Eddy detection

Author: Bence Dudás
Supervisor: Dávid Visontai

Table of contest

1. Abstract

In 1993 the California University and started to observe Eddies in the ocean. They had a theory, that the size of the Eddies has correlation with their longevity. This project goes on today, so we have a structured data even in 2018.

The project leader provided us the data through the university server k8plex. My goal was to analyze the given data, try to find the link between these 2 features, and do it so, that other will be able to reproduce the work.

2. Workflow

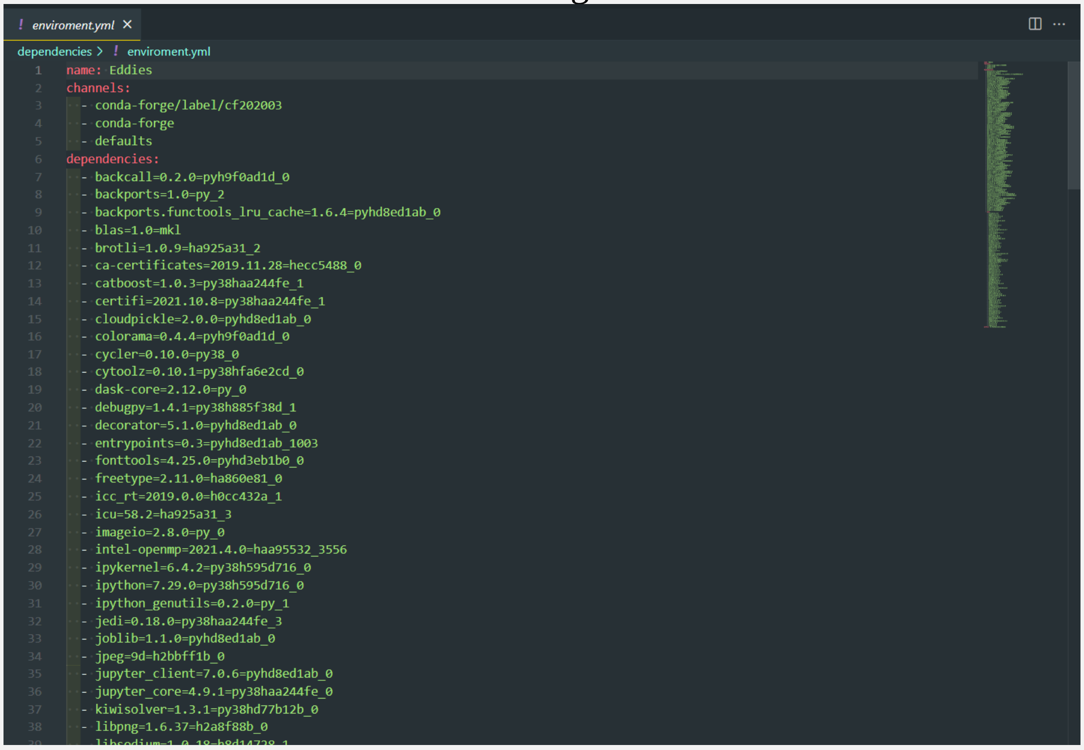
2.1 Virtual environment

Since reproducibility was one of the main requirements of the project, I've decided to use a virtual environment. The only programming language I used is Python. This language is not always backwards compatible, which means a new version of a library may have different structure, the location of some functions, variables might change.

This can cause warnings, errors, or just simply wrong results in the same algorithm run in different version of libraries. If one wants to avoid this kind of incompatibility, one can up- or downgrade specific libraries inside a notebook, or in the computer/server. Changing in the computer/server might cause more and more problems later. Specifying in the notebooks is a much safer way to create projects, but also more circumstantial.

Using many notebooks, files mean we must write down multiple times the same command. Virtual environments are used to avoid this unnecessary work. Creating a virtual environment means, all library version, path, etc. are saved. A project with a provided (and correctly created) virtual environment can be re-run always.

Anaconda provides us the opportunity to create virtual environments and store their values in files (in .yml format). The structure of the file is the following:



```yaml
name: Eddies
channels:
  - conda-forge/label/cf202003
  - conda-forge
  - defaults
dependencies:
  - backcall=0.2.0=pyh9f0ad1d_0
  - backports=1.0=py_2
  - backports.functools_lru_cache=1.6.4=pyhd8ed1ab_0
  - blas=1.0=mkl
  - brotli=1.0.9=ha925a31_2
  - ca-certificates=2019.11.28=hecc5488_0
  - catboost=1.0.3=py38haa244fe_1
  - certifi=2021.10.8=py38haa244fe_1
  - cloudpickle=2.0.0=pyhd8ed1ab_0
  - colorama=0.4.4=pyh9f0ad1d_0
  - cycler=0.10.0=py38_0
  - cytoolz=0.10.1=py38hfa6e2cd_0
  - dask-core=2.12.0=py_0
  - debugpy=1.4.1=py38h885f38d_1
  - decorator=5.1.0=pyhd8ed1ab_0
  - entrypoints=0.3=pyhd8ed1ab_1003
  - fonttools=4.25.0=pyhd3eb1b0_0
  - freetype=2.11.0=ha860e81_0
  - icc_rt=2019.0.0=h0cc432a_1
  - icu=58.2=ha925a31_3
  - imageio=2.8.0=py_0
  - intel-openmp=2021.4.0=haa95532_3556
  - ipykernel=6.4.2=py38h595d716_0
  - ipython=7.29.0=py38h595d716_0
  - ipython_genutils=0.2.0=py_1
  - jedi=0.18.0=py38haa244fe_3
  - joblib=1.1.0=pyhd8ed1ab_0
  - jpeg=9d=h2bbff1b_0
  - jupyter_client=7.0.6=pyhd8ed1ab_0
  - jupyter_core=4.9.1=py38haa244fe_0
  - kiwisolver=1.3.1=py38hd77b12b_0
  - libpng=1.6.37=h2a8f88b_0
  - libsodium=1.0.18=h8d14728_1
```

The whole file is uploaded into github.

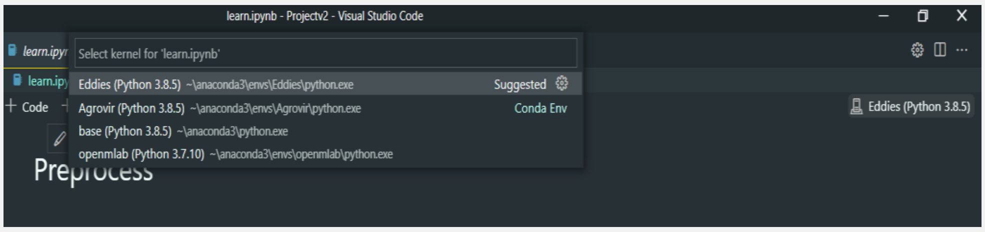
If you want to run the project without any problems, you only need to download the project. Navigate into dependencies folder with anaconda prompt and write the following:

```
conda env create -f environment.yml
```

Then it depends on your interpreter. If you use only the jupyter notebook, you have to write:

```
conda activate Eddies
```

After this, you will be able to use the environment. But using a modern IDE, like Visual Studio Code makes it easier, because you do not need to activate all the times, the environment. You can click on it, from a list.



learn.ipynb - Projectv2 - Visual Studio Code

Select kernel for 'learn.ipynb'

Eddies (Python 3.8.5) ~\anaconda3\envs\Eddies\python.exe Suggested

Agrovir (Python 3.8.5) ~\anaconda3\envs\Agrovir\python.exe Conda Env

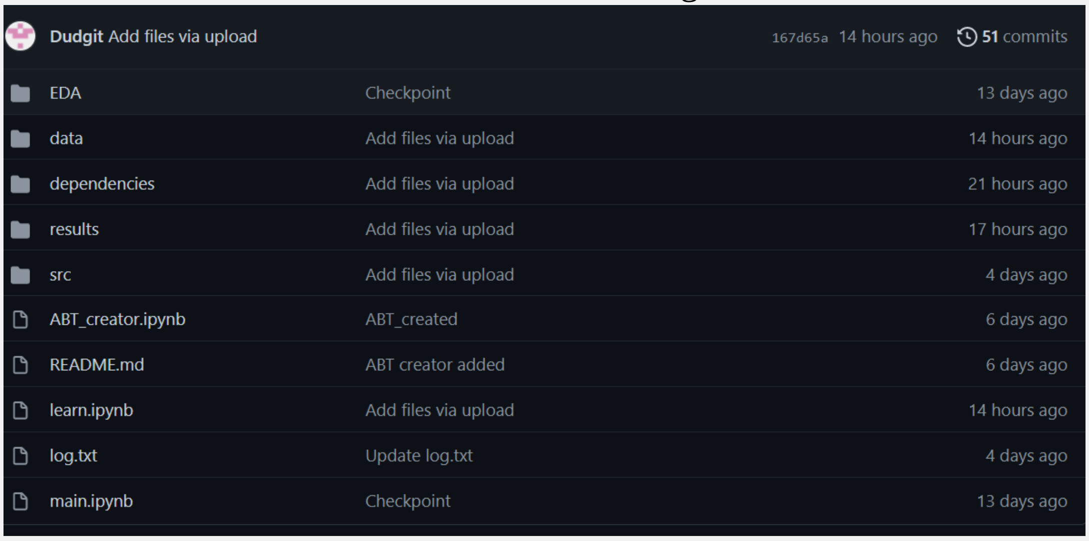
base (Python 3.8.5) ~\anaconda3\python.exe

openmlab (Python 3.7.10) ~\anaconda3\envs\openmlab\python.exe

2.2 Version control

The codes and files are stored in my local computer, for me it is the easiest and the most convenient way. But every programmer’s best friend is the version control. Since it is the most popular, I used github and its extension to follow, save and share the project.

The data structure is the following:



Dudgit Add files via upload 167d65a 14 hours ago **51** commits

EDA	Checkpoint	13 days ago
data	Add files via upload	14 hours ago
dependencies	Add files via upload	21 hours ago
results	Add files via upload	17 hours ago
src	Add files via upload	4 days ago
ABT_creator.ipynb	ABT_created	6 days ago
README.md	ABT creator added	6 days ago
learn.ipynb	Add files via upload	14 hours ago
log.txt	Update log.txt	4 days ago
main.ipynb	Checkpoint	13 days ago

2.3 Libraries

The first library I needed to install was Xarray, it is an alternative for NetCDF. This package were used to load the given dataset. The quality of data is discussed later.

Pandas is one of the most popular packages to work with data tables. I used it to process the data, group it and create some new features.

Matplotlib is a common package to create visualizations. I used it to visualize data distributions and to plot necessary figures.

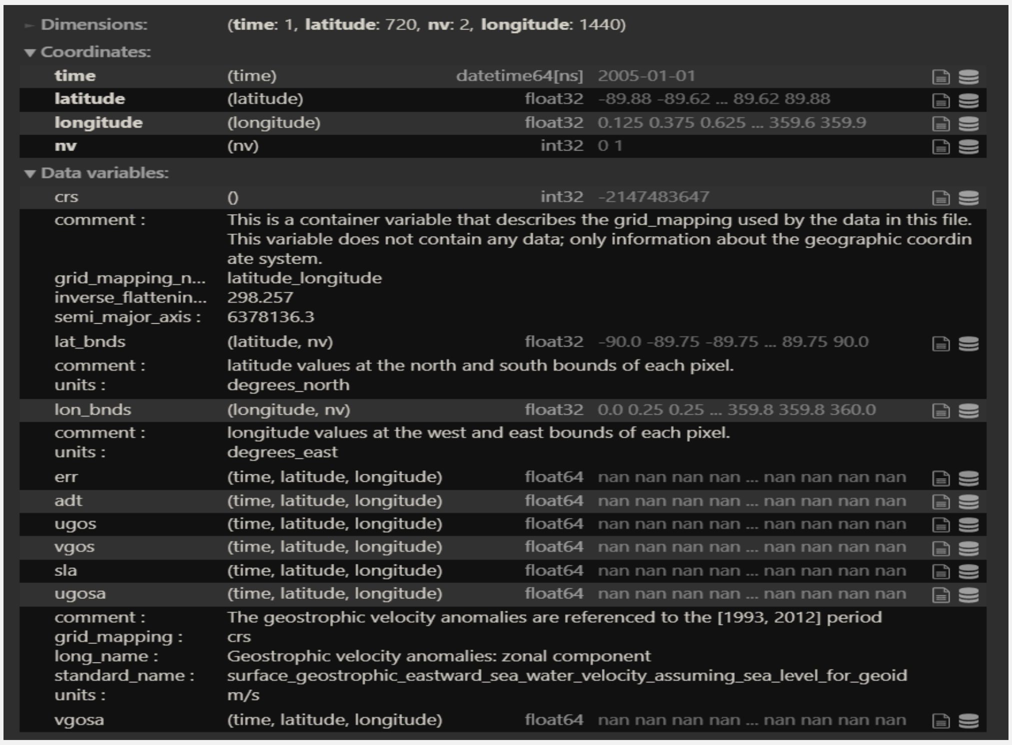
Catboost is an open-source gradient boosting library. Very easy and convenient to use. With built-in dashboard it can visualize it’s results and the process of learning. The two reason I chose it is because it’s performance and compatibility to Shap.

Shap is machine learning visualization tool. After creating the model, we will be able to understand it much better. Since the goal of this project was not to predict eddies longevity, but to understand some of their aspects. Shap is can visualize how the model predict and some relations between the features.

3. Project

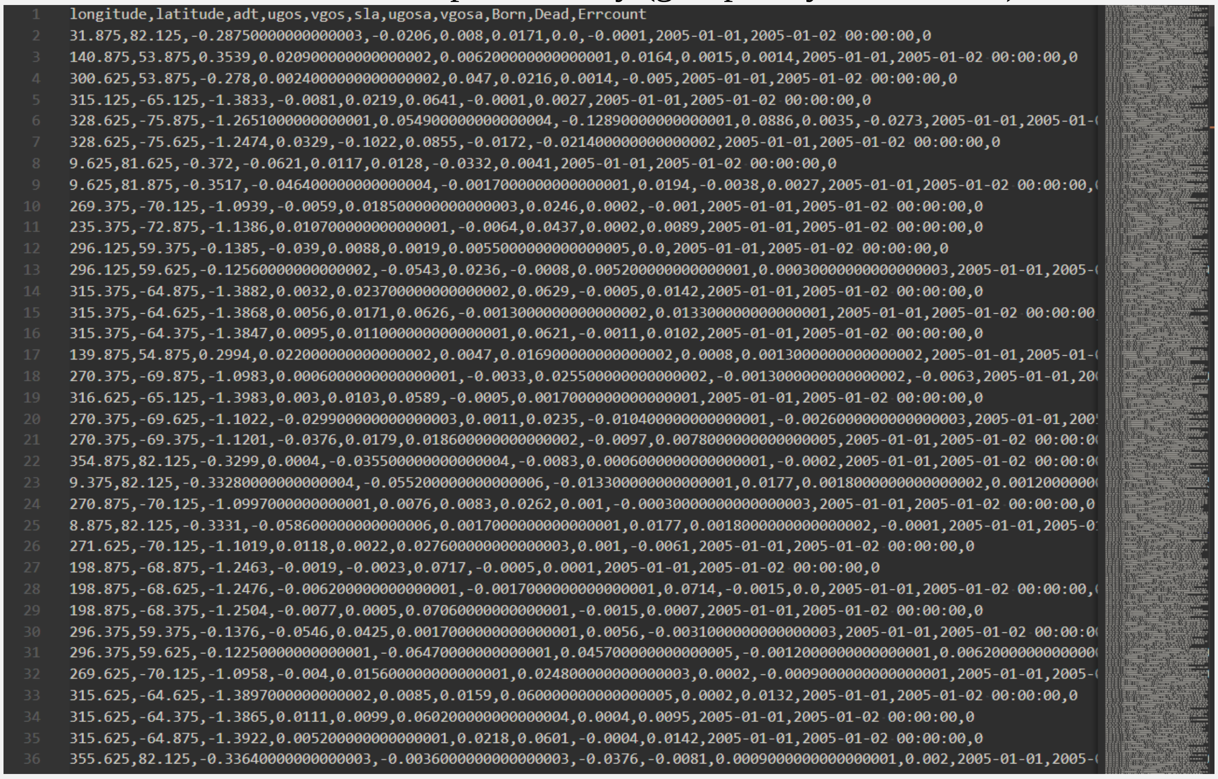
3.1 Data description

The Aviso observed eddies from 1993 to 2018. These were saved in .nc (Network Common Data Form) format, which is a multi-indexed file format.



```
← Dimensions:          (time: 1, latitude: 720, nv: 2, longitude: 1440)
▼ Coordinates:
    time               (time)                       datetime64[ns]  2005-01-01
    latitude           (latitude)                   float32  -89.88 -89.62 ... 89.62 89.88
    longitude          (longitude)                  float32  0.125 0.375 0.625 ... 359.6 359.9
    nv                 (nv)                         int32  0 1
▼ Data variables:
    crs                ()                           int32  -2147483647
    comment :          This is a container variable that describes the grid_mapping used by the data in this file.
                       This variable does not contain any data; only information about the geographic coordin
                       ate system.
    grid_mapping_n...  latitude_longitude
    inverse_flattenin...  298.257
    semi_major_axis :  6378136.3
    lat_bnds           (latitude, nv)               float32  -90.0 -89.75 -89.75 ... 89.75 90.0
    comment :          latitude values at the north and south bounds of each pixel.
    units :            degrees_north
    lon_bnds           (longitude, nv)              float32  0.0 0.25 0.25 ... 359.8 359.8 360.0
    comment :          longitude values at the west and east bounds of each pixel.
    units :            degrees_east
    err                (time, latitude, longitude)  float64  nan nan nan nan ... nan nan nan nan
    adt                (time, latitude, longitude)  float64  nan nan nan nan ... nan nan nan nan
    ugos               (time, latitude, longitude)  float64  nan nan nan nan ... nan nan nan nan
    vgos               (time, latitude, longitude)  float64  nan nan nan nan ... nan nan nan nan
    sla                (time, latitude, longitude)  float64  nan nan nan nan ... nan nan nan nan
    ugosa              (time, latitude, longitude)  float64  nan nan nan nan ... nan nan nan nan
    comment :          The geostrophic velocity anomalies are referenced to the [1993, 2012] period
    grid_mapping :     crs
    long_name :        Geostrophic velocity anomalies: zonal component
    standard_name :    surface_geostrophic_eastward_sea_water_velocity_assuming_sea_level_for_geoid
    units :            m/s
    vgosa              (time, latitude, longitude)  float64  nan nan nan nan ... nan nan nan nan
```

Every file contains the information of one specific year. That implies each folder contains 365 files (366 in every 4 year). This data was merged into 1 large table, which contains the first and last observation of a specific eddy (grouped by coordinates).



```
longitude,latitude,adt,ugos,vgos,sla,ugosa,vgosa,Born,Dead,Errcount
31.875,82.125,-0.2875000000000003,-0.0206,0.008,0.0171,0.0,-0.0001,2005-01-01,2005-01-02 00:00:00,0
140.875,53.875,0.3539,0.020900000000000002,0.006200000000000001,0.0164,0.0015,0.0014,2005-01-01,2005-01-02 00:00:00,0
300.625,53.875,-0.278,0.0024000000000000002,0.047,0.0216,0.0014,-0.005,2005-01-01,2005-01-02 00:00:00,0
315.125,-65.125,-1.3833,-0.0081,0.0219,0.0641,-0.0001,0.0027,2005-01-01,2005-01-02 00:00:00,0
328.625,-75.875,-1.2651000000000001,0.054900000000000004,-0.12890000000000001,0.0886,0.0035,-0.0273,2005-01-01,2005-01-
328.625,-75.625,-1.2474,0.0329,-0.1022,0.0855,-0.0172,-0.021400000000000002,2005-01-01,2005-01-02 00:00:00,0
9.625,81.625,-0.372,-0.0621,0.0117,0.0128,-0.0332,0.0041,2005-01-01,2005-01-02 00:00:00,0
9.625,81.875,-0.3517,-0.046400000000000004,-0.0017000000000000001,0.0194,-0.0038,0.0027,2005-01-01,2005-01-02 00:00:00,
269.375,-70.125,-1.0939,-0.0059,0.018500000000000003,0.0246,0.0002,-0.001,2005-01-01,2005-01-02 00:00:00,0
235.375,-72.875,-1.1386,0.010700000000000001,-0.0064,0.0437,0.0002,0.0089,2005-01-01,2005-01-02 00:00:00,0
296.125,59.375,-0.1385,-0.039,0.0088,0.0019,0.005500000000000005,0.0,2005-01-01,2005-01-02 00:00:00,0
296.125,59.625,-0.12560000000000002,-0.0543,0.0236,-0.0008,0.005200000000000001,0.00030000000000000003,2005-01-01,2005-
315.375,-64.875,-1.3882,0.0032,0.023700000000000002,0.0629,-0.0005,0.0142,2005-01-01,2005-01-02 00:00:00,0
315.375,-64.625,-1.3868,0.0056,0.0171,0.0626,-0.0013000000000000002,0.013300000000000001,2005-01-01,2005-01-02 00:00:00
315.375,-64.375,-1.3847,0.0095,0.011000000000000001,0.0621,-0.0011,0.0102,2005-01-01,2005-01-02 00:00:00,0
139.875,54.875,0.2994,0.022000000000000002,0.0047,0.016900000000000002,0.0008,0.0013000000000000002,2005-01-01,2005-01-
270.375,-69.875,-1.0983,0.0006000000000000001,-0.0033,0.025500000000000002,-0.0013000000000000002,-0.0063,2005-01-01,20
316.625,-65.125,-1.3983,0.003,0.0103,0.0589,-0.0005,0.0017000000000000001,2005-01-01,2005-01-02 00:00:00,0
270.375,-69.625,-1.1022,-0.029900000000000003,0.0011,0.0235,-0.010400000000000001,-0.0026000000000000003,2005-01-01,200
270.375,-69.375,-1.1201,-0.0376,0.0179,0.018600000000000002,-0.0097,0.007800000000000005,2005-01-01,2005-01-02 00:00:0
354.875,82.125,-0.3299,0.0004,-0.035500000000000004,-0.0083,0.0006000000000000001,-0.0002,2005-01-01,2005-01-02 00:00:0
9.375,82.125,-0.33280000000000004,-0.055200000000000006,-0.013300000000000001,0.0177,0.0018000000000000002,0.0012000000
270.875,-70.125,-1.0997000000000001,0.0076,0.0083,0.0262,0.001,-0.00030000000000000003,2005-01-01,2005-01-02 00:00:00,0
8.875,82.125,-0.3331,-0.058600000000000006,0.0017000000000000001,0.0177,0.0018000000000000002,-0.0001,2005-01-01,2005-0
271.625,-70.125,-1.1019,0.0118,0.0022,0.027600000000000003,0.001,-0.0061,2005-01-01,2005-01-02 00:00:00,0
198.875,-68.875,-1.2463,-0.0019,-0.0023,0.0717,-0.0005,0.0001,2005-01-01,2005-01-02 00:00:00,0
198.875,-68.625,-1.2476,-0.006200000000000001,-0.0017000000000000001,0.0714,-0.0015,0.0,2005-01-01,2005-01-02 00:00:00,
198.875,-68.375,-1.2504,-0.0077,0.0005,0.07060000000000001,-0.0015,0.0007,2005-01-01,2005-01-02 00:00:00,0
296.375,59.375,-0.1376,-0.0546,0.0425,0.0017000000000000001,0.0056,-0.0031000000000000003,2005-01-01,2005-01-02 00:00:0
296.375,59.625,-0.12250000000000001,-0.06470000000000001,0.045700000000000005,-0.0012000000000000001,0.006200000000000000
269.625,-70.125,-1.0958,-0.004,0.015600000000000001,0.024800000000000003,0.0002,-0.0009000000000000001,2005-01-01,2005-
315.625,-64.625,-1.3897000000000002,0.0085,0.0159,0.060000000000000005,0.0002,0.0132,2005-01-01,2005-01-02 00:00:00,0
315.625,-64.375,-1.3865,0.0111,0.0099,0.060200000000000004,0.0004,0.0095,2005-01-01,2005-01-02 00:00:00,0
315.625,-64.875,-1.3922,0.005200000000000001,0.0218,0.0601,-0.0004,0.0142,2005-01-01,2005-01-02 00:00:00,0
355.625,82.125,-0.33640000000000003,-0.0036000000000000003,-0.0376,-0.0081,0.0009000000000000001,0.002,2005-01-01,2005-
```

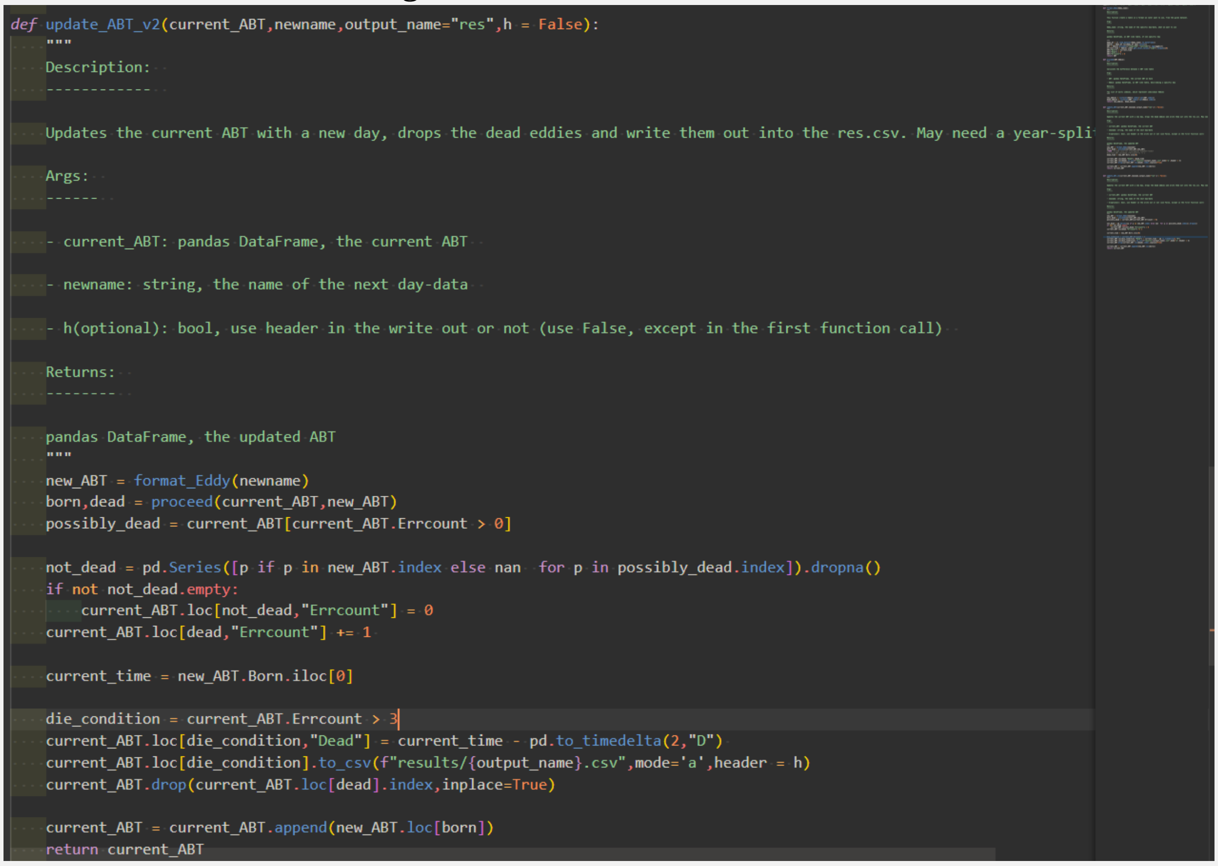
3.2 Steps

My first step was to initialize a github repository and a virtual environment. Since it is a project requirement to be reproducible, this was the best way to achieve it. A docker file could have been provided, but since these packages are independent from the operation system.

After this I started to get to know the data. For it is one of the most crucial part of a data science project. If we do not understand our data, we won't be able to say anything about it. For me it was quite easy, because as I've shown before, the .nc formatted files have a feature-wise description.

Step three was the creation of the ABT (Analytic Base Table). I grouped the data by their positions in the initial day. Compared it to the next days observations, if there were no value for a few days (it was a changeable threshold) I declared the eddy "dead", so I saved it into a .csv file with its first and last observation date. If there were new eddies, not contained in the

ABT, I added them into it. Every dead eddy was dropped from the ABT to not run out of memory. The function I wrote for this is the following:



```python
def update_ABT_v2(current_ABT,newname,output_name="res",h = False):
    """
    Description:
    ------------

    Updates the current ABT with a new day, drops the dead eddies and write them out into the res.csv. May need a year-spli

    Args:
    ------

    - current_ABT: pandas DataFrame, the current ABT

    - newname: string, the name of the next day-data

    - h(optional): bool, use header in the write out or not (use False, except in the first function call)

    Returns:
    --------

    pandas DataFrame, the updated ABT
    """
    new_ABT = format_Eddy(newname)
    born,dead = proceed(current_ABT,new_ABT)
    possibly_dead = current_ABT[current_ABT.Errcount > 0]

    not_dead = pd.Series([p if p in new_ABT.index else nan  for p in possibly_dead.index]).dropna()
    if not not_dead.empty:
        current_ABT.loc[not_dead,"Errcount"] = 0
    current_ABT.loc[dead,"Errcount"] += 1

    current_time = new_ABT.Born.iloc[0]

    die_condition = current_ABT.Errcount > 3
    current_ABT.loc[die_condition,"Dead"] = current_time - pd.to_timedelta(2,"D")
    current_ABT.loc[die_condition].to_csv(f"results/{output_name}.csv",mode='a',header = h)
    current_ABT.drop(current_ABT.loc[dead].index,inplace=True)

    current_ABT = current_ABT.append(new_ABT.loc[born])
    return current_ABT
```

The last step was to load these files and do some analysis on them. There are 3 phases of the experiment. From 1993 to 2005, from 2005 to 20012 and from 2012 to 2020 (we have data only to 2018). I decided to use the middle phase because its quality is the best. In the end I feed them into a catboost model, but results will be discussed later.

3.3 Experiences

The project itself was interesting. Working with large dataset can always have surprises. Due to this project, I met with the Net Common Datafile Format, which is handy tool. I also learned about Xarray.

I had 2 bigger problems during this project.

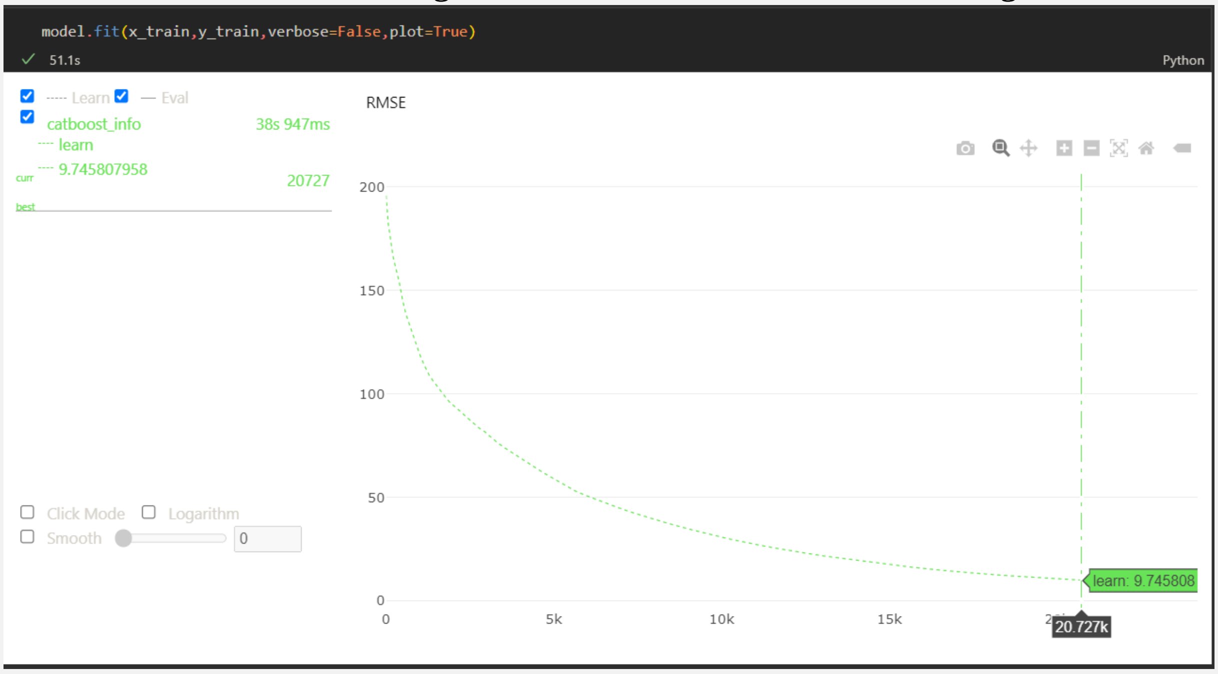
The first, since the database was huge, and I wanted to work locally. To explore the data, I've downloaded a smaller portion of it, from the aviso's website. Unfortunately, the data structured changed so the data provided was not the same as the current one. It took a few days back from the work, but it was a good lesson too.

The second problem was the clusters in the data. There were Eddies with verry small radius, or with small size. These may have not lasted a whole day, but they were detected. Feeding them to a model, or just do statistics on them gave me a false image. After I decided to use only those eddies which has a size greater than a certain amount, the results were much more promising.

3.4 Results

I've fed the data into a machine learning model. This one was a Catboost regressor. Fortunately, Shap package is made to make it easy to understand machine learning models, and how they make their decision. The following results were created during the project.

The learning curve of the model was the following:

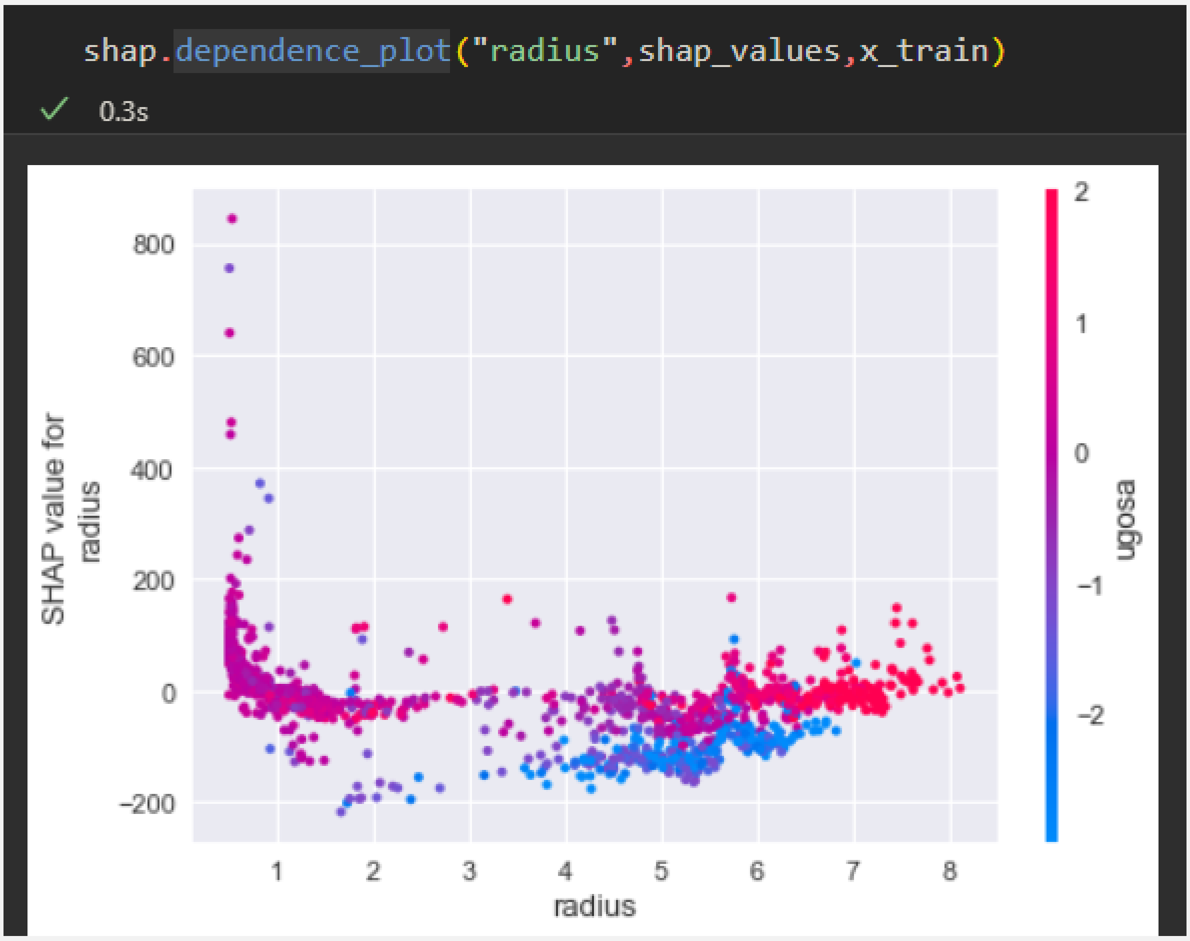


```
model.fit(x_train,y_train,verbose=False,plot=True)
✓ 51.1s                                                    Python
```

Our goal was not to predict all longevity of the eddies, but as it is seen in the figure, this kind of task is clearly not impossible. I've the following figures to make it clearer, that there is connection in between the radius and the longevity.

Dependence plot:

In this figure we can see that there are a some kind of dependence in between the longevity and the radius. Though it is not too strong connection. We can also see that during the decision radius has a connection with ugosa, which is the velocity component, so that make sense also.

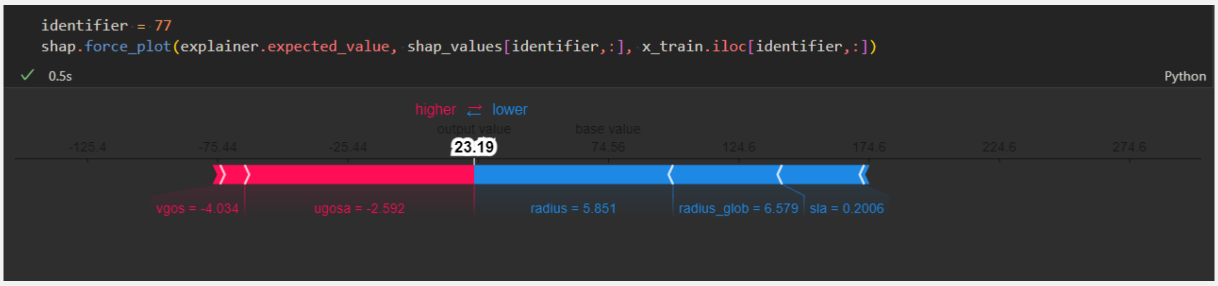


```
shap.dependence_plot("radius",shap_values,x_train)
```

```
✓ 0.3s
```

The force plot:

As we can se in that specific case, radius was one of the most important features.



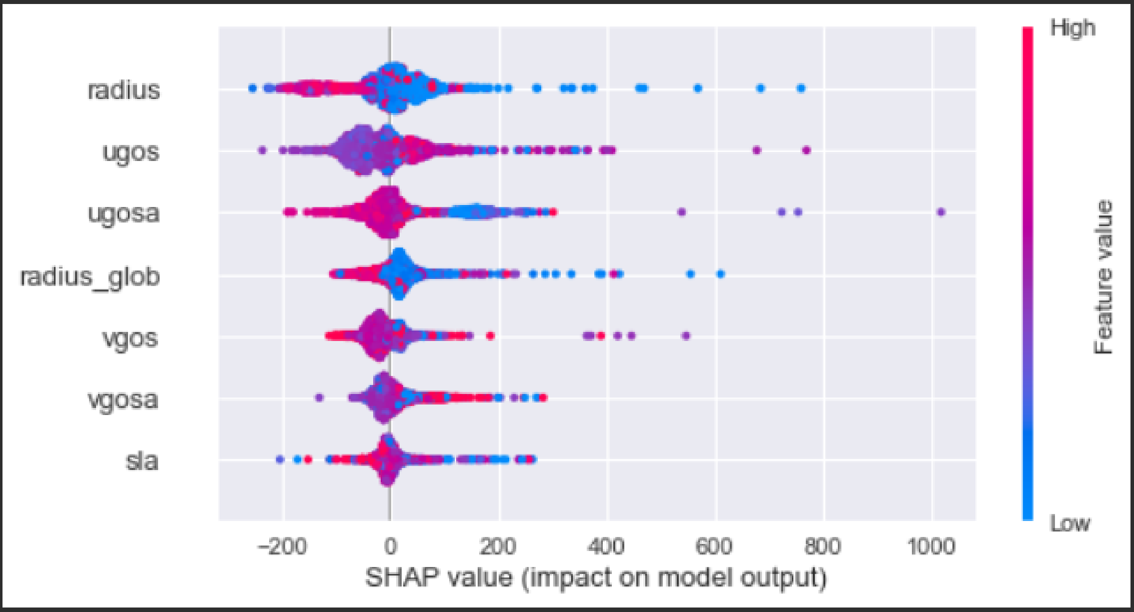
```
identifier = 77
shap.force_plot(explainer.expected_value, shap_values[identifier,:], x_train.iloc[identifier,:])
```

```
✓ 0.5s                                                                  Python
```

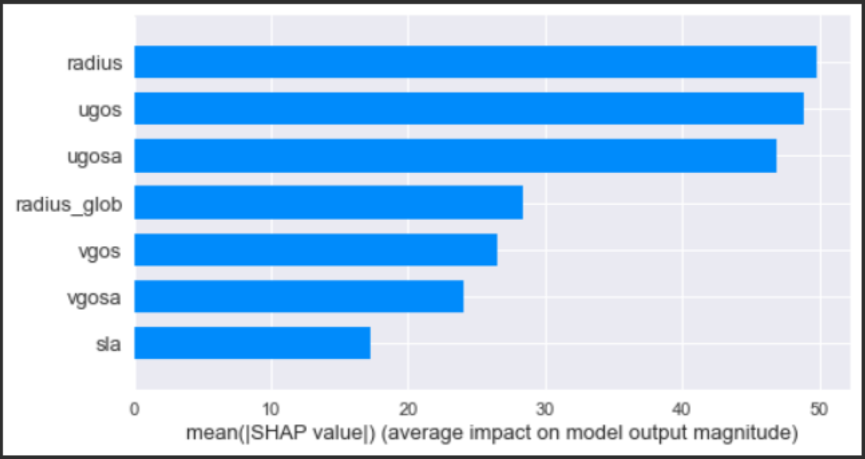
Summary plot:

Here we can see, that radius is the most important feature during the decision.

```
shap.summary_plot(shap_values,x_train)
```



```
shap.summary_plot(shap_values,x_train,plot_type="bar")
```



We can say that the project was successful, I've learned some useful skill during it.

4. References

- https://www.aviso.altimetry.fr/en/data/products/value-added-products/global-mesoscale-eddy-trajectory-product.html
- http://xarray.pydata.org/en/stable/
- https://en.wikipedia.org/wiki/NetCDF
- https://shap.readthedocs.io/en/latest/index.html
- https://catboost.ai/