MSE404-MAC

Computational Materials Science - Macroscale

Materials Band Gap Prediction Using Machine Learning Models

Saagar Kolachina

12/15/23

Abstract

Using python code, various machine learning models will be implemented as a tool to predict the band gap of electronic materials. Computationally, the band gap of a given structure could be calculated using Density Functional Theory (DFT). However, DFT calculations require a certain degree of convergence and more complicated structures may require larger sampled k-spaces and complex functionals that can be extremely costly computationally. As such, machine learning from compiled data of material characteristics serves as a promising and efficient approach towards band-gap prediction. Success, as measured by the efficacy of the ML model to predict band-gap will depend on how well we define the model as well as the extent to which the model is trained, however with a robust dataset and effective use of ML tools, such success can be achieved. If a model can be trained such that a band-gap could be predicted given a certain material structure to an achievable degree of accuracy, we can potentially circumvent or augment costly DFT calculations. Risks include creation of an ill-trained and ineffective model, poor pre-processing of datasets leading to models that are unsuccessful descriptors, as well as potential overfitting of data. Simulation time will ultimately be low as most time will be spent pre-processing data, sorting data, defining descriptors, and running each respective algorithm to output a trained model that can be used on a testing dataset. Folding all these tasks into data analysis, then most of the time will be spent in data analysis. Further analysis will be done on the trained models in order to determine MSE, R^2, and goodness of fit. In testing how well our trained model works with a testing dataset, we can produce these metrics and define a criterion for success based on relative performance with these calculated metrics.

Introduction

As a data-driven methodology towards material science started to develop, so did using machine learning to take advantage of these advancements in data. Machine learning, in a general sense, takes in data and can perform regression, classification, and clustering tasks to great accuracy. Moreover, machine learning models can take high dimensionality data as inputs as well as large datasets. Materials science, traditionally, is experiment-forward. Experimentation has served as an important benchmark in assessing the specific properties of a given material, yet can be limited by resources, time, and availability of material structures. [5]

Data-driven materials science takes advantage of high-throughput computing and data in order to use given and expansive material datasets as a way to train models that can predict the qualities of an unknown material of specific interest. Such an approach looks to bring about a more robust approach towards material discovery allowing for more rapid prediction of stable material structures, unknown materials, interatomic description of elements, and first-principle calculation [5].

Machine learning is an algorithmic process that uses these new and expansive materials data to aid in the prediction of these materials. Generally, a machine learning problem boils down to classification and regression. Classification uses machine learning to predict whether a certain element belongs to one group or another. Regression uses trained data in order to predict a target value of an unknown data point. Both are useful in materials science to help group materials together or predict an unknown property of a material given a certain structure. Moreover, given the computational cost needed for performing accurate DFT calculations, machine learning offers a more efficient alternative to obtaining these materials properties.

In this work, we aim to gain an overview of how machine learning is being applied to materials datasets in direct application. We seek to accomplish this by establishing and following a framework commonly used in these cases in order to build models that can predict the band-gap of a material given a formula or structure. As such, we are primarily interested in the regressive capabilities of machine learning though a similar problem could be tackled by a classification outlook - testing how well machine learning can predict if a given structure is a metal or nonmetal.

We follow a simple framework for development of these models. Given a dataset of known structures and computed band-gaps (either experimentally or through density functional theory), we encode descriptors for each material structure that can associate a given formula with a vector of computed values specific to that formula. There are many common descriptors that use material orientation, atomic character, electronegativity, space group, and other structure specific properties to correlate between formula and values that a model can evaluate. We then split between the descriptors and the target value (in this case the band-gap) and given a certain regressor model, train the model with a training subset of the total dataset. We then use the trained model to validate performance with a test dataset containing data the model has yet to see and evaluate how close the model prediction is to the actual value.

In this study, we will primarily be looking at the Random Forest model. The model uses a number of decision trees corresponding to an uncorrelated set of features, where each decision tree evaluates an estimate on its own. The estimate from each decision tree is then averaged to the final model quantity. [7].

Random forest is a useful algorithm as it is adjustable based on the input parameters (number of trees, depth of trees, and number of features sampled), and since it uses an ensemble average of uncorrelated trees, reduces the risk of overfitting while also lowering variance. [7]. For our simple case here, random forest looks promising given the large number of features to sample from and the single target value, making overall prediction less inherently complex.

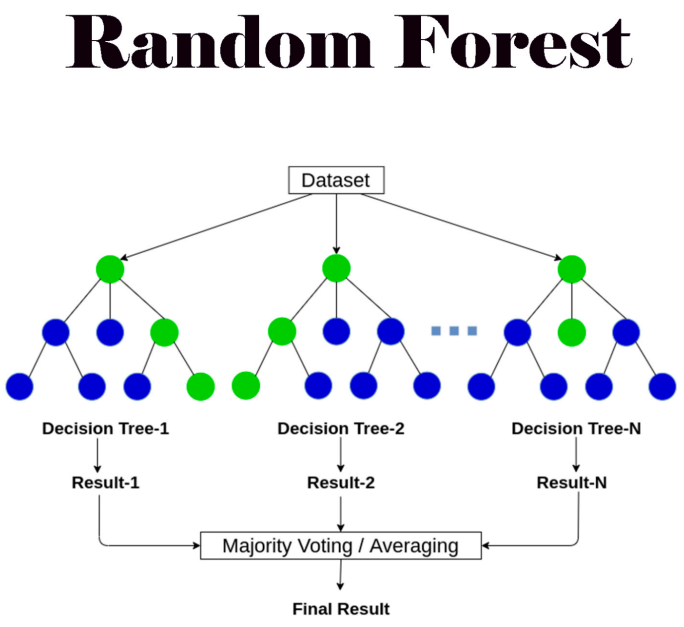


Figure 1. Random Forest Model Graphic [6]

Other models we will also briefly look at include the Gradient Boosting algorithm and the Support Vector Regression algorithm.

Overall, we want to look at the way in which a number of factors manifest and impact the machine learning algorithm and workflow in materials science. Specifically, we will look at two datasets - one from DFT and on experimental, look at how these datasets are featurized into descriptors, assess base model performance using our descriptors, and perform hyperparameter optimization to try to improve base model performance. The goal of this work is to provide a broad overview into machine learning applications in materials science with a direct result - training efficient models that can accurately predict the band-gap of unknown materials.

Methods

I. Random Forest Model and Optimization Using JARVIS-DFT3D Dataset

The JARVIS-DFT3D [4] dataset was chosen to featurize, train, and analyze towards generating our first model. The data was loaded in via Matminer [2] as a Pandas dataframe. Because we are interested in prediction over classification, we delete any rows in the dataframe where the band-gap equals 0, eliminating any data corresponding to a metal.

The JARVIS-DFT3D dataset has a column specifying structure and composition. First we use the composition of each sample in the dataset and featurize each data according to the ‘magpie’ preset [3]. Using the structure information of each sample, we then add ‘density’, ‘vpa’, and ‘packing fraction’ descriptors to our list of features using Matminer structure featurizers.

Following this procedure, we have a total of 134 descriptors for each material sample. Before we can start model regression and prediction, we need to separate X and y data and split the data into training and testing sets. We remove the ‘composition’, ‘structure’, and ‘mpid’ columns from the data and assign the remaining descriptors to the ‘X’ array. We then assign the ‘y’ array to the column of band-gap data describing our target. We then split the data in a 90/10 ratio, meaning the model is trained over 90% of the data and then tested over 10%. This split remains consistent for every model in this work.

First we use the Random Forest Regressor from Scikit-Learn using parameters of n_estimators = 150. We fit the model to our training data and use the model to predict from our testing set. We then calculate MSE and R^2 using scikit-learn metrics and our actual y data.

From this we obtain a preliminary model that we then do further testing and optimization on. Before optimizing, the same training and testing procedure was done using a Gradient Boosting regressor and the same sample metrics were obtained and compared.

Continuing with our Random Forest model, we then create another model using n_estimators = 250. From this, we then proceed with more rigorous hyperparameter optimization. A grid-search model was created that uses the Shuffle Split method with a 10% test size and a parameter grid specifying:

```
param_grid = {
   'n_estimators': [100, 200, 300, 400],
   'max_depth': [None, 10, 25, 50],
   'max_features' : range(8,15) }
```

Running this search, an optimal model was obtained and the performance statistics of the model was reported.

Finally, using scikit-learn functions, a 10-fold KFold cross validation procedure was run on this determined optimal model and the mean and standard deviation of the CV score at each fold was reported.

II. Random Forest Model and Optimization Using Dataset of Experimental Bandgaps

As part of this study, we are interested in the role dataset selection can play in the overall machine learning scheme. To accomplish this, a similar procedure to Method I was conducted using the ‘expt_gap’ dataset.

The dataset was imported into a pandas DataFrame using matminer and all samples corresponding to metals were removed. We end up with 3896 data points. Unlike the Jarvis-DFT dataset, we only have the formula of the semiconductor and the corresponding experimentally reported band-gap. As such, we use Matminer StrToComposition() to convert formulas to readable composition values. We then use composition featurizers set to the ‘magpie’ [3] preset to generate magpie composition descriptors from these composition descriptions. We end with 132 columns of descriptors. We then remove any rows where a value reads ‘NaN’ to avoid problems with the regressors.

We remove the ‘formula’ and ‘composition’ columns and assign remaining descriptors to the X_ex data. The experimental band-gap data was then assigned as y_ex data. The data was then split 90/10 between training and testing and from this split an initial Random Forest regressor model using n_estimators = 300 was trained and sample metrics recorded.

We then once again perform a grid search with a Shuffle Split CV with 10% test data and the following parameter grid:

```
param_grid = {
   'n_estimators': [200, 300, 400],
   'max_depth': [None, 10, 25, 50],
   'max_features' : range(8,15) }
```

The search was conducted and an optimal model was found, the features noted, and the metrics recorded.

III. Comparing Models using Experimental Band Gap Dataset

Finally, we have been primarily focused on the Random Forest Regressor in creating these models. However, there are many potential models to use and model selection remains a critical factor in these machine learning workflows.

As such, using the same ‘expt_gap’ dataset, with the same set of features as listed above, an algorithm was implemented directly from Wang et al [1] that created a model for the data using the Dummy, Ridge, AdaBoost, GradientBoosting, Random Forest, ExtraTrees, SVR, LinearSVR, and k-nearest neighbors models and recorded RMSE, R^2, and MSE for the training and testing data for each model alongside time needed to fit the data, launching a discussion of model performance and relative costs. The X and y data in this case are normalized and scaled.

Results and Discussion

I. Jarvis-DFT3D

The Jarvis DFT dataset computes the 3D materials properties of 75993 materials. Figure 2 plots the distribution of the OPTB88 Band-Gap values for this dataset with all the metals removed. We see that the majority of the data is distributed from 0-2.5 eV in a very left-tailed distribution.

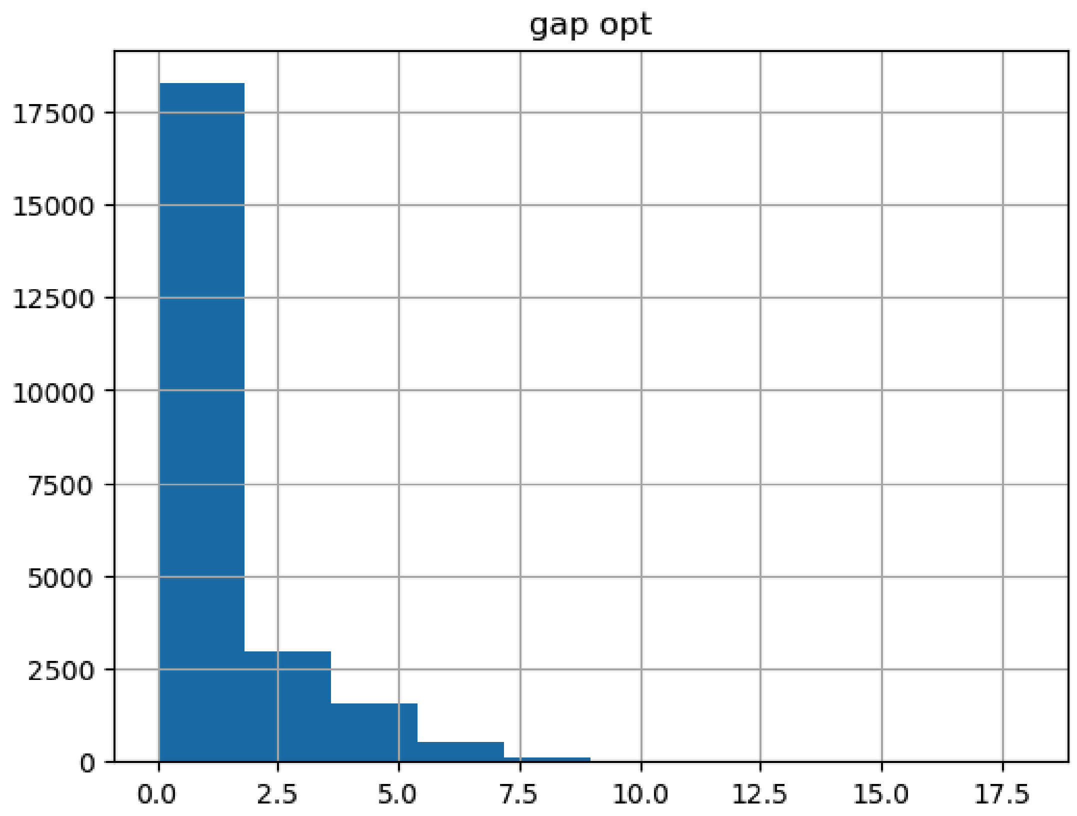


Figure 2. Histogram describing data distribution of the OPTB88 BandGap values of the Jarvis-DFT3D Dataset.

Creating a base model using the Random Forest regressor with n_estimators = 150. We obtain the following testing MSE and R2 values.

Testing MSE:	0.3459964660926981
Testing R2	0.867766902254969

Figure 3 depicts the parity plot for this model.

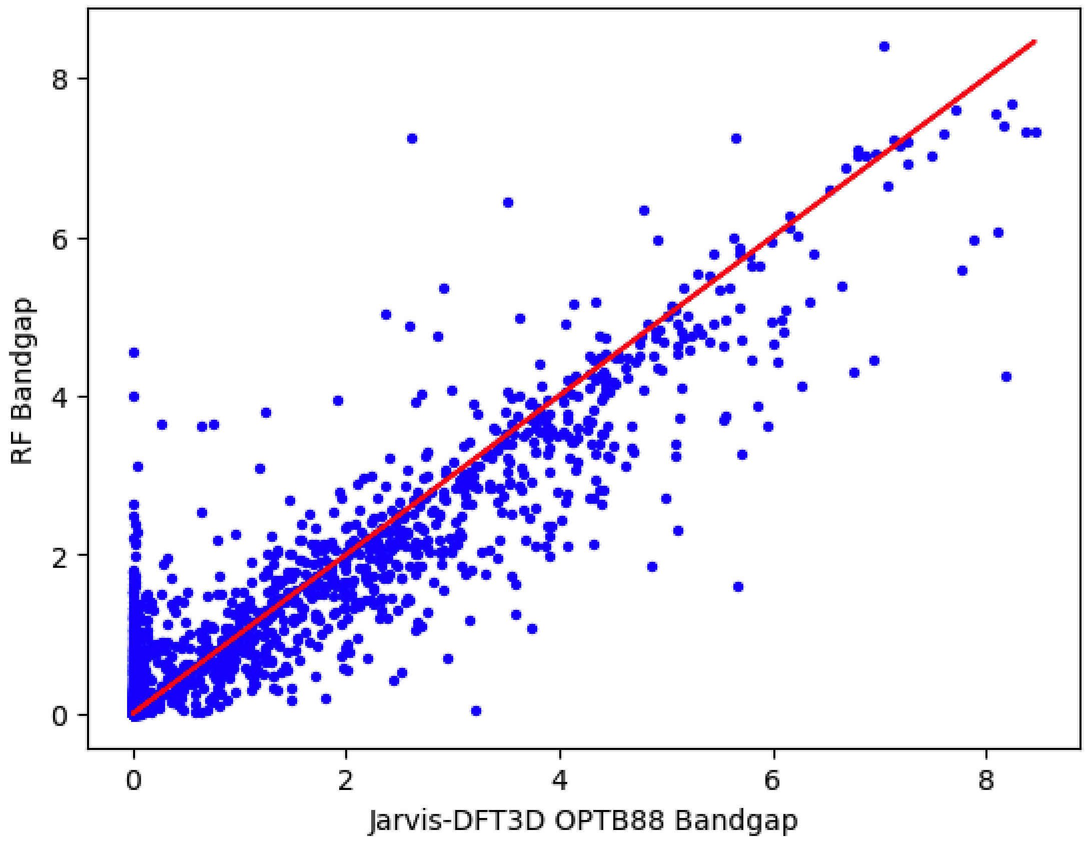


Figure 3. Parity Plot for RF Model (n_estimators = 150)

Relatively, the R2 above 0.8 indicates a good fit. We also have relatively low mean-squared error (MSE). So we determine that as a base model this functions well, though we want to optimize the parameters of the model in order to see if we can create a model that results in lower MSE with higher R2. One possible avenue is to explore other regressors. We perform a more extensive study of this in Method III though for this dataset, the Gradient Boosting model was performed on the same training/testing data with the same parameters. We obtain the following statistics:

Testing MSE:	0.5736713631372953
Testing R2	0.7807539993343877

Compared to our random forest model, we see noticeably higher MSE with a lower R2 indicating worse fit/prediction. This step is informative as it allows us to perform greater analysis on our Random Forest model for optimization of our model. Figure 4 depicts this parity plot.

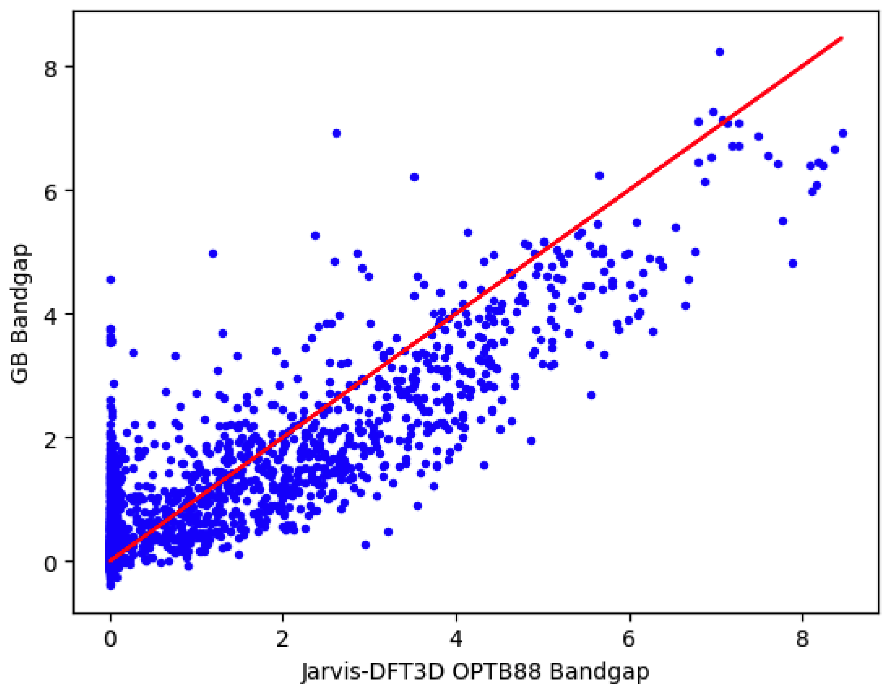


Figure 4. Parity Plot for GB Model

One parameter we can optimize is the number of estimators which describes the number of decision trees used in the models. We expect that a higher number of estimators can result in better performance though using too many parameters may result in converged statistics with greater computational cost. As a preliminary step, we run the same model using a greater number of parameters and result in the following statistics.

Testing MSE:	0.34594961851478934
Testing R2	0.8677848064850864

Compared to our original model, we observe that the R2 is only incrementally higher. Though we do notice a slight increase in model performance, the change in performance may not justify the higher cost. We want to then look at other features to see if tuning them can lead to greater model performance. This procedure is generally known as hyperparameter optimization.

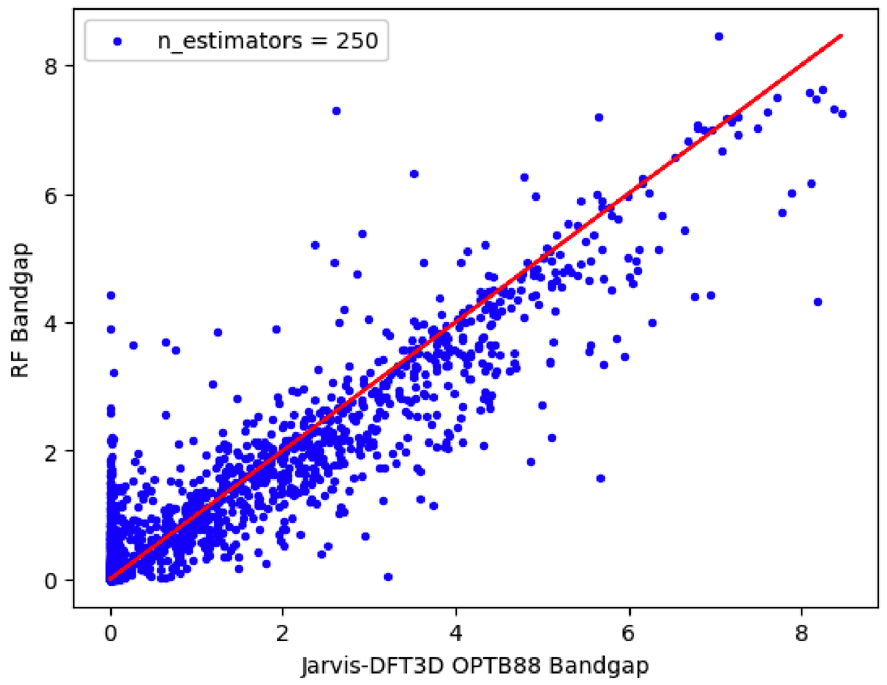


Figure 5. Parity Plot for RF Model with increased estimators.

As an initial step, we can hyper parameterize the number of features used. As a default, the square root of the total number of features is used but we can adjust this parameter for each model. As such, a grid-search is employed on this parameter ranging from 8 to 15 features (capped at the sqrt of the total number which may be less than 15). The MSE of each model using set n_estimators = 175 is calculated and the model with the lowest MSE is selected. This procedure is adapted from the "formation_e" example from the Matminer package. Figure 6 plots MSE as a function of the model highlighting the model with the lowest at 11 features with an MSE of around 0.316.

Using these parameters to train a new model, we obtain the following statistics:
Testing MSE: 0.34364858563402884

Testing R2 0.8686642163509507

We see a noticeable improvement in both the MSE and the R2 as compared to solely increasing the n_estimators indicating a degree of success with this specific parameter optimization.

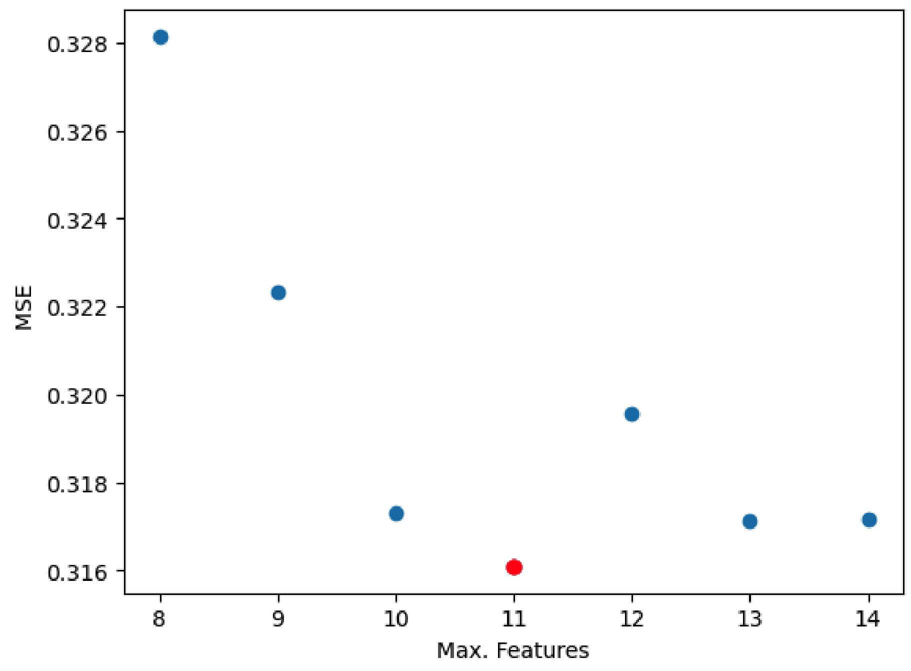


Figure 6. Example of Hyperparameter Optimization of Number of Features

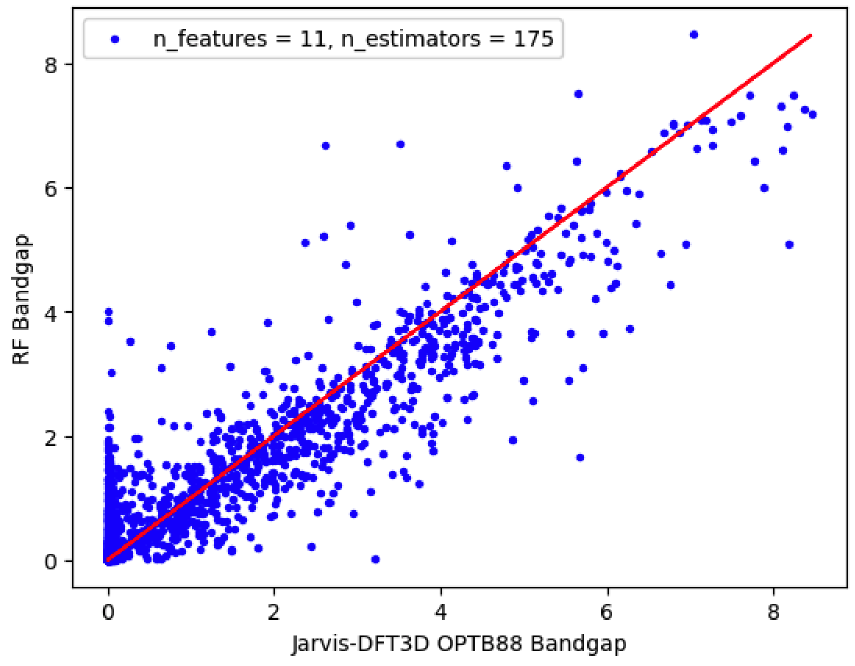
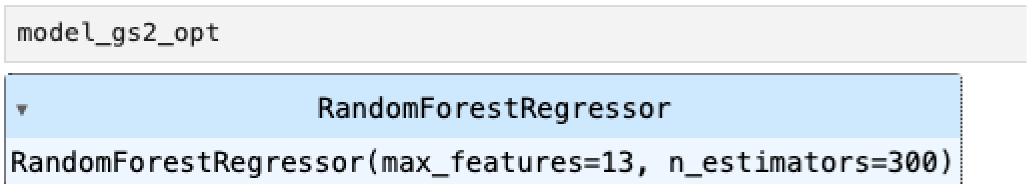


Figure 7. Parity Plot With Optimized Features

Seeing an improvement in our model performance with simple hyperparameter optimization, we want to see if a more expansive grid-search method can result in a better performing model. Using a grid-search with the parameter grid as defined in Method I, we obtain the following optimized model and statistics:



```
model_gs2_opt
```

```
▾ RandomForestRegressor
RandomForestRegressor(max_features=13, n_estimators=300)
```

Testing MSE: 0.34388555453224456

Testing R2 0.8685736514621408

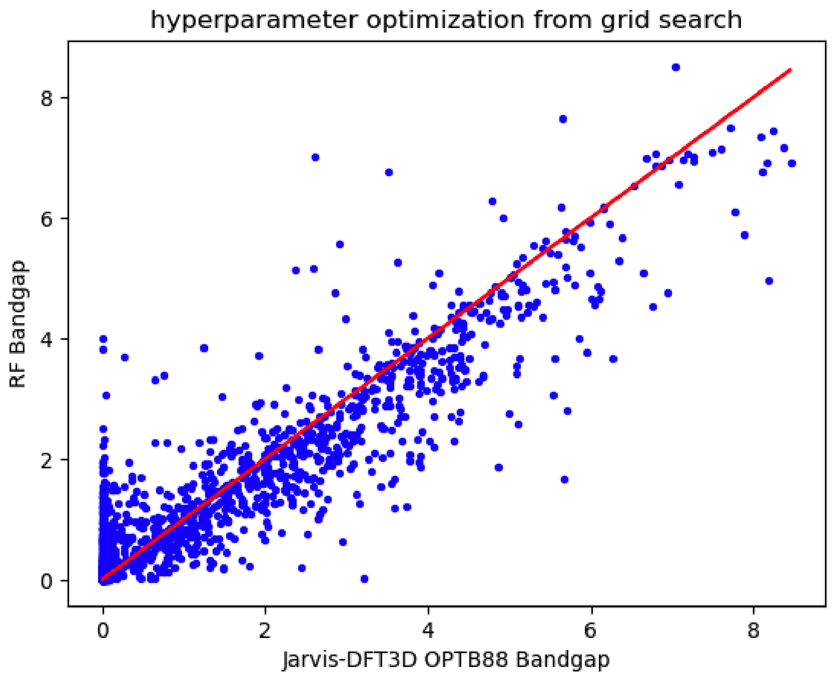


Figure 8. Parity Plot of Optimized Model from GridSearchCV

The statistics are certainly an improvement to our original RF model, though interestingly compared to our simple grid-search for the number of features we obtain a model here that performs similarly if slightly worse. This suggests that our more expansive model search, while effective and successful may be more work than necessary and more specific and intentful parameterization is more important. Such a distinction is a common balancing act when designing these machine learning models.

Finally, as a last test of model performance, we employ a 10-fold KFold cross validation and measure the accuracy at each fold. We result in the following mean accuracy and standard deviation of the metrics of each fold.

Cross Validation Mean and Accuracy:
0.8475158101271048
0.017851447345007952

We have a relatively high accuracy with low standard deviation helps reinforce model performance. With the issue of overfitting, since we still obtain high model performance between the training and testing sets alongside low variance is the CV, we can rationalize that our model is not overfit to the data though future analysis may be required.

Model II. Experimentally Determined Band-Gap ML Regression

Figure 9 shows the distribution of experimentally determined band-gaps. We notice that, as compared to the JARVIS dataset, there are fewer data points and we have more data centered around 2-4 eV. Here we notice how inherently dataset selection matters as an initial step. Comparing two datasets that have the same base target we notice differences in data distribution and ranges.

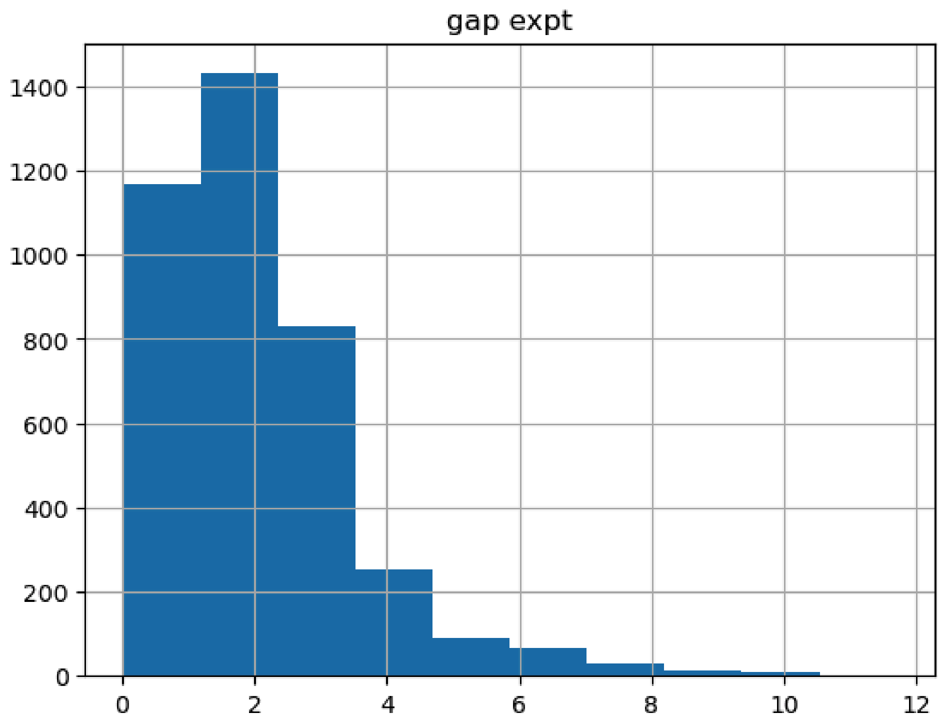


Figure 9. Distribution of experimentally determined band-gaps.

Using the initial models detailed in Method II, we obtain the following testing statistics:

Testing MSE: 0.2114629906188201

Testing R2   0.8857442771810636

We report an even lower MSE and higher R2 values indicating that the Random Forest regressor is able to adequately predict testing band-gaps after being trained on the training set. We also notice better model performance as compared to the JARVIS dataset which can be rationalized by fewer variance and disparity between data points and fewer data points closer to 0 as we saw the JARVIS data points struggling to predict these low band-gap values from each individual parity plot. Qualitatively, we see less of that with this model.

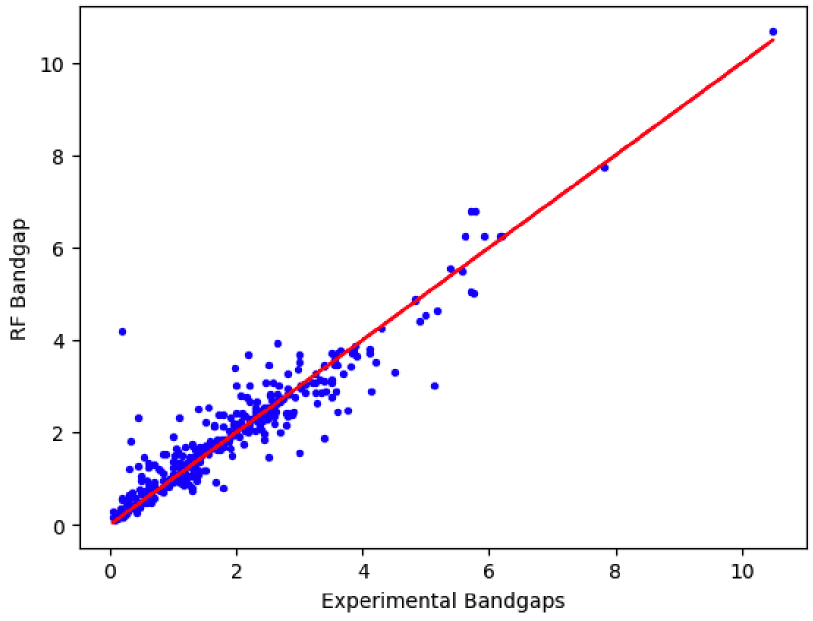
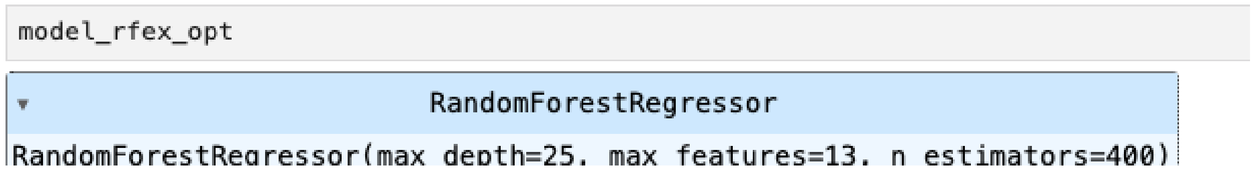


Figure 10. Parity plot of initial experimental RF model.

We still want to perform a general extensive hyperparameter optimization on these properties to benchmark improvements to model performance. From our grid search detailed in Method II, we obtain the following optimized model with statistics:

Testing MSE: 0.21019056193035737

Testing R2 0.8864317840545379



```
model_rfex_opt
```

```
RandomForestRegressor
RandomForestRegressor(max_depth=25, max_features=13, n_estimators=400)
```

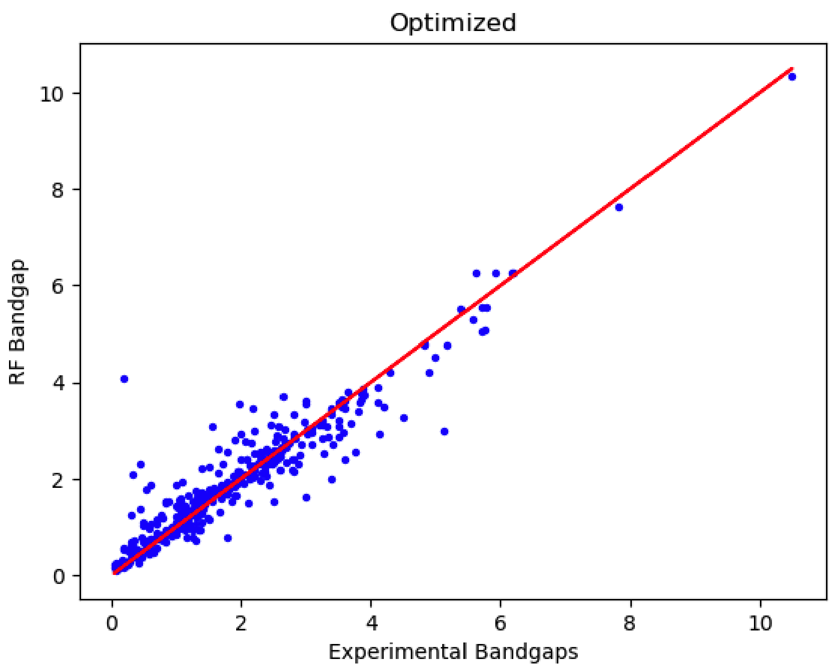
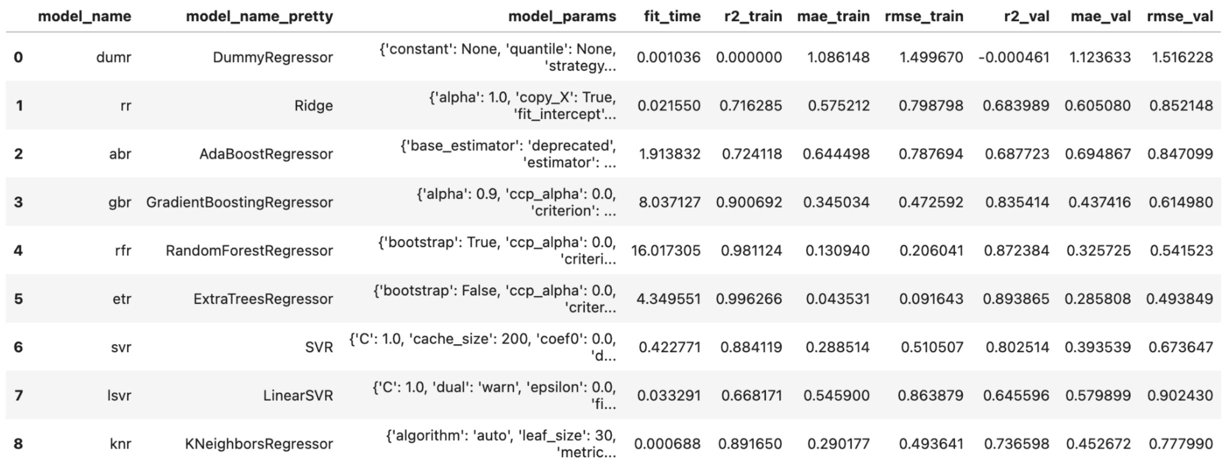


Figure 11. Parity plot for optimized experimental model.

We do indicate a drop in MSE and slightly higher R2 (though again, in a relatively minute way) indicating that our hyperparameter optimization did result in a better performing model. With n_estimators at 400 it is worth training a similar model with lower n_estimators to see if that has any impact on model performance.

III.

Performing the procedure indicated in Method III using code from Wang et al[1], we obtain the following statistics.



	model_name	model_name_pretty	model_params	fit_time	r2_train	mae_train	rmse_train	r2_val	mae_val	rmse_val
0	dumr	DummyRegressor	{'constant': None, 'quantile': None, 'strategy...	0.001036	0.000000	1.086148	1.499670	-0.000461	1.123633	1.516228
1	rr	Ridge	{'alpha': 1.0, 'copy_X': True, 'fit_intercept'...	0.021550	0.716285	0.575212	0.798798	0.683989	0.605080	0.852148
2	abr	AdaBoostRegressor	{'base_estimator': 'deprecated', 'estimator': ...	1.913832	0.724118	0.644498	0.787694	0.687723	0.694867	0.847099
3	gbr	GradientBoostingRegressor	{'alpha': 0.9, 'ccp_alpha': 0.0, 'criterion': ...	8.037127	0.900692	0.345034	0.472592	0.835414	0.437416	0.614980
4	rfr	RandomForestRegressor	{'bootstrap': True, 'ccp_alpha': 0.0, 'criteri...	16.017305	0.981124	0.130940	0.206041	0.872384	0.325725	0.541523
5	etr	ExtraTreesRegressor	{'bootstrap': False, 'ccp_alpha': 0.0, 'criter...	4.349551	0.996266	0.043531	0.091643	0.893865	0.285808	0.493849
6	svr	SVR	{'C': 1.0, 'cache_size': 200, 'coef0': 0.0, 'd...	0.422771	0.884119	0.288514	0.510507	0.802514	0.393539	0.673647
7	lsvr	LinearSVR	{'C': 1.0, 'dual': 'warn', 'epsilon': 0.0, 'fi...	0.033291	0.668171	0.545900	0.863879	0.645596	0.579899	0.902430
8	knr	KNeighborsRegressor	{'algorithm': 'auto', 'leaf_size': 30, 'metric...	0.000688	0.891650	0.290177	0.493641	0.736598	0.452672	0.777990

Figure 12. Dataframe of model name, time it took to fit, and training/testing RMSE, MAE, and R2

From Figure 12, we see that the ExtraTreesRegressor performs the best resulting in the highest R2 and lowest RMSE, followed by the RandomForestRegressor. The Gradient Boosting also reports a higher R2 compared to the SVR and KNN models.

The dummy regressor performing by far the worst indicates that our model is being fit in likely the correct way. Moreover, we see that the Gradient Boosting and Random Forest models take the longest to fit - orders above models like SVR and Linear SVR.

What this study illuminates is how, for different types of datasets and different complexities of predictiveness, appropriate models should be chosen in order to get the best results. While Random Forest is useful for these target based regression prediction problems, models like SVR could be more useful in cases with more complex or nonlinear data.

Conclusions and Future Study

Python along with Scikit-Learn and Matminer were used in order to investigate and overview ways to use Machine Learning to predict material properties. Dataset selection was studied using both a general Jarvis DFT dataset of a larger size containing many properties against a smaller dataset composed of experimentally determined bandgaps. Descriptors for this dataset were generated and the datasets were fit to Random Forest Regression Models. These models then underwent iterative hyperparameter optimization outlining how critical model selection and optimization of these parameters can be in the overall efficacy of the model. Finally the experimental band-gap dataset was used to benchmark performance and time for many different machine learning models to better understand how well some models perform against others given a specific dataset and how organization and complexity of data can impact model selection.

This work was intended to provide a general and introductory overview into how machine learning is being applied to material science based prediction and aiding in the future of materials discovery. As such we look at defining a general workflow for designing these models as well as the role material composition and structure descriptors, parameter optimization, and model selection play in tuning these models. Further study could be done on more complex models such as Graph Convolution Neural Networks as well as more refined workflows for better model creation. This includes more rigorous cross-validation, splitting data into training/testing/validation sets, and studying more about how these descriptors are generated based on material structure, phase, and composition.

Code Availability

Code used in analysis and in Method I, II, and III are referenced, adapted, or directly applied from Matminer example Jupyter notebooks [2] as well as example notebooks created by Wang et al [1]. Personal jupyter notebooks used to conduct all analysis and plot figures are available in the link below and can be sent directly on request via email.

https://drive.google.com/drive/folders/1ThL_h8TSWq-kMq5Htx81wJzmNq8cWBcj?usp=sharing

References

[1] Wang, A.; Murdock, R.; Kauwe, S. K.; Oliynyk, A. O.; Gurlo, A.; Brgoch, J.; Persson, K. A.; Sparks, T. D. Machine Learning for Materials Scientists: An Introductory Guide toward Best Practices. *Chemistry of Materials* 2020, *32* (12), 4954–4965. https://doi.org/10.1021/acs.chemmater.0c01907.
[2] Hackingmaterials. *matminer_examples/matminer_examples/machine_learning-nb at main · hackingmaterials/matminer_examples*. GitHub.
https://github.com/hackingmaterials/matminer_examples/tree/main/matminer_examples/machine_learning-nb.
[3] *Bitbucket*. Bitbucket. https://bitbucket.org/wolverton/magpie/src/master/.
[4] *Table of Datasets — matminer 0.9.0 documentation*.
https://hackingmaterials.lbl.gov/matminer/dataset_summary.html.
[5] Schmidt, J.; Marques, M. R. G.; Botti, S.; Marques, M. a. L. Recent Advances and Applications of Machine Learning in Solid-State Materials Science. *Npj Computational Materials* 2019, *5* (1). https://doi.org/10.1038/s41524-019-0221-0.
[6] - *Anas Brital | Random Forest algorithm explained* .
https://anasbrital98.github.io/blog/2021/Random-Forest/.
[7] *What is Random Forest? | IBM*. https://www.ibm.com/topics/random-forest.