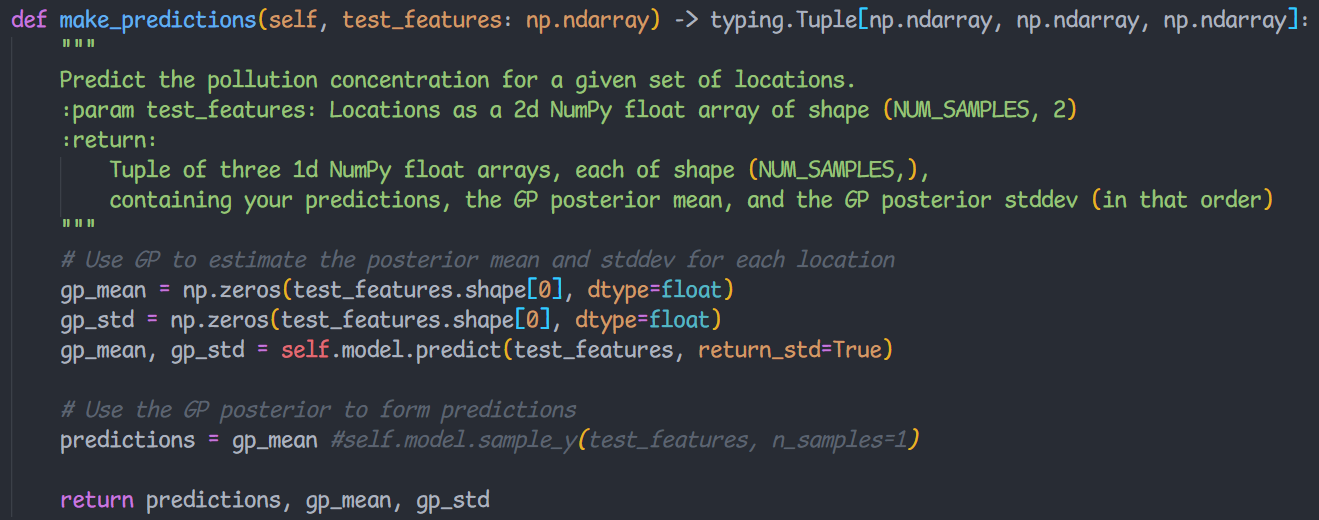
**Description of implementation of Task1-PAI**

dengzi ([dengzi@student.ethz.ch](mailto://dengzi@student.ethz.ch))

1. **Init model and select kernel**
2. **Randomly sample from training data and fit model using GPR**
3. **Test**



**I define and implement a Gaussian kernel, and then pass the train data and the test data to the Gaussian kernel. Using the K\_AA, k\_xA, and k\_xx calculate by the gaussian kernel, and add some noise, I get the mean and the variance of the GP model. I choose the model parameter by adjusting the sigma\_p and length\_scale of the kernel and the observation noise sigma\_n.**

**Ps: my computer is too slowly to process the GPR(Gaussian Process Regression), so I don’t use this method.**

To implement Gaussian Process Regression, I choose an RBF kernel with an output scale (by times a constant kernel), in order to train faster, I set the length scale of the RBF kernel as 0.05 and fixed. As for the GPR model, I add a noise 1e-8 and set the restart times of the optimizer as 1. The code is as below:

self.kernel = ConstantKernel(constant\_value=0.3, constant\_value\_bounds=(1e-3, 1)) \* RBF(length\_scale=0.05, length\_scale\_bounds='fixed')

self.model = GaussianProcessRegressor(kernel=self.kernel, alpha=1e-8, n\_restarts\_optimizer=1)

Then I set a random p = 0.42, and randomly sampled data using this probability p.

choice\_p = 0.42

train\_set = np.random.choice([True, False], len(train\_GT), p=[choice\_p, 1.0-choice\_p])

self.train\_X = train\_features[train\_set,:]

self.train\_Y = train\_GT[train\_set]

Finally, fitting the GPR model and processing prediction on the test data.

self.model.fit(self.train\_X, self.train\_Y)

gp\_mean, gp\_std = self.model.predict(test\_features, return\_std=True)