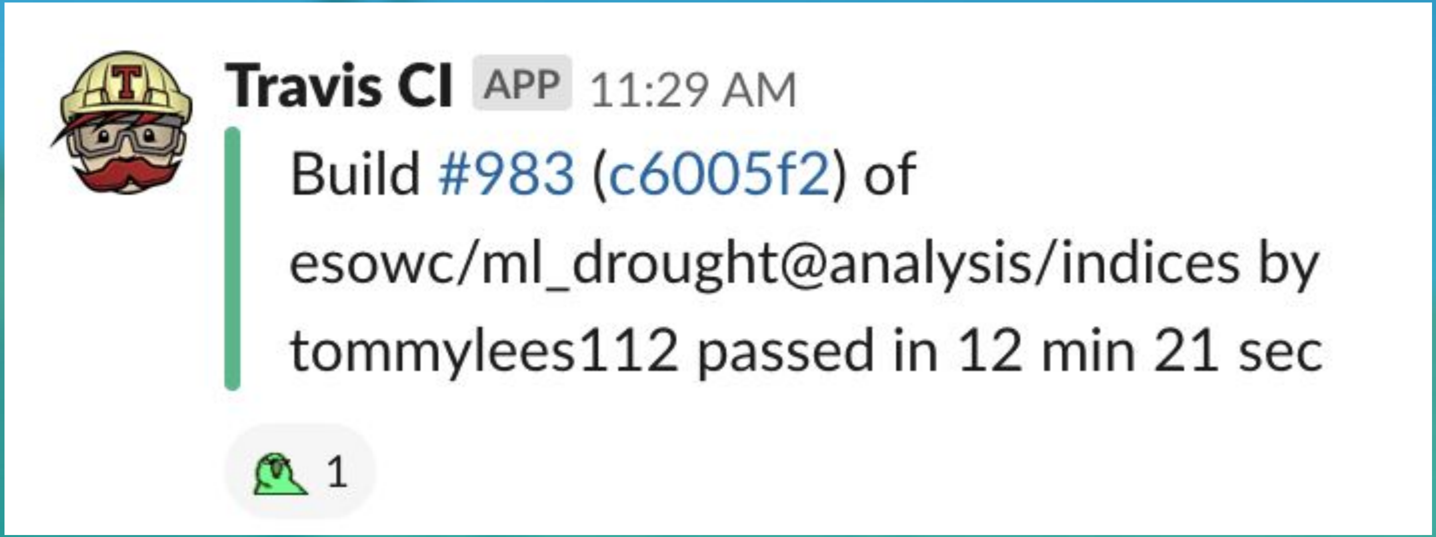
We wanted to utilise the tools from Software Engineering 🤓



T

We wanted to allow other people (and future us) to **train**, **use** and **reproduce** our models 🤩

Unit testing allows us to be confident our pipeline does what we expect



We used type hints to better communicate what functions did, and to leverage type checkers

```python
    def preprocess(self, subset_str: Optional[str] = 'kenya',
                   regrid: Optional[Path] = None,
                   resample_time: Optional[str] = 'M',
                   upsampling: bool = False,
                   parallel: bool = False,
                   cleanup: bool = True) -> None:
        """ Preprocess all of the era5 POS .nc files to produce
        one subset file.
```

We use json configurations to keep track of what experiments are run (WIP)

```
    },
    "models": {
      "Persistence": {
        "init_args":  {"experiment":  "one_month_forecast"},
        "train_args":  {},
        "evaluate_args": {"save_preds":  true}},
      "EARecurrentNetwork": {
        "init_args": {"hidden_size": 128, "experiment": "one_month_forecast", "include_pred_month": true},
        "train_args": {"num_epochs": 50, "early_stopping": 5},
        "evaluate_args": {"save_preds": true}
      }
    }
}
```

```
(esowc-drought) Gabriels-MacBook-Pro:ml_drought gabrieltseng$ python run.py --help
usage: run.py [-h] [--config CONFIG] [--run-from RUN_FROM]

Run the drought prediction pipeline

optional arguments:
  -h, --help           show this help message and exit
  --config CONFIG      path to configuration file describing the pipeline to
                       be run
  --run-from RUN_FROM  Which step to start the pipeline from
```

G

Making scientific workflows fully reproducible is hard...

- Unit Testing
- Experimental vs. Library code
- Source Control
- Parameters as Arguments

- Documentation
- Instructions

B+

Lessons learned:

- Infrastructure **communicates what code does**.
- The **initial time investment** is worth investing.
- There is a tension between **experimenting quickly**, and maintaining **well-tested robust code**.

Our Summer in Numbers

- **131** commits to master branch
- **37** Github issues
- **87** Pull Requests
- **+15,600** slack messages
- **18,544** lines of Python code
- **188** tests written

15,600 messages

131 commits

18,544 lines of code

To the future ...

- Work with **forecast data**
- Test for different problems
- ***ml_climate.readthedocs.io*** documentation!
- Improve our VCI predictions

Usability Performance Analysis

Let's innovate together.

SCAN ME

ECMWF

#ESoWC2019

Appendix

We focused on predicting an agricultural drought index in Kenya 🔥

National Drought Management Authority
NDMA
Committed to ending drought emergencies

ECMWF

#ESoWC2019

T

Lessons learned:

- The biggest benefit of all this infrastructure is **easier communication of what code is supposed to do**
- A little overhead at the beginning of the project (setting up CI) reaps big rewards later
- It is challenging to manage the tension between experimenting quickly, but also keeping well-tested robust code

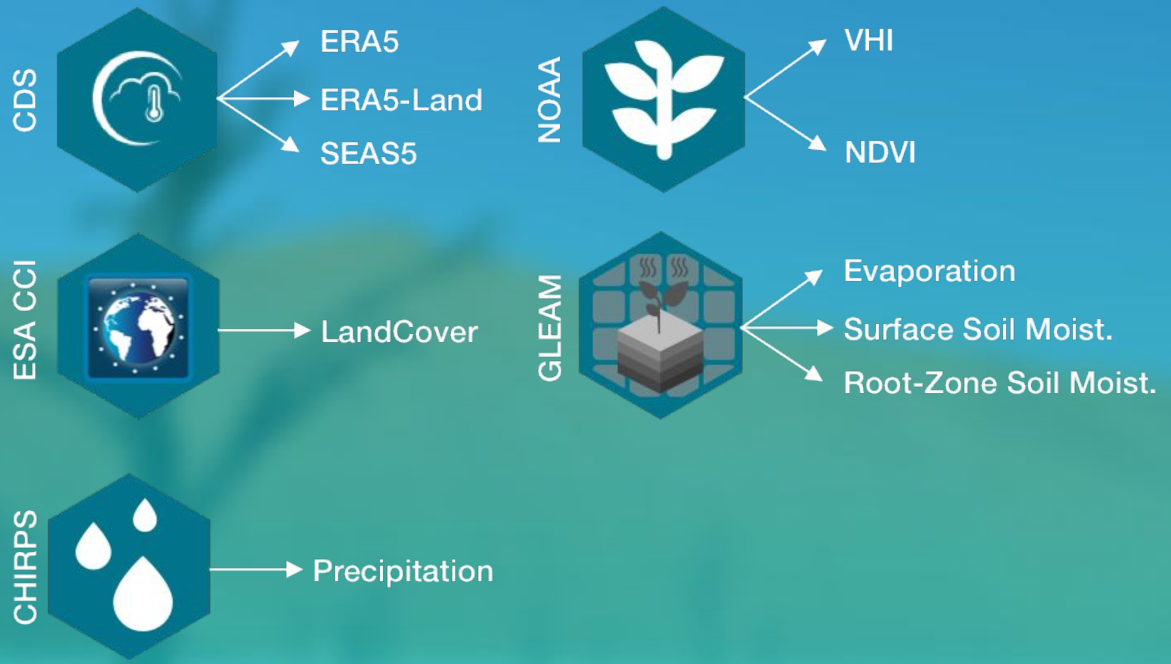
- data /
  - raw
    - { dataset }
  - interim
    - { dataset }_interim
    - { dataset }_preprocessed
  - models
    - { experiment }
      - { model }
        - results.json
        - preds_{ year }_{ month }.nc
  - features
    - { experiment }
      - normalizing_dict.pkl
      - test
        - 2017
          - x.nc
          - y.nc
      - train
        - 2016
          - x.nc
          - y.nc

ECMWF

#ESoWC2019

G

Our data sources include satellite data and model outputs



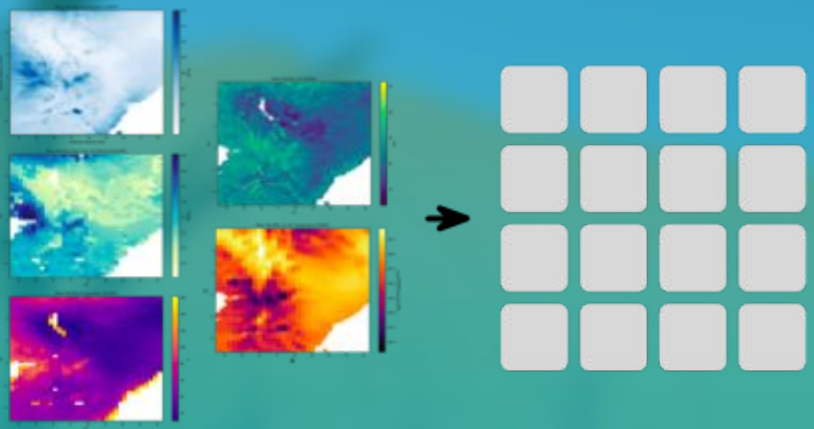
exporters

The Exporters are responsible for **downloading** data from external sources. They have a common `exporter.export()` method for downloading data.

```
def export_vhi():
    # if the working directory is alread ml_drought don't need
    ../data
    if Path('.').absolute().as_posix().split('/')[-1] ==
    'ml_drought':
        data_path = Path('data')
    else:
        data_path = Path('../data')
    exporter = VHIExporter(data_path)

    exporter.export()


def export_chirps():
    # if the working directory is alread ml_drought don't need
    ../data
    if Path('.').absolute().as_posix().split('/')[-1] ==
    'ml_drought':
        data_path = Path('data')
    else:
        data_path = Path('../data')
    exporter = CHIRPSExporter(data_path)

    exporter.export(years=None, region='global',
    period='monthly')
```

preprocessors

The Preprocessors work to convert these different datasets into a **unified data format**. This makes testing and developing different models much more straightforward.



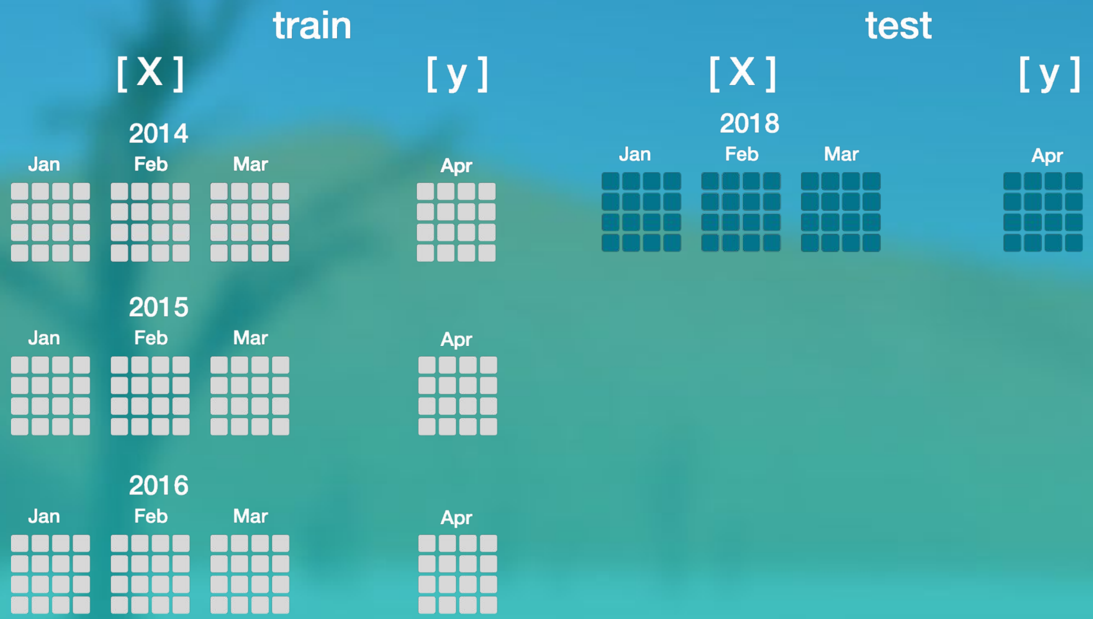
preprocessors

The Preprocessors work to convert these different datasets into a **unified data format**. This makes testing and developing different models much more straightforward.

```python
def process_precip_2018():
    # if the working directory is alread ml_drought don't need
    ../data
    if Path('.').absolute().as_posix().split('/')[-1] ==
    'ml_drought':
        data_path = Path('data')
    else:
        data_path = Path('../data')

    processor = CHIRPSPreprocesser(data_path)

    processor.preprocess(subset_str='kenya',
                         parallel=False)
```

engineer

The Engineer class works to create **train and test data**. This class reads preprocessed data and writes out X, y pairs of data. This class allows us enormous flexibility to choose input and output variables.



engineer

The Engineer class works to create **train and test data**. This class reads preprocessed data and writes out X, y pairs of data. This class allows us enormous flexibility to choose input and output variables.

```
def engineer(experiment='one_month_forecast', process_static=True,
             pred_months=12, test_year=2018):
    # if the working directory is alread ml_drought don't need ../
    data
    if Path('.').absolute().as_posix().split('/')[-1] ==
    'ml_drought':
        data_path = Path('data')
    else:
        data_path = Path('../data')

    engineer = Engineer(data_path, experiment=experiment,
    process_static=process_static)
    engineer.engineer(
        test_year=test_year, target_variable='VCI',
        pred_months=pred_months, expected_length=pred_months,
    )
```

models

The Models implement our **predictions**. They take the train / test data output by the engineer for fitting and produce predicted values. They are currently mostly machine learning models.

- Persistence (previous month)
- Linear Regression
- Linear (classical) Neural Network
- Recurrent Neural Network - LSTM
- Entity Aware LSTM (Hydrology Specific Architecture!)

models

The Models implement our **predictions**. They take the train / test data output by the engineer for fitting and produce predicted values. They are currently mostly machine learning models.

```
def regression(
    experiment='one_month_forecast',
    include_pred_month=True,
    surrounding_pixels=1,
    ignore_vars=None,
    include_static=True
):
    # if the working directory is alread ml_drought don't need
    ../data
    if Path('.').absolute().as_posix().split('/')[-1] ==
    'ml_drought':
        data_path = Path('data')
    else:
        data_path = Path('../data')

    predictor = LinearRegression(
        data_path, experiment=experiment,
        include_pred_month=include_pred_month,
        surrounding_pixels=surrounding_pixels,
        include_static=include_static,
    )
    predictor.train()
    predictor.evaluate(save_preds=True)
```

```
def linear_nn(
    experiment='one_month_forecast',
    include_pred_month=True,
    surrounding_pixels=1,
    ignore_vars=None,
    include_static=True,
):
    # if the working directory is alread ml_drought don't need
    ../data
    if Path('.').absolute().as_posix().split('/')[-1] ==
    'ml_drought':
        data_path = Path('data')
    else:
        data_path = Path('../data')

    predictor = LinearNetwork(
        layer_sizes=[100], data_folder=data_path,
        experiment=experiment,
        include_pred_month=include_pred_month,
        surrounding_pixels=surrounding_pixels,
        include_static=include_static,
    )
    predictor.train(num_epochs=50, early_stopping=5)
    predictor.evaluate(save_preds=True)
    predictor.save_model()
```

models

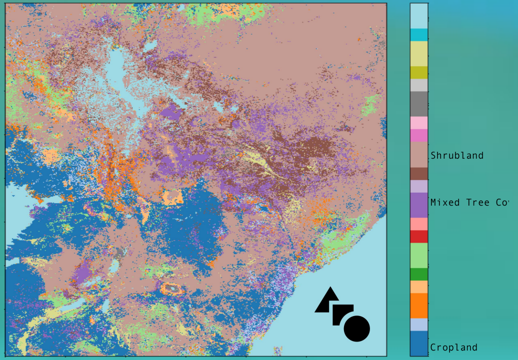
Incorporate static variables and dynamic variables

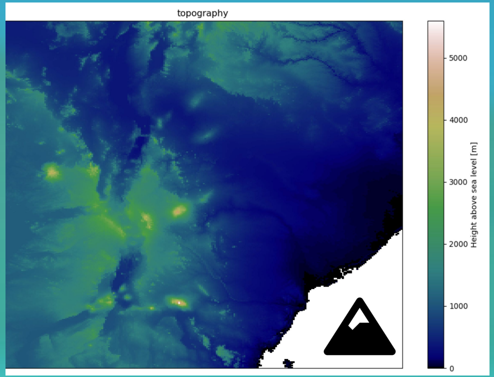
Benchmarking a Catchment-Aware Long Short-Term Memory Network (LSTM) for Large-Scale Hydrological Modeling

Frederik Kratzert[1], Daniel Klotz[1], Guy Shalev[2], Günter Klambauer[1], Sepp Hochreiter[1,*], and Grey Nearing[3,*]

[1]LIT AI Lab & Institute for Machine Learning, Johannes Kepler University Linz, Austria
[2]Google Research
[3]Department of Geological Sciences, University of Alabama, Tuscaloosa, AL United States
[*]Shared last author

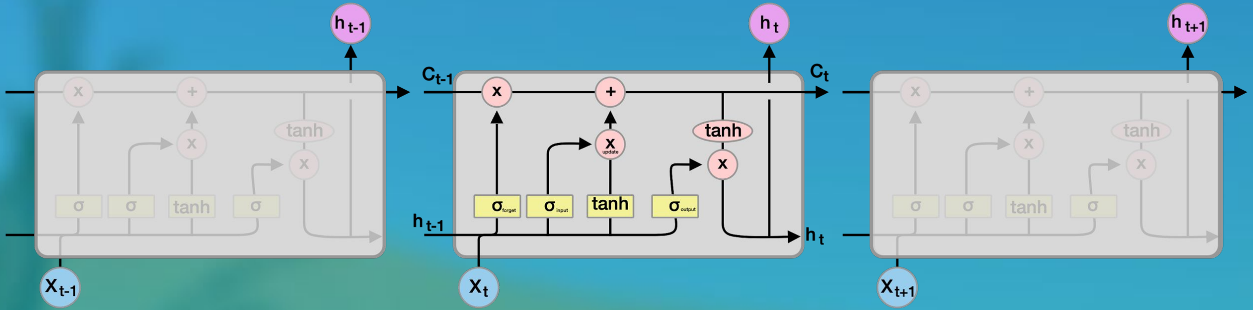
Correspondence: Frederik Kratzert (kratzert@ml.jku.at)



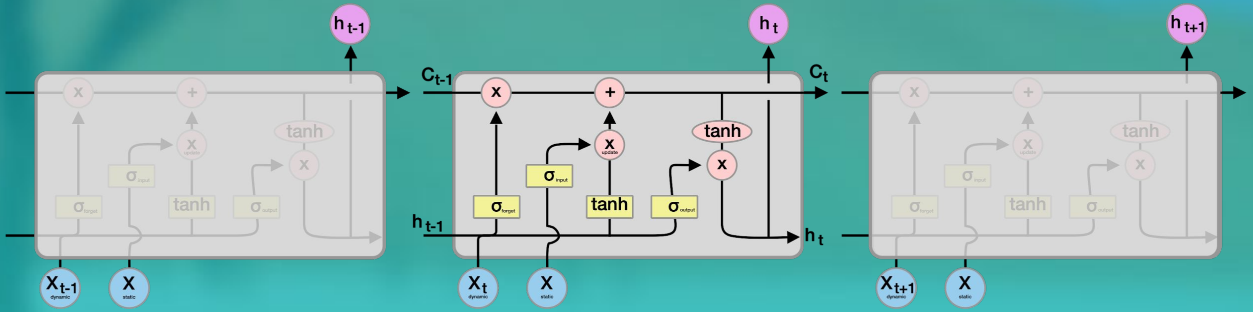


models

Classic LSTM



Entity Aware LSTM

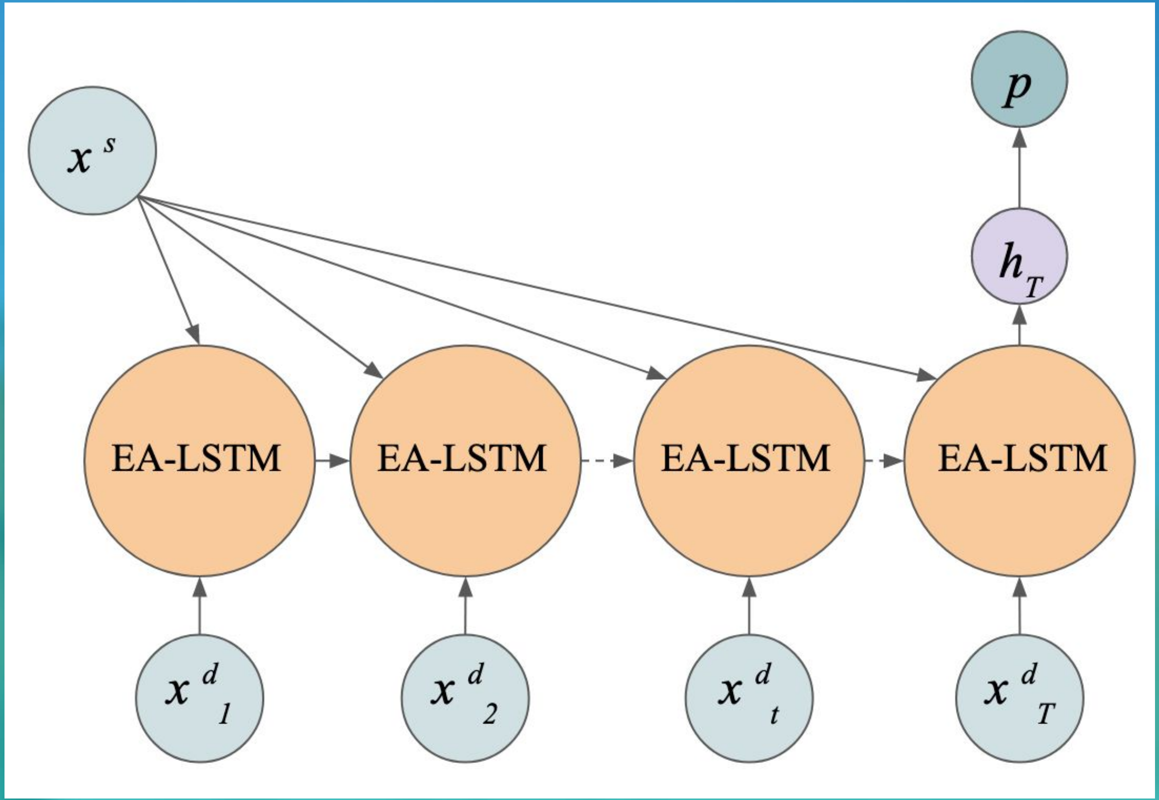


ECMWF

#ESoWC2019

G

models



Initial Experiments

t = 0

t = 1

train = 1981 - 2017
test = 2018
n previous months = 12

Initial Experiments

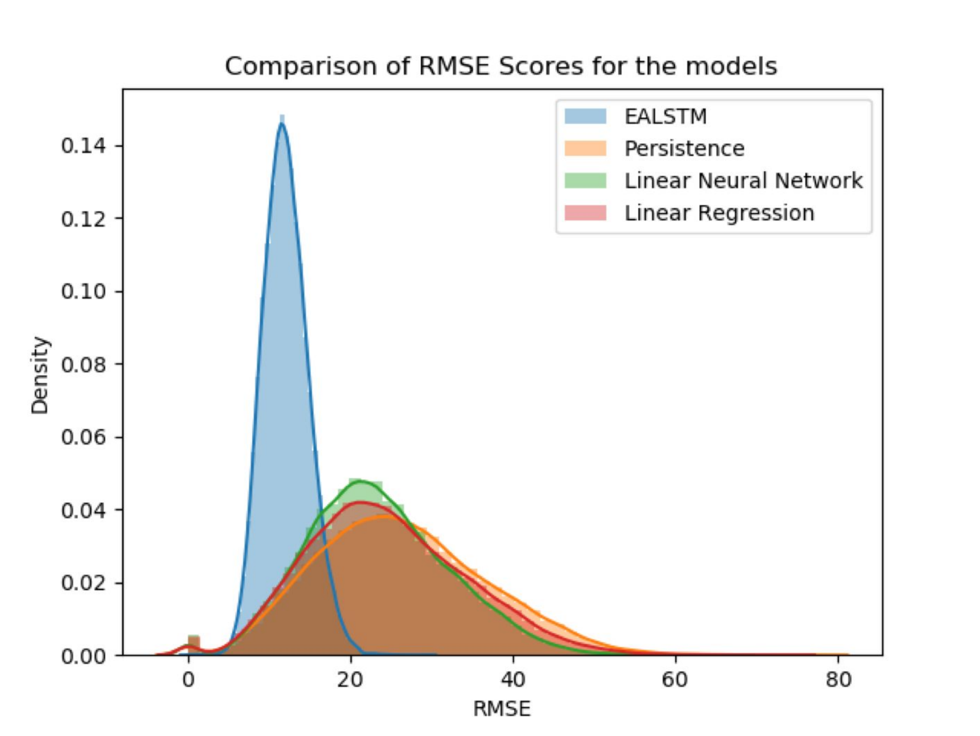
t = 0

t = 1

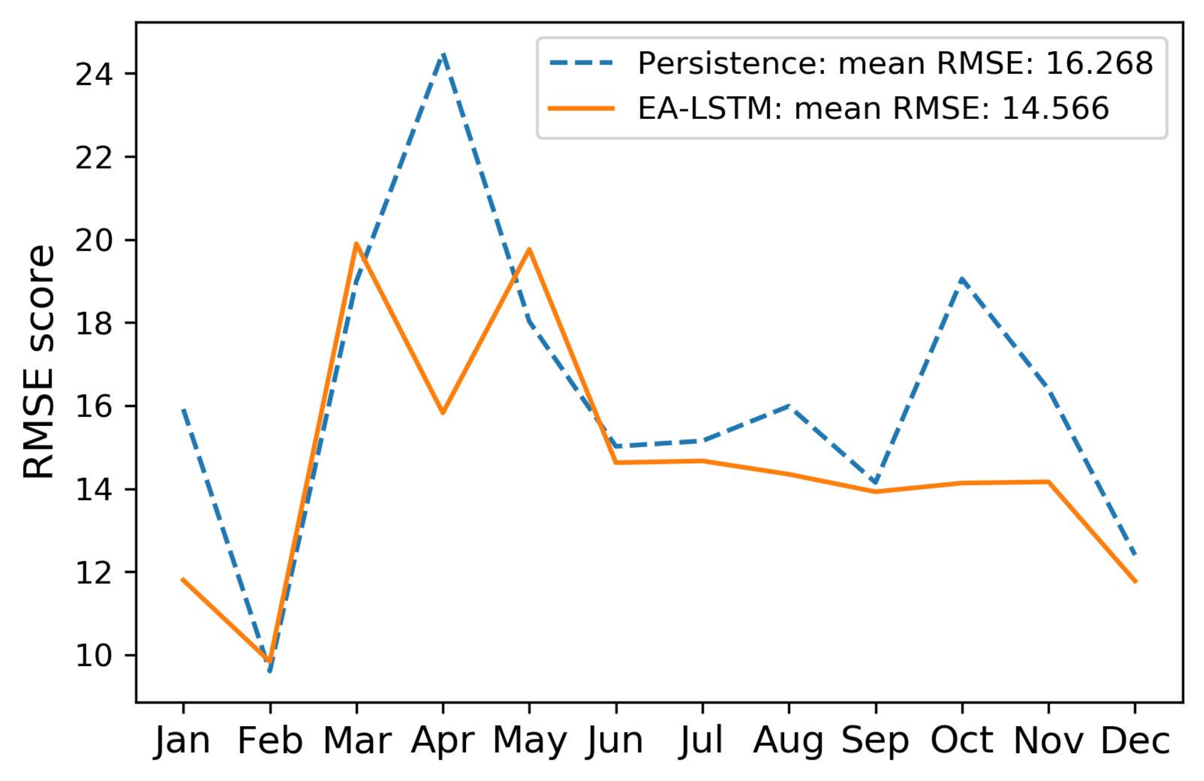
train = 1981 - 2017
test = 2018
n previous months = 12

Initial Results - EALSTM

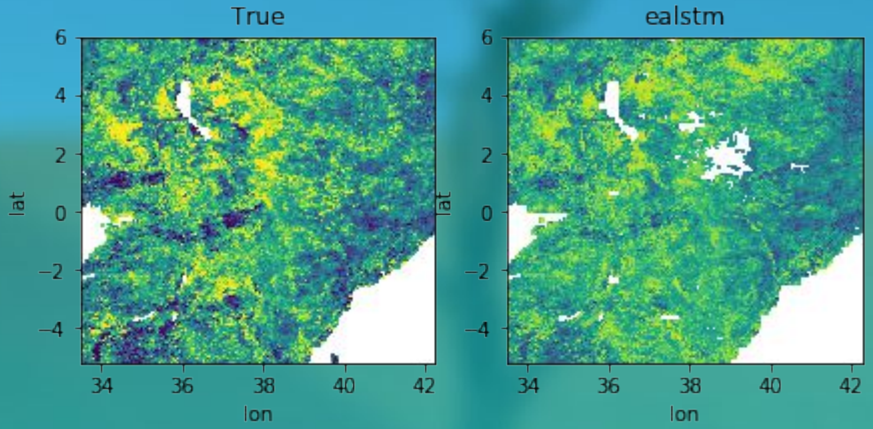
- We fit a number of different machine learning models.
- EALSTM seems to have performed significantly better than the other models.

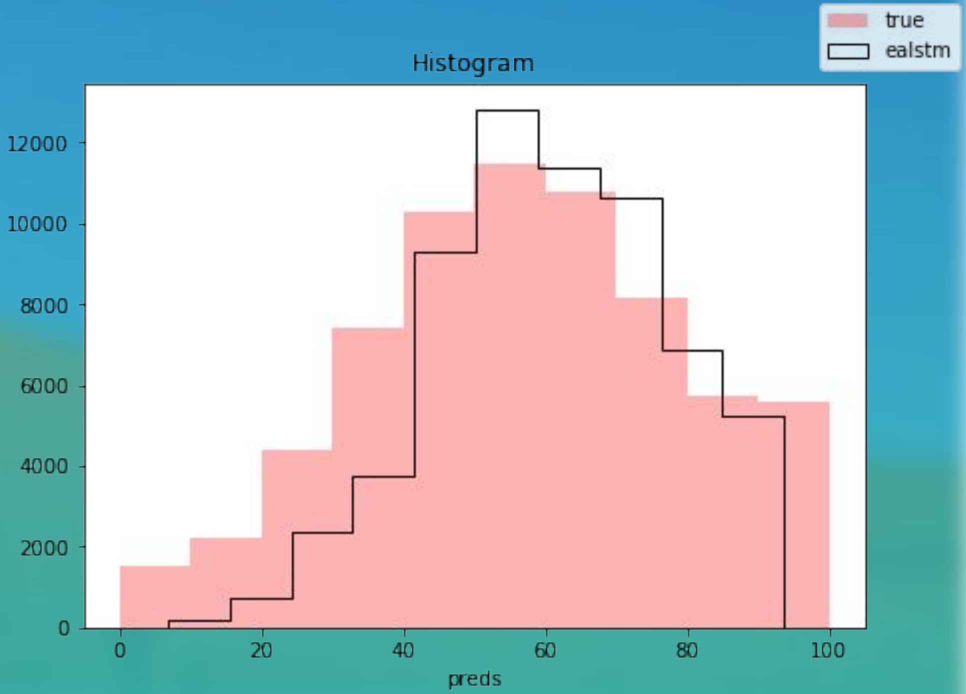


Initial Results

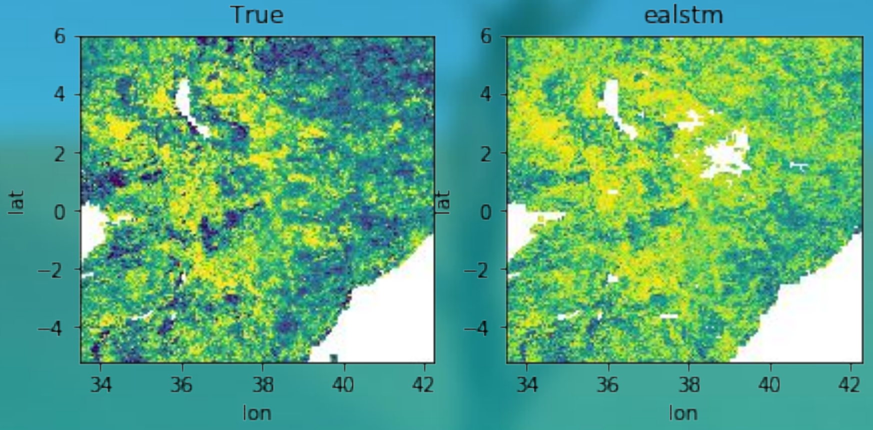


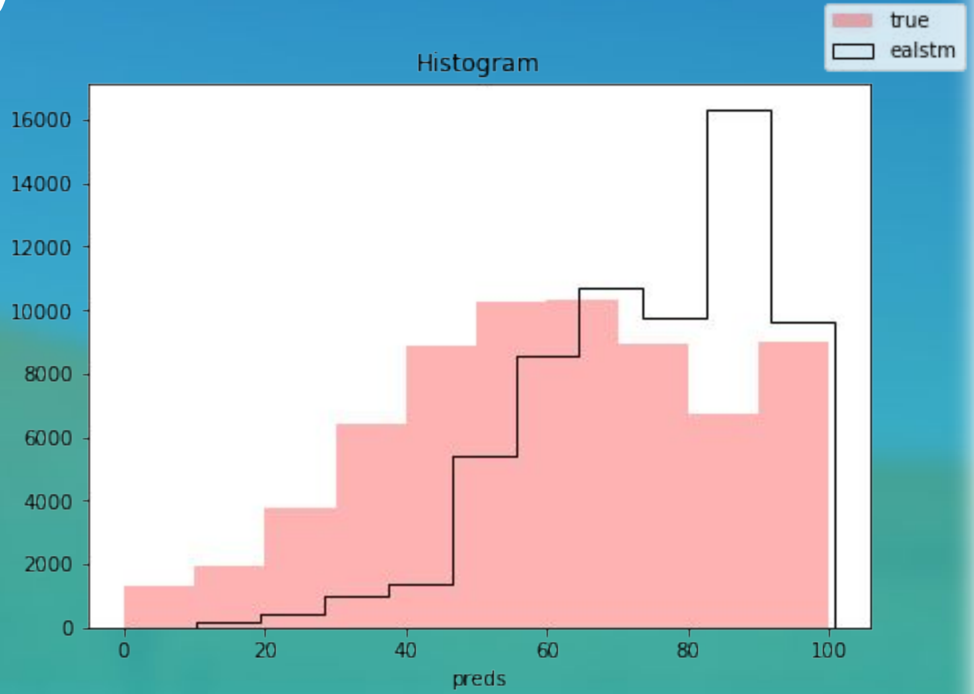
Preliminary Results in April





Preliminary Results in May

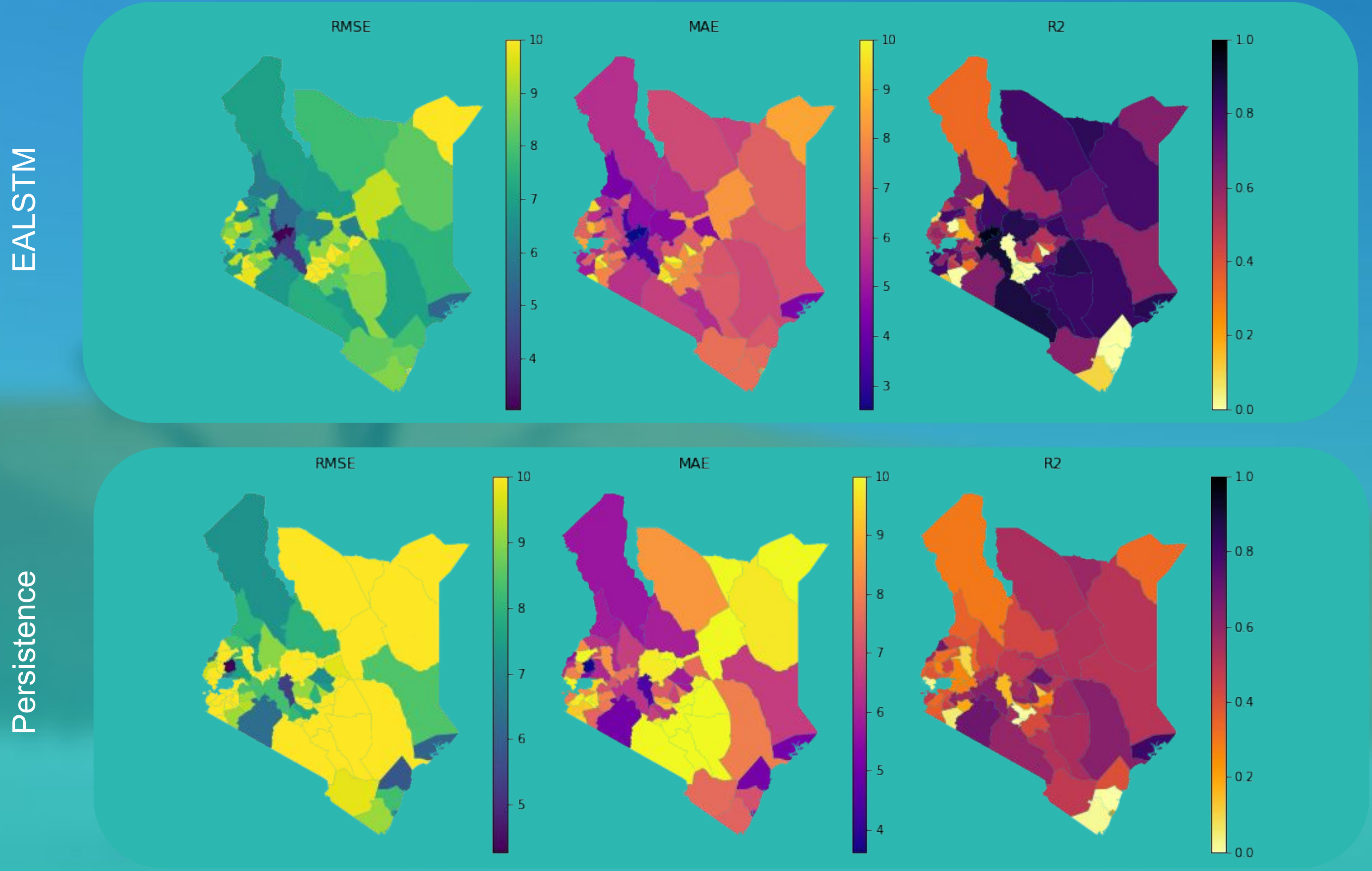




ECMWF

#ESoWC2019

T


EALSTM
RMSE
MAE
R2
10
9
8
7
6
5
4
3
1.0
0.8
0.6
0.4
0.2
0.0
Persistence
RMSE
MAE
R2

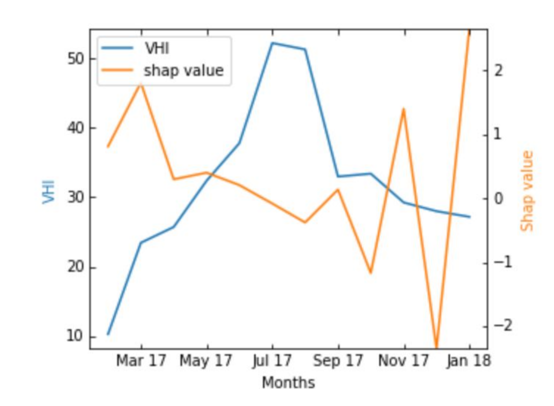
Initial Results

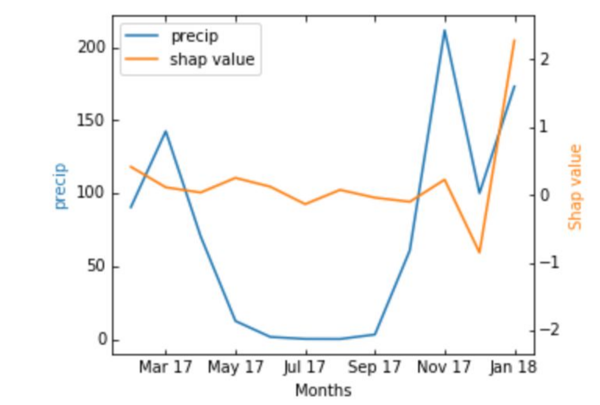
District	VCI3M Adede et al. [2019]	VCI3M EA-LSTM	VCI3M Persistence	VCI EA-LSTM	VCI Persistence
Mandera	0.71	0.91	0.57	0.80	0.34
Marsabit	0.77	0.75	0.50	0.77	0.54
Turkana	0.83	0.39	0.41	0.27	0.30
Wajir	0.71	0.82	0.52	0.83	0.51

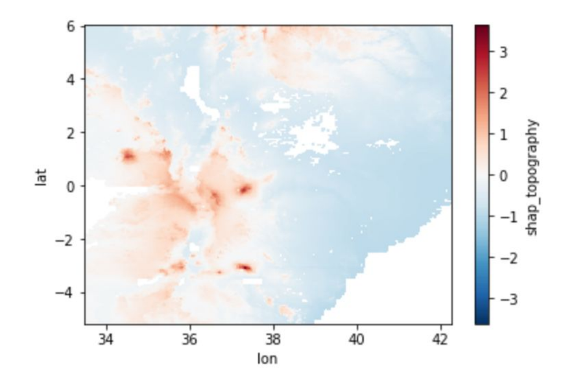
Table 1: R^2 values reported by Adede et al. [2019], our EA-LSTM model and the baseline persistence model. For the Adede et al. [2019] model, VCI is aggregated over three months - results in both the aggregated and unaggregated case are presented. The performance of the model in Turkana compared to the other districts suggests there is still region-specific information the model is missing.

Initial Results

https://github.com/esowc/ml_drought/blob/master/notebooks/draft/15_gt_ealstm.ipynb







ECMWF

#ESoWC2019

G

Results - We do best in Cropland areas

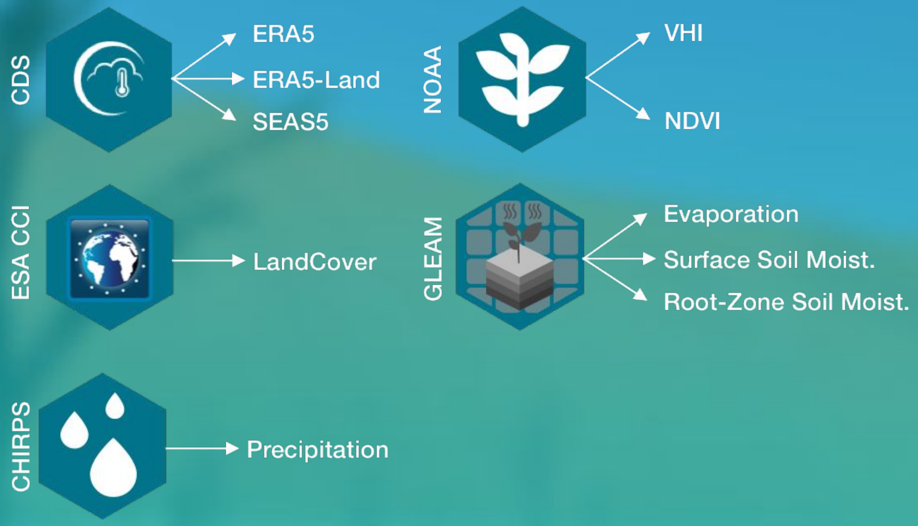
	model	admin_level_name	region_name	rmse	mae	r2
0	rnn	landcover	cropland_rainfed_one_hot	8.143682	5.395350	0.199256
1	rnn	landcover	tree_or_shrub_cover_one_hot	8.618315	6.953503	-13.611413
2	rnn	landcover	cropland_irrigated_or_postflooding_one_hot	9.545614	6.901316	-0.237254
3	rnn	landcover	no_data_one_hot	NaN	NaN	NaN
4	rnn	landcover	herbaceous_cover_one_hot	8.157337	5.716190	0.149336
5	ealstm_surrounding_0	landcover	cropland_rainfed_one_hot	5.404216	4.189554	0.647371
6	ealstm_surrounding_0	landcover	tree_or_shrub_cover_one_hot	5.068028	4.075745	-4.052724
7	ealstm_surrounding_0	landcover	cropland_irrigated_or_postflooding_one_hot	5.509399	4.098703	0.587846
8	ealstm_surrounding_0	landcover	no_data_one_hot	NaN	NaN	NaN
9	ealstm_surrounding_0	landcover	herbaceous_cover_one_hot	5.374018	4.210546	0.630802

Appendix - Administrative Level performance

	Unnamed: 0	model	admin_level_name	rmse	mae	r2
0	0	rnn	region_kenya	10.456015	7.960484	0.296722
1	1	ealstm_surrounding_0	region_kenya	7.250155	5.744490	0.662912
2	2	rnn	division_l3_kenya	11.729663	9.034048	0.362789
3	3	ealstm_surrounding_0	division_l3_kenya	8.297145	6.545985	0.679212
4	4	rnn	location_l4_kenya	12.860918	9.986623	0.388949
5	5	ealstm_surrounding_0	location_l4_kenya	9.696184	7.643787	0.653535
6	6	rnn	district_l2_kenya	10.456015	7.960484	0.296722
7	7	ealstm_surrounding_0	district_l2_kenya	7.250155	5.744490	0.662912
8	8	rnn	province_l1_kenya	9.904732	7.494977	0.353457
9	9	ealstm_surrounding_0	province_l1_kenya	6.613191	5.059705	0.711774

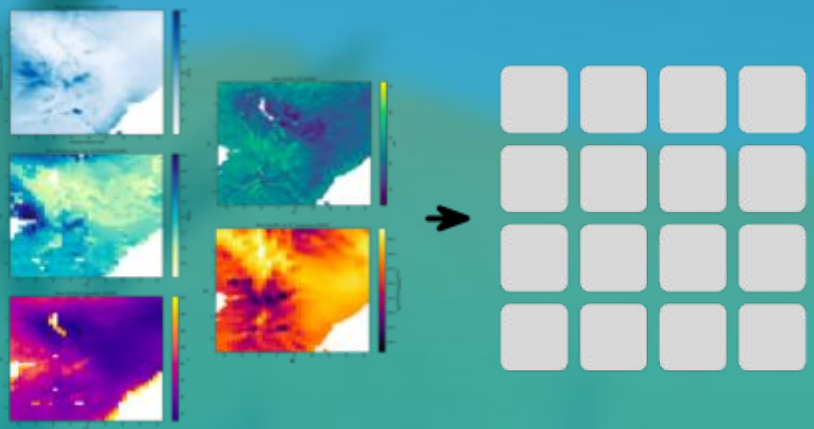
exporters

The Exporters are responsible for **downloading** data from external sources. They have a common `exporter.export()` method for downloading data.



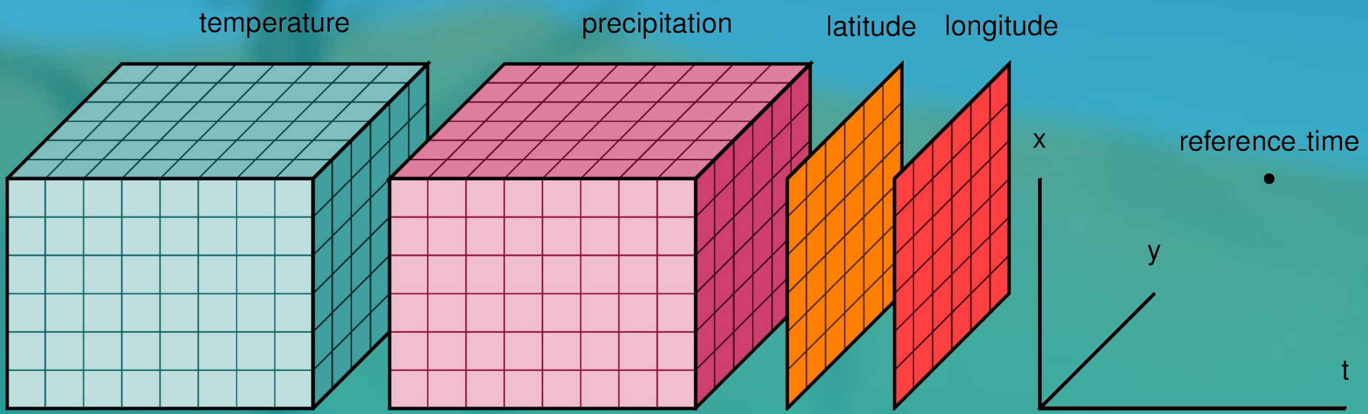
preprocessors

The Preprocessors work to convert these different datasets into a **unified data format**. This makes testing and developing different models much more straightforward.



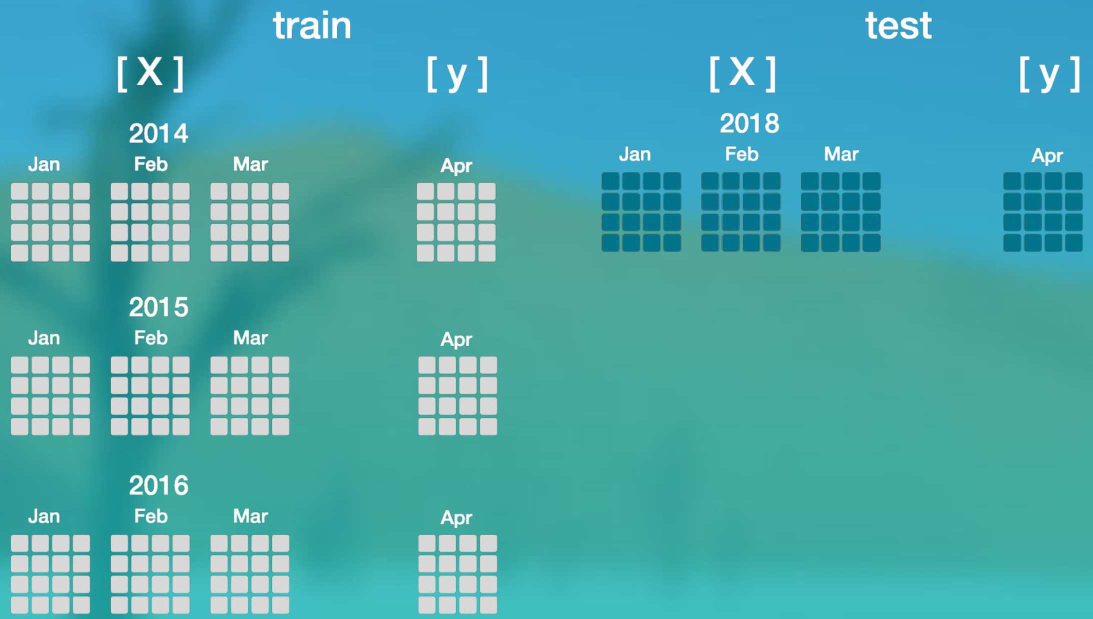
preprocessors

The Preprocessors work to convert these different datasets into a **unified data format**. This makes testing and developing different models much more straightforward.



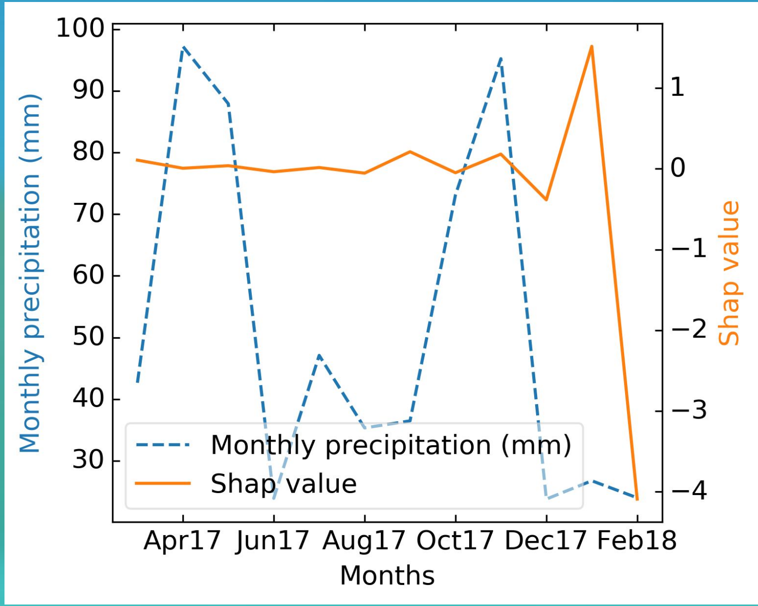
engineer

The Engineer class works to create **train and test data**. This class reads preprocessed data and writes out X, y pairs of data. This class allows us enormous flexibility to choose input and output variables.



Model interpretability was a key component of our pipeline 🤩

- We were particularly interested in understanding the **patterns being learnt by the model**.
- We used the DeepSHAP implementation of DeepLIFT to interpret how an **input data points affected a model's final prediction**



models

The Models implement our **predictions**. They take the train / test data output by the engineer for fitting and produce predicted values. They are currently mostly machine learning models.

- Persistence (previous month)
- Linear Regression
- Linear (classical) Neural Network
- Recurrent Neural Network - LSTM
- Entity Aware LSTM (Hydrology Specific Architecture!)

scikit learn

G

In addition to the pixel wise climate values, we fed the model a few additional inputs 🥳

- Model extras
  - Surrounding Pixels
  - One hot encoded month
  - Spatial Climatology (for each month)
  - Spatial Mean of input timesteps

analyzers

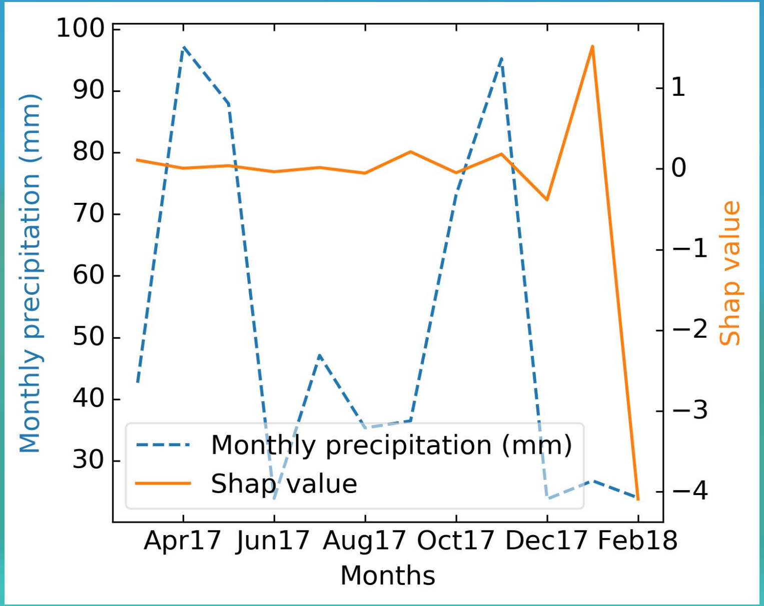
The Analyzers are a broad set of classes for **plotting and understanding** our predictions. They help us explore regional trends, better understand the patterns models are learning and evaluate our performance.

- Climate Indices.
- Identify ‘runs’ of drought events.
- Aggregate results by region or landcover.
- Diagnose feature contributions (SHAP).
- Calculate performance metrics (RMSE, R^2).
- Plotting functions.

analyzers

The Analyzers are a broad set of classes for **plotting and understanding** our predictions. They help us explore regional trends, better understand the patterns models are learning and evaluate our performance.

- We were particularly interested in understanding the **patterns being learnt by the model**.
- We used the DeepSHAP implementation of DeepLIFT to interpret how an **input data points affected a model's final prediction**



ECMWF

#ESoWC2019