NOTES FOR EXERCISES FOR “SUPERVISED MACHINE LEARNING FOR POROUS MATERIALS”

SEYED MOHAMAD MOOSAVI[1] AND KEVIN M. JABLONKA[2]

[1] seyedmohamad.moosavi@epfl.ch

[2] kevin.jablonka@epfl.ch

MOLSIM 2020

In this exercise we explore how machine learning can be used to expedite the discovery of porous materials for gas storage—which is a field of very active research.

These notes guide through the full machine learning pipeline from exploring the data, over optimizing a model to actually interpreting the model.

Note that the Jupyter Notebook contains all exercises. These document only contains supplementary notes.

More background information can be found at `https://figshare.com/s/b5bd6a3cddd0dfd1f199` and the references therein.

Contents

Use the margins of this document to take notes!

1 Setting the Stage

1.1 Porous Materials for Gas Storage and Separation

If we want to mitigate the effect of rising global greenhouse gas emissions[3] we can either switch the source of our energy or develop techniques that are able to capture the greenhouse gases from air or flue gases. [4]

For both processes, porous materials such as metal-organic frameworks (MOFs) or zeolites are promising candidates.

One separation that received particular interest is the capture of CO_2 from wet flue gases, where the material needs to be selective for CO_2 over N_2 and H_2O. In practice, one would hope to use this material in a process like pressure- or vacuum swing adsorption, where the gas is adsorbed at high pressure and desorbed at lower pressure. Obviously, the ideal materials should have a large uptake at adsorption conditions and a lower uptake at desorption conditions. Typically, one uses the difference between the uptake at the two pressure points,[5] the working capacity, as performance indicator.

Another way to mitigate greenhouse gas emission is to switch to fuels that emit less CO_2 per unit of energy, like CH_4. The challenge with CH_4 is its low volumetric energy density. One potential way to remedy this is to adsorb the gas in porous materials to increase the energy density of the fuel. Also here, one key performance indicator is the deliverable capacity (difference between uptake at adsorption and desorption condition).

1.2 Accelerating Materials Genomics with Machine Learning

As aforementioned, MOFs are materials that receive a lot of attention with respect to these separations. MOFs are build out of metal nodes that are connected by organic linkers (cf. Figure 1) on some net topology. Due to this building block principle (reticular chemistry[6]) there is a nearly infinite amount of possible combinations of linkers and nodes—which makes experimental investigation for all materials for any given application completely infeasible.

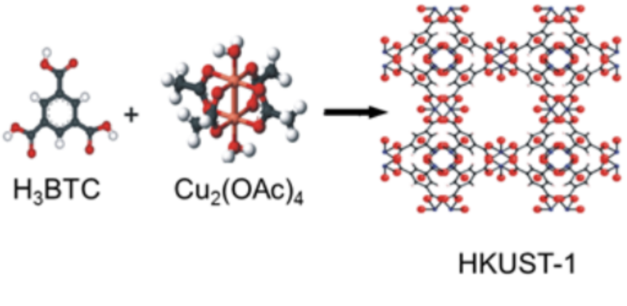


Figure 1: Building principle of MOFs.

But also computational screenings[7] can become unfeasible or too expensive at this scale, for which reason there is significant effort in replacing or accelerating such computational screenings using

[3] L. Jonkers, H. Hillebrand, and M. Kucera. Global change drives modern plankton communities away from the pre-industrial state. *Nature*, page 1, 2019

[4] ; M. Bui, C. S. Adjiman, A. Bardow, E. J. Anthony, A. Boston, S. Brown, P. S. Fennell, S. Fuss, A. Galindo, L. A. Hackett, J. P. Hallett, H. J. Herzog, G. Jackson, J. Kemper, S. Krevor, G. C. Maitland, M. Matuszewski, I. S. Metcalfe, C. Petit, G. Puxty, J. Reimer, D. M. Reiner, E. S. Rubin, S. A. Scott, N. Shah, B. Smit, J. P. M. Trusler, P. Webley, J. Wilcox, and N. Mac Dowell. Carbon capture and storage (CCS): the way forward. *Energy & Environmental Science*, 11(5):1062–1176, 2018; and K. S. Lackner, S. Brennan, J. M. Matter, A.-H. A. Park, A. Wright, and B. v. d. Zwaan. The urgency of the development of CO2 capture from ambient air. *Proceedings of the National Academy of Sciences*, 109(33):13156–13162, 2012

[5] sometimes, also the temperature is varied

[6] O. M. Yaghi. Reticular Chemistry in All Dimensions. *ACS Central Science*, 5(8):1295–1300, 2019

[7] A. Sturluson, M. T. Huynh, A. R. Kaija, C. Laird, S. Yoon, F. Hou, Z. Feng, C. E. Wilmer, Y. J. Colón, Y. G. Chung, D. W. Siderius, and C. M. Simon. The role of molecular modelling and simulation in the discovery and deployment of metal-organic frameworks for gas storage and separation. *Molecular Simulation*, 45(14-15):1082–1121, 2019; and P. G. Boyd, Y. Lee, and B. Smit. Computational development of the nanoporous materials genome. *Nature Reviews Materials*, 2(8):17037, 2017

machine learning techniques.

In this exercise, we will guide you trough some of the basic techniques that are needed to develop a model that predicts gas adsorption properties based on data from GCMC simulations.

2 *Setting Up Your Environment*

You can find all course material on Github[8]. You can clone the repository with

```
git clone https://github.com/kjappelbaum/ml\_molsim2020
```

to have all material on your machine.

2.1 *Using Your Own Machine or the Remote Cluster*

We recommend that you set up a virtual environment for this exercise using `conda`.

1. If you did not already do so, download the Anaconda distribution[9]. Be careful to install the Python 3.7 version.
2. Verify that the installation was successful by opening a new shell and verifying that

   ```
   conda list
   ```

   returns no error
3. Use the `environment.yml` file in the repository to create a new environment

   ```
   conda create --name molsim_ml --file=environment.yml
   ```

4. Activate the environment with

   ```
   conda activate molsim_ml
   ```

5. Enter the folder with the Jupyter Notebook and open a notebook by typing

   ```
   Jupyter notebook
   ```

2.2 *Using Google Colab*

If you do not want to use your own machine, you can also use Google Colab, which provides you with free computational resources.[10]

To do so, open the shared notebook at *and make a copy into your Google drive and work on this copy* (File › Save a copy in drive ...).

If you use the notebook on Colab you will have to uncomment and run the first code cell.

2.3 *Using Binder*

Binder does not require to set up anything. Simply visit `https://bit.ly/2N6ixu5`. Unfortunately, the computing resources there are much more limited and it can take some time to start up an image.

We will use this as a last resort for this course.

[8] `https://github.com/kjappelbaum/ml_molsim2020`

[9] `https://www.anaconda.com/distribution/`. If you want to save disk space you can also go for miniconda `https://docs.conda.io/en/latest/miniconda.html`

[10] We recommend you to play with this service if you can profit from TPU/GPU acceleration.

3 *Python Basics*

This exercise requires minimal programming in Python for which reason we review some basics in the following.[11]

3.1 *Iterables*

Lists, tuples, dictionaries, and sets are all iterable objects which you can loop over. Lists are denoted by square brackets (`['apple', 'banana', 123, 1355.354, 'tree']`), tuples by parenthesis (`('apple', 'banana', 123, 1355.354, 'tree')`) and dictionaries are a set of key-value pairs (e.g., `dict = {'manufacturer': 'Porsche', 'year': 2018}`).

One elegant way of looping over iterables is list comprehension. This is, we can rewrite a for loop such as

```python
column_names_old = list(df.columns)
column_names_new = []

for name in column_names_new:
    column_names_new.append(name.lower().replace(' ', '_'))
```

in one line of code

```python
column_names_new = [name.lower().replace(' ', '_') for name in column_names_old]}.
```

3.2 *Dealing With Data*

Good introductory material for (exploratory) data analysis can be found in the BE/BI 103 course at Caltech.[12]

The most popular library to deal with data in Python is `pandas`[13]. It implements `dataframe` objects, which are tabular data structures that allow for efficient operations also on larger datasets. A typical workflow to load a dataset is shown in Listing 1.

Listing 1: Loading data with pandas

```python
import pandas as pd # import the pandas library

df = pd.read_csv('data.csv') # load the data, pandas can read a variety of formats

df.head() # peak at the first rows

df.info() # get some basic info

df.describe() # get some basic statistics

df.columns # get the columns
```

To select specific elements, you can use the `loc` and `iloc` methods.

3.3 *Matplotlib*

The most widely used plotting library in Python is `matplotlib`. Listings 2 and 3 show examples of how to create a basic scatterplot and a 3D plot, respectively.

[11] you can find a cheat sheet for example at `https://perso.limsi.fr/pointal/_media/python:cours:mementopython3-english.pdf`

[12] `http://bebi103.caltech.edu.s3-website-us-east-1.amazonaws.com/2018/index.html`

[13]

Listing 2: A basic scatterplot.

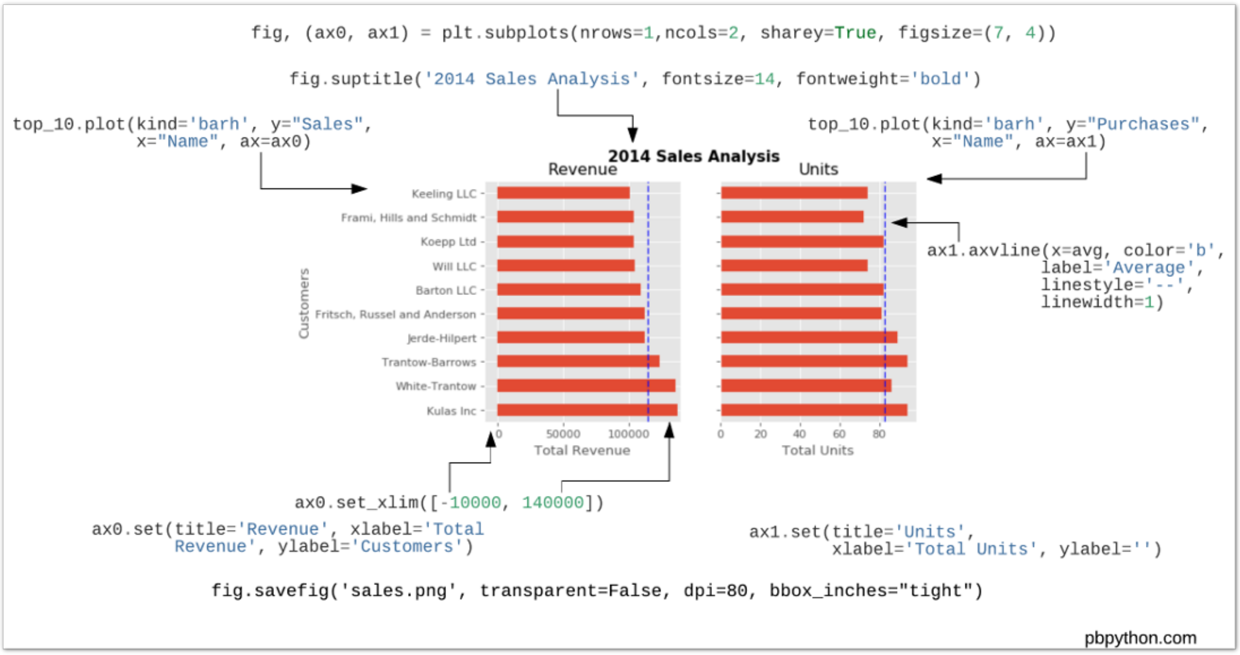
```
plt.figure(figsize=(8,6))
plt.scatter(X[:, 0], X[:, 1], c=color, cmap=plt.cm.rainbow)

plt.title('Plot tilte')
plt.xlabel('X axis label')
plt.ylabel('Y axis label')
plt.show()
```

Listing 3: Plotting in 3D.

```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(7,7))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=color, cmap=plt.cm.rainbow)
```

Matplotlib allows for nearly unlimited customization of plots. An example is shown in Figure 2. In case one wants to customize the plots, it is usually necessary to use the object oriented programming (oop) version with the figure and axis objects, as shown in Figure 2.



```
fig, (ax0, ax1) = plt.subplots(nrows=1,ncols=2, sharey=True, figsize=(7, 4))

fig.suptitle('2014 Sales Analysis', fontsize=14, fontweight='bold')

top_10.plot(kind='barh', y="Sales",
            x="Name", ax=ax0)

top_10.plot(kind='barh', y="Purchases",
            x="Name", ax=ax1)

ax1.axvline(x=avg, color='b',
            label='Average',
            linestyle='--',
            linewidth=1)

ax0.set_xlim([-10000, 140000])

ax0.set(title='Revenue', xlabel='Total
        Revenue', ylabel='Customers')

ax1.set(title='Units',
        xlabel='Total Units', ylabel='')

fig.savefig('sales.png', transparent=False, dpi=80, bbox_inches="tight")
```

Figure 2: Example of customization of plots using `matplotlib`. We took this figure from the practical business python blog (`https://pbpython.com/effective-matplotlib.html`) that gives more details and tips.

3.4 Holoviews

[14]

Often, it is handy to have interactive plots to explore the data. One library to do this is holoviews, which builds a high-level interface on top of plotly[15] and Bokeh.[16] We will use holoviews as the default plotting tool for this exercise.

One of the design principles behind holoviews is that you do not plot the data itself but that you annotate it—and the library figures out how to make the plot. This declarative approach might be familiar to you from “grammar of graphics” packages (e.g., Altair [17] or `ggplot`)[18].

This means that if you want to create a scatter plot you create a `hv.Scatter` element, to which you give the dataframe as a first argument and then declare the key and the value variables, as done in Listing 4.

Listing 4: Plotting with holoviews

```python
hv.Scatter(df, kdims=['density'], vdims=['uptake'])
```

To style the plots, you can create a dictionary with display options, see Listing 5.

Listing 5: Styling holoviews plots

```python
plot_opts = {'show_grid':  True,
             'width': 450,
             'height': 350}
style_opts = {'color': '#1f77b4',
              'size': 5}
scatter.opts(style=style_opts, plot=plot_opts)
```

In Jupyter notebooks you can also use magics, see Listing 6.

Listing 6: Styling holoviews with cell magics.

```python
%%opts Scatter [show_grid=True, width=450, height=350] (color='#1f77b4',
     size=5)
```

One power of holoviews is that it allows to create complex figures, e.g. with dropdown selection menus, with only one function call.[19] For example, assume that we want to group the previous scatter plot by the topology of the net, we can do this using the code from Listing 7.

Listing 7: Groupby with holoviws.

```python
# Make groupby object
gb = scatter.groupby('topology')
gb
```

You can add multiple plots as subplots next to each other (and coupled if they use the same data) using the + operator. We can overlay plots using the ∗ operator.

[14] We follow the notes from the BE/Bi 103 course at Caltech. `http://bebi103.caltech.edu.s3-website-us-east-1.amazonaws.com/2017/tutorials/t2b_exploratory_data_analysis.html`. We recommend that you use this tutorial if you want to learn more about this library. Of similar excellence is this tutorial from one holoviews developer `https://pyvideo.org/scipy-2017/interactive-data-visualization-with-holoviews-bo html`.

[15] `https://plot.ly`

[16] `https://bokeh.org`

[17] `https://altair-viz.github.io`

[18] `https://ggplot2.tidyverse.org`

[19] But we still have the full customization options of the underlying libraries if we need it.

3.5 *Sklearn Basics*

`Sklearn`[20] is the most widely used python library for machine learning, with an incredibly well-crafted API.[21]

A typical, basic, workflow in `sklearn` is shown in Listing 8.[22]

Listing 8: A typical `sklearn` workflow.

```python
# Import the packages
from sklearn import neighbors, datasets, preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# load the data
iris = datasets.load_iris()

# split the data
X, y = iris.data[:, :2], iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=33)

# scale the data
scaler = preprocessing.StandardScaler()
 # fit on training set and directly scale the data
X_train = scaler.fit_transform(X_train)
# scale the test set using mean and standard deviation from training set
X_test = scaler.transform(X_test)

# train a nearest neighbors classification model
knn = neighbors.KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# perform a prediction
y_pred = knn.predict(X_test)

# measure the accuracy
accuracy_score(y_test, y_pred)
```

A useful object to group subsequent operations, that for example can be cross-validated together, are `Pipelines`. An example of how to use them is shown in Listing 9.

Listing 9: Example use of `sklearn` pipelines.

```python
# This pipeline scales the data, then reduces the dimensionality of the data using PCA and then fits a knn classifier
pipeline = Pipeline([
    (’scaler’, StandardScaler()),
    (’pca’, PCA(n_components=10)),
    (’clf’, KNeighborsClassifier()),
])
pipeline.fit(X_train)

predicted = pipeline.predict(X_test)
```

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011

[21] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. Vanderplas, A. Joly, B. Holt, and G. Varoquaux. API design for machine learning software: experiences from the scikit-learn project. *arXiv:1309.0238 [cs]*, 2013

[22] a more detailed cheat sheet can be found at `https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Scikit_Learn_Cheat_Sheet_Python.pdf`

3.6 Jupyter Notebooks

For this exercise, we will use Jupyter notebooks.[23] It is important to realize that each notebook has its kernel which needs to use the correct conda environment (which is usually shown in the upper right corner, you should read Python 3 there).

In the notebook, you have cells which you can evaluate by either clicking the run symbol or by hitting ⇧ + Enter. You can run the cells in any order you want—and the result of each evaluation can depend on which cell you previously evaluated.[24]

You should be aware that the cells have two modes, between which you can toggle using Esc, in Jupyter.

If you have some issues with some code that does not stop you can either interrupt the kernel or restart it. Note that in latter case you have to evaluate all cells again.

Useful Shortcuts/Commands

- Esc switch between edit and command mode
- b in command mode to create a cell below the current cell
- a in command mode to create a cell above the current cell
- you can also use the insert menu to insert new cells
- ⇥ (Tab) to get command completion
- ⇧ + ⇥ (Shift + Tab) in a function to get the docstring

[23] `https://Jupyter.org` A basic tutorial is available at `https://Jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Notebook%20Basics.html`

[24] Variables in Jupyter notebooks live globally "between" cells.

3.7 *Kaggle Competition*

For this exercise we host a Kaggle competition. Kaggle is a community of (aspiring) data scientists that regularly host competitions on various fields. We encourage you to explore Kaggle if you want to strengthen your data science skills![25]

For our "classroom competition" we put some data aside. You can download the features and feed them into your model to make predictions and then upload these predictions where these predictions will be evaluated based on the mean absolute error.

To join the competition you need to

1. Create an account on `www.kaggle.com`
2. Visit the site of the competition `http://www.kaggle.com/c/molsim2020)`
3. Download the file with the features (`features.csv`) (you also find the files in the `data` directory)
4. Feed the features into your model
5. Create `.csv` with your submission containing a id column and the prediction
6. Upload this submission file on the competition site[26]
7. You will then see your score in the leaderboard

You can also form teams on Kaggle (click on "My Team" in the dashboard).

Please share your code with a Kaggle notebook!

[25] T. Simonite. Solve These Tough Data Problems and Watch Job Offers Roll In. *Wired*, Oct. 2017

[26] `https://www.kaggle.com/c/molsim2020/submit`

4 *Introduction to Kernel Ridge Regression*

[27]

4.1 *Linear Regression*

In linear regression we try to find the weights $\mathbf{w}$ of a line such that $\mathbf{w}^T\mathbf{x}_i \approx y_i$ for each data point (x_i, y_i). To find this line, one often minimizes the mean square error

$$\min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2. \tag{1}$$

This can be solved[28] analytically (normal equation):

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{2}$$

4.2 *Ridge Regression (Primal)*

To avoid overfitting, one can add a regularization term that penalizes large weights

$$\min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - y\|^2 + \lambda \, \|\mathbf{w}\|^2 \tag{3}$$

and we find that in the analytical solution we add the regularization factor λ to each diagonal element:

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda \, \mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}. \tag{4}$$

Using the fact that

$$(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\mathbf{X}^T = \mathbf{X}^T\mathbf{X}\mathbf{X}^T + \lambda\mathbf{X}^T = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I}) \tag{5}$$

$$(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\mathbf{X}^T(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1} \tag{6}$$

we can also write eq. 4 as

$$\mathbf{w} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\mathbf{y}. \tag{7}$$

When we make a prediction for $\mathbf{X}^*$ we have to evaluate $y = \mathbf{X}^{*T}\mathbf{w}$ (for simplicity, we set $\lambda = 0$):

$$\hat{\mathbf{y}} = \mathbf{X}^{*T}\mathbf{w} = \mathbf{X}^{*T}\mathbf{X}^T \underbrace{(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\mathbf{y}}_{:=\boldsymbol{\alpha}} \tag{8}$$

$$\hat{\mathbf{y}} = \mathbf{X}^{*T}\mathbf{X}^T\boldsymbol{\alpha} \tag{9}$$

$$= \sum_i^m \alpha_i \mathbf{X}^{*T} x_i \tag{10}$$

4.3 *Kernel Ridge Regression in Lagrangian Form*

To derive kernel ridge regression (KRR) we rewrite what we did for ridge regression in Lagrangian form, using the dual feature vector $x_i \to \phi(x_i)$:

$$\min_{\xi,\mathbf{w}} \sum_{i=1}^n \xi_i^2, \tag{11}$$

[27] W. N. van Wieringen. Lecture notes on ridge regression. *arXiv:1509.09169 [stat]*, 2019

[28] taking the derivative and setting it to zero

subject to $\xi_i = y_i - \mathbf{w}^T\phi(x_i)$ and $\|\mathbf{w}\|^2 \leq C^2$, which can be written in terms of the following Lagrangian

$$\mathcal{L}_p(\mathbf{w}, \mathbf{e}; C) = \sum_i^m \xi^2 + \sum_{i=1}^m \beta_i(y_i - \mathbf{w}^T\phi(x_i) - \boldsymbol{\xi}) + \lambda\left(\|\mathbf{w}\|^2 - C^2\right), \tag{12}$$

where λ and β are now Lagrange multipliers. For optimality, the partial derivatives of $\mathcal{L}_p$ with respect to the primal variables $(\boldsymbol{\xi}, \mathbf{w})$ have to vanish:

$$\begin{aligned} \partial_{\boldsymbol{\xi}}\mathcal{L}_p &= 2\|\boldsymbol{\xi}\| - \boldsymbol{\beta} = 0 \quad \Leftrightarrow \quad \boldsymbol{\xi} = \frac{1}{2}\boldsymbol{\beta} = \boldsymbol{\alpha} & (13) \\ \partial_{\mathbf{w}}\mathcal{L} &= -\sum_{i=1}^m \beta_i\phi(x_i) + 2\lambda\mathbf{w} = 0 \quad \Leftrightarrow \quad \mathbf{w} = \frac{1}{2\lambda}\boldsymbol{\phi}\boldsymbol{\beta}. & (14) \end{aligned}$$

Plugging this back into the Lagrangian, we find the dual Lagrangian

$$\mathcal{L}_D = \sum_i \left(-\frac{1}{4}\beta_i + \beta_i y_i\right) + \frac{1}{4\lambda}\sum_{ij}\left(\beta_j\beta_i K_{ij}\right) - \lambda C^2, \tag{15}$$

with $K = \phi(x_i)^T\phi(x_j)$. Using α, as defined in eq. 13, we can write

$$-\lambda^2\sum_i \alpha_i^2 + 2\lambda\sum_i \alpha_i y_i - \lambda\sum_{ij}\alpha_i\alpha_j K_{ij} - \lambda C^2, \tag{16}$$

which we can optimize by taking the derivative with respect to $\boldsymbol{\alpha}$

$$\boldsymbol{\alpha} = (\mathbf{K} - \lambda\mathbf{I})^{-1}\mathbf{y}. \tag{17}$$

We see that for $K(x_i, x_j) = \langle x_i, x_i \rangle$ (and $\lambda = 0$) eq. 10 is equivalent eq. 17. We call K kernel. And we used it to express each dot product in terms of the kernel.

4.4 *Kernels*

Kernels are an elegant way to introduce non-linearities. A trivial way to introduce non-linearities is to add feature transformations, such as products of features, as new features. But this explicit mapping easily gives rise to a computational explosion for larger feature spaces.

Kernels can solve this problem via implicit mapping (cf. Figure 3).

For a function to be a kernel function, $K(\cdot, \cdot)$, it has to satisfy Mercer's condition which, less formally,[29] says that a kernel function needs to satisfy

$$\int_{\mathcal{X}\times\mathcal{X}} K(x, x')f(x)f(x')\,\mathrm{d}x\mathrm{d}x' \geq 0, \tag{18}$$

for all square-integrable functions f.

Instead of the dot product $\langle \cdot, \cdot \rangle$, one often chooses Gaussian or Laplacian kernels (cf. Figure 4 for some popular kernel functions).

[29] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004

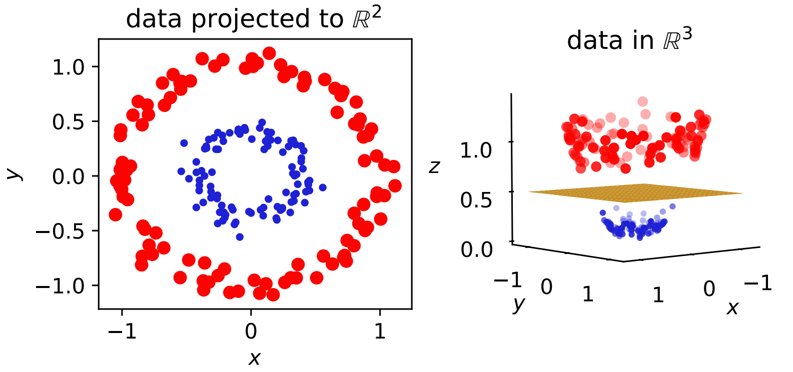


Figure 3: Visualization of one idea behind the kernel trick—mapping to higher-dimensional spaces to make problems linearly separable. In two dimension, the data (two different classes, colored in red and blue, respectively) is not linearly separable, but after applying the kernel $K(x,y) = x \cdot y + \|x\|^2 = \|y\|^2$ we can draw a plane to separate the classes (three-dimensional plot on the right).

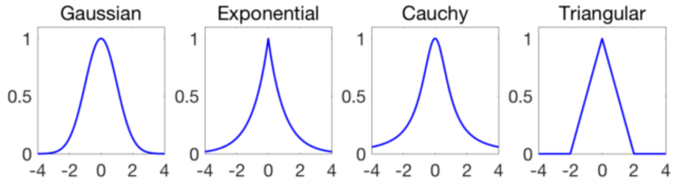


Figure 4: Examples for kernel functions. Taken from `https://francisbach.com/cursed-kernels/`.

What Do These Kernels Do? It is useful to think of them of a way to measure similarity between data points and then rewrite the regression problem in terms of these similarities (cf. Figure 5). For example, we can think of the width of the Gaussian as a measure of how local our reasoning is. Typically we will find that for zero width, the regression will interpolate the data.

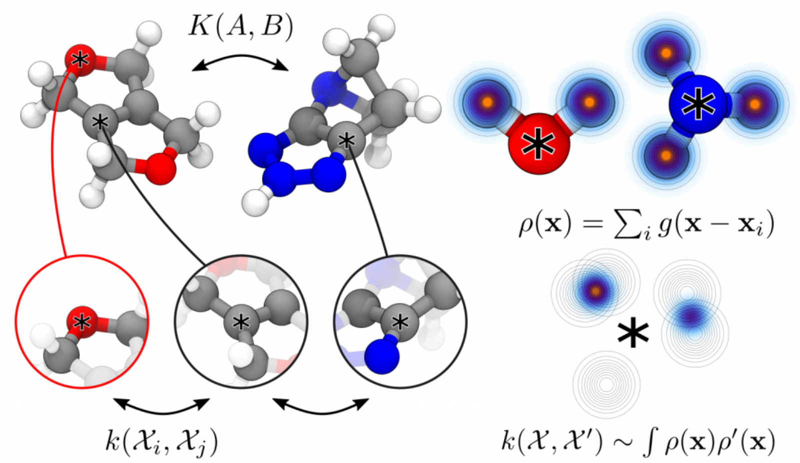

$K(A,B)$
$\rho(\mathbf{x}) = \sum_i g(\mathbf{x} - \mathbf{x}_i)$
$k(\mathcal{X}_i, \mathcal{X}_j)$
$k(\mathcal{X}, \mathcal{X}') \sim \int \rho(\mathbf{x})\rho'(\mathbf{x})$

Figure 5: Kernels as measures of similarity of molecules. Illustration of the COSMO group (Michele Ceriotti, `https://www.epfl.ch/labs/cosmo/research/page-148963-en-html/`). Often, chemical environments χ are represented in terms of atom-centered densities ρ, which are local "flavors" (projections onto some basis) of correlation functions.[22]

5 Predicting CO_2 Uptake in MOFs Using KRR

5.1 Splitting the Data

Unless you have a really small dataset, the first thing to do is to put data aside. The most trivial way to do so is to perform a random split of the data into a training and test set, for which one often chooses a ratio of 70/30.[30]

The most important principle in data science is to avoid any data leakage (i.e., to avoid that any information from the test data enters the model before we test it). Therefore, we carry out all transformations (like scaling), feature selection as well as hyperparameter optimization using only data from the training set.

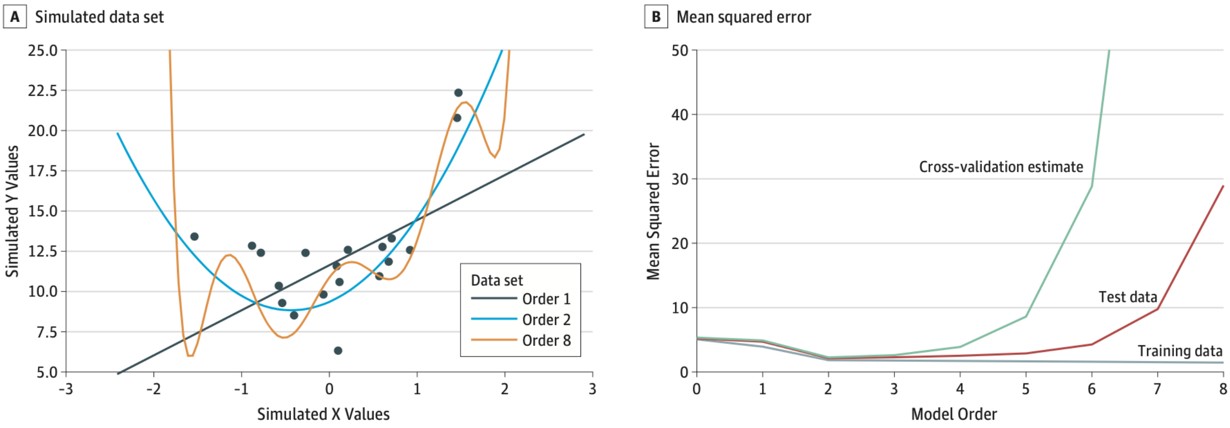


Figure 6: **A**, Simulated data set. The data set was generated from a quadratic model.**B**, Mean squared error. Mean squared error for the model was assessed against the data set used to train the model and against a separate test data set sampled from the same generative process with different random measurement error. Figure and Caption taken from [14].

The reason why we need to split the data in disjoint sets for training and testing is observable from Figure 6: One typically observes that when one increases the order of model complexity, one can fit the dataset arbitrarily well (if the number of parameters is equal to the number of datapoints we will find zero error (cf. Fig. 6). But our model might not (only) learn the underlying, physical, relationship between the features and the labels but also learn the noise or some errors which are specific to the dataset.

If one has so little data that one cannot afford to put data aside, one should apply k-fold cross-validation techniques in which the data is splitted into k subsets and the model is subsequently trained on all but one of the k folds and tested on the holdout fold.[31]

The choice of k is generally a trade-off between bias and variance. For small test sets (large k) the the variance in the test errors will be high, but the pessimism of the estimate will be lower due to the larger training set. In most cases $k = 10$ is a good default choice.[32]

The optimism associated with testing on the training set is shown in Fig. 6 **B**: With increasing model complexity the test error rapidly approaches zero, whereas we see that the test error for data that was generated using the same statistical process rapidly increases with increasing model complexity. We say that the difference between

[30] A. Ng. *Machine Learning Yearning*. 2018

[31] $k = n$ is known leave-one-out cross-validation for particularly small datasets (cf. Figure 7)

[32] S. Raschka. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. *arXiv:1811.12808 [cs, stat]*, 2018; and R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th international joint conference on artificial intelligence - volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc

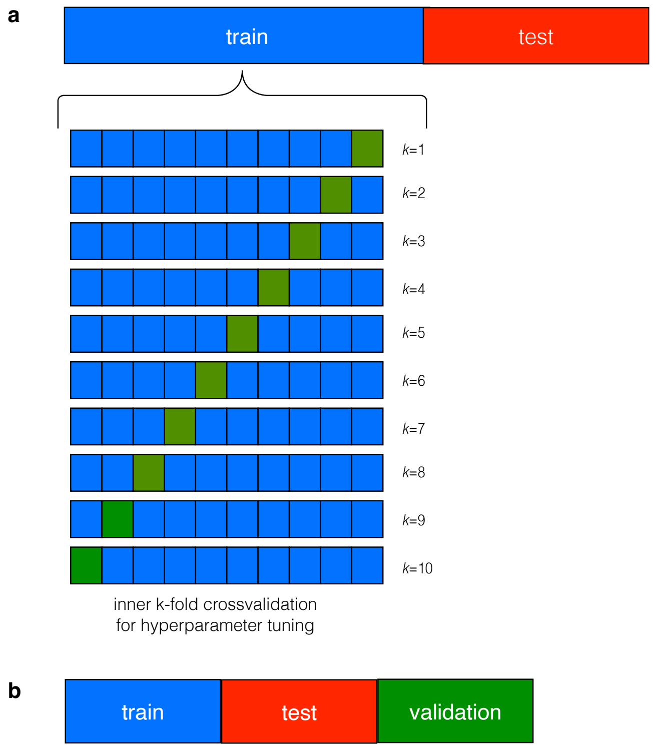


Figure 7: Comparison of two popular model selection techniques. **a** shows 10-fold cross validation with external test set and **b** show the split into three disjoint sets for training (blue), testing (red) and validation (green).

the training and test error is due to *overfitting*. The cross-validation error is typically pessimistically biased as we decrease the size of the training set by putting data aside.

Note that all the testing techniques assume that the data in the training and test set are independent, which is not always the case. Time series, like trajectories of molecular simulations, have autocorrelation (observations close in time are more similar) and other data might be clustered. In these cases there are special techniques (e.g., leave-one-cluster-out cross-validation that one should use.[33])

5.2 *Looking and Thinking About the Data*

The first stage in any data-driven project is to explore the data to see if there are obvious correlations, if the data needs some further cleaning or if we should discard some parts of it. This is also known as *exploratory data analysis (EDA)*.[34]

Especially in cases where we do not understand the physics of the problem yet, this is the most important stage as we have to find the proper features, or feature combinations to learn form. This stage is known as *feature engineering*.

In the case of gas adsorption in (micro)porous materials the physics is well understood and we also have an understanding of which features are needed to predict the target property. For this reason we will not perform extensive feature engineering.

This exploratory stage is also important to understand if there are *potential biases* in the underlying dataset.[35] We need to understand how the data is distributed, how diverse it is and if there are some

[33] B. Meredig, E. Antono, C. Church, M. Hutchinson, J. Ling, S. Paradiso, B. Blaiszik, I. Foster, B. Gibbons, J. Hattrick-Simpers, A. Mehta, and L. Ward. Can machine learning identify the next high-temperature superconductor? Examining extrapolation performance for materials discovery. *Molecular Systems Design & Engineering*, 3(5):819–825, 2018

[34] J. W. Tukey. *Exploratory data analysis*. Addison-Wesley series in behavioral science. Addison-Wesley Pub. Co, Reading, Mass, 1977

[35] X. Jia, A. Lynch, Y. Huang, M. Danielson, I. Lang'at, A. Milder, A. E. Ruby, H. Wang, S. A. Friedler, A. J. Norquist, and J. Schrier. Anthropogenic biases in chemical reaction data hinder exploratory inorganic synthesis. *Nature*, 573(7773):251–255, 2019

minority classes. Often, the interesting cases are the rare ones and we have to pay special care that our model also performs well for them.

5.3 Getting Some Baselines

Before we fit any machine learning model, we should also get an understanding what "good performance" actually means. In the machine learning community, this is known as *getting a baseline*. This can either be a model that is much simpler than the one we plan to build, the current state of the art (if we have access to it) or simply some heuristics like the median or mean of the training set. It is a trivial concept, but still people often forget to report this metric.

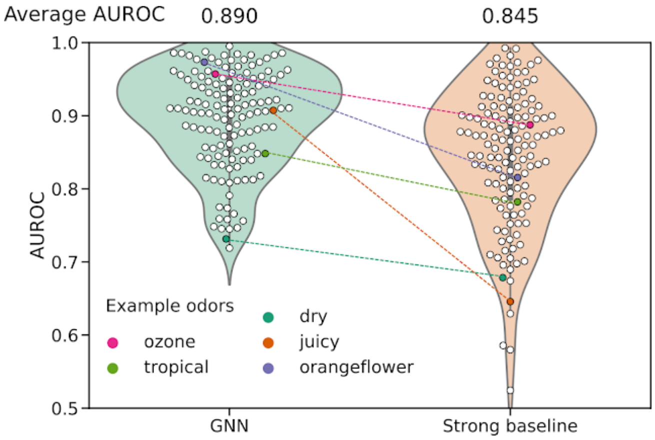


Figure 8: Example of the performance of a GNN (Google's submission to the odor prediction challenge[16], see `https://ai.googleblog.com/2019/10/learning-to-smell-using-deep-learning.html`) on odor descriptors against a baseline. Closer to 1.0 means better.

Especially for classification, this can be important. One such example is shown in Figure 8 for the case of odor predictions, where researchers tried to train a graph neural network to predict the odor of a molecule based on its structure. But how do we know that the use of such a complicated model really is useful and needed? To answer this question, the authors built a state-of-the-art model and performed the classification using this model—without comparison to the baseline, we would have no clue how good an AUC score 0.89 for this task is.

In Figure 9 we show an example from our own research in which we try to assign the oxidation state of copper using a machine learning model and different baseline models (some of them using only random [uniform] guessing or guessing based on the distribution of the training set [stratified]). We measure the performance using different metrics like accuracy, recall and precision and observe that without using a proper baseline a high score for some metric might easily be confused with good predicitve performance.

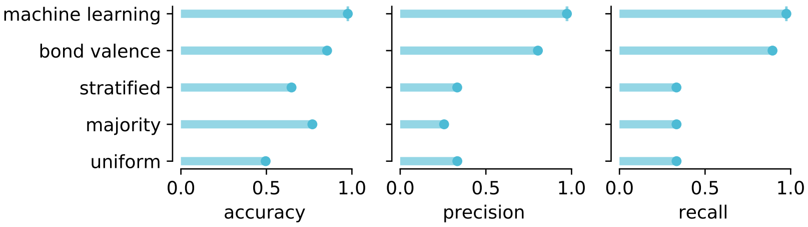


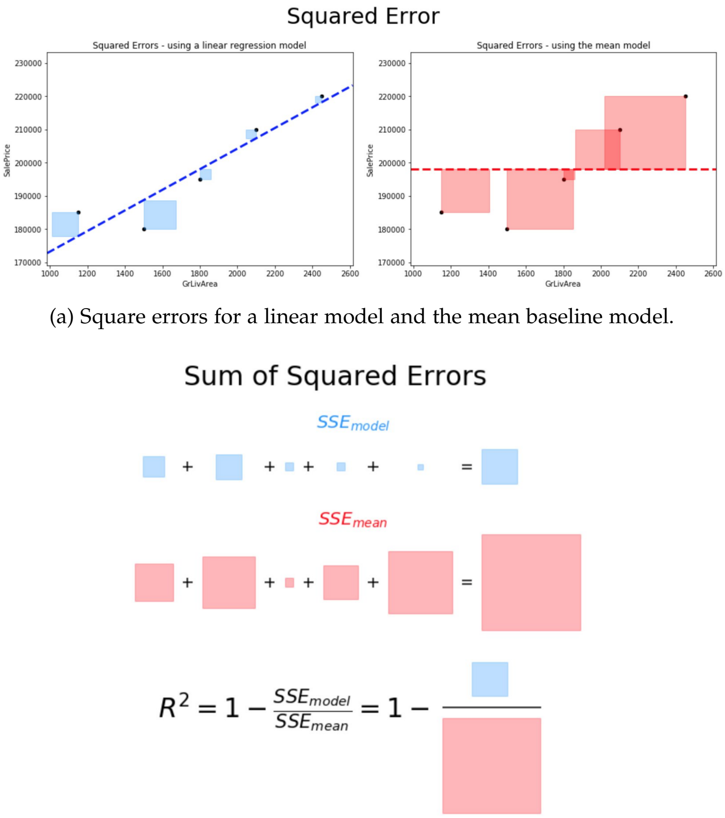
Figure 9: Classification metric scores for metrics like accuracy (number of correct predict over total predictions), the precision (true positive rate) or recall (inverse false positive rate) for our model and different baselines, including the state-of-the-art (bond-valence) and models that randomly guess the label (uniform) or that randomly select the label from the training distribution (stratified) or by only predicting the most common label.

Aside: Baselines and R^2 The R^2 score is a widely used metric, even though there are a lot of issues associated with it and it certainly shouldn't be used as a loss function.[36].

One of the definitions of the score is the ratio of variance of the predictions (generated by some model m) to the variances of the labels (y)

$$R^2 = \frac{\operatorname{var}(m(X))}{\operatorname{var}(y)}. \tag{19}$$

One can hence interpret the R^2 as measure of the decrease in error that your model provides over a baseline of guessing the mean, this is shown in Figure 10.



(a) Square errors for a linear model and the mean baseline model.

(b) The ratio of the squared errors can be used to calculate the R^2 score.

Figure 10: The use of baselines to define the R^2 score. Visualizations from Ted Petrou (https://twitter.com/TedPetrou/status/1205897381356683264?s=09).

[36] read these lecture notes if you want to learn more http://www.stat.cmu.edu/~cshalizi/mreg/15/lectures/10/lecture-10.pdf
Think about what could go wrong with eq. 19 before going there: when is it large, when is it small, is it continuous?

5.4 Standard Scaling

Many methods, like kernel methods or instance based learners like *k*nn, explicitly depend on the distance or similarity between different features. If we would simply use the unmodified raw data, the different features might be in completely different units or on completely different scales. This might bias those model to some features (e.g., as their numerical values are much larger, this effect is illustrated for the assignment of nearest neighbors in Figure 11). The most common way to avoid this bias is to use *z*-score normalization (`StandardScaler` in `sklearn`). Here, one centers the data by subtracting the mean (μ) and scales the width of the distribution by dividing by the standard deviation (σ).

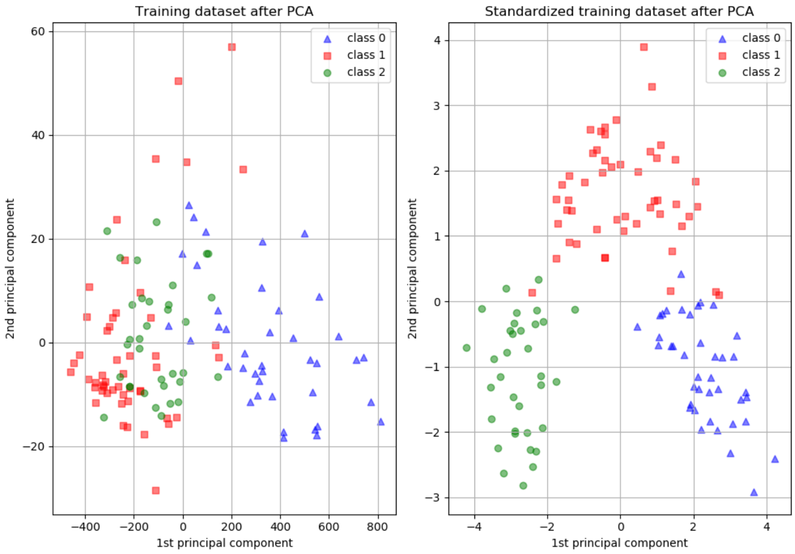


Figure 11: Effect of feature scaling. PCA is performed on scaled and unscaled data. It is evident that it is easier to separate the classes after scaling. We took this figure from the `sklearn` documentation (`https://scikit-learn.org/stable/auto_examples/preprocessing/plot_scaling_importance.html`).

To reduce the effect of outliers, it can be useful to replace the mean with the median (more robust measure of centrality) and the standard deviation by the interquartile range (`RobustScaler` in `sklearn`).

For distributions that are not normal, it can be beneficial to perform feature transformations like log transform to remove skew in the data.

5.5 Optimizing the Model

Optimizing the Hyperparameters Machine learning models typically have many knobs one can tune. And one should tune those knobs to get a good predictive performance.

Also here, we need to be careful to avoid too optimistic estimates as our overarching goal is to predict on datasets that are different from the training sets. This is, if we change a hyperparameter and test on the training set we get a too positive estimate (as we discussed in Section 5.1). If, on the other hand, we already use data from the test set, we leak information as we will bias the selection of the hyperparameters based on some particular (noise) pattern in the test set. For this reason, we need to split once more. For large datasets, one can typically just split the data randomly in three sets (training set, validation set for hyperparameter optimization and test set for the final estimate of the predictive performance). In case of k-fold cross validation, one can use a nested version in which the inner loop is used to search optimal hyperparameters and the outer loop is used to estimate the predictive performance (generalization error).

Grid Search For this exercise, we will use grid search, which is a popular due to its conceptual simplicity. But, there are several techniques that are more efficient in grid search (even random search was shown to be more efficient than grid search).[37]

Optimizing the Feature Set KRR is particularly sensitive to the feature set as it is based on similarities in this space. Also, one typically wants to compress the number of features to avoid redundancy and increase computational efficiency.

There is no silver bullet to perform this feature selection. One could use the first few principal components, filter out features that have low variance (`sklearn.feature_selection.VarianceThreshold`) or add (or eliminate) one after another features to the model until the performance does not change anymore. This is known as recursive feature addition (or elimination).

[37] One popular library that implements more efficient search techniques is `hyperopt`. Have a look at the blog post at `http://steventhornton.ca/blog/hyperparameter-tuning-with-hyperopt-in-python.html` if you want to learn more about how to use this library.

5.6 *Analysing What the Model Learnt*

Interpretable and explainable machine-learning[38] is a topic of active research. The scope of the terms themselves are ill-defined[39] and current approaches always come with a trade-off.

Some models are inherently interpretable, e.g., one can analyze the weights in a linear model to estimate the importance of different variables. In cases where this is not possible (neural networks, kernel methods) one can fall back to post-hoc models to explain the predictions made by the model. As this basically means building a model to explain the predictions of a model, there are limitations.[40]

Permutation Feature Importance One of the most intuitive ways to calculate feature importance is to use the permutation technique.[41] Here, one tries to estimate the importance of a feature by permuting a feature column and running the model with this permuted feature column. By measuring the difference in test error for non-permuted feature and permuted features on can estimate the importance of a particular feature or feature combination. For example, if a feature is not important, permuting the values for it will not affect model performance. On the other hand, permuting an important feature will drastically affect the model performance[42].

[38] C. Molnar. *Interpretable Machine Learning*. 2019. `https://christophm.github.io/interpretable-ml-book/`

[39] Z. C. Lipton. The Mythos of Model Interpretability. *arXiv:1606.03490 [cs, stat]*, June 2016

[40] If the post-hoc model would be perfect, then there would be no need for the primary model that we try to explain[**?**].

[41] See also the explanations at `https://christophm.github.io/interpretable-ml-book/feature-importance.html`

[42] Keep in mind that one needs to be careful with using permutation feature importance for correlated features [1].

References

[1] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer. Permutation importance: A corrected feature importance measure. *Bioinformatics*, 26(10):1340–1347, May 2010.

[2] P. G. Boyd, Y. Lee, and B. Smit. Computational development of the nanoporous materials genome. *Nature Reviews Materials*, 2(8):17037, 2017.

[3] M. Bui, C. S. Adjiman, A. Bardow, E. J. Anthony, A. Boston, S. Brown, P. S. Fennell, S. Fuss, A. Galindo, L. A. Hackett, J. P. Hallett, H. J. Herzog, G. Jackson, J. Kemper, S. Krevor, G. C. Maitland, M. Matuszewski, I. S. Metcalfe, C. Petit, G. Puxty, J. Reimer, D. M. Reiner, E. S. Rubin, S. A. Scott, N. Shah, B. Smit, J. P. M. Trusler, P. Webley, J. Wilcox, and N. Mac Dowell. Carbon capture and storage (CCS): the way forward. *Energy & Environmental Science*, 11(5):1062–1176, 2018.

[4] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. Vanderplas, A. Joly, B. Holt, and G. Varoquaux. API design for machine learning software: experiences from the scikit-learn project. *arXiv:1309.0238 [cs]*, 2013.

[5] X. Jia, A. Lynch, Y. Huang, M. Danielson, I. Lang'at, A. Milder, A. E. Ruby, H. Wang, S. A. Friedler, A. J. Norquist, and J. Schrier. Anthropogenic biases in chemical reaction data

hinder exploratory inorganic synthesis. *Nature*, 573(7773):251–255, 2019.

[6] L. Jonkers, H. Hillebrand, and M. Kucera. Global change drives modern plankton communities away from the pre-industrial state. *Nature*, page 1, 2019.

[7] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th international joint conference on artificial intelligence - volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[8] K. S. Lackner, S. Brennan, J. M. Matter, A.-H. A. Park, A. Wright, and B. v. d. Zwaan. The urgency of the development of CO2 capture from ambient air. *Proceedings of the National Academy of Sciences*, 109(33):13156–13162, 2012.

[9] Z. C. Lipton. The Mythos of Model Interpretability. *arXiv:1606.03490 [cs, stat]*, June 2016.

[10] B. Meredig, E. Antono, C. Church, M. Hutchinson, J. Ling, S. Paradiso, B. Blaiszik, I. Foster, B. Gibbons, J. Hattrick-Simpers, A. Mehta, and L. Ward. Can machine learning identify the next high-temperature superconductor? Examining extrapolation performance for materials discovery. *Molecular Systems Design & Engineering*, 3(5):819–825, 2018.

[11] C. Molnar. *Interpretable Machine Learning*. 2019. `https://christophm.github.io/interpretable-ml-book/`.

[12] A. Ng. *Machine Learning Yearning*. 2018.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[14] R. A. Poldrack, G. Huckins, and G. Varoquaux. Establishment of Best Practices for Evidence for Prediction: A Review. *JAMA Psychiatry*, 2019.

[15] S. Raschka. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. *arXiv:1811.12808 [cs, stat]*, 2018.

[16] B. Sanchez-Lengeling, J. N. Wei, B. K. Lee, R. C. Gerkin, A. Aspuru-Guzik, and A. B. Wiltschko. Machine Learning for Scent: Learning Generalizable Perceptual Representations of Small Molecules. *arXiv:1910.10685 [physics, stat]*, 2019.

[17] T. Simonite. Solve These Tough Data Problems and Watch Job Offers Roll In. *Wired*, Oct. 2017.

[18] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[19] A. Sturluson, M. T. Huynh, A. R. Kaija, C. Laird, S. Yoon, F. Hou, Z. Feng, C. E. Wilmer, Y. J. Colón, Y. G. Chung, D. W. Siderius, and C. M. Simon. The role of molecular modelling and simulation in the discovery and deployment of metal-organic frameworks for gas storage and separation. *Molecular Simulation*, 45(14-15):1082–1121, 2019.

[20] J. W. Tukey. *Exploratory data analysis*. Addison-Wesley series in behavioral science. Addison-Wesley Pub. Co, Reading, Mass, 1977.

[21] W. N. van Wieringen. Lecture notes on ridge regression. *arXiv:1509.09169 [stat]*, 2019.

[22] M. J. Willatt, F. Musil, and M. Ceriotti. Atom-density representations for machine learning. *J. Chem. Phys.*, 150(15):154110, Apr. 2019.

[23] O. M. Yaghi. Reticular Chemistry in All Dimensions. *ACS Central Science*, 5(8):1295–1300, 2019.