# *Data Science in Materials Science – Lab 27.07.22*

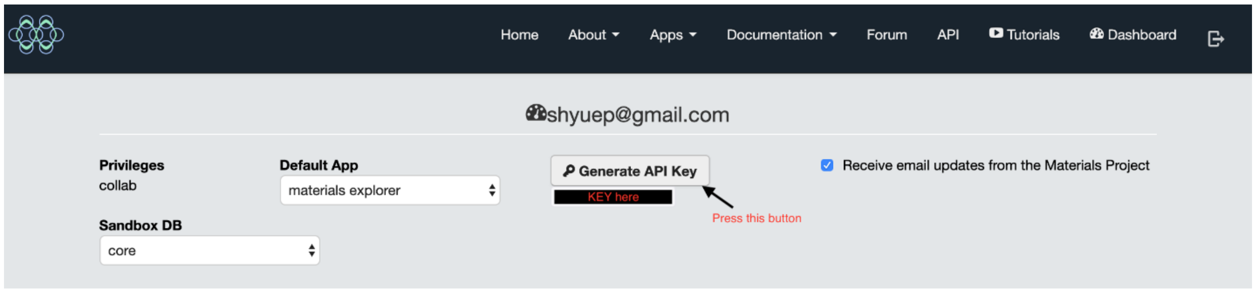
## Part A: Data Exploration

### Q1 - Materials Project data

Using pymatgen (or any alternative approach), query the Materials Project for the following properties of all ABO3 type compounds (hint: look at this example notebook and figure out what is the best way to do this):

- materials project identifier (this is called task_id)
- formula of the compound
- number of sites in the unit cell
- band gap
- formation energy per atom
- icsd ids (note that `icsd_id` key is no longer used, use `icsd_ids` instead)
- energy above hull

You will need to sign up for a free account at https://www.materialsproject.org and get an API_KEY from the https://www.materialsproject.org/dashboard.



shyuep@gmail.com

Privileges
collab

Default App
materials explorer

Sandbox DB
core

[✓] Receive email updates from the Materials Project

Answer the following questions:

1. Perform the query and convert the data into a Pandas DataFrame.
2. How many ABO3 compounds in total are there in the Materials Project? How many unique ABO3 formulae are there? What is the average number of crystals (also known as polymorphs) per ABO3 formula?
3. Typically, the existence of an ICSD (Inorganic Crystal Structure Database) id is a rough indication of whether a compound is an experimentally-known compound

or a theoretical compound. What fraction of the compounds have at least one icsd id?

4. The formation energies in the Materials Project are given in eV/atom. Create an additional column in your dataset that has the formation energies in J/mol.
5. Let us assume that materials with energy above hull of >0.03 eV/atom are `unstable` and are `potentially stable` otherwise. Furthermore, band gaps of 0, (0, 1], [1, +inf) are indicative of `metallic`, `small band gap`, `large band gap` for the materials. Create a table of the number of ABO3 crystals in each joint category, e.g., `(unstable, metallic)`, `(unstable, small band gap)`, etc.
6. Plot the distribution of (a) the formation energies per atom (in eV/atom) and (b) the band gaps of all the materials. Annotate the plots with the average and standard deviation of each quantity. Ensure that all axes are labelled appropriately with units, i.e., something that you can potentially put in a scientific paper.

## Q2 - Publicly available research data

Researchers frequently share the datasets they have via various online platforms. Figshare is one such online platform. We will use this example to illustrate some of the challenges in working with datasets.

1. Query for the data from https://ndownloader.figshare.com/files/9158587, which is in the csv format. This is a dataset of from high-throughput DFT calculations of formation energy, stability and oxygen vacancy formation energy of ABO3 perovskites (https://www.nature.com/articles/sdata2017153) available in the Open Quantum Materials Database (OQMD). This dataset, which we will call the OQMD dataset, comprises computed data on compounds known as perovskites. Parse the data into a Pandas DataFrame.
2. How many compounds in total are there?
3. How many total columns are there in the dataset? Print the column names.
4. Unfortunately, the dataset contains invalid data and some of the data are also not properly tagged in the right data type. For example, the formation energy column contains strings, rather than floating point numbers. There are also strings that indicate whether a particular data point is valid. Filter the DataFrame to remove all invalid data points, i.e., those that contain just "-" in the formation energy column. and convert the formation energy column to the proper floats. How many data points remain?
5. Plot the distribution of the formation energies per atom. Annotate the plot with the average and standard deviation.

## Q3 - Comparing data sets

It is often useful to compare similar datasets to check them against each other. The simplest form of the perovskite crystal structure has formula ABO3, and such compounds are present in both the dataset you queried from the Materials Project in Q1 and the OQMD data you downloaded from figshare in Q2.

1. Identify the subset of formulas that are present in both the Materials Project dataset and OQMD dataset (hint: look at the Python built-in `set` object). How many formulas are there?
2. Plot the distribution of the formation energies per atom of this subset of formulas for (a) the Materials Project dataset, and (b) the OQMD data, overlaying the two distributions on top of each other. Annotate your plot with the mean and standard deviation for each data set.
3. Perform a hypothesis test at the 95% level to determine if there is a significant difference between the formation energies reported in the Materials Project and the OQMD dataset (hint: check out the scipy.stats.ttest_ind method). Discuss your findings, including providing any possible explanations for any discrepancy between the two datasets.

Source: https://github.com/materialsvirtuallab/nano281

## Part B: Machine Learning

Download `data.csv` file in this repo. It contains basic data for all materials in the Materials Project.

## Q1 - Exploratory data analysis

Load the `data.csv` in variable `orig_data` using `pandas.read_csv` with `na_filter=False` option, and perform the following analysis.

1. How many elements are there in this data set?
2. What is the maximum number of elements (`max_el`) in a single structure?
3. Plot a barplot of the number of materials having 1, 2, 3, ... `max_els`. Use a log scale for the y-axis.
4. Count the number of materials where each element is present. Sort this count and answer the following questions. What are the 10 most common elements in this data set? and what are the 10 least common elements in this data set?

Hint: When dealing with formula, you may use `pymatgen.core.Composition` to speed up the process. For example, the following code snippet shows the use of Composition to process formula. For more usage, you may visit https://matgenb.materialsvirtuallab.org/2013/01/01/Basic-functionality.html

```
from pymatgen.core import Composition
comp = Composition('Al2O3')
print(comp.elements)  # this will give you the elements
print(comp.to_data_dict['unit_cell_composition'])  # this will give you the
elementstr-stoichiometry dictionary.
```

## Q2 - Data cleaning and feature computations

About 80% of the effort in ML modelling is in data processing. The goal is to develop ML models to predict the formation energy per atom and band gap of the material from the formula. To do that, we will first convert the formula to numeric vectors (descriptors) for model inputs.

1. Filter out materials that contain noble gas elements and save it in variable `data`. How many materials are left? Store this number in the variable `n`.
2. Load the element property data file `element_properties.csv` in variable `element_data` using pandas by setting `index_col=0` in `pandas.read_csv` function. How many NaN (Not a Number) are there in each column?

3. Compute the mean values for each column, ignoring the NaNs. For each column, fill the NaN with the mean value of that column. This is a common data imputation technique.
4. Compute the composition-averaged `AtomicRadius` for all materials and store the results in variable `atomic_radius`. For example, averaged `AtomicRadius` for `Li2O` can be computed as `(2 * 1.45 + 0.6) / 3`, where `1.45` is the `AtomicRadius` for `Li` and `0.6` is the `AtomicRadius` for `O`.
5. Compute the composition-averaged properties for all properties in `element_data` and for all materials. Store the results in the variable `average_properties`. `average_properties` should have a dimension of `(n, 16)` where `n` is the number of materials and 16 is the number of properties.
6. Similar to the previous computations of average properties, compute the maximum properties and minimum properties for all properties and all materials, and store them in variables `max_properties` and `min_properties` respectively. Both variables should have dimension `(n, 16)`.
7. Concatenate `average_properties`, `max_properties` and `min_properties`, and store the result in variable `design_matrix` with dimension `(n, 48)`.

## Q3 - Regression and classification modeling

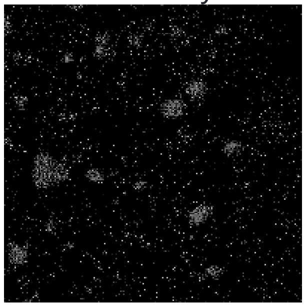
We are going to use `band_gap` and `formation_energy_per_atom` in `data` as the targets, and store them in variable `targets`. `targets` should be a dataframe with dimension of `(n, 2)`.

1. Split the data (`design_matrix` as X, and `targets` as y) into training and test sets in the ratio 90%:10%. Store the training data in variables `train_X` and `train_y` and the test data as variables `test_X` and `test_y`. To make sure the data is reproducible, set the `random_state=42` in `sklearn.model_selection.train_test_split`.
2. Compute the mean and standard deviation of columns in `train_X`. Both of them should be length 48 vectors. Use them to normalize `train_X` and `test_X`, so that each column has a mean of 0 and standard deviation of 1. Store the normalized design matrices to `norm_train_X`, `norm_test_X`.
3. Train a linear model to predict `formation_energy_per_atom`. What are the mean absolute error (MAE) and root mean squared error (RMSE) on the test data?
4. Train a Ridge regression model and a LASSO regression model for the `formation_energy_per_atom` using `alpha=0.1`, what are the test MAE and RMSE?
5. Let's define `band_gap < 0.001` as metallic and `band_gap >= 0.001` as nonmetallic. Construct linear discriminant analysis, quadratic discriminant analysis, and logistic regression models on train data and predict the accuracy of the models on test data.
6. What are the problems of using only the compositions to predict material properties?

Source: https://github.com/materialsvirtuallab/nano281

## Q4 - Clustering

In this problem, we will be looking at catalyst clusters. The image file `catalyst.png` below is extracted from a figure shared on figshare by Gomez-Bolivar et al. (Front. Microbiol., 20 June 2019, DOI: 10.3389/fmicb.2019.01276). It is an energy dispersive X-ray (EDX) microanalysis of Pd/Ru bimetallic nanoparticle catalysts synthesized by Escherichia coli. For this whole exercise, it is recommended that you use the `hot` colormap in matplotlib.



1. Read in the image as a numpy array using matplotlib. Show the image in your Jupyter notebook. What are the dimensions of the array?
2. Plot the distribution of the values in the numpy array representing the image. Note that the values in the numpy array are between zero and 1.
3. Measured images has a variety of levels. Sometimes, we want to label each pixel at pre-specified levels, e.g., 0 representing the background, and fixed values representing certain features. This is known as vector quantization. Here, we will quantize the image using K-means. We know for a fact that there are two elements (Pd and Ru) in the system. Using K-means, quantize the image such that there are three levels representing the background, and one level for each element. Ensure that 0 corresponds to the background (this should be the cluster with the largest number of data points) and non-zero levels correspond to the elements. Plot the quantized image.
4. For the purposes of this exercise, we will not attempt to distinguish between different elements. Any value within the numpy array that is > 0 is considered a catalyst particle. Use K-means clustering to distinguish identify clusters of metal particles (you will need to figure out what a good value of K is). Plot your clustered image, ensuring that each cluster has a different color. Comment on how you chose your value of K.
5. Finally, we will use a density-based clustering method called DBSCAN. Similar to part 4, any value in the numpy array that is > 0 is considered a catalyst particle. Use DBSCAN clustering to distinguish identify clusters of metal particles (you will need to figure out what a good value of `eps` is). Plot your clustered image, ensuring that each cluster has a different color. Comment on how you chose your value of `eps`.
6. Discuss on the differences between the K-means and DBSCAN results, and which method is more appropriate for the purpose we are using it for.

Source: https://github.com/materialsvirtuallab/nano281