WildfiresKangarooIsland

September 2, 2022

1 Crash course on machine learning on WEkEO data

2 OR

3 Random forest on a forest…?

Welcome to this notebook! Here you will learn how to train a popular class of machine learning model and use WEkEO data to estimate forest canopy.

Photograph by Geekstreet. CC BY-SA 4.0

3.1 The steps…

We will access and analyse data from the WEkEO HDA API Client using the following steps:

1. Section **??** data: We will demonstrate how to download data from the Wekeo API client, then download some pre-prepared data from the Wekeo data viewer.
2. Section **??** data: Prepare the data by reformatting and averaging it. Visualize the data and begin to explore corellations semi-manually.
3. Section **??** the data: Build and run a random forest model.

[35]:
```python
# import all packages

import glob
import os
import datetime
import logging
from zipfile import ZipFile
import numpy as np
import pandas as pd
import netCDF4 as nc
from netCDF4 import Dataset
import xarray as xr
import rasterio
import rasterio.plot
import rasterio.merge
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import sklearn
```

```
from sklearn.ensemble import RandomForestRegressor

# set up logging
logger = logging.getLogger()
logger.setLevel(logging.WARNING)
```

[36]:
```
# we require the module seaborn in version >= 0.11
%conda install --yes -q seaborn=0.11
```

```
Collecting package metadata (current_repodata.json): …working… done
Solving environment: …working… done

# All requested packages already installed.


Note: you may need to restart the kernel to use updated packages.
```

[37]:
```
# In WEkEO Jupyterhub this is needed to make sure we import the correct version␣
 ↪of seaborn
import importlib.util
import sys
spec = importlib.util.spec_from_file_location("seaborn", "/opt/conda/lib/
 ↪python3.8/site-packages/seaborn/__init__.py")
sns = importlib.util.module_from_spec(spec)
sys.modules["seaborn"] = sns
spec.loader.exec_module(sns)

sns.set_style()
sns.set(font_scale=1.15)
```

[38]:
```
bbox = [136.4, -36.1, 138.2, -35.5]  # bounding box of our area of interest
```

3.2 Part 1: Download the data

We will get data about how vegetated the island we are studying is from FCover, the 10-daily fraction-of-vegetation-cover product. ### Download your first data Firstly, input your WEkEO credentials below. Alternatively, if you have set up a .hdarc file (instructions) you can skip this step. Then we can set up the API client that lets us run WEkEO downloads

[39]:
```
user = 'YourName'    # your WekEO username in quotes
password = 'password' # your WekEO password in quotes
```

[40]:
```
pip install -U hda
```

```
Requirement already up-to-date: hda in /opt/conda/lib/python3.8/site-packages
(0.2.2)
Requirement already satisfied, skipping upgrade: tqdm in
```

```
/opt/conda/lib/python3.8/site-packages (from hda) (4.50.2)
Requirement already satisfied, skipping upgrade: requests>=2.5.0 in
/opt/conda/lib/python3.8/site-packages (from hda) (2.24.0)
Requirement already satisfied, skipping upgrade: certifi>=2017.4.17 in
/opt/conda/lib/python3.8/site-packages (from requests>=2.5.0->hda) (2022.6.15)
Requirement already satisfied, skipping upgrade: chardet<4,>=3.0.2 in
/opt/conda/lib/python3.8/site-packages (from requests>=2.5.0->hda) (3.0.4)
Requirement already satisfied, skipping upgrade: idna<3,>=2.5 in
/opt/conda/lib/python3.8/site-packages (from requests>=2.5.0->hda) (2.10)
Requirement already satisfied, skipping upgrade:
urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /opt/conda/lib/python3.8/site-
packages (from requests>=2.5.0->hda) (1.25.11)
Note: you may need to restart the kernel to use updated packages.
```

[41]:
```python
from hda import Client

try:
    c = Client(url='https://wekeo-broker.apps.mercator.dpi.wekeo.eu/
 ↪databroker', user=user, password=password, debug=True)
except NameError:
    c = Client(debug=True)
```

This is the query we'll send to the client. Use the WEkEO DATA viewer as a helper if you ever want to make other queries for your own projects.

[42]:
```python
query = {
  "datasetId": "EO:CLMS:DAT:CGLS_GLOBAL_FCOVER300_V1_333M",
  "dateRangeSelectValues": [
    {
      "name": "dtrange",
      "start": "2018-01-01T00:00:00.000Z",
      "end": "2018-01-11T23:59:59.999Z"
    }
  ]
}
```

[43]:
```python
matches = c.search(query)
print(matches)
matches.download()
```

```
SearchResults[items=1,volume=1.8G,jobId=FtrocVUMrfJPjzXo8fvsxg9kQTA]
```

```
/opt/conda/lib/python3.8/site-packages/urllib3/connectionpool.py:981:
InsecureRequestWarning: Unverified HTTPS request is being made to host
'eumetsat.dpi.wekeo.eu'. Adding certificate verification is strongly advised.
See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-warnings
  warnings.warn(
2022-09-02 18:45:00,620 WARNING Oops, downloaded 1964808539 byte(s), was
```

```
supposed to be 1963982848 (extra 825691)
```

3.2.1 Find the file, subset and plot it

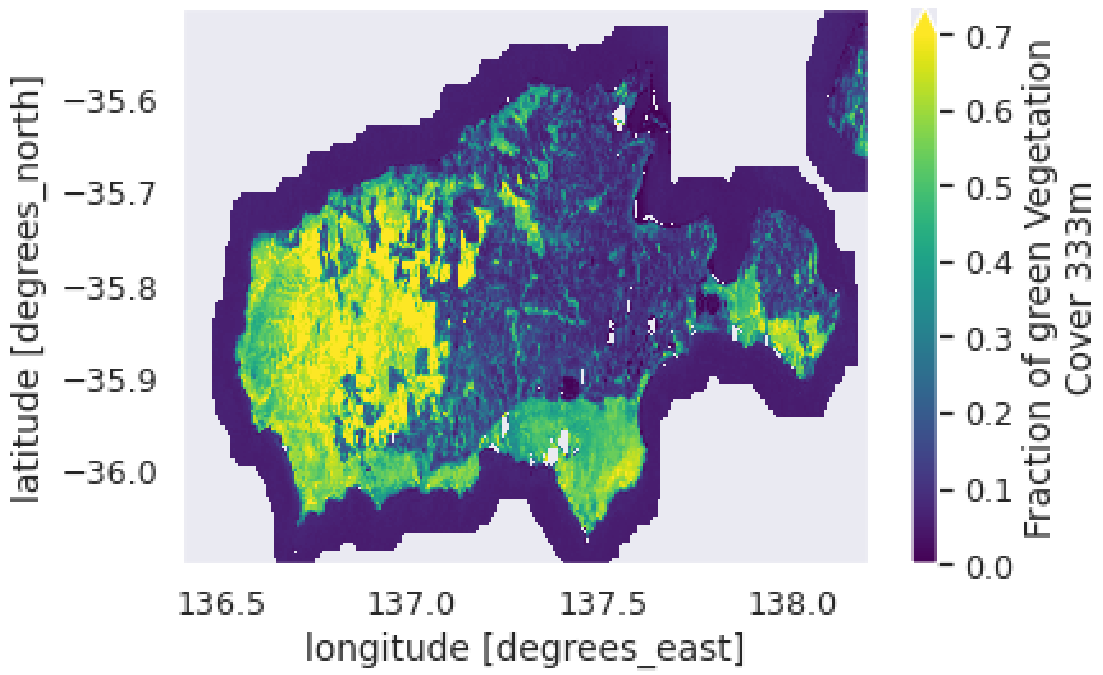
[44]:
```
fcover_file = glob.glob("*FCOVER300_20180310*.nc")[0]
ds_canopy = xr.open_dataset(fcover_file)
ds_canopy = ds_canopy.sortby(['lat', 'lon'])
```

[45]:
```
ds_canopy = ds_canopy["FCOVER"].sel(lon=slice(bbox[0], bbox[2]),␣
 ↪lat=slice(bbox[1], bbox[3]))
```

[46]:
```
ds_canopy.plot(vmax=0.7)
```

```
/home/jovyan/.local/lib/python3.8/site-packages/xarray/plot/plot.py:1451:
FutureWarning: Conversion of the second argument of issubdtype from `str` to
`str` is deprecated. In future, it will be treated as `np.str_ ==
np.dtype(str).type`.
  and not np.issubdtype(x.dtype, str)
/home/jovyan/.local/lib/python3.8/site-packages/xarray/plot/plot.py:1466:
FutureWarning: Conversion of the second argument of issubdtype from `str` to
`str` is deprecated. In future, it will be treated as `np.str_ ==
np.dtype(str).type`.
  and not np.issubdtype(y.dtype, str)
```

[46]:
```
<matplotlib.collections.QuadMesh at 0x7fa74ee8d9a0>
```



3.2.2 Download the rest of the data

So far we have downloaded data for one 10-day period. We need more data because we want to use all of the Forest Cover data from 2011 to 2019. To save time on this step, we have pre-prepared the data we need. We can download it in two csv files and load them directly into pandas.

The canopy.csv file is 98 MB and the meteo.csv file is 343 MB so it will take a few minutes to complete the downloads.

Use the dropbox link below to download

[47]:
```
canopy_dt = pd.read_csv('https://www.dropbox.com/s/d1kzizhlou79w2q/canopy.csv?
 ↪dl=1')
canopy_dt = canopy_dt[['time','lon','lat','FCOVER']]
```

That's the vegetation cover data, now get meteorological data for the island:

[48]:
```
meteo_dt = pd.read_csv('https://www.dropbox.com/s/7813uo23x1o1vjr/meteo.csv?
 ↪dl=1')
meteo_dt = meteo_dt[['time','longitude','latitude','t2m','tp','u10','v10']]
```

The meteo.csv data contains data from the ERA5 product downloaded from WEkEO. We downloaded hourly data from 2011-Oct-01 to 2019-Dec-31 for four variables: temperature (t2m), total precipitation (tp), and the eastwards (u10) and northwards (v10) components of the wind vector.

We converted the temperature from Kelvin to Celcius and saved it all to a csv file.

4 Part 2: Prepare and explore the data

View the first ten data records:

[49]:
```
meteo_dt.head()
```

[49]:
```
                  time   longitude   latitude        t2m        tp       u10  \
0  2011-10-01 00:00:00  136.399994 -35.599998  15.003021  0.000027  0.763832
1  2011-10-01 01:00:00  136.399994 -35.599998  15.315308  0.000036  0.258831
2  2011-10-01 02:00:00  136.399994 -35.599998  15.486237  0.000029 -0.586782
3  2011-10-01 03:00:00  136.399994 -35.599998  15.637054  0.000031 -0.878011
4  2011-10-01 04:00:00  136.399994 -35.599998  15.728516  0.000041 -0.999779

        v10
0  8.570405
1  8.050105
2  7.643668
3  7.249296
4  7.025718
```

This data needs some preparation before we analyse it.

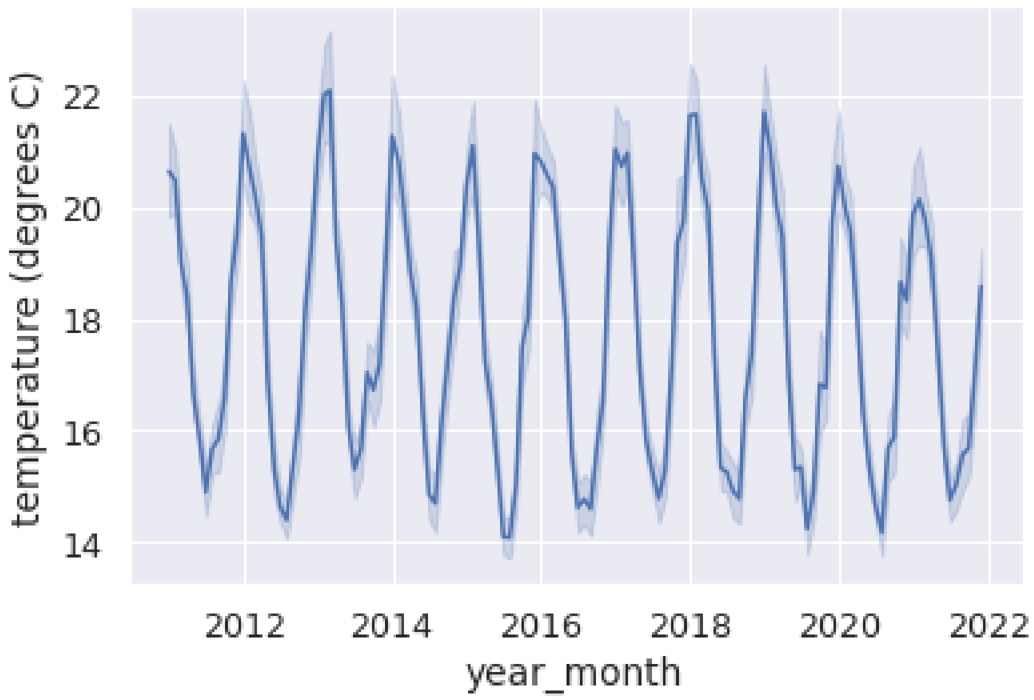
```
[50]: # find the mean for each hourly timestep (average over the 24 pixels with␣
       ↪different latitudes and longitudes)
      meteo_dt = meteo_dt.groupby(['time']).mean()
      meteo_dt.index = pd.to_datetime(meteo_dt.index)
```

```
[51]: # We need to decide how we will aggregate to daily scale from hourly,
      # for the temperature & wind we calculate the daily mean and for the␣
       ↪precipitation the sum
      t2m_mean = meteo_dt['t2m'].resample('1D').mean()  # 1D means one day
      tp_sum = meteo_dt['tp'].resample('1D').sum()
      u10_mean = meteo_dt['u10'].resample('1D').mean()
      v10_mean = meteo_dt['v10'].resample('1D').mean()
```

Merge the new meteo daily variables together

```
[52]: meteo_daily = pd.concat([t2m_mean, tp_sum, u10_mean, v10_mean], axis=1)
```

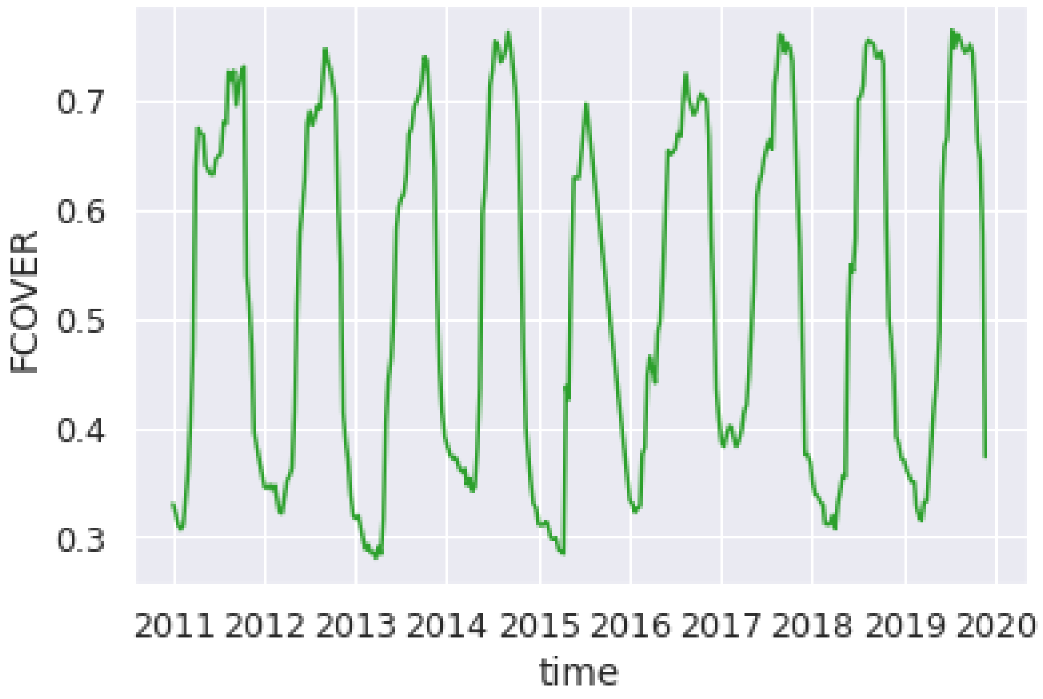
```
[53]: # the graph looks very full if we plot every datapoint, so we'll plot the mean␣
       ↪and bounds for each month
      meteo_to_plot = meteo_daily.copy()
      meteo_to_plot['year_month'] = meteo_to_plot.index.to_period('M').to_timestamp()
      sns.lineplot(x='year_month', y='t2m', data=meteo_to_plot)
      plt.ylabel('temperature (degrees C)');
```



[54]:
```
# find the mean of the canopy cover by time (so instead of having one value per␣
↪time per longitude per latitude we have one per time)
canopy_dt = canopy_dt.groupby(['time'])[['FCOVER']].mean().reset_index()
canopy_dt['time'] =  pd.to_datetime(canopy_dt['time'], format='%Y-%m-%d')
```

[55]:
```
sns.lineplot(x='time', y='FCOVER', data=canopy_dt, color='tab:green')
```

[55]:
```
<AxesSubplot:xlabel='time', ylabel='FCOVER'>
```

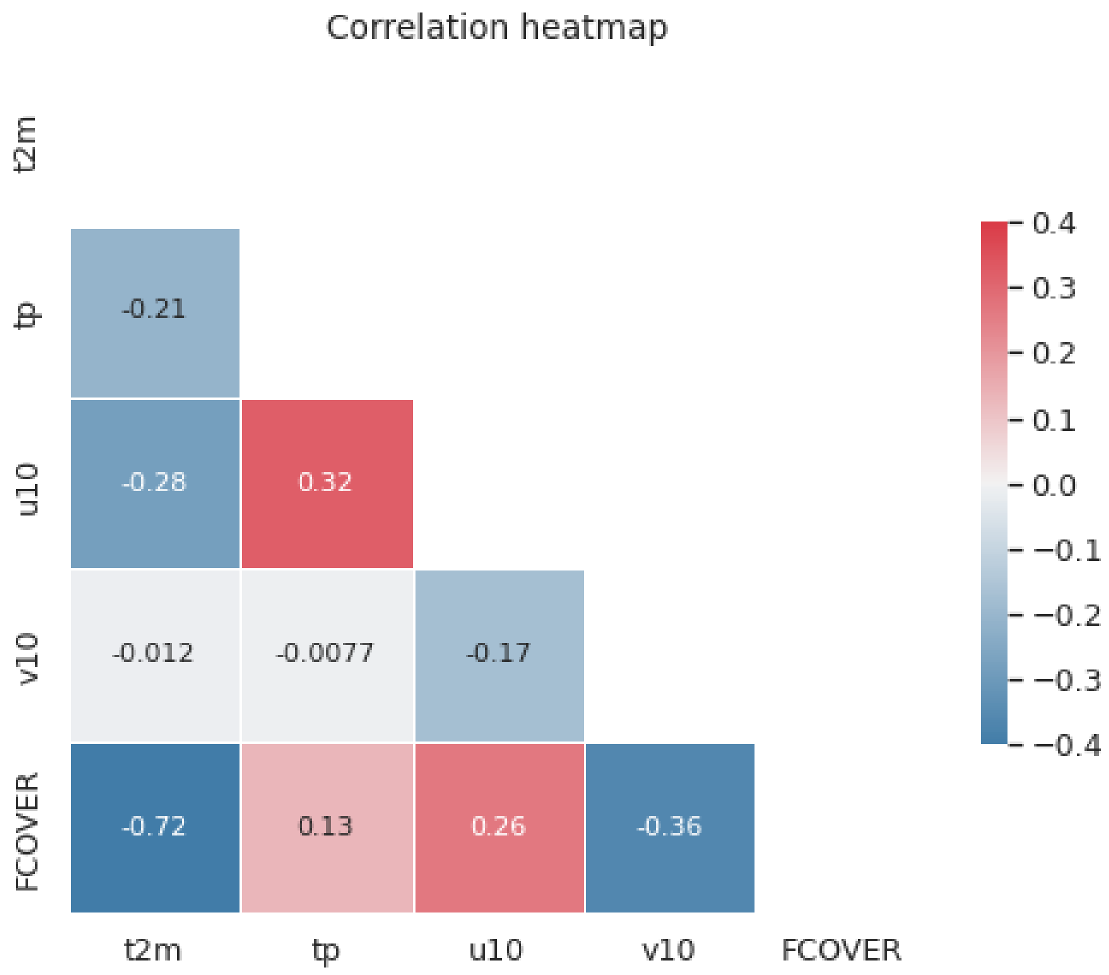


[56]:
```
# merge meteo dt and canopy_dt by time
dt_combined = pd.merge(meteo_daily, canopy_dt, on=['time'])
```

4.0.1 Start exploring correlations between the variables

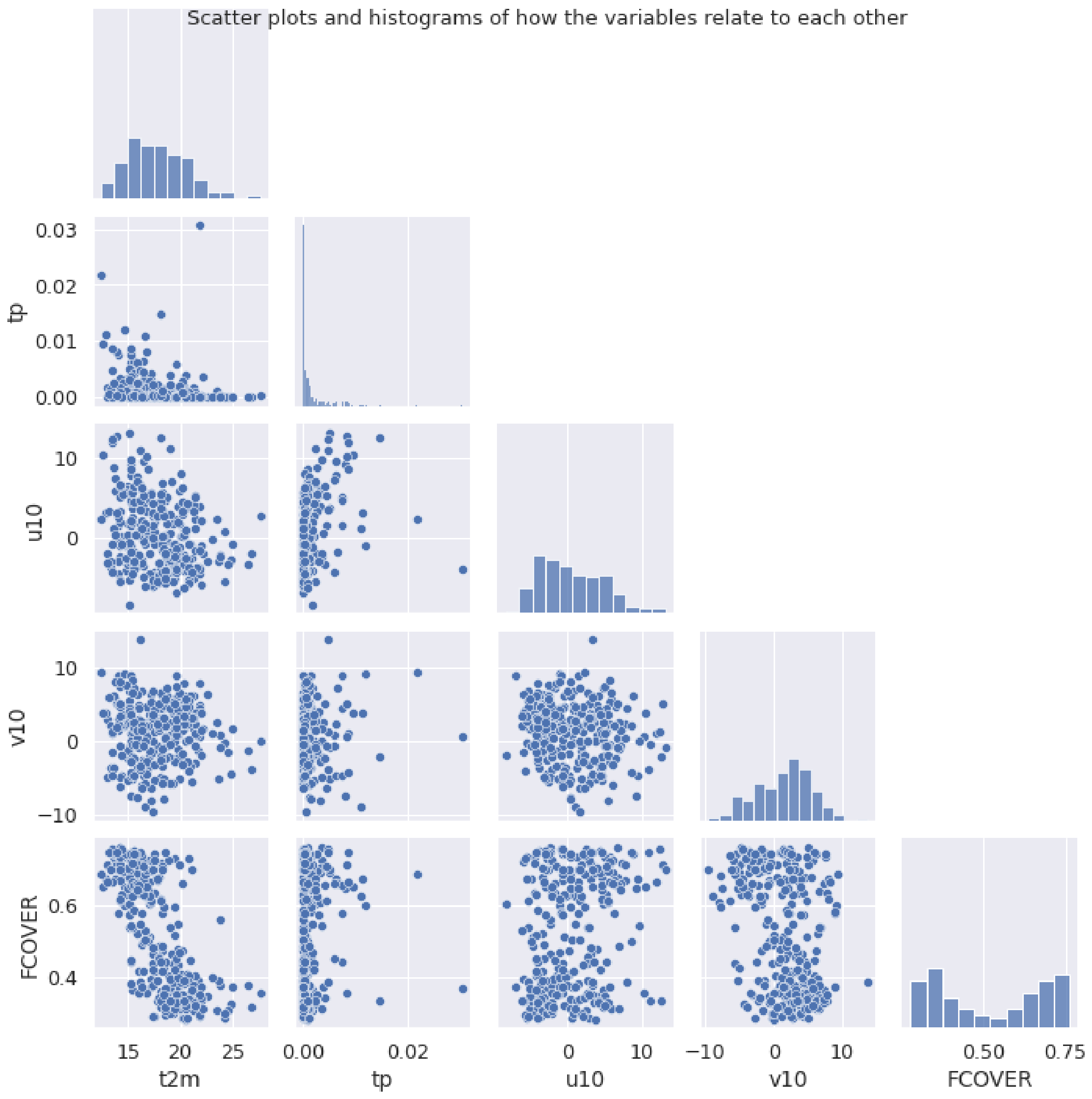
Can we see any relationships between the variables? Start by plotting a heatmap of their corellation.

[57]:
```
corr = dt_combined.corr()
mask = np.triu(np.ones_like(corr, dtype=bool))
cmap = sns.diverging_palette(240, 10, as_cmap=True)
plt.figure(figsize = (8, 8))
with sns.axes_style("white"):
    sns.heatmap(corr, mask=mask, cmap=cmap, vmax=0.4, vmin=-0.4,
                square=True, linewidths=.5, cbar_kws={"shrink": .5},
                annot=True, annot_kws={'size':11})
    plt.title('Correlation heatmap')
```



Now look at these relationships for the individual datapoints:

[58]:
```
# if this throws an error your environment has an old version of seaborn.
sns.pairplot(dt_combined, corner=True, height=2)
plt.suptitle('Scatter plots and histograms of how the variables relate to each␣
 ↪other', size=13);
```



5 Part 3: Build a random forest model

We have seen some correlations, in this part of the notebook we will move to more advanced methods. We will apply a random forest to see if we can predict the fraction of canopy cover in the area using only the 4 meteorological variables.

First we need to divide our dataset into train and validation sets. We split the data set based on time, at the moment we have data from 2011 to 2019/12. We use data from 2011 to 2016 to train our model and data from 2017 to 2019 to validate the performance of the model.

[59]:
```
train_data = dt_combined[dt_combined['time'] <= '2016-12-31']
valid_data = dt_combined[dt_combined['time'] > '2016-12-31']
```

[60]:
```
# first we need to create a Random forest  object
regressor = RandomForestRegressor()
```

[61]:
```
# then we fit the model using the training dataset
regressor.fit(train_data[['t2m','tp','u10','v10']], train_data['FCOVER']);
```

```
/home/jovyan/.local/lib/python3.8/site-packages/sklearn/ensemble/forest.py:244:
FutureWarning: The default value of n_estimators will change from 10 in version
0.20 to 100 in 0.22.
  warn("The default value of n_estimators will change from "
```

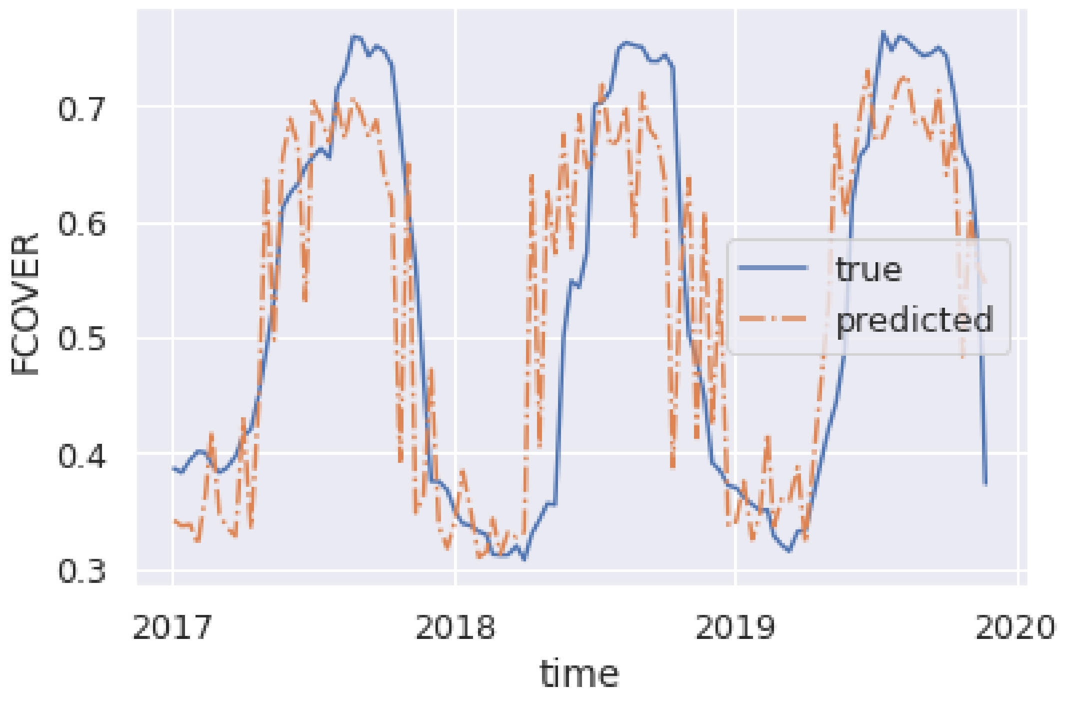
Then we predict canopy for 2017 onwards (the validation dataset)

[62]:
```
canopy_pred = regressor.predict(valid_data[['t2m','tp','u10','v10']])
```

Plot the predicted versus the actual FCOVER

[63]:
```
ax = sns.lineplot(x=valid_data['time'], y=valid_data['FCOVER'], label="true")
sns.lineplot(x=valid_data['time'], y=canopy_pred, label="predicted",␣
 ↪linestyle="-.", ax=ax)

myFmt = mdates.DateFormatter('%Y')
locator = mdates.YearLocator()
ax.xaxis.set_major_formatter(myFmt)
ax.xaxis.set_major_locator(locator)
```



[64]:
```python
# to validate how good the prediction of random forest is we calculate the RMSE
rmse = np.sqrt(np.sum((canopy_pred-valid_data['FCOVER'].values)**2)/
 ↪len(canopy_pred))
print("RMSE = ", rmse)
```

```
RMSE =  0.0998059379926272
```

5.0.1 Qustions?

Question: What factors stop the RMSE from being zero?
Question: Can you think of other applications of random forest models in Earth Observation?

5.0.2 That's the end! What other ways can we use this ML method?