We are a PhD Student and a Machine Learning Engineer

#ESoWC2019
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!

#ESoWC2019
We developed a modular and extensible pipeline to apply machine learning to climate science.
First results - using machine learning to predict vegetation health

Table 1: $R^2$ values reported by Adee et al. [2019], our EA-LSTM model and the baseline persistence model. For the Adee 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!
We were strongly influenced by this talk 😐

Reproducibility
@ ICLR 2019

ICLR 2019
"Reproducibility in ML" workshop

Joel Grus
@joelgrus

ECMWF

#ESoWC2019
Reproducibility is central to the ideals of scientific research.
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 😎

"reproducibility"
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 😊

Travis CI

Build #983 (c6005f2) of esowc/ml_drought@analysis/indices by tommylees112 passed in 12 min 21 sec

#ESoWC2019
We used type hints to better communicate what functions did, and to leverage type checkers.

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).
Making scientific workflows fully reproducible is hard...

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

- Documentation
- Instructions

#ESoWC2019
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

#ESoWC2019
To the future ...

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

Usability  Performance  Analysis

ECMWF  #ESoWC2019
Let’s innovate together.
Appendix
We focused on predicting an agricultural drought index in Kenya 🔥
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
Our data sources include satellite data and model outputs.

- **CDS**
  - ERA5
  - ERA5-Land
  - SEAS5

- **NOAA**
  - VHI
  - NDVI

- **ESA CCI**
  - LandCover

- **GLEAM**
  - Evaporation
  - Surface Soil Moist.
  - Root-Zone Soil Moist.

- **CHIRPS**
  - Precipitation

ECMWF

#ESoWC2019
The Exporters are responsible for **downloading** data from external sources. They have a common `exporter.export()` method for downloading data.

```python
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')
```
The Preprocessors work to convert these different datasets into a **unified data format**. This makes testing and developing different models much more straightforward.
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 already 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 = CHIRPSPreprocessor(data_path)

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

```python
def engineer(experiment='one_month_forecast', process_static=True, pred_months=12, test_year=2018):
    # if the working directory is already 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,
    )
```
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!)
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.

```python
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()
```
Incorporate static variables and dynamic variables.
models

Classic LSTM

Entity Aware LSTM

ECMWF

#ESoWC2019
models

\[ x^s \]

\[ \text{EA-LSTM} \rightarrow \text{EA-LSTM} \rightarrow \text{EA-LSTM} \rightarrow \text{EA-LSTM} \]

\[ x^d_1 \rightarrow x^d_2 \rightarrow x^d_t \rightarrow x^d_T \]

\[ p \]

\[ h_T \]
Initial Experiments

$t = 0$
- Water droplets
- Temperature
- Water level
- Temperature increase
- Rain

$t = 1$
- Plant growth

Training period: 1981 - 2017
Test period: 2018
Number of previous months: 12

#ESoWC2019
Initial Experiments

$t = 0$

$t = 1$

train = 1981 - 2017

test = 2018

n previous months = 12

#ESoWC2019
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

- Persistence: mean RMSE: 16.268
- EA-LSTM: mean RMSE: 14.566
Preliminary Results in April

[Images of maps and histograms showing comparisons between True and ealstm models]
Preliminary Results in May

ECMWF

#ESoWC2019
## Initial Results

<table>
<thead>
<tr>
<th>District</th>
<th>Adede et al. [2019]</th>
<th>VCI3M</th>
<th>EA-LSTM</th>
<th>Persistence</th>
<th>VCI</th>
<th>EA-LSTM</th>
<th>Persistence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mandera</td>
<td>0.71</td>
<td>0.91</td>
<td>0.57</td>
<td></td>
<td>0.80</td>
<td>0.77</td>
<td>0.54</td>
</tr>
<tr>
<td>Marsabit</td>
<td>0.77</td>
<td>0.75</td>
<td>0.50</td>
<td></td>
<td>0.27</td>
<td>0.77</td>
<td>0.54</td>
</tr>
<tr>
<td>Turkana</td>
<td>0.83</td>
<td>0.39</td>
<td>0.41</td>
<td></td>
<td>0.27</td>
<td>0.77</td>
<td>0.54</td>
</tr>
<tr>
<td>Wajir</td>
<td>0.71</td>
<td>0.82</td>
<td>0.52</td>
<td></td>
<td>0.83</td>
<td>0.30</td>
<td>0.51</td>
</tr>
</tbody>
</table>

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
## Results - We do best in Cropland areas

<table>
<thead>
<tr>
<th>model</th>
<th>admin_level_name</th>
<th>region_name</th>
<th>rmse</th>
<th>mae</th>
<th>r2</th>
</tr>
</thead>
<tbody>
<tr>
<td>rnn</td>
<td>landcover</td>
<td>cropland_rainfed_one_hot</td>
<td>8.143482</td>
<td>5.395350</td>
<td>0.199256</td>
</tr>
<tr>
<td>rnn</td>
<td>landcover</td>
<td>tree_or_shrub_cover_one_hot</td>
<td>8.618315</td>
<td>6.953503</td>
<td>-13.611413</td>
</tr>
<tr>
<td>rnn</td>
<td>landcover</td>
<td>cropland_irrigated_or_postflooding_one_hot</td>
<td>9.545614</td>
<td>6.901316</td>
<td>-0.237254</td>
</tr>
<tr>
<td>rnn</td>
<td>landcover</td>
<td>no_data_one_hot</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>rnn</td>
<td>landcover</td>
<td>herbaceous_cover_one_hot</td>
<td>8.157337</td>
<td>5.716190</td>
<td>0.149336</td>
</tr>
<tr>
<td>ealstm_surrounding_0</td>
<td>landcover</td>
<td>cropland_rainfed_one_hot</td>
<td>5.404216</td>
<td>4.189554</td>
<td>0.647371</td>
</tr>
<tr>
<td>ealstm_surrounding_0</td>
<td>landcover</td>
<td>tree_or_shrub_cover_one_hot</td>
<td>5.068028</td>
<td>4.075745</td>
<td>-4.052724</td>
</tr>
<tr>
<td>ealstm_surrounding_0</td>
<td>landcover</td>
<td>cropland_irrigated_or_postflooding_one_hot</td>
<td>5.509399</td>
<td>4.098703</td>
<td>0.587846</td>
</tr>
<tr>
<td>ealstm_surrounding_0</td>
<td>landcover</td>
<td>no_data_one_hot</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>ealstm_surrounding_0</td>
<td>landcover</td>
<td>herbaceous_cover_one_hot</td>
<td>5.374018</td>
<td>4.210546</td>
<td>0.630802</td>
</tr>
</tbody>
</table>
# Appendix - Administrative Level performance

<table>
<thead>
<tr>
<th>Unnamed: 0</th>
<th>model</th>
<th>admin_level_name</th>
<th>rmse</th>
<th>mae</th>
<th>r2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>rnn</td>
<td>region_kenya</td>
<td>10.456015</td>
<td>7.960484</td>
<td>0.296722</td>
</tr>
<tr>
<td>1</td>
<td>ealstm_surrounding_0</td>
<td>region_kenya</td>
<td>7.250155</td>
<td>5.744490</td>
<td>0.662912</td>
</tr>
<tr>
<td>2</td>
<td>rnn</td>
<td>division_13_kenya</td>
<td>11.729663</td>
<td>9.034048</td>
<td>0.362789</td>
</tr>
<tr>
<td>3</td>
<td>ealstm_surrounding_0</td>
<td>division_13_kenya</td>
<td>8.297145</td>
<td>6.545985</td>
<td>0.679212</td>
</tr>
<tr>
<td>4</td>
<td>rnn</td>
<td>location_14_kenya</td>
<td>12.860918</td>
<td>9.986623</td>
<td>0.388949</td>
</tr>
<tr>
<td>5</td>
<td>ealstm_surrounding_0</td>
<td>location_14_kenya</td>
<td>9.696184</td>
<td>7.643787</td>
<td>0.653353</td>
</tr>
<tr>
<td>6</td>
<td>rnn</td>
<td>district_12_kenya</td>
<td>10.456015</td>
<td>7.960484</td>
<td>0.296722</td>
</tr>
<tr>
<td>7</td>
<td>ealstm_surrounding_0</td>
<td>district_12_kenya</td>
<td>7.250155</td>
<td>5.744490</td>
<td>0.662912</td>
</tr>
<tr>
<td>8</td>
<td>rnn</td>
<td>province_11_kenya</td>
<td>9.904732</td>
<td>7.494977</td>
<td>0.353457</td>
</tr>
<tr>
<td>9</td>
<td>ealstm_surrounding_0</td>
<td>province_11_kenya</td>
<td>6.613191</td>
<td>5.059705</td>
<td>0.711774</td>
</tr>
</tbody>
</table>

---

ECMWF          #ESoWC2019
The Exporters are responsible for **downloading** data from external sources. They have a common `exporter.export()` method for downloading data.
The Preprocessors work to convert these different datasets into a **unified data format**. This makes testing and developing different models much more straightforward.
The Preprocessors work to convert these different datasets into a **unified data format**. This makes testing and developing different models much more straightforward.
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.
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!)
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
● 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.

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.