Contents lists available at ScienceDirect

Materials Today: Proceedings

journal homepage: www.elsevier.com/locate/matpr

Virtual Prediction of Material Properties

Arpan Kumar

Dept. of Computer Sc., Ramakrishna Mission Vidyamandira, Belur, Howrah, India

ARTICLE INFO

Article history:
Available online 7 February 2022

Keywords:
Materials
MAPI
Pymatgen
Matminer
Python
Sklearn
Scikit-Learn
Machine Learning
Support Vector Machine
Classification
Matplotlib
Seaborn

ABSTRACT

To start working with materials, it is most important to determine the properties to ensure its suitability but sometimes it is not only difficult but also time consuming and costly affair to arrange an experimentation with the materials. To overcome this problem some free libraries of python like pymatgen, matminer etc. are used with Materials Application Programming Interface (API) to gather and process datasets and when combined with Machine learning libraries like Sklearn, a machine Learning Model can be built. In the present work, with the aid of glass_ternary_hipt dataset, a metallic glass formation dataset for Co-Fe-Zr, Co-Ti-Zr, Co-V-Zr and Fe-Ti-Nb ternary alloy systems, a Support Vector Machine Classifier, is built as an example to predict the glass forming ability of the alloys. The accuracy of the model is checked and a heatmap is generated to show the correlation between the features and the target. The development of these program is very intuitive with these packages and python for prediction of material properties virtually.

1. Introduction

Mankind is using materials from the day of their inception and presently there is enormous number of materials to choose from. In order to select a particular one best suitable to the application, sufficient database is needed. These data can be obtained by conducting different tests. But sometimes conducting these tests are difficult, time taking, risky and costly so the researchers need to find alternative means to gather required data to complete the work. There are many Frameworks and libraries that provides the same and mathematical tools to use on the data in order to get data. Some of those are unstable, high priced where as some are free to use, robust and reliable. Using these, one can estimate and predict the results of an experiment without actually conducting them. Current work reviews the endless possibilities with the open-source robust packages and libraries based on Python in the field of Material Science with Machine Learning.

Pymatgen [1] (PYthon MATerials GENomics), developed from works of many researchers [2–13], has many optimized Inbuilt mathematical and statistical tools for computation. These computations allow one to build a class or structure of a material inside the program. Materials Project [14] also makes available all of its data and scientific analysis through the open access Materials Application Programming Interface (MAPI). It is a Robust API managed by scientists and researchers and developers. Normally Matminer [15] is used to get the datasets and use it for machine learning. When combined with pymatgen and Scikit-learn a robust machine learning model can be built.

For Researchers who do not have the time to learn a programming language from the scratch Python comes into picture with its intuitive syntax. Python is a general-purpose programming language developed by Guido van Rossum and others [16]. Python is well-known for its simplicity, and vast developer community.

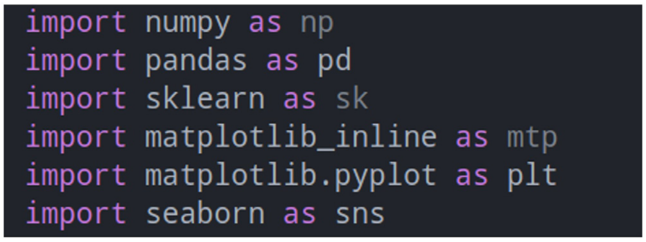
Scikit-learn (Sklearn) is a machine learning library that is developed by Pedregosa and others [17] for Python. It has all the tools of machine learning packaged inside of it. It has been used in the machine learning part of the current work and also, some packages like NumPy [18], matplotlib [19], pandas [20], seaborn [21] are used for general purpose work.

Present work is a review of existing python libraries and APIs in the materials field and implementation of the same to train a machine learning model easily using python and for the same Python 3.10 has been used.

2. Methodology

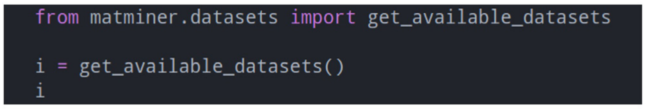
The glass_ternary_hipt dataset of Matminer database is used for testing a Machine Learning algorithm in the present work. A sup-

E-mail address: kumararpan227@gmail.com



```
import numpy as np
import pandas as pd
import sklearn as sk
import matplotlib_inline as mtp
import matplotlib.pyplot as plt
import seaborn as sns
```

Fig. 1. Importing Library Code.



```
from matminer.datasets import get_available_datasets

i = get_available_datasets()
i
```

Fig. 2. Getting Dataset Code.

port vector machine classifier is used in this work. To form and train machine learning model following steps are followed and the Python Code and its respective outputs are shown:

2.1. Importing the required python libraries

At first of any ML program all the needed general-purpose libraries like NumPy, pandas, matplotlib, seaborn etc. are included at the start of the code to ensure all the packages are integrated into the code.

All the important libraries needed for the program is included in this step is shown in Fig. 1 above.

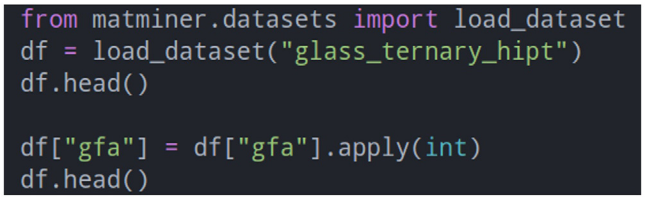
2.2. Selecting the dataset

A sufficient dataset is needed for good execution, fitting and accuracy of the program.

Now the available datasets are checked with the code of Fig. 2. The output is in Fig. 3.

2.3. Loading the dataset

The dataset glass_ternary_hipt is chosen for this program. It is a Metallic glass formation dataset for ternary alloys, collected from the high-throughput sputtering experiments measuring whether it is possible to form a glass using sputtering. The hipt experimental data are of the Co-Fe-Zr, Co-Ti-Zr, Co-V-Zr and Fe-Ti-Nb ternary systems. In this data set the target is the column named "gfa"



```
from matminer.datasets import load_dataset
df = load_dataset("glass_ternary_hipt")
df.head()

df["gfa"] = df["gfa"].apply(int)
df.head()
```

Fig. 4. Importing Dataset to a Data Frame.

In the figure below (Fig. 4) first two lines loads the dataset into a pandas data Frame. The third line prints the first 5 rows of the dataset. As the dataset has over 5 k rows the whole dataset is not shown. All the features of the dataset are described here [22]. The next two lines sets the target value to an integer from Boolean. The output is shown in Fig. 5.

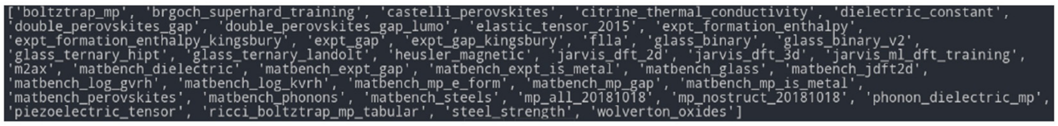
2.4. Data preparation

In this step the data is prepared for fitting it into the machine learning model. The data is cleaned the extra features are neglected to stop overfitting as shown by Ying [23] in his work. This step is vital for the accuracy of the model. So, at first the Composition function from pymatgen core is imported and the formula is changed from type 'Str' to Pymatgen composition object for featurization (constructing features from composition object). Then the element fraction featurizers is used in modified dataset to extract features. (Fig. 6)

Now in Fig. 7 the elements which have 0 fraction through dataset is removed. The string values of Cells system and phase are given an Integer id to be able to fit in the model. Output of Figs. 6 and 7 is given in Fig. 8.

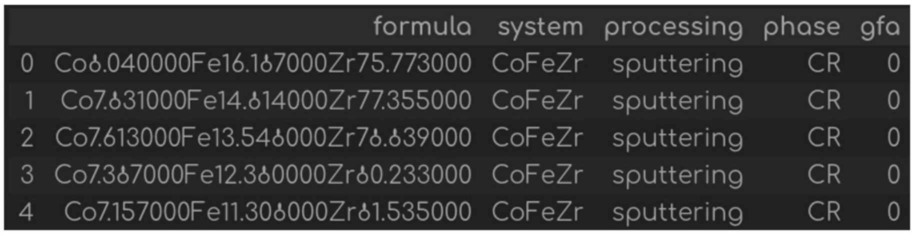
2.5. Splitting the test and train sets

Now when the Data model Is prepared it is necessary to split the dataset into target (gfa) and feature sets and split those sets further into Training and testing feature sets and their correspond-



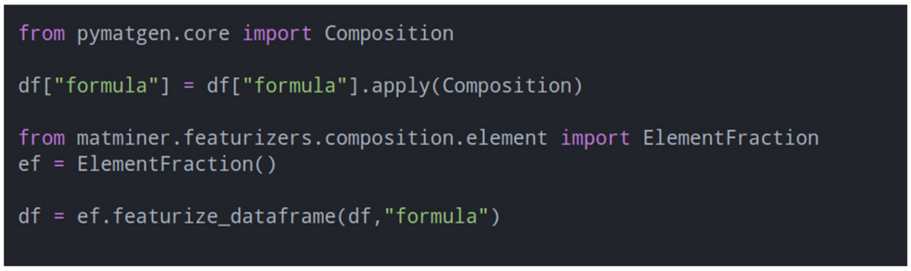
```
['boltztrap_mp', 'brgoch_superhard_training', 'castelli_perovskites', 'citrine_thermal_conductivity', 'dielectric_constant',
'double_perovskites_gap', 'double_perovskites_gap_lumo', 'elastic_tensor_2015', 'expt_formation_enthalpy',
'expt_formation_enthalpy_kingsbury', 'expt_gap', 'expt_gap_kingsbury', 'flla', 'glass_binary', 'glass_binary_v2',
'glass_ternary_hipt', 'glass_ternary_landolt', 'heusler_magnetic', 'jarvis_dft_2d', 'jarvis_dft_3d', 'jarvis_ml_dft_training',
'm2ax', 'matbench_dielectric', 'matbench_expt_gap', 'matbench_expt_is_metal', 'matbench_glass', 'matbench_jdft2d',
'matbench_log_gvrh', 'matbench_log_kvrh', 'matbench_mp_e_form', 'matbench_mp_gap', 'matbench_mp_is_metal',
'matbench_perovskites', 'matbench_phonons', 'matbench_steels', 'mp_all_20181018', 'mp_nostruct_20181018', 'phonon_dielectric_mp',
'piezoelectric_tensor', 'ricci_boltztrap_mp_tabular', 'steel_strength', 'wolverton_oxides']
```

Fig. 3. Output of Fig. 2.



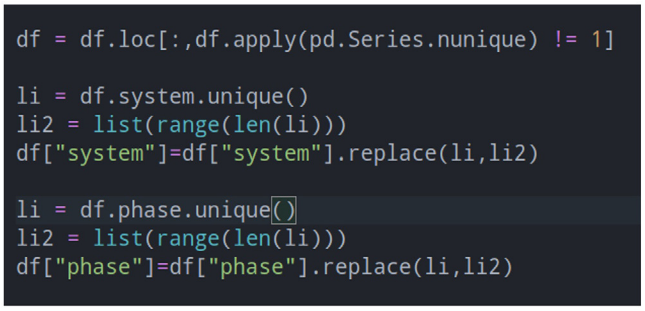
```
                              formula  system  processing  phase  gfa
0   Co8.040000Fe16.187000Zr75.773000   CoFeZr  sputtering     CR    0
1   Co7.831000Fe14.814000Zr77.355000   CoFeZr  sputtering     CR    0
2   Co7.613000Fe13.548000Zr78.839000   CoFeZr  sputtering     CR    0
3   Co7.387000Fe12.380000Zr80.233000   CoFeZr  sputtering     CR    0
4   Co7.157000Fe11.308000Zr81.535000   CoFeZr  sputtering     CR    0
```

Fig. 5. First 5 Entries of the Data Frame.



```
from pymatgen.core import Composition

df["formula"] = df["formula"].apply(Composition)

from matminer.featurizers.composition.element import ElementFraction
ef = ElementFraction()

df = ef.featurize_dataframe(df,"formula")
```

Fig. 6. Data Preparation – 1 (Featurizers).



```
df = df.loc[:,df.apply(pd.Series.nunique) != 1]

li = df.system.unique()
li2 = list(range(len(li)))
df["system"]=df["system"].replace(li,li2)

li = df.phase.unique()
li2 = list(range(len(li)))
df["phase"]=df["phase"].replace(li,li2)
```

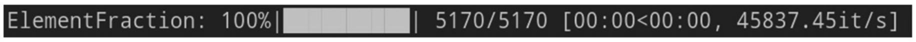
Fig. 7. Data Preparation – 2 (Cleaning).

ing target Set. In Fig. 9 the code splits the data sets into the Target and the feature Sets and Fig. 10 has its Output.

Now this data will be sliced further into training sets and testing sets 75% of the data will be used for training the model and the rest will be used for testing the model. Fig. 11 shows the code for this part.

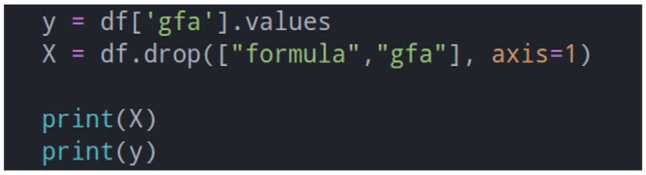
2.6. Fitting the model into the datasets

Now the data set is ready to be fitted into the model. The machine learning algorithm should be chosen Depending on the Model. The model used in current work is a binary-classification model. So, using a support vector machine classifier from Sklearn is enough to yield good accuracy in this model.



```
ElementFraction: 100%|██████████| 5170/5170 [00:00<00:00, 45837.45it/s]
```

Fig. 8. Output of Figs. 6 and 7.



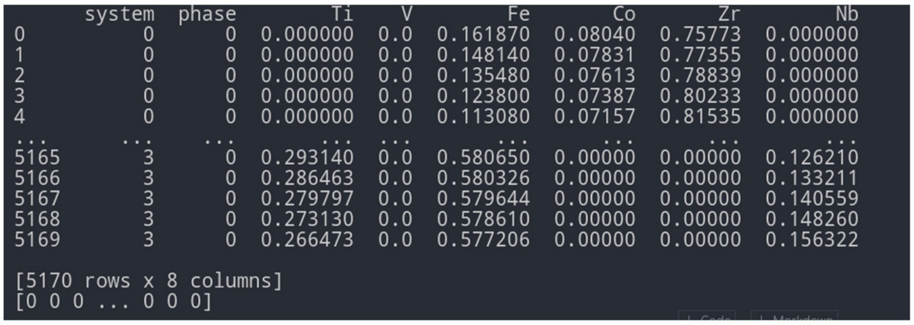
```
y = df['gfa'].values
X = df.drop(["formula","gfa"], axis=1)

print(X)
print(y)
```

Fig. 9. Splitting Dataset.

At first After the selection of the algorithm, the Sklearn machine learning libraries will be imported (Fig. 12). Then an instance of the support vector machine classifier is created and the training dataset is fitted into the Model (Fig. 13) and with the output (Fig. 14) it is ensured that the model is trained successfully.

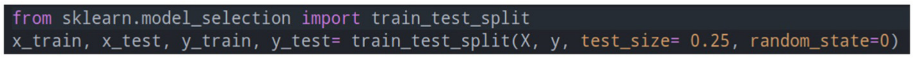
2.7. Results and discussion

So far, the Model is trained but its resultant accuracy is not calculated nor the model is tested. So, to get the results of training the first thing to do is to make a confusion matrix which will provide



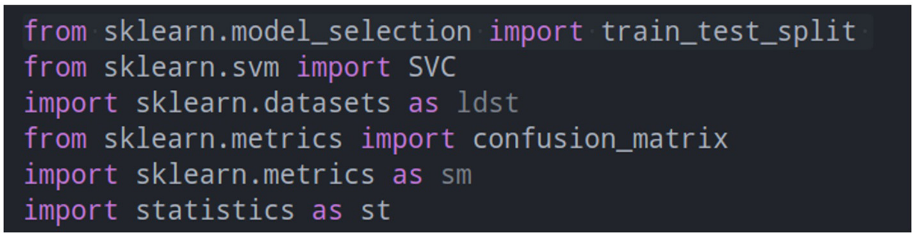
```
      system  phase        Ti    V        Fe       Co       Zr        Nb
0          0      0  0.000000  0.0  0.161870  0.08040  0.75773  0.000000
1          0      0  0.000000  0.0  0.148140  0.07831  0.77355  0.000000
2          0      0  0.000000  0.0  0.135480  0.07613  0.78839  0.000000
3          0      0  0.000000  0.0  0.123800  0.07387  0.80233  0.000000
4          0      0  0.000000  0.0  0.113080  0.07157  0.81535  0.000000
...      ...    ...       ...  ...       ...      ...      ...       ...
5165       3      0  0.293140  0.0  0.580650  0.00000  0.00000  0.126210
5166       3      0  0.286463  0.0  0.580326  0.00000  0.00000  0.133211
5167       3      0  0.279797  0.0  0.579644  0.00000  0.00000  0.140559
5168       3      0  0.273130  0.0  0.578610  0.00000  0.00000  0.148260
5169       3      0  0.266473  0.0  0.577206  0.00000  0.00000  0.156322

[5170 rows x 8 columns]
[0 0 0 ... 0 0 0]
```

Fig. 10. Output of Fig. 9.



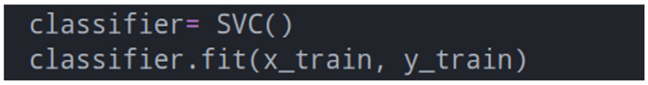
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(X, y, test_size= 0.25, random_state=0)
```

Fig. 11. Splitting Test Train Dataset.



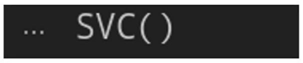
```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
import sklearn.datasets as ldst
from sklearn.metrics import confusion_matrix
import sklearn.metrics as sm
import statistics as st
```

Fig. 12. Importing Sklearn Sub Libraries.



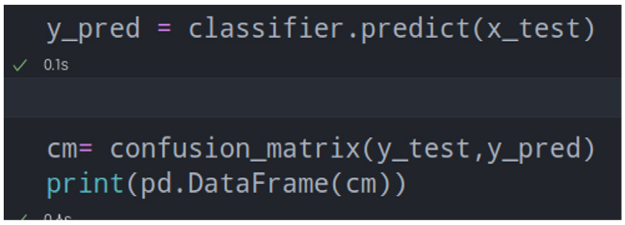
```
classifier= SVC()
classifier.fit(x_train, y_train)
```

Fig. 13. Training the Dataset.



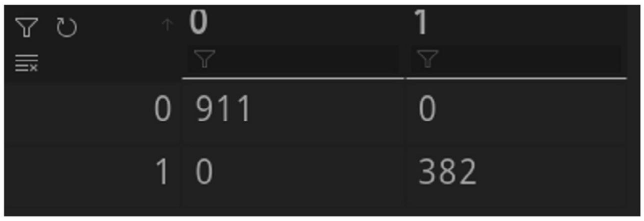
```
... SVC()
```

Fig. 14. Output of Fig. 13.



```
y_pred = classifier.predict(x_test)
0.1s

cm= confusion_matrix(y_test,y_pred)
print(pd.DataFrame(cm))
```

Fig. 15. Code for Generating Confusion Matrix.



```
     0     1
0  911     0
1    0   382
```

Fig. 16. Resultant Confusion Matrix.

us with the number of errors following code (Fig. 15) predicts the test target, and then the prediction is compared with the actual result to create the confusion matrix (Fig. 16).

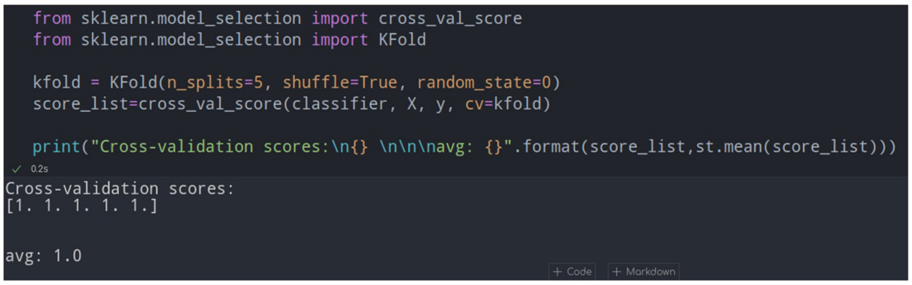
From Fig. 16, it can be seen that all the data are predicted successfully. Now the next step is to ensure that the model is not overfitted cross validation is needed on the model. In this work 5-fold and stratified 5-fold methods are shown in Fig. 17 and Fig. 18 with their outputs depicting 100% accuracy in the results.

For visualizing the data frame, a Heatmap is generated with the Seaborn and Matplotlib packages with the code shown below in Fig. 19.

The Heatmap generated from the dataset is given in the Fig. 20. In the same, all the features are equally important as the values of correlation are within ± 0.5 i.e., medium correlation, with the target 'GFA'.

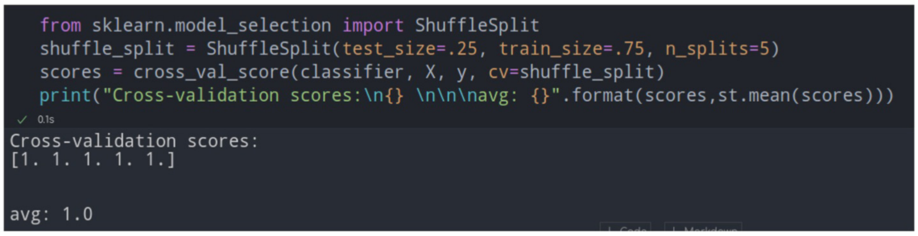
3. Conclusion

In the field of material science and machine learning prediction of a property of a material requires data which can be collected from the MAPI or matminer or pymatgen. Pymatgen and Matminer are packed with tools to represent molecules by their properties and the different featurizers to extract features from the formula, composition, structure or other properties of the molecules. combining those outputs with Sklearn, where all the machine learning algorithms are pre-implemented. A robust model can be trained to predict and estimate properties of materials. The same is done in this work, which shows the necessary steps to easily implement a machine learning model and to visualize the model and dataset, the data is plotted with the help of seaborn and matplotlib libraries. Current work also shows that how easily these complex concepts are implemented without a hassle in python. Programming in python also makes it really intuitive to the researcher. In



```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold

kfold = KFold(n_splits=5, shuffle=True, random_state=0)
score_list=cross_val_score(classifier, X, y, cv=kfold)

print("Cross-validation scores:\n{} \n\n\navg: {}".format(score_list,st.mean(score_list)))
0.2s
Cross-validation scores:
[1. 1. 1. 1. 1.]


avg: 1.0
```

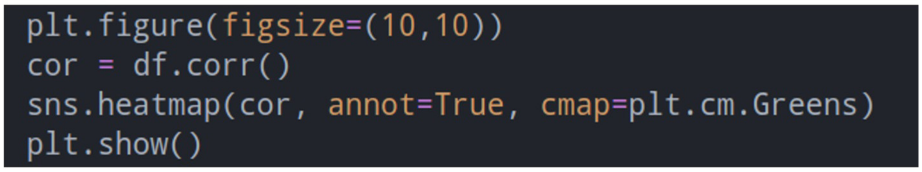
Fig. 17. 5-Fold Cross Validation Code & Results.



```
from sklearn.model_selection import ShuffleSplit
shuffle_split = ShuffleSplit(test_size=.25, train_size=.75, n_splits=5)
scores = cross_val_score(classifier, X, y, cv=shuffle_split)
print("Cross-validation scores:\n{} \n\n\navg: {}".format(scores,st.mean(scores)))
```

```
Cross-validation scores:
[1. 1. 1. 1. 1.]


avg: 1.0
```

Fig. 18. Stratified 5-Fold Cross Validation Code & Results.



```
plt.figure(figsize=(10,10))
cor = df.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Greens)
plt.show()
```

Fig. 19. Heatmap Generation.

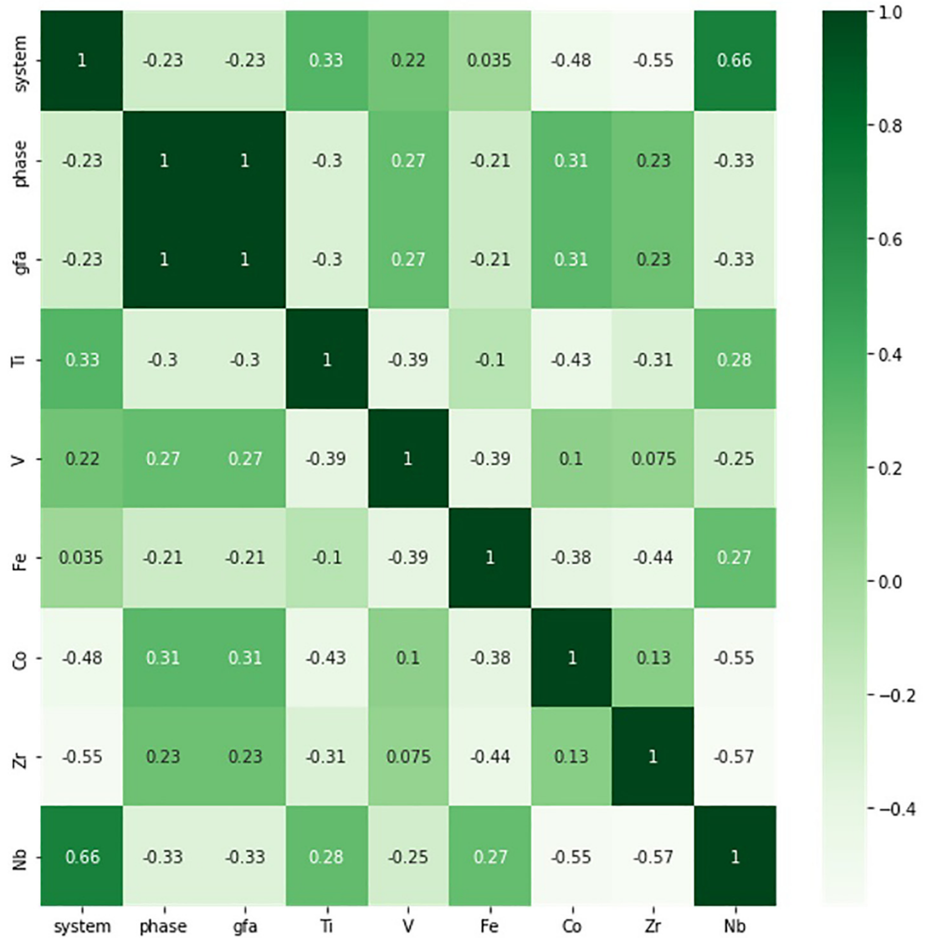


Fig. 20. Generated Heatmap.

the current focuses to help those researchers who either don't have sufficient lab to do the test on materials or are locked in their homes in the pandemic situation to continue their research remotely based on the machine predicted estimation. The current work will also help enthusiasts of different fields to work with materials, machine learning and Python.

CRediT authorship contribution statement

Arpan Kumar: Data curation, Conceptualization, Methodology, Writing – original draft, Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] https://pymatgen.org/introduction.html
[2] Z. Rong, D. Kitchaev, P. Canepa, W. Huang, G. Ceder, An efficient algorithm for finding the minimum energy path for cation migration in ionic materials, J. Chem. Phys. 145 (7) (2016) 74112, https://doi.org/10.1063/1.4960790.
[3] R. Tran, Z. Xu, B. Radhakrishnan, D. Winston, W. Sun, K.A. Persson, S.P. Ong, Surface energies of elemental crystals, Sci. Data 3 (2016), https://doi.org/10.1038/sdata.2016.80 160080.
[4] W. Sun, G. Ceder, Efficient creation and convergence of surface slabs, Surf. Sci. 617 (2013) 53–59, https://doi.org/10.1016/j.susc.2013.05.016.
[5] A. Jain, G. Hautier, C.J. Moore, S. Ping Ong, C.C. Fischer, T. Mueller, K.A. Persson, G. Ceder, A high-throughput infrastructure for density functional theory calculations, Comput. Mater. Sci. 50 (8) (2011) 2295–2310, https://doi.org/10.1016/j.commatsci.2011.02.023.
[6] S.P. Ong, L. Wang, B. Kang, G. Ceder, Li–Fe–P–O2 Phase Diagram from First Principles Calculations, Chem. Mater. 20 (5) (2008) 1798–1807, https://doi.org/10.1021/cm702327g.
[7] S.P. Ong, A. Jain, G. Hautier, B. Kang, G. Ceder, Thermal stabilities of delithiated olivine MPO4 (M=Fe, Mn) cathodes investigated using first principles calculations, Electrochem commun. 12 (3) (2010) 427–430, https://doi.org/10.1016/j.elecom.2010.01.010.
[8] A. Jain, G. Hautier, S.P. Ong, C.J. Moore, C.C. Fischer, K.A. Persson, G. Ceder, Formation enthalpies by mixing GGA and GGA+Ucalculations, Phys. Rev. B 84 (2011) 45115, https://doi.org/10.1103/PhysRevB.84.045115.
[9] A. Jain, S.P. Ong, G. Hautier, W. Chen, W.D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, K.A. Persson, Commentary: The Materials Project: A materials genome approach to accelerating materials innovation, APL Mater. 1 (1) (2013) 011002, https://doi.org/10.1063/1.4812323.
[10] S.P. Ong, S. Cholia, A. Jain, M. Brafman, D. Gunter, G. Ceder, K. Persson, The Materials Application Programming Interface (API): A simple, flexible and efficient API for materials data based on Representational State Transfer (REST) principles, Comput. Mater. Sci. 97 (2015) 209–215, https://doi.org/10.1016/j.commatsci.2014.10.037.
[11] G. Henkelman, A. Arnaldsson, H. Jónsson, A fast and robust algorithm for Bader decomposition of charge density, Comput. Mater. Sci. 36 (3) (2006) 354–360, https://doi.org/10.1016/j.commatsci.2005.04.010.
[12] T.F. Willems, C.H. Rycroft, M. Kazi, J.C. Meza, M. Haranczyk, Algorithms and tools for high-throughput geometry- based analysis of crystalline porous materials, Microporous and Mesoporous Materials 149 (1) (2012) 134–141, https://doi.org/10.1016/j.micromeso.2011.08.020.
[13] R.L. Martin, B. Smit, M. Haranczyk, Addressing Challenges of Identifying Geometrically Diverse Sets of Crystalline Porous Materials, J. Chem. Inf. Model. 52 (2) (2012) 308–318.
[14] https://materialsproject.org/open
[15] L. Ward, A. Dunn, A. Faghaninia, N.E.R. Zimmermann, S. Bajaj, Q. Wang, J.H. Montoya, J. Chen, K. Bystrom, M. Dylla, K. Chard, M. Asta, K. Persson, G.J. Snyder, I. Foster, A. Jain, Matminer: An open-source toolkit for materials data mining, Comput. Mater. Sci. 152 (2018) 60–69.
[16] G. Van Rossum, F.L. Drake, Python 3 Reference Manual, CreateSpace, Scotts Valley, CA, 2009.
[17] 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, E. Duchesnay, Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research. 12 (2011) 2825–2830.
[18] C.R. Harris, K.J. Millman, S.J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N.J. Smith, R. Kern, M. Picus, S. Hoyer, M.H. van Kerkwijk, M. Brett, A. Haldane, J.F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, T.E. Oliphant, Array programming with NumPy, Nature 585 (7825) (2020) 357–362, https://doi.org/10.1038/s41586-020-2649-2.
[19] J.D. Hunter, Matplotlib: A 2D graphics environment, Comput. Sci. Eng. 9 (3) (2007) 90–95, https://doi.org/10.1109/MCSE.2007.55.
[20] McKinney, W. et. al. Data structures for statistical computing in python, in: Proceedings of the 9th Python in Science Conference, Austin, TX, 2010: pp. 51–56.
[21] Waskom M., Botvinnik O., O'Kane D., Hobson P., Ostblom J., Lukauskas S., Gemperline D.C., Augspurger T., Halchenko Y., Cole J.B., Warmenhoven J., d. Ruiter J., Pye C., Hoyer S., Vanderplas J., Villalba S., Kunter G., Quintero E., Bachant P., Martin M., Meyer K., Miles A., Ram Y., Brunner T., Yarkoni T., Williams M.L., Evans C., Fitzgerald C., Brian, Qalieh A. https://hackingmaterials.lbl.gov/matminer/dataset_summary.html#glass-ternary-hipt
[22] X. Ying, An Overview of Overfitting and its Solutions, J. Phys. Conf. Ser. 1168 (2019) 022022, https://doi.org/10.1088/1742-6596/1168/2/022022.