

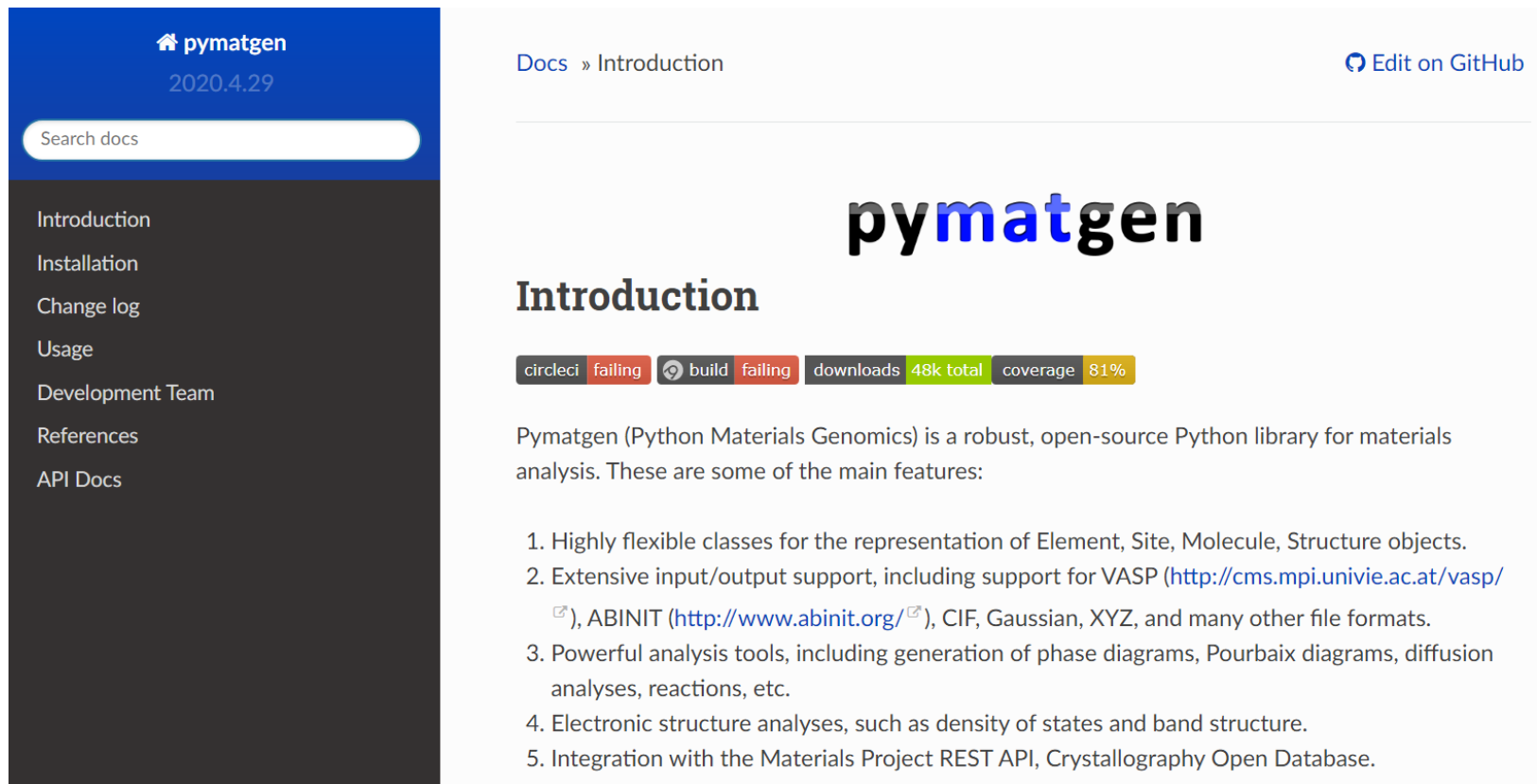
# MSE1065: Lab 5

## Case study #3: Crystal structure prediction through Logistic Regression

# Learning objectives for this lab

- Utilize Logistic regression model to predict the crystal structure
- Learn to utilize materials database and python utility, **pymatgen**

# Source



The screenshot displays the pymatgen website. The left sidebar is dark blue with a search bar and a list of navigation links: Introduction, Installation, Change log, Usage, Development Team, References, and API Docs. The main content area is white and features the pymatgen logo, the title 'Introduction', and a status bar with metrics: circleci failing, build failing, downloads 48k total, and coverage 81%. Below this, a paragraph describes pymatgen as a robust, open-source Python library for materials analysis. A list of five features follows: 1. Highly flexible classes for the representation of Element, Site, Molecule, Structure objects. 2. Extensive input/output support, including support for VASP (<http://cms.mpi.univie.ac.at/vasp/>), ABINIT (<http://www.abinit.org/>), CIF, Gaussian, XYZ, and many other file formats. 3. Powerful analysis tools, including generation of phase diagrams, Pourbaix diagrams, diffusion analyses, reactions, etc. 4. Electronic structure analyses, such as density of states and band structure. 5. Integration with the Materials Project REST API, Crystallography Open Database.

pymatgen  
2020.4.29

Search docs

Introduction  
Installation  
Change log  
Usage  
Development Team  
References  
API Docs

Docs » Introduction [Edit on GitHub](#)

## pymatgen

### Introduction

circleci failing build failing downloads 48k total coverage 81%

Pymatgen (Python Materials Genomics) is a robust, open-source Python library for materials analysis. These are some of the main features:

1. Highly flexible classes for the representation of Element, Site, Molecule, Structure objects.
2. Extensive input/output support, including support for VASP (<http://cms.mpi.univie.ac.at/vasp/>), ABINIT (<http://www.abinit.org/>), CIF, Gaussian, XYZ, and many other file formats.
3. Powerful analysis tools, including generation of phase diagrams, Pourbaix diagrams, diffusion analyses, reactions, etc.
4. Electronic structure analyses, such as density of states and band structure.
5. Integration with the Materials Project REST API, Crystallography Open Database.

Data availability:

<https://pymatgen.org/>

# Dataset

- Downloaded from pymatgen:  
<https://pymatgen.org/>
- Inputs
  - Number of features 18
  - Features include Atomic mass, Ionic radius, Lattice constant, Young's Modulus etc.
- Output
  - Crystal structure – FCC, BCC, HCP

# What will you accomplish in the lab?

- In this lab, we will implement different logistic regression models on the dataset provided
- We will be using SKLearn's LogisticRegression class to fit classification models and compute mean misclassification error.
- We will extend our model by including regularization using SKLearn
- In the last section, we will implement feature selection under Logistic regression by using 'L-1' penalty term

# Lab notebook

Jupyter Lab5\_LogRegression Last Checkpoint: Yesterday at 1:54 PM (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted

Python 3 C

Run

Markdown

## Lab - 5 Logistic Regression

In this lab, we are going to perform classification of elements as 'FCC', 'BCC', 'HCP', given their elemental attributes. The attributes or input features to our model include atomic mass, electronegativity, ionic radius etc.

Detailed list is provided below

```
In [43]: element_attributes = ["atomic_number", "atomic_volume", "boiling_point", "en_ghosh", "evaporation_heat", "heat_of_form",  
                             "lattice_constant", "melting_point", "specific_heat", "atomic_mass", "atomic_radius",  
                             "electrical_resistivity", "molar_volume", "bulk_modulus", "youngs_modulus",  
                             "average_ionic_radius", "density_of_solid", "coefficient_of_linear_thermal_expansion"]
```

```
In [44]: # import libraries  
import numpy as np  
import pandas as pd
```

The data for 47 element is provided in the file 'inputX.csv' and corresponding structure values are provided in 'origSt.csv'. Load the csv files into appropriate data frames

```
In [45]: inputX = np.loadtxt(open("inputX.csv", "rb"), delimiter=",") #pd.read_csv('inputX.csv')
```

```
In [46]: outY = np.loadtxt(open("origSt.csv", "rb"), dtype='str', delimiter=",") #pd.read_csv('origSt.csv')
```