Machine Learning for Data Driven Prediction of Electronic Band Gaps of Inorganic Compounds

Palak Mahendru

The University of British Columbia

Introduction

The electronic band gap (Band Gap - Energy Education, 2015) is the energy difference between the top of the valence band and the bottom of the conduction band of a solid. Accurate prediction of electronic band gap (Eg) for an inorganic compound is critical since it directly determines the electronic as well as optical properties of a material of considerable importance in many applications like solar cells, light-emitting diodes, and transistors. Precise band gap knowledge enables efficient design and development of new materials with desired functionalities. Experimentally measuring the band gaps of a vast array of materials can be both expensive and time-consuming. Hence, reliable computational methods to predict band gaps efficiently are highly valuable.

This project addresses the scientific question of how effectively machine learning (ML) algorithms can predict the experimental band gaps of materials based solely on compositional and structural features. Specifically, we investigate the predictive performance of various ML models, including Random Forests, Gradient Boosting, and a two-step approach (Zhuo et al., 2018) combining Support vector Classification (SVC) and Support Vector Regression (SVR), on a dataset initially explored by Zhuo and others.

Zhuo et al. previously demonstrated that ML techniques could predict band gaps using compositional data with significant accuracy, leveraging a combination of SVC and SVR. They used compositional features derived from material properties such as stoichiometry, elemental properties (through Magpie library), and valence orbital characteristics. Building up on their methodology, this project aims to further examine the performance of these ML predictions by employing rigorous validation techniques such as cross-validation and hyperparameter optimization.

The dataset utilized comprises experimentally measured band gaps of various inorganic compounds. Our primary goal is to evaluate and compare the accuracy and reliability of different ML approaches in predicting these band gaps, explore the interpretability of model predictions through feature importance analysis, and provide insights into which compositional attributes most significantly influence the band gap. Ultimately, this project seeks to not only validate previous findings but also to enhance the predictive capabilities of ML models in materials science, thereby facilitating faster and more cost-effective discovery of new functional materials.

Methods

In this project, I employed a combination of ML algorithms, starting with SVM and SVR tested by Zhuo et al., followed by Random Forest (RF) and Gradient Boosting Regression (GBR) methods. The SVM classifier was first used to distinguish between metallic and non-metallic compounds. I used a threshold of 0.1 eV on the band gap values; in line with the argument that materials with near-zero band gaps exhibit metallic conductivity. Compounds with experimental band gap values below this threshold were considered metals, while those above were treated as non-metals. This binary classification target

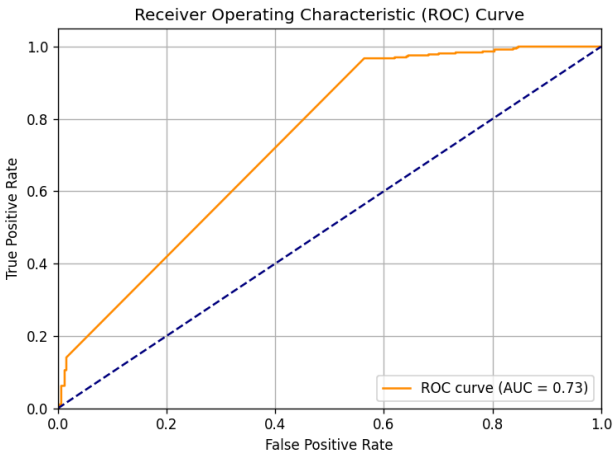
was used to train an SVM classifier based on features describing each compound's composition, extracted using materials science-specific featurization methods. For input features, I used compositional descriptors generated via the *matminer* library. Specifically, I applied the Magpie preset within the ElementProperty featurizer to get elemental statistics (average atomic number, atomic radius, electronegativity, etc). These were also combined with Stoichiometry and ValenceOrbital descriptors. After experimenting a little, I found that using only Magpie-derived features and scaling our composition data led to a substantial performance improvement in the classification part.

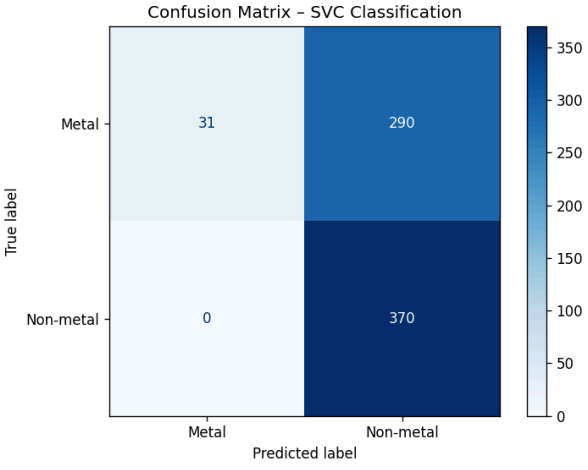
Compounds classified as nonmetals underwent further quantitative analysis through SVR to predict their band gap values. A Receiver Operating Characteristic (ROC) curve and confusion matrix was used for data interpretation. During data cleaning, I also removed duplicated and all-NaN columns and imputed missing values with feature-wise means. Removal of the target variable from the feature matrix was also ensured.

For an improved predictive capability, Random Forest and Gradient Boosting regression models were developed using hyperparameter optimization and evaluated using cross-validation to ensure robustness. The hyperparameters included the number of estimators (trees) and maximum features to split at each node, optimized to achieve the best balance between performance and computational efficiency. Additionally, hyperparameter optimization via GridSearchCV was conducted for the GBR model to tune n_estimators, max_depth, and learning_rate. Lastly, for data interpretation and model assessment of both our tree-based models, various plots including learning curves, residual plots, and QQ plots were created.

Results

The classification performance of the SVM model for distinguishing metals from nonmetals yielded an accuracy of ~58%, demonstrating moderate differentiation capability. The ROC analysis produced an Area Under Curve (AUC) of 0.73, suggesting a reasonable but not significant discriminating power of the model. The confusion matrix highlights that while the SVM model detects many true non-metals, there isn't a notable number of false negatives and false positives, limiting precision. For band gap predictions, our initial approach with SVC + SVR achieved a mean absolute error (MAE) of 0.945 eV and root mean squared error (RMSE) of 1.669 eV, indicating moderate predictive ability.





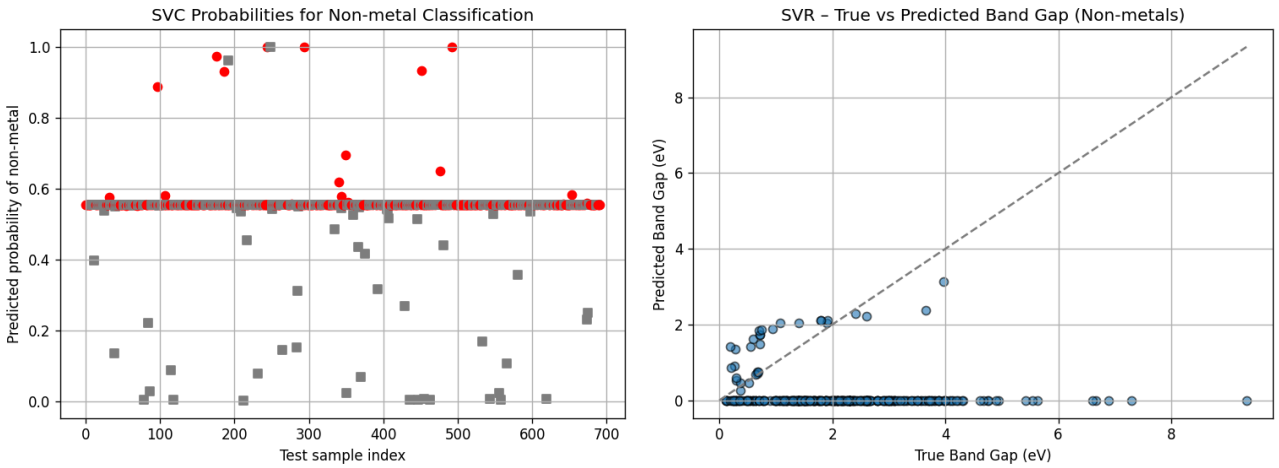
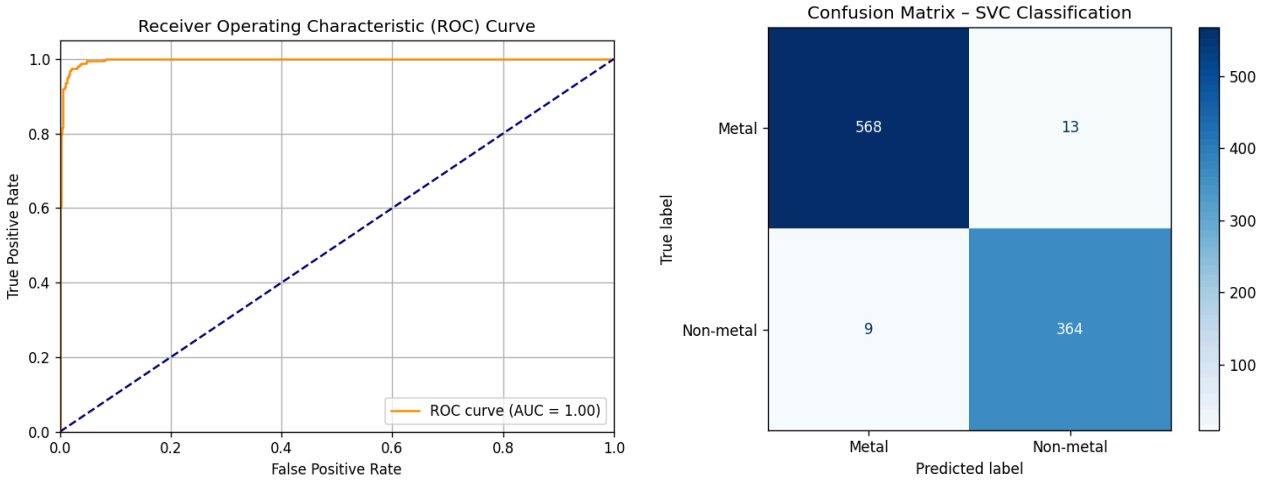


Figure 1. Evaluation of SVC→SVR Hybrid Pipeline Performance. Top left: ROC curve for SVC classifier with AUC = 0.73, indicating moderate classification performance in distinguishing metals from nonmetals. Top right: Confusion matrix for SVC model, showing correct identification of non-metals but a large number of false positives for metals. Bottom left: Predicted probability plot from SVC model; non-metals (red circles) are clustered near high predicted probabilities, while metals (gray squares) show wider dispersion. Bottom right: SVR-predicted vs. true band gap values for non-metals, with noticeable underprediction and deviation from the ideal dashed parity line.

While these results were consistent with the original methodology described by Zhuo et al. (2018), I was unable to fully replicate their higher regression performance metrics. This discrepancy likely arose due to differences in data preprocessing, feature selection, or handling missing values. After adjusting to account for only Magpie-derived features and scaling the feature matrix, I was able to surpass the results in the literature. The new SVM model yielded an accuracy of 98%, with the ROC analysis supporting an AUC of 1.00, which represents perfect separation. New SVR model achieved very strong predictive performance, with a low MAE of 0.055 eV and RMSE of 0.124 eV.



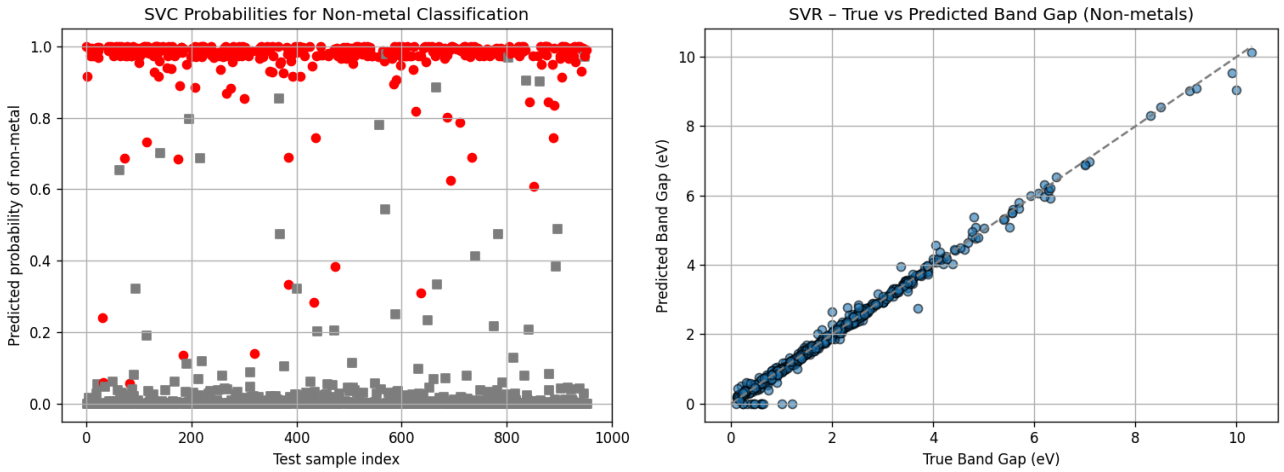


Figure 1. Evaluation of new SVC→SVR Pipeline Performance. Top left: ROC curve for SVC classifier with AUC = 1.00, indicating perfect classification performance in distinguishing metals from nonmetals. Top right: Confusion matrix for SVC model, showing strong identification of nonmetals and metals. Bottom left: Predicted probability plot from SVC model; confident separation of non-metals (red circles) clustered near high predicted probabilities, from metals (gray squares) clustered at the bottom. Bottom right: SVR-predicted vs. true band gap values for nonmetals closely align with the parity line and indicate excellent regression accuracy (98%)..

The Random Forest model significantly improved prediction accuracy as compared to our initial SVC + SVR approach and got a much lower MAE of 0.555 eV, RMSE of 0.933 eV, and R^2 of 0.583, after hyperparameter tuning and cross-validation. Learning curve analysis further validated the model's behavior: as training size increased, the MAE dropped sharply before gradually leveling off, suggesting the model benefits from additional data but is approaching performance saturation. The curve indicates limited overfitting and consistent generalization. Residual diagnostic plots revealed a roughly normal distribution of errors centered around zero.

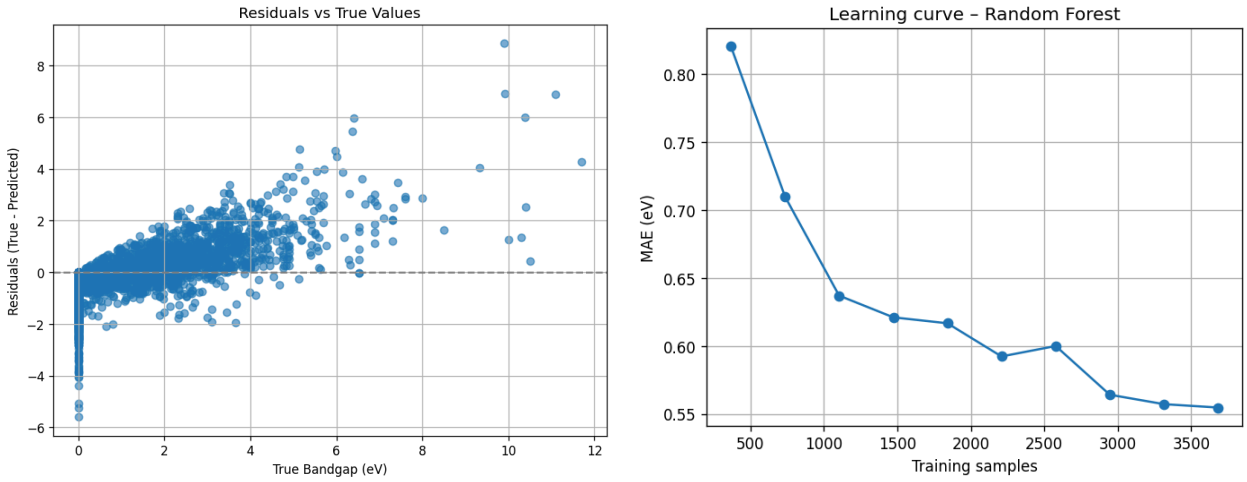
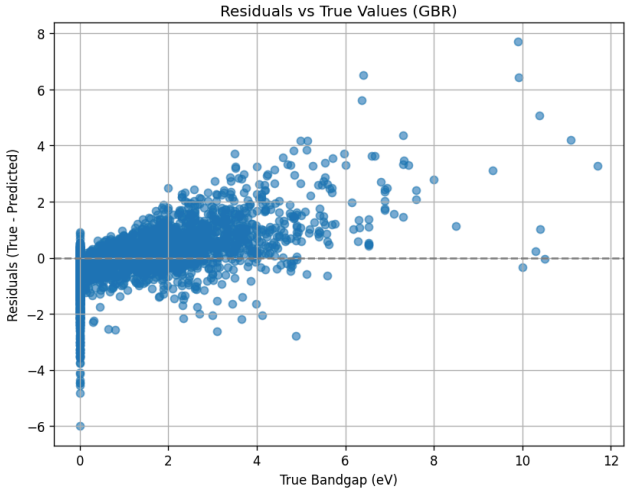
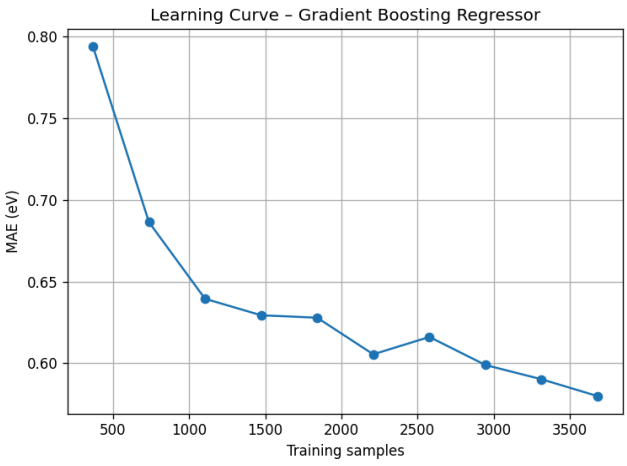


Figure 2. Random Forest Model Diagnostics. Top left: Residuals vs. true band gaps showing heteroscedasticity (fan-shape). Top right: Learning curve indicating steady improvement with more training data. Bottom: Q-Q plot suggests residuals are approximately normally distributed but with heavy tails.

Further improvement was observed using Gradient Boosting Regression, which achieved a cross-validated MAE of 0.512 eV, RMSE of 0.858 eV. Hyperparameter tuning via GridSearchCV confirmed the best configuration out of given options (considering time-constraints) to be a learning rate of 0.05, maximum depth of 5, and 300 estimators. The model's R^2 score was 0.647, outperforming Random Forest. Learning Curve analysis for GBR shows a decline in validation MAE as training size increases, indicating the model is benefitting from more data and is not seeming to plateau yet. This suggests it could improve further with a larger dataset or estimators.





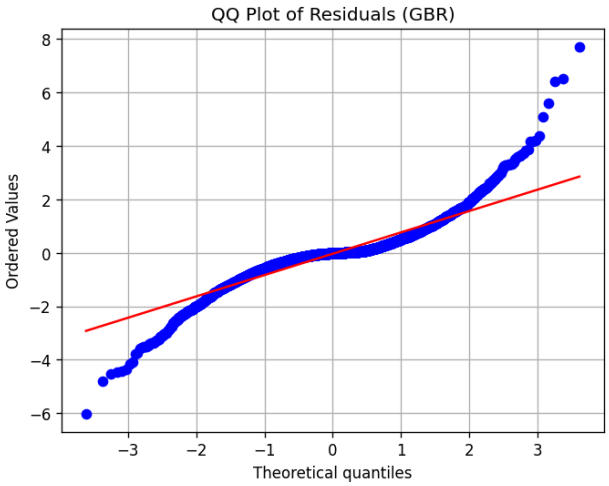
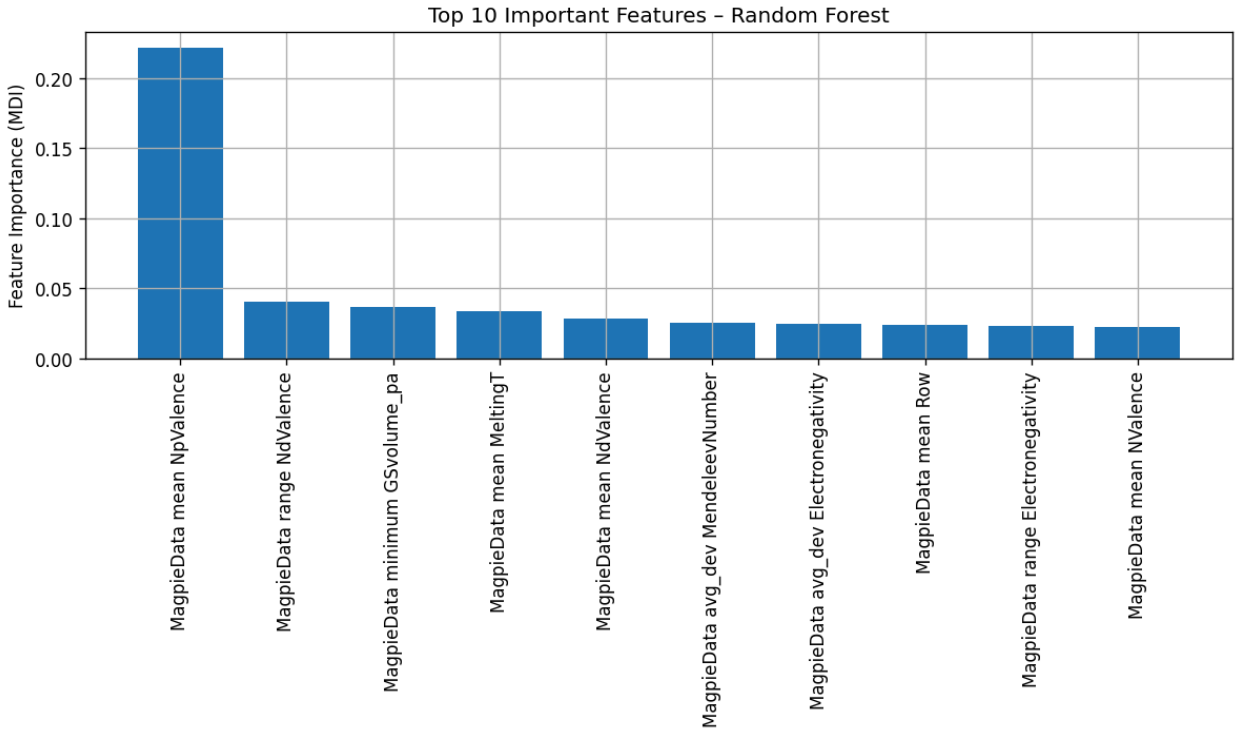


Figure 3. Gradient Boosting Model Diagnostics. Top left: Residuals vs. true band gaps show similar spread as RF but slightly tighter clustering. Top right: Learning curve indicates ongoing performance gains with more data. Bottom: QQ plot shows roughly normal residuals with mild deviation in tails.

Feature importance analysis revealed key contributors to the prediction accuracy for both RF and GBR models, excluding confounding features like the experimentally measured band gap, to ensure model reliability and robustness. Figure 4 shows the top 10 most influential compositional features that contributed to band gap prediction for both these models. MagpieData mean NpValence was found to be the most important feature, reaffirming the key role of valence electron configuration in determining electronic structures. Other features like range NdValence, mean MeltingT, and Electronegativity also contributed meaningfully.



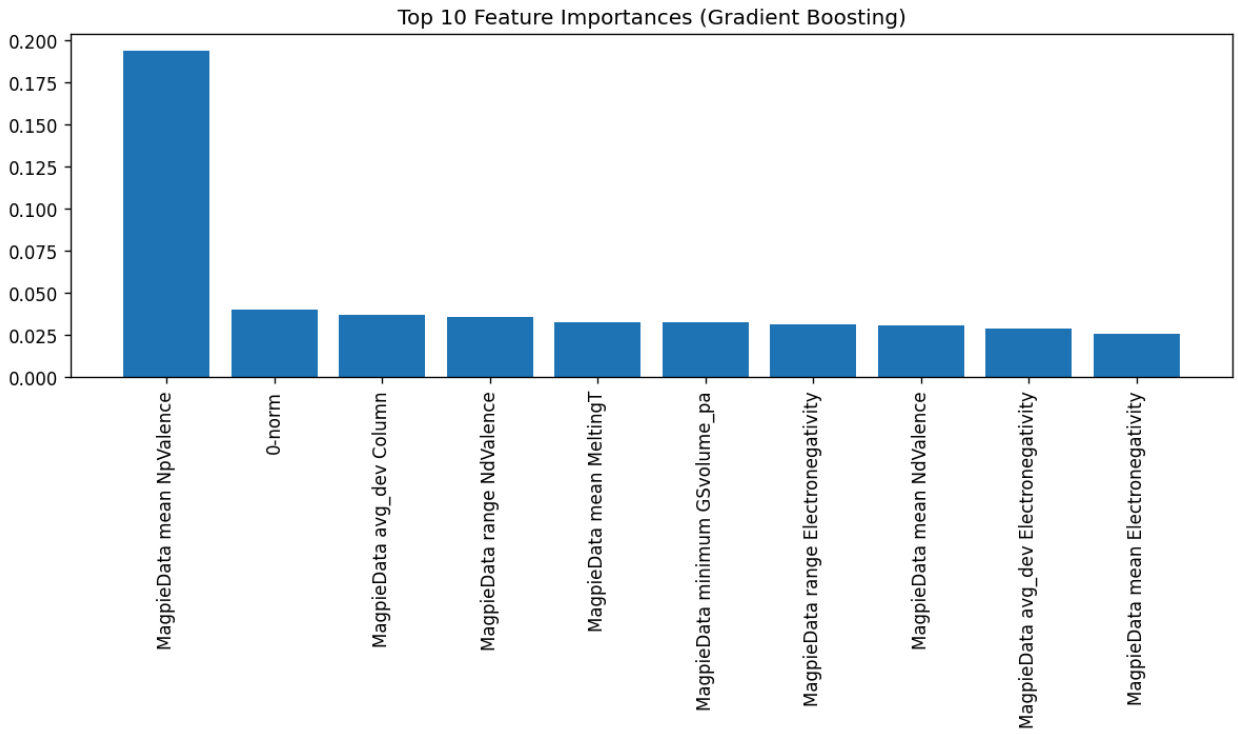


Figure 4. Feature Importance Rankings: for (top) Random Forest Regression, and (bottom) Gradient Boosting Regression models.

Discussion and Conclusion

This project successfully demonstrated the efficacy of integrating classification and regression approaches to accurately predict material band gaps using only compositional features. The baseline pipeline of SVC and SVR achieved a moderate MAE of 0.945 eV and RMSE of 1.669 eV, with a predictive accuracy of only 58%. This highlighted the necessity for advanced modeling techniques but also got us to question the disparity in the results achieved by us and Zhuo et al. The subsequent employment of a Random Forest and Gradient Boosting regression model significantly enhanced predictive accuracy respectively, demonstrating superior generalization capabilities with optimized hyperparameters. RF achieved a reasonable and better MAE of 0.555 eV (RMSE = 0.933). After hyperparameter tuning, GBR topped this performance, achieving an even lower MAE of 0.512 eV (RMSE = 0.858 eV). Learning curve analysis for GBR showed that model performance continued to improve with more training data, suggesting that generalization could further benefit from a bigger dataset.

Jumping back to the reasons behind our poor SVC + SVR results, I decided to account for only Magpie-derived features and also scaled the feature matrix, X, before performing the regression. Accordingly, I was able to top the accuracy obtained by Zhou et al. and get a value of 98%, with a very low MAE of 0.055 eV and RMSE of 0.124 eV. These new results were impressive, allowing us to infer the following:

- The importance of scaling in SVM, which is sensitive to the scale of the features. Features on a very different scale (e.g. atomic number vs. melting point), distance metrics used by SVMs can become distorted.

- Narrowing down to Magpie features helped us avoid noisy, irrelevant features and deal with redundant information.
- Using only a more relevant subset of features likely reduces the risk of overfitting and makes optimization easier and faster.

Achieving substantially improved values that topped our reference literature with the same SVC → SVR hybrid pipeline they used was a key success of this project. This highlighted the importance of comprehensive feature selection and model optimization. However, the study identified challenges as well, such as the initially unnoticed confounding inclusion of the target variable ("gap expt") revealed by the feature importance plot (see Figure 5), which likely artificially inflated performance metrics in earlier models that were doing suspiciously well. Correcting this issue confirmed our models' realistic predictive ability. Residual diagnostics and learning curve plots provided deeper interpretability, showing that model errors were approximately normally distributed and centered around zero, and that performance improved with more data. This analysis supports the validity and robustness of the final GBR model.

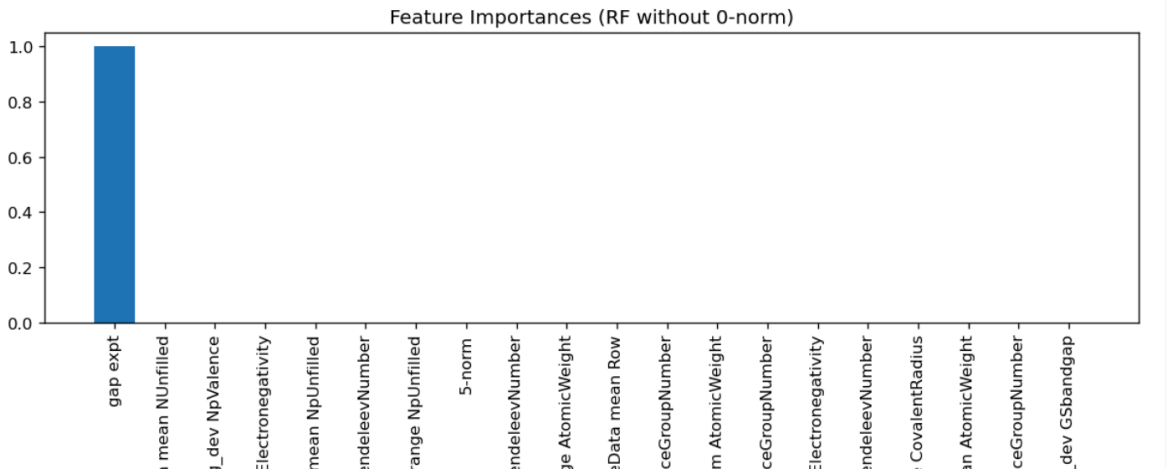


Figure 5. Feature Importance Rankings for the very first Random Forest model, showing the inadvertent inclusion of the target variable "gap expt", which prompted me to forcefully remove it in all further analyses.

Our analyses show that segmenting the problem into classification (metal vs. non-metal) and performing a targeted regression improves prediction abilities. Hence, the two-step SVC → SVR approach outperformed both RF and GBR models. These still play a meaningful role as they give us a general-purpose model that doesn't require a classification step first. They are also easier to interpret and let us explore feature importances more efficiently. However, we can conclude that a cleaner regression problem is a much better choice for such prediction tasks. Our project succeeds in providing informative comparisons with different machine learning strategies to predict electronic band gaps in inorganic compounds.

Future studies could explore deeper feature engineering, alternative ML techniques such as Extreme Gradient Boosting (XGBoost), Gaussian Processes, deep learning architectures like Neural Networks, or broader datasets to further improve model performance and reliability. This project makes way for more sophisticated computational techniques in predictive materials science, offering significant potential for rapid materials discovery and characterization.

References

Band gap - Energy Education. (2015). Energyeducation.ca. https://energyeducation.ca/encyclopedia/Band_gap#cite_ref-RE2_1-1

Zhuo, Y., Mansouri Tehrani, A., & Brgoch, J. (2018). Predicting the Band Gaps of Inorganic Solids by Machine Learning. *The Journal of Physical Chemistry Letters*, *9*(7), 1668–1673. https://doi.org/10.1021/acs.jpclett.8b00124

Electronic Band-Gap Prediction via Machine Learning

Author: Palak Mahendru | Date: 2025-04-16

Workflow:

1. Data load & EDA
2. Feature engineering (composition descriptors via matminer)
3. Baseline SVC → SVR model
4. Tree-based models (Random-Forest, GradientBoosting + GridSearchCV)
5. Interpretation with feature_importance, residual, qq-plots
6. Learning curves

```
pip install matminer
```

```
Requirement already satisfied: plotly>=5.0.0 in /usr/local/lib/python3.11/dist-packages (from pymatgen>=2023->matminer) (5.24.1)
Collecting pybtex>=0.24.0 (from pymatgen>=2023->matminer)
  Downloading pybtex-0.24.0-py2.py3-none-any.whl.metadata (2.0 kB)
Requirement already satisfied: scipy>=1.13.0 in /usr/local/lib/python3.11/dist-packages (from pymatgen>=2023->matminer) (1.14.1)
Collecting spglib>=2.5 (from pymatgen>=2023->matminer)
  Downloading spglib-2.6.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (4.2 kB)
Requirement already satisfied: tabulate>=0.9 in /usr/local/lib/python3.11/dist-packages (from pymatgen>=2023->matminer) (0.9.0)
Collecting uncertainties>=3.1.4 (from pymatgen>=2023->matminer)
  Downloading uncertainties-3.2.2-py3-none-any.whl.metadata (6.9 kB)
Collecting dnspython<3.0.0,>=1.16.0 (from pymongo~=4.5->matminer)
  Downloading dnspython-2.7.0-py3-none-any.whl.metadata (5.8 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests~=2.31->matminer) (3
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests~=2.31->matminer) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests~=2.31->matminer) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests~=2.31->matminer) (2025.1.
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn~=1.3->matminer) (3.
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy~=1.11->matminer) (1.3.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.8->pymatgen>=2023->mat
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.8->pymatgen>=2023->matmine
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.8->pymatgen>=2023->ma
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.8->pymatgen>=2023->ma
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.8->pymatgen>=2023->matm
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.8->pymatgen>=2023->matminer)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.8->pymatgen>=2023->mat
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.11/dist-packages (from plotly>=5.0.0->pymatgen>=2023->matmir
Requirement already satisfied: PyYAML>=3.01 in /usr/local/lib/python3.11/dist-packages (from pybtex>=0.24.0->pymatgen>=2023->matminer
Collecting latexcodec>=1.0.4 (from pybtex>=0.24.0->pymatgen>=2023->matminer)
  Downloading latexcodec-3.0.0-py3-none-any.whl.metadata (4.9 kB)
Requirement already satisfied: six in /usr/local/lib/python3.11/dist-packages (from pybtex>=0.24.0->pymatgen>=2023->matminer) (1.17.0
Collecting ruamel.yaml.clib>=0.2.7 (from ruamel.yaml->monty>=2023->matminer)
  Downloading ruamel.yaml.clib-0.2.12-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (2.7 kB)
Requirement already satisfied: typing-extensions>=4.9.0 in /usr/local/lib/python3.11/dist-packages (from spglib>=2.5->pymatgen>=2023-
Downloading matminer-0.9.3-py3-none-any.whl (5.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.5/5.5 MB 50.7 MB/s eta 0:00:00
Downloading monty-2025.3.3-py3-none-any.whl (51 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 51.9/51.9 kB 3.0 MB/s eta 0:00:00
Downloading pymatgen-2025.4.17-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (5.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.1/5.1 MB 90.3 MB/s eta 0:00:00
Downloading pymongo-4.12.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.4/1.4 MB 48.9 MB/s eta 0:00:00
Downloading dnspython-2.7.0-py3-none-any.whl (313 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 313.6/313.6 kB 20.6 MB/s eta 0:00:00
Downloading palettable-3.3.3-py2.py3-none-any.whl (332 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 332.3/332.3 kB 22.2 MB/s eta 0:00:00
Downloading pybtex-0.24.0-py2.py3-none-any.whl (561 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 561.4/561.4 kB 32.3 MB/s eta 0:00:00
Downloading ruamel.yaml-0.18.10-py3-none-any.whl (117 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 117.7/117.7 kB 9.0 MB/s eta 0:00:00
Downloading spglib-2.6.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (809 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 809.0/809.0 kB 34.7 MB/s eta 0:00:00
Downloading uncertainties-3.2.2-py3-none-any.whl (58 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.3/58.3 kB 3.6 MB/s eta 0:00:00
Downloading latexcodec-3.0.0-py3-none-any.whl (18 kB)
Downloading ruamel.yaml.clib-0.2.12-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (739 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 739.1/739.1 kB 37.4 MB/s eta 0:00:00
Installing collected packages: uncertainties, spglib, ruamel.yaml.clib, palettable, latexcodec, dnspython, ruamel.yaml, pymongo, pybt
Successfully installed dnspython-2.7.0 latexcodec-3.0.0 matminer-0.9.3 monty-2025.3.3 palettable-3.3.3 pybtex-0.24.0 pymatgen-2025.4.
```

```
pip install lightgbm shap
```

```
Requirement already satisfied: lightgbm in /usr/local/lib/python3.11/dist-packages (4.5.0)
Requirement already satisfied: shap in /usr/local/lib/python3.11/dist-packages (0.47.1)
Requirement already satisfied: numpy>=1.17.0 in /usr/local/lib/python3.11/dist-packages (from lightgbm) (2.0.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from lightgbm) (1.14.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (from shap) (1.6.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (from shap) (2.2.2)
Requirement already satisfied: tqdm>=4.27.0 in /usr/local/lib/python3.11/dist-packages (from shap) (4.67.1)
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.11/dist-packages (from shap) (24.2)
Requirement already satisfied: slicer==0.0.8 in /usr/local/lib/python3.11/dist-packages (from shap) (0.0.8)
Requirement already satisfied: numba>=0.54 in /usr/local/lib/python3.11/dist-packages (from shap) (0.60.0)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.11/dist-packages (from shap) (3.1.1)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.11/dist-packages (from shap) (4.13.2)
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/python3.11/dist-packages (from numba>=0.54->shap) (0.43.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas->shap) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas->shap) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas->shap) (2025.2)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn->shap) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn->shap) (3.6.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas->shap) (1.17.0)
```

```
import os, random, json, joblib, warnings, math
from pathlib import Path

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, cross_val_predict, learning_curve, GridSearchCV
from sklearn.metrics import mean_absolute_error, root_mean_squared_error, r2_score, accuracy_score
from sklearn.svm import SVC, SVR
from sklearn.ensemble import RandomForestRegressor


from matminer.datasets import load_dataset
from matminer.featurizers.composition import ElementProperty, Stoichiometry, ValenceOrbital
from matminer.featurizers.base import MultipleFeaturizer
from pymatgen.core import Composition

def str_to_composition(formula_str):
    """Fallback if matminer.utils.conversions is missing."""
    try:
        return Composition(formula_str)
    except Exception:
        return None

SEED = 42
np.random.seed(SEED)
random.seed(SEED)
pd.set_option('display.max_columns', 100)
plt.rcParams['figure.dpi'] = 120
```

1. Load Dataset from Matminer

```
df = load_dataset('matbench_expt_gap')
df.head()
```

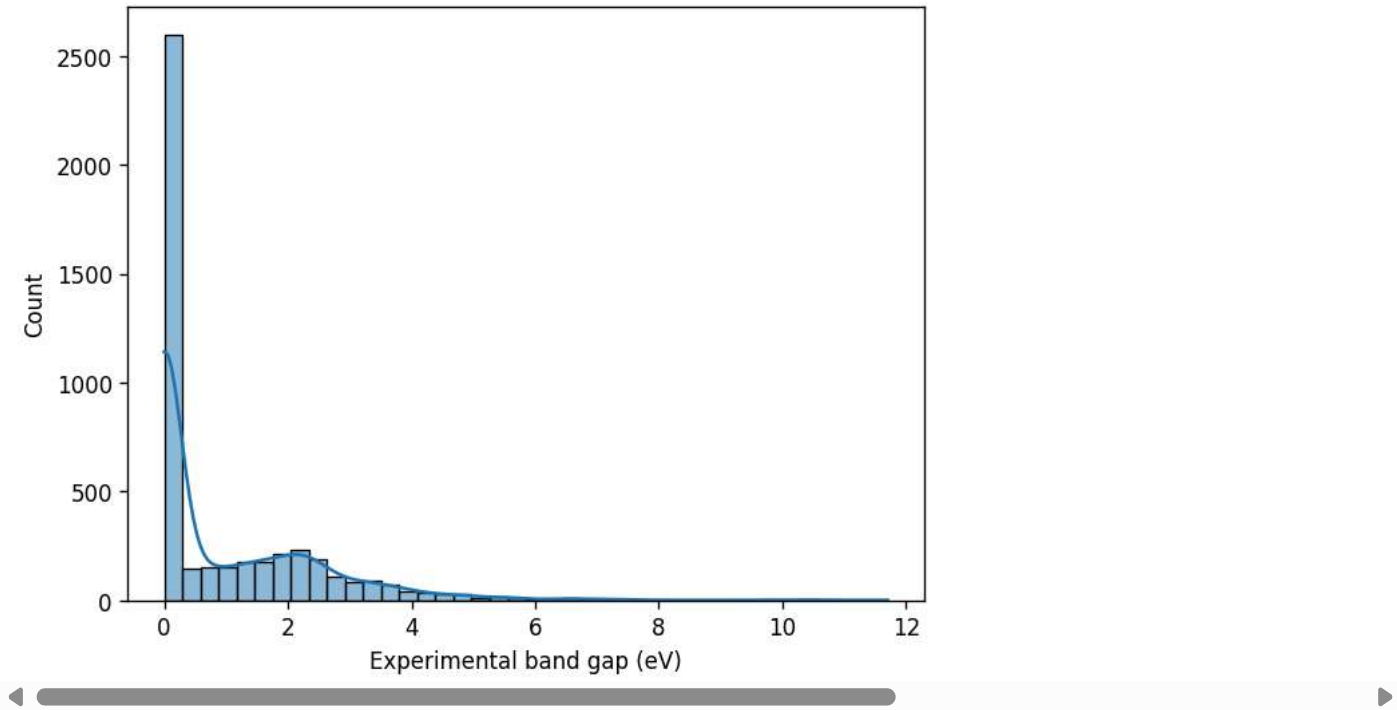
```
Fetching matbench_expt_gap.json.gz from https://ml.materialsproject.org/projects/matbench_expt_gap.json.gz to /usr/local/lib/python3.11/
Fetching https://ml.materialsproject.org/projects/matbench_expt_gap.json.gz in MB: 0.038911999999999995MB [00:00, 21.43MB/s]
```

	composition	gap expt
0	Ag(AuS)2	0.00
1	Ag(W3Br7)2	0.00
2	Ag0.5Ge1Pb1.75S4	1.83
3	Ag0.5Ge1Pb1.75Se4	1.51
4	Ag2BBr	0.00

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
print(df['gap expt'].describe())
sns.histplot(df['gap expt'], bins=40, kde=True)
plt.xlabel('Experimental band gap (eV)');
```

```
count    4604.000000
mean        0.975951
std         1.445034
min         0.000000
25%         0.000000
50%         0.000000
75%         1.812500
max        11.700000
Name: gap expt, dtype: float64
```



2. Feature Engineering (Composition descriptors)

```
from sklearn.impute import SimpleImputer

featurizer = MultipleFeaturizer([
    Stoichiometry(),
    ElementProperty.from_preset('magpie'),
    ValenceOrbital('average')
])

df['composition_obj'] = df['composition'].apply(str_to_composition)
feature_df = featurizer.featurize_dataframe(df, 'composition_obj', ignore_errors=True)
numeric_df = feature_df.select_dtypes(include=[np.number])
numeric_df.replace([np.inf, -np.inf], np.nan, inplace=True)
numeric_df = numeric_df.drop(columns=['gap expt'])

numeric_df = numeric_df.loc[:, ~numeric_df.columns.duplicated()]
all_nan_cols = numeric_df.columns[numeric_df.isna().all()]
numeric_df = numeric_df.drop(columns=all_nan_cols)

imputer = SimpleImputer(strategy="mean")
X = imputer.fit_transform(numeric_df)
y = feature_df['gap expt'].values
print('Feature matrix:', X.shape)
```

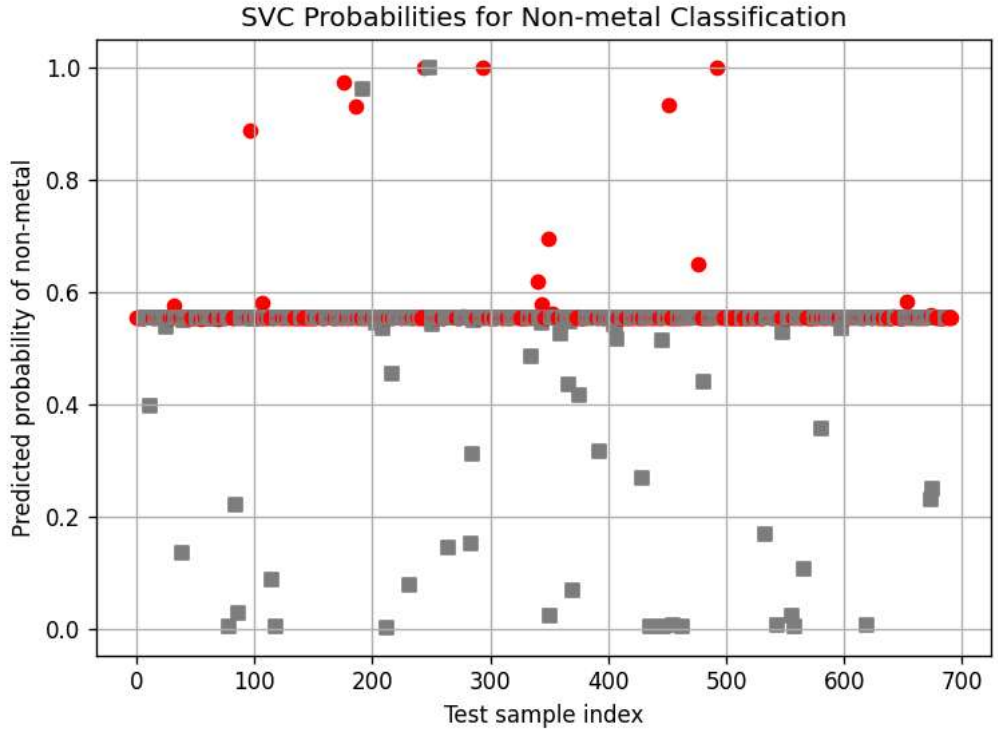
```
/usr/local/lib/python3.11/dist-packages/matminer/utils/data.py:326: UserWarning: MagpieData(impute_nan=False):
In a future release, impute_nan will be set to True by default.
                    This means that features that are missing or are NaNs for elements
                    from the data source will be replaced by the average of that value
                    over the available elements.
                    This avoids NaNs after featurization that are often replaced by
                    dataset-dependent averages.
  warnings.warn(f"{self.__class__.__name__}(impute_nan=False):\n" + IMPUTE_NAN_WARNING)
/usr/local/lib/python3.11/dist-packages/matminer/featurizers/composition/orbital.py:115: UserWarning: ValenceOrbital(impute_nan=False):
In a future release, impute_nan will be set to True by default.
                    This means that features that are missing or are NaNs for elements
                    from the data source will be replaced by the average of that value
                    over the available elements.
                    This avoids NaNs after featurization that are often replaced by
                    dataset-dependent averages.
  warn(f"{self.__class__.__name__}(impute_nan=False):\n" + IMPUTE_NAN_WARNING)
MultipleFeaturizer: 100%                                  4604/4604 [00:49<00:00, 170.33it/s]
Feature matrix: (4604, 138)
```

3. Baseline: Zhou pipeline SVC -> SVR

```
metals = y < 0.1
X_train, X_test, y_train, y_test, m_train, m_test = train_test_split(
    X, y, metals, test_size=0.15, stratify=metals, random_state=SEED)

# 1: classifying metals vs non-metals
svc = SVC(kernel='rbf', C=10, gamma=0.01, probability=True, random_state=SEED)
svc.fit(X_train, m_train)

# 2: regression on non-metals
svr = SVR(kernel='rbf', C=100, gamma=0.01, epsilon=0.1)
svr.fit(X_train[~m_train], y_train[~m_train])

m_pred = svc.predict(X_test)
preds = np.where(m_pred, 0.0, svr.predict(X_test))
mae = mean_absolute_error(y_test, preds)
rmse = root_mean_squared_error(y_test, preds)
print(f'Baseline SVC→SVR  MAE={mae:.3f} eV  RMSE={rmse:.3f} eV')
```

```
Baseline SVC→SVR  MAE=0.945 eV  RMSE=1.669 eV
```

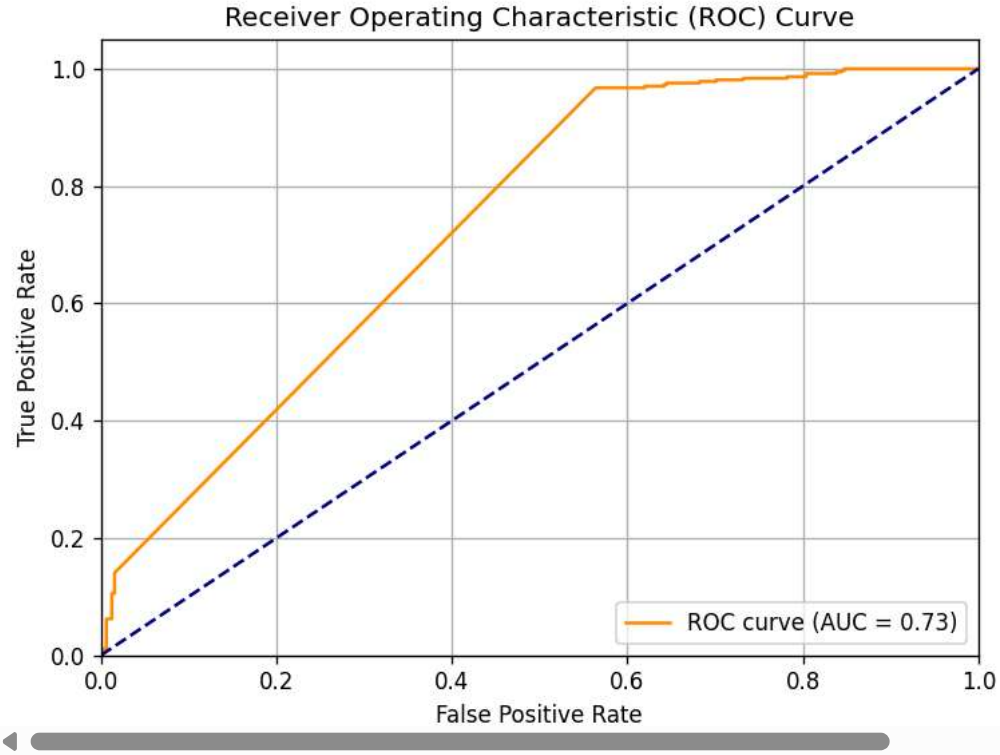
```
probs = svc.predict_proba(X_test)[:, 1]

# Scatter plot: red for non-metals, gray for metals
plt.figure()
for i in range(len(probs)):
    color = 'red' if m_test[i] else 'gray'
    marker = 'o' if m_test[i] else 's'
    plt.scatter(i, probs[i], color=color, marker=marker)
plt.xlabel('Test sample index')
plt.ylabel('Predicted probability of non-metal')
plt.title('SVC Probabilities for Non-metal Classification')
plt.grid(True)
plt.tight_layout()
plt.show()
```



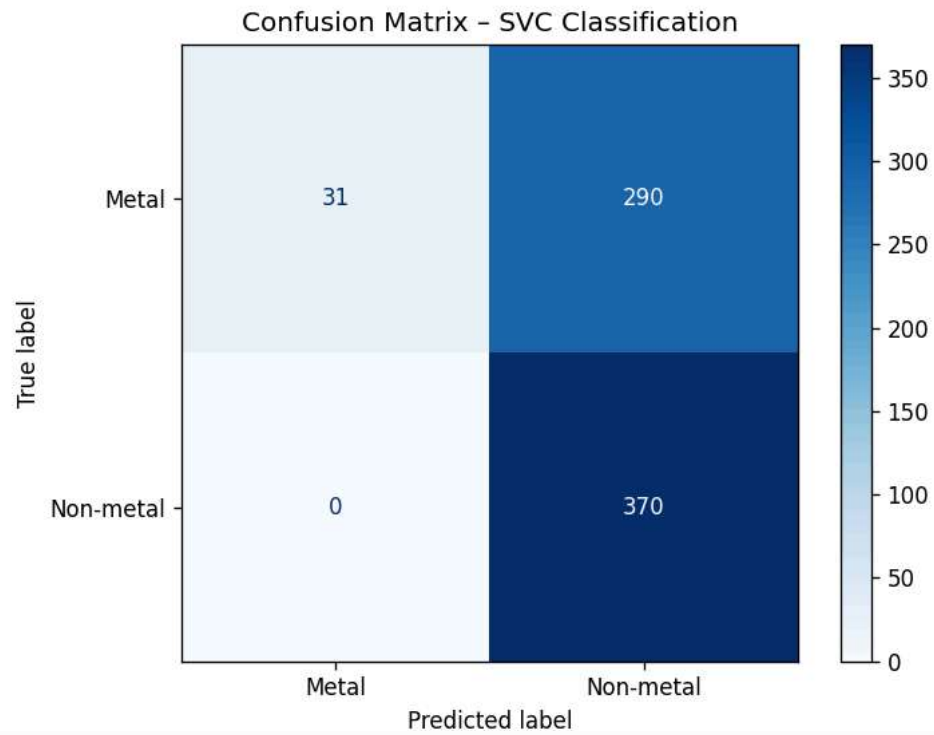

```
accuracy = accuracy_score(m_test, m_pred)
print(f"SVC Accuracy: {accuracy:.2f}")
```

```
SVC Accuracy: 0.58
```

```
from sklearn.metrics import roc_curve, auc

fpr, tpr, _ = roc_curve(m_test, probs)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.grid(True)
plt.tight_layout()
plt.show()
```

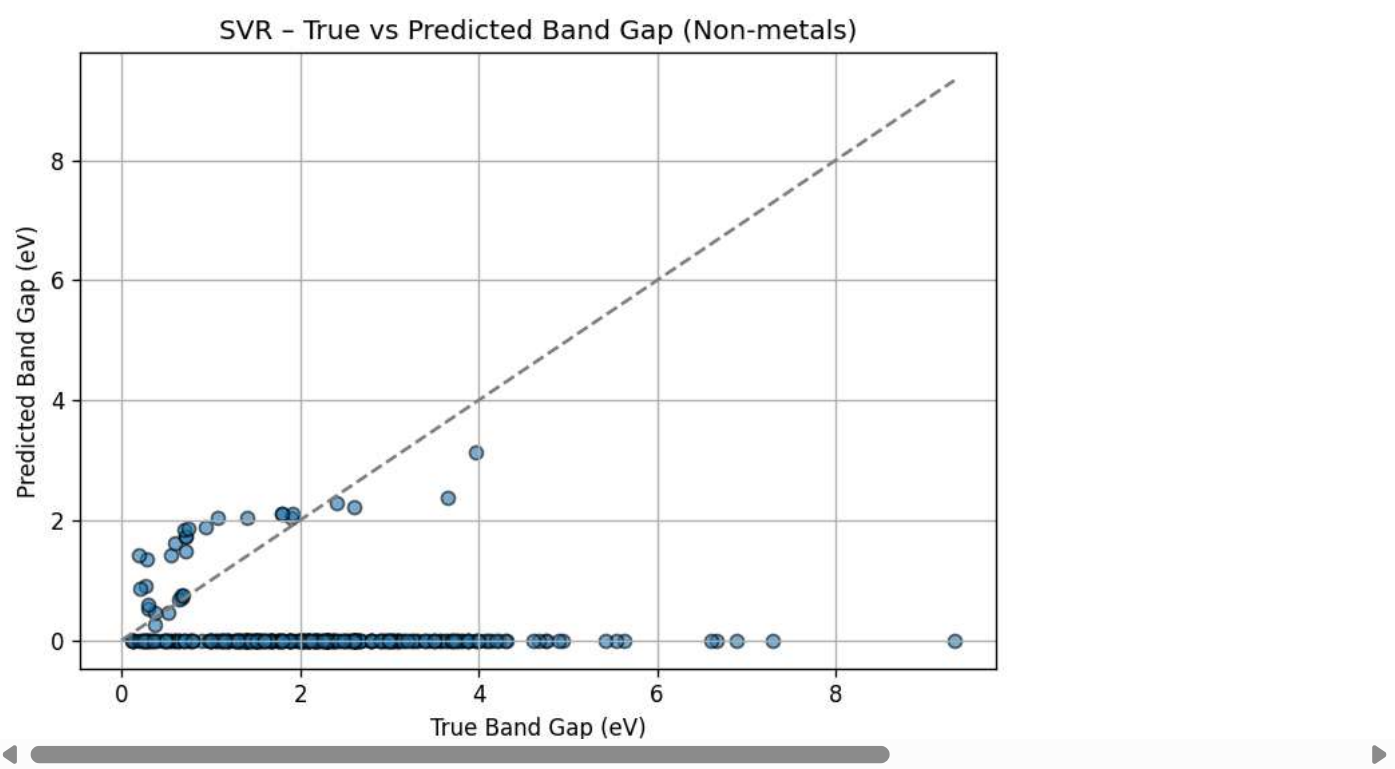



```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

cm = confusion_matrix(m_test, m_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Metal", "Non-metal"])
disp.plot(cmap='Blues')
plt.title("Confusion Matrix – SVC Classification")
plt.grid(False)
plt.tight_layout()
plt.show()
```

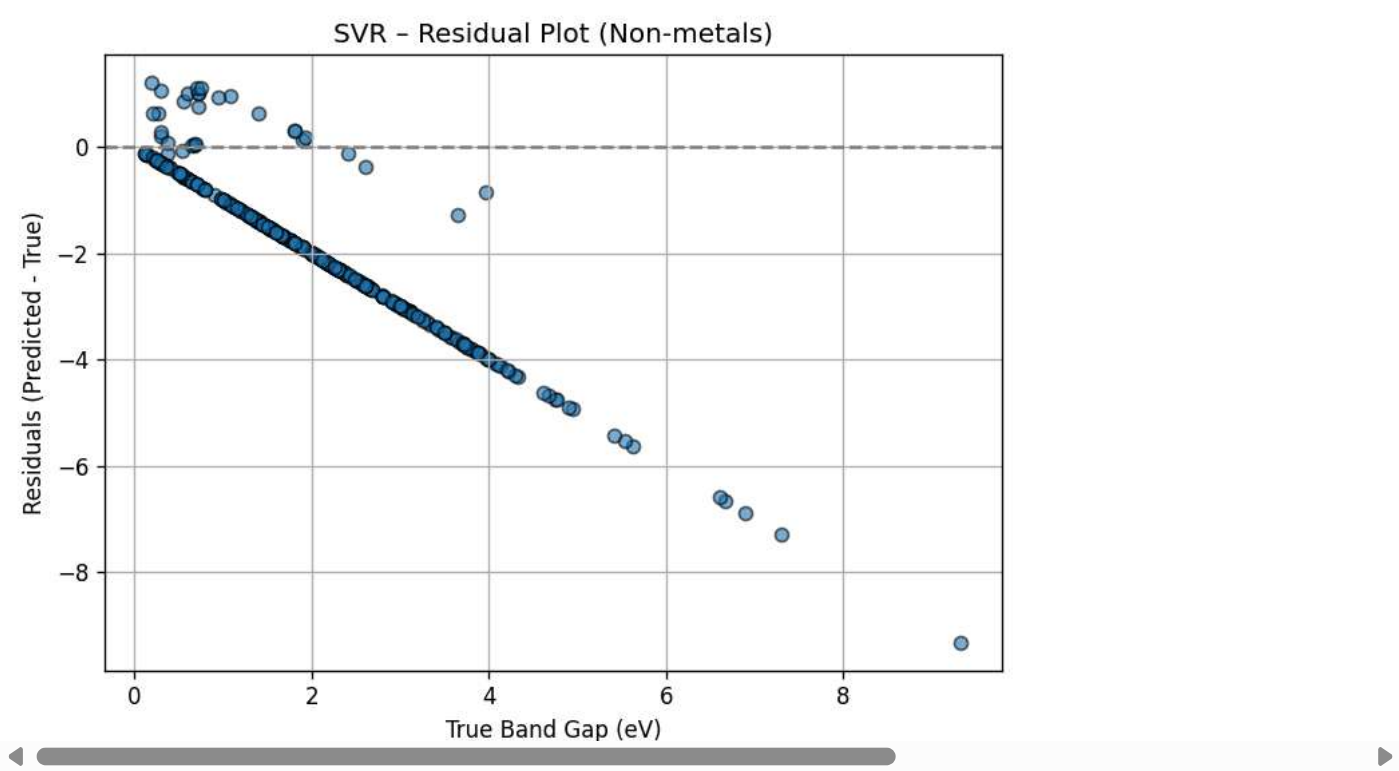



```
# Only for non-metals
true_nonmetals = y_test[~m_test]
pred_nonmetals = preds[~m_test]
```

```
plt.figure()
plt.scatter(true_nonmetals, pred_nonmetals, alpha=0.6, edgecolor='k')
plt.plot([0, max(true_nonmetals)], [0, max(true_nonmetals)], '--', color='gray')
plt.xlabel('True Band Gap (eV)')
plt.ylabel('Predicted Band Gap (eV)')
plt.title('SVR – True vs Predicted Band Gap (Non-metals)')
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
# Residual Plot
residuals = pred_nonmetals - true_nonmetals

plt.figure()
plt.scatter(true_nonmetals, residuals, alpha=0.6, edgecolor='k')
plt.axhline(0, linestyle='--', color='gray')
plt.xlabel('True Band Gap (eV)')
plt.ylabel('Residuals (Predicted - True)')
plt.title('SVR – Residual Plot (Non-metals)')
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
mae = mean_absolute_error(true_nonmetals, pred_nonmetals)
rmse = root_mean_squared_error(true_nonmetals, pred_nonmetals)
print(f'SVR Performance on Non-metals\nMAE = {mae:.3f} eV | RMSE = {rmse:.3f} eV')
```

```
SVR Performance on Non-metals
MAE = 2.034 eV | RMSE = 2.448 eV
```

4. Tree-Based Models

```
rf = RandomForestRegressor(n_estimators=500, random_state=SEED)
y_pred_rf = cross_val_predict(rf, X, y, cv=5, verbose=1, n_jobs=-1)
mae_rf = mean_absolute_error(y, y_pred_rf)
print(f'Random Forest CV-MAE = {mae_rf:.3f} eV')
rf.fit(X, y)
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed: 10.2min finished
Random Forest CV-MAE = 0.555 eV
```

```
RandomForestRegressor
RandomForestRegressor(n_estimators=500, random_state=42)
```

```
rmse_rf = root_mean_squared_error(y, y_pred_rf)
r2_rf = r2_score(y, y_pred_rf)

print(f'Random Forest CV RMSE = {rmse_rf:.3f} eV')
print(f'Random Forest CV R² = {r2_rf:.3f}')
```

```
Random Forest CV RMSE = 0.933 eV
Random Forest CV R² = 0.583
```

```
from sklearn.ensemble import GradientBoostingRegressor

gbr = GradientBoostingRegressor(
    n_estimators=300,
    learning_rate=0.1,
    max_depth=3,
    random_state=SEED
)

# cross-validated predictions
```

```
y_pred_gbr = cross_val_predict(gbr, X, y, cv=5, n_jobs=-1)

mae_gbr = mean_absolute_error(y, y_pred_gbr)
rmse_gbr = root_mean_squared_error(y, y_pred_gbr)
print(f'Gradient Boosting CV-MAE = {mae_gbr:.3f} eV')
print(f'Gradient Boosting CV-RMSE = {rmse_gbr:.3f} eV')
```

```
Gradient Boosting CV-MAE = 0.548 eV
Gradient Boosting CV-RMSE = 0.874 eV
```

```
# with GridSearchCV
param_grid = {
    'n_estimators': [100, 300],
    'learning_rate': [0.01, 0.05],
    'max_depth': [3, 5]
}

gbr = GradientBoostingRegressor(random_state=SEED)
grid_search = GridSearchCV(gbr, param_grid, cv=5, scoring='neg_mean_absolute_error', n_jobs=-1, verbose=1)
grid_search.fit(X, y)

# using best estimator
best_gbr = grid_search.best_estimator_
y_pred = cross_val_predict(best_gbr, X, y, cv=5, n_jobs=-1)
mae = mean_absolute_error(y, y_pred)
rmse = root_mean_squared_error(y, y_pred)

print(f"Gradient Boosting CV-MAE = {mae:.3f} eV")
print(f"Gradient Boosting RMSE = {rmse:.3f} eV")
print("Best Params:", grid_search.best_params_)
```

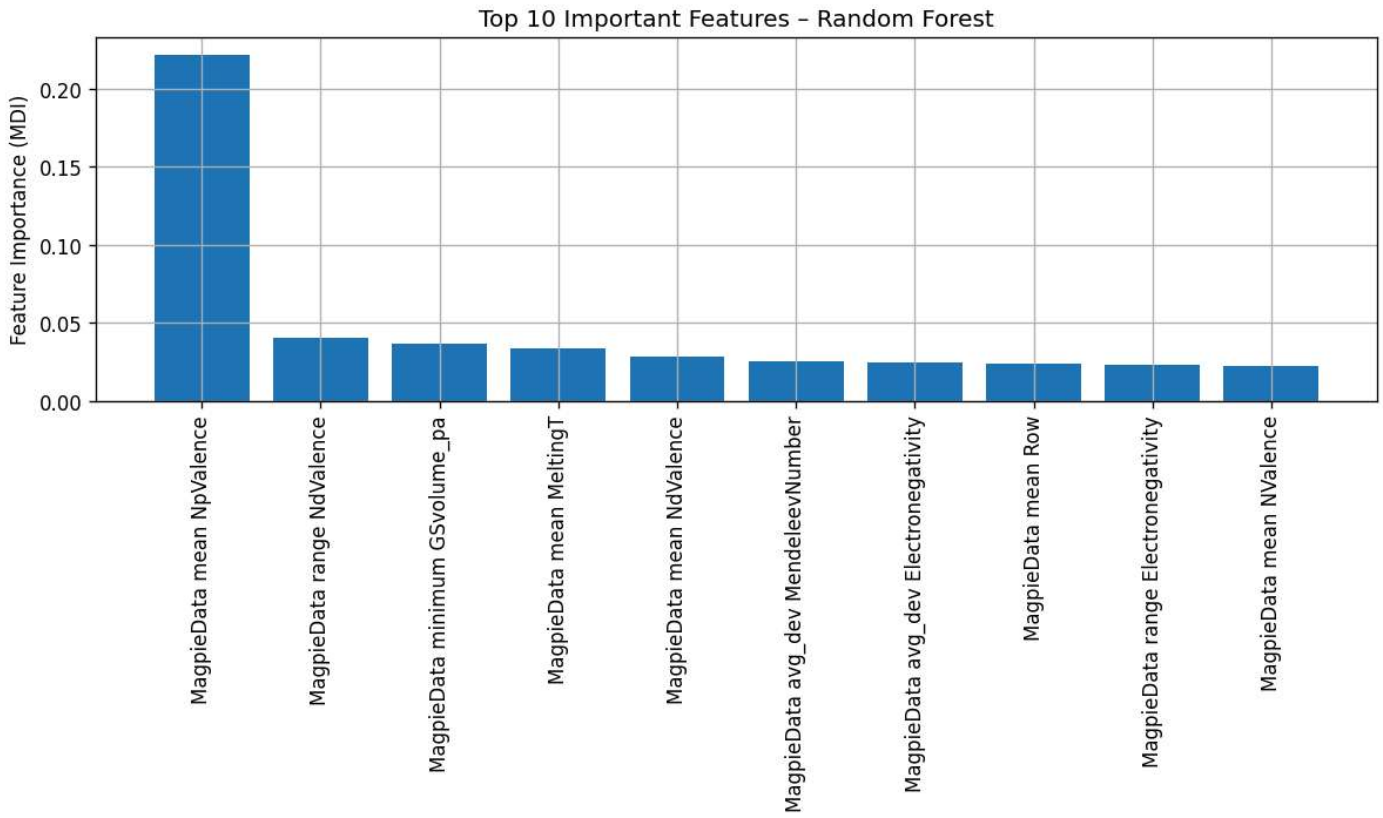
```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
Gradient Boosting CV-MAE = 0.512 eV
Gradient Boosting RMSE = 0.858 eV
Best Params: {'learning_rate': 0.05, 'max_depth': 5, 'n_estimators': 300}
```

```
r2_gbr = r2_score(y, y_pred)
print(f'Gradient boosting Regression R² = {r2_gbr:.3f}')
```

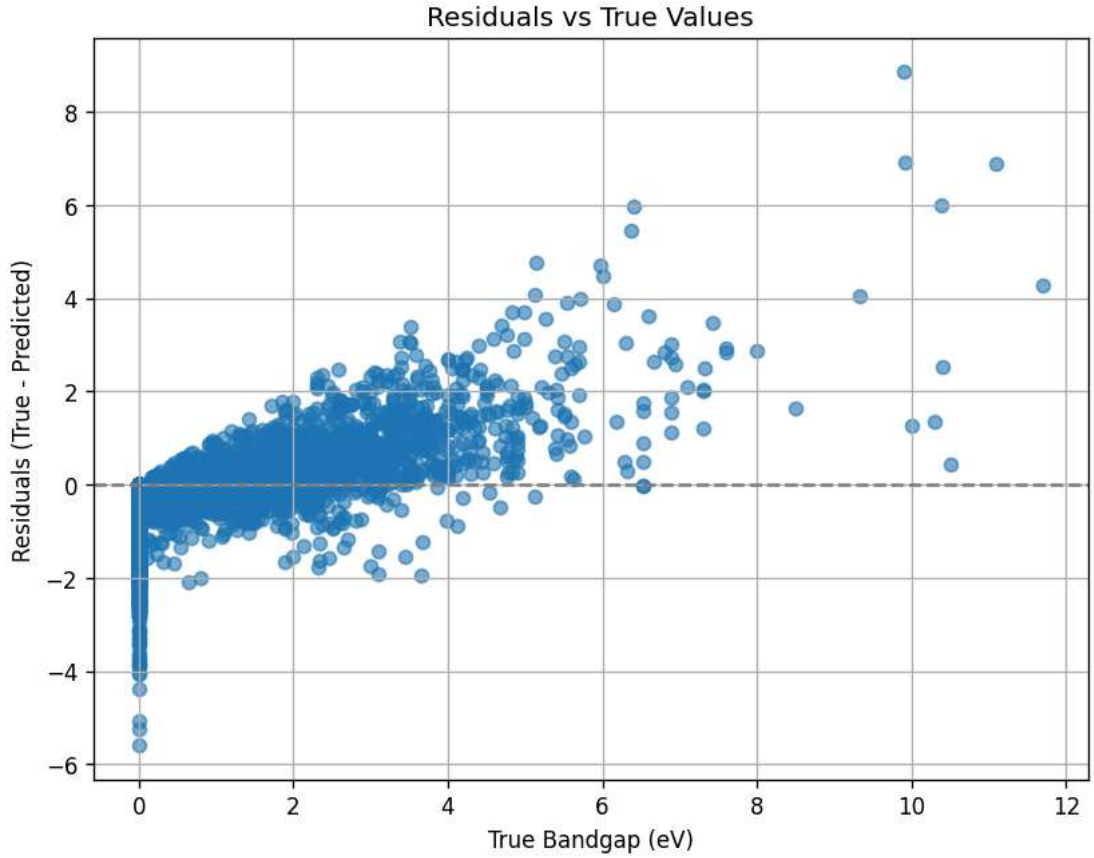
```
Gradient boosting Regression R² = 0.647
```

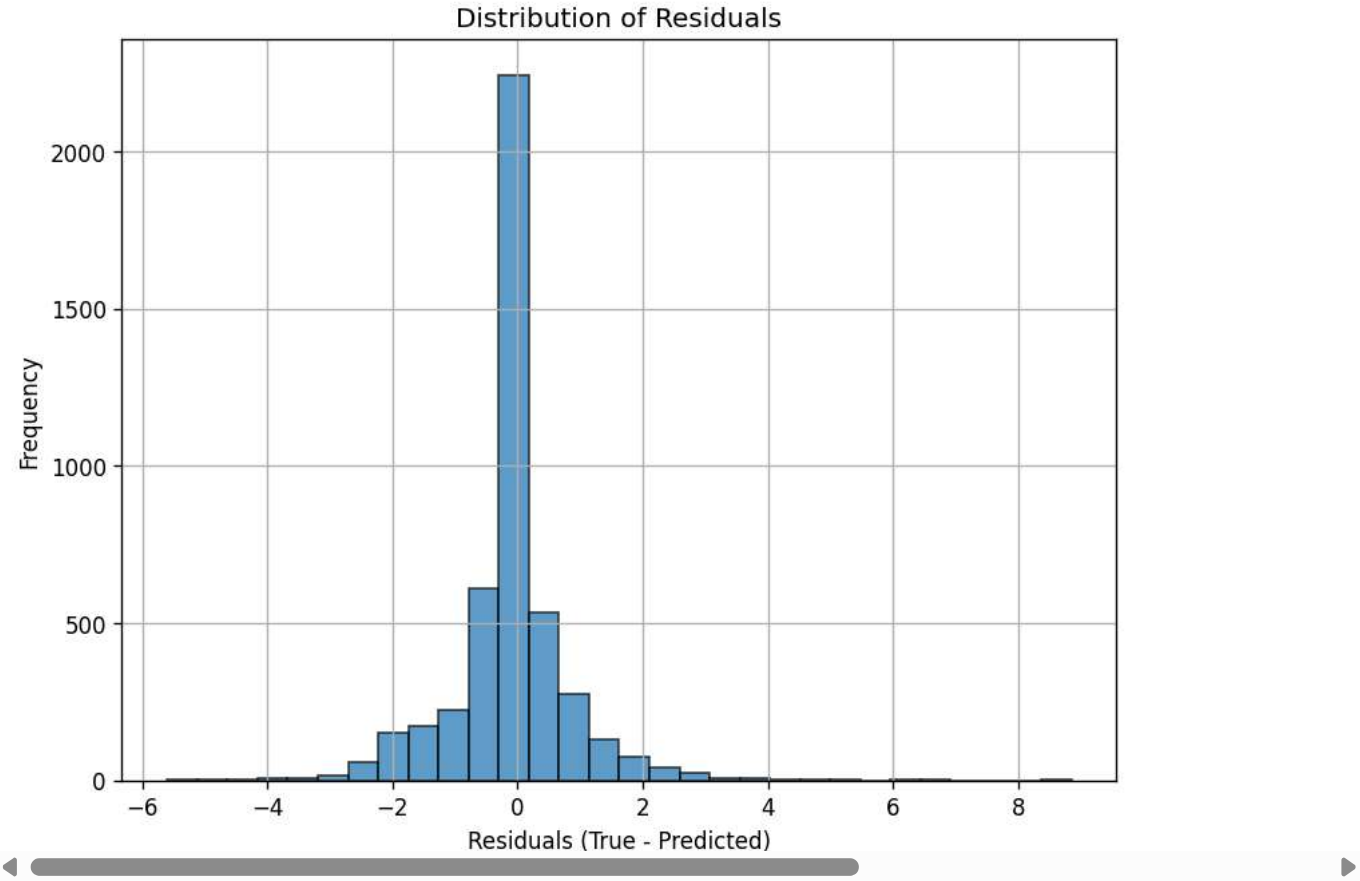
5. Interpretation and Evaluation of RF and GBR

```
# Feature Importances for RF
importances = rf.feature_importances_
feature_names = featurizer.feature_labels()
indices = np.argsort(importances)[::-1]

# top 10 important features
plt.figure(figsize=(10, 6))
plt.bar(range(10), importances[indices[:10]], align="center")
plt.xticks(range(10), [feature_names[i] for i in indices[:10]], rotation=90)
plt.ylabel("Feature Importance (MDI)")
plt.title("Top 10 Important Features – Random Forest")
plt.tight_layout()
plt.grid(True)
plt.show()
```



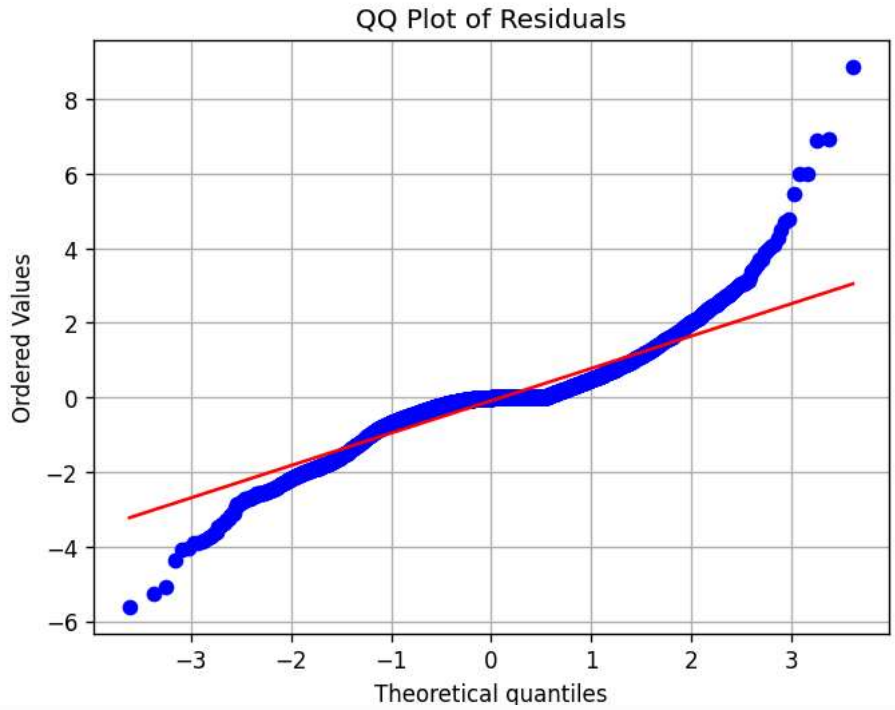

```
# Residual Plot for RF
residuals = y - y_pred_rf

plt.figure(figsize=(8, 6))
plt.scatter(y, residuals, alpha=0.6)
plt.axhline(0, color='gray', linestyle='--')
plt.xlabel('True Bandgap (eV)')
plt.ylabel('Residuals (True - Predicted)')
plt.title('Residuals vs True Values')
plt.grid(True)
plt.show()

# Distribution of Residuals
plt.figure(figsize=(8, 6))
plt.hist(residuals, bins=30, edgecolor='k', alpha=0.7)
plt.xlabel('Residuals (True - Predicted)')
plt.ylabel('Frequency')
plt.title('Distribution of Residuals')
plt.grid(True)
plt.show()
```



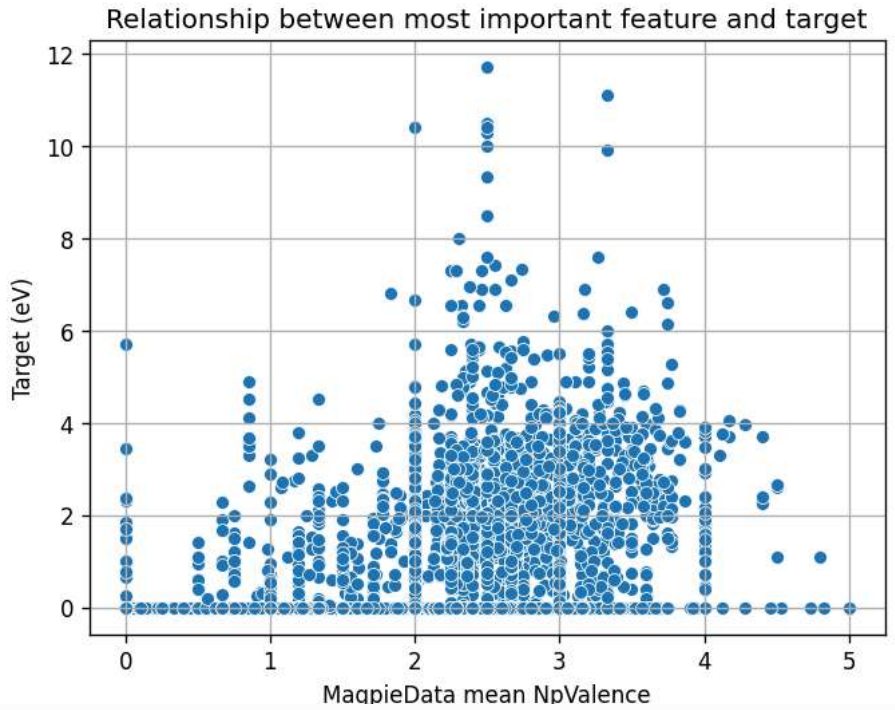



```
# Q-Q Plot of Residuals (RF)

import scipy.stats as stats

stats.probplot(residuals, dist="norm", plot=plt)
plt.title("QQ Plot of Residuals")
```

```
plt.grid(True)
plt.show()
```

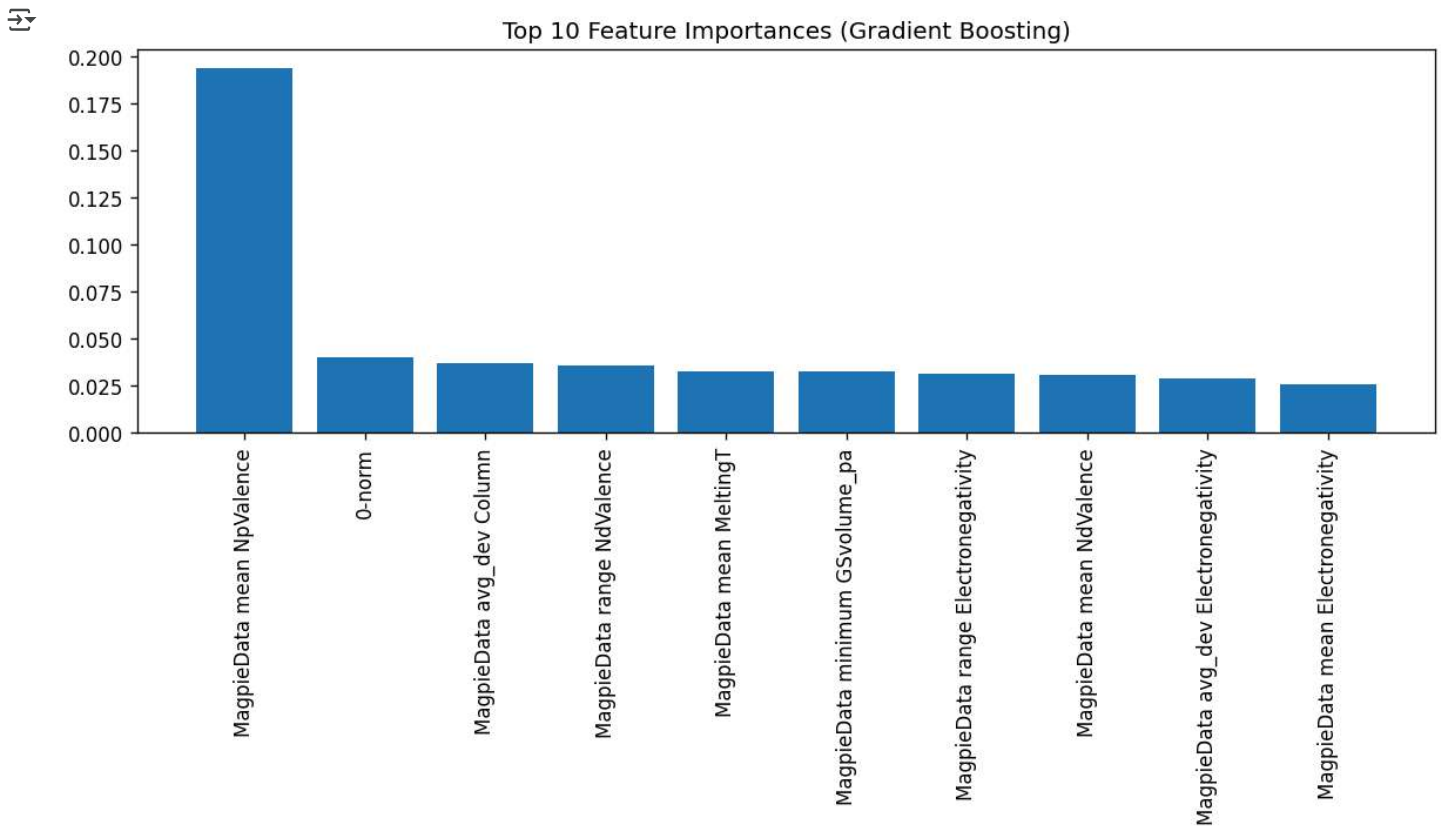



```
import seaborn as sns

sns.scatterplot(x=X[:, feature_names.index('MagpieData mean NpValence')], y=y)
plt.xlabel('MagpieData mean NpValence')
plt.ylabel('Target (eV)')
plt.title('Relationship between most important feature and target')
plt.grid(True)
plt.show()
```

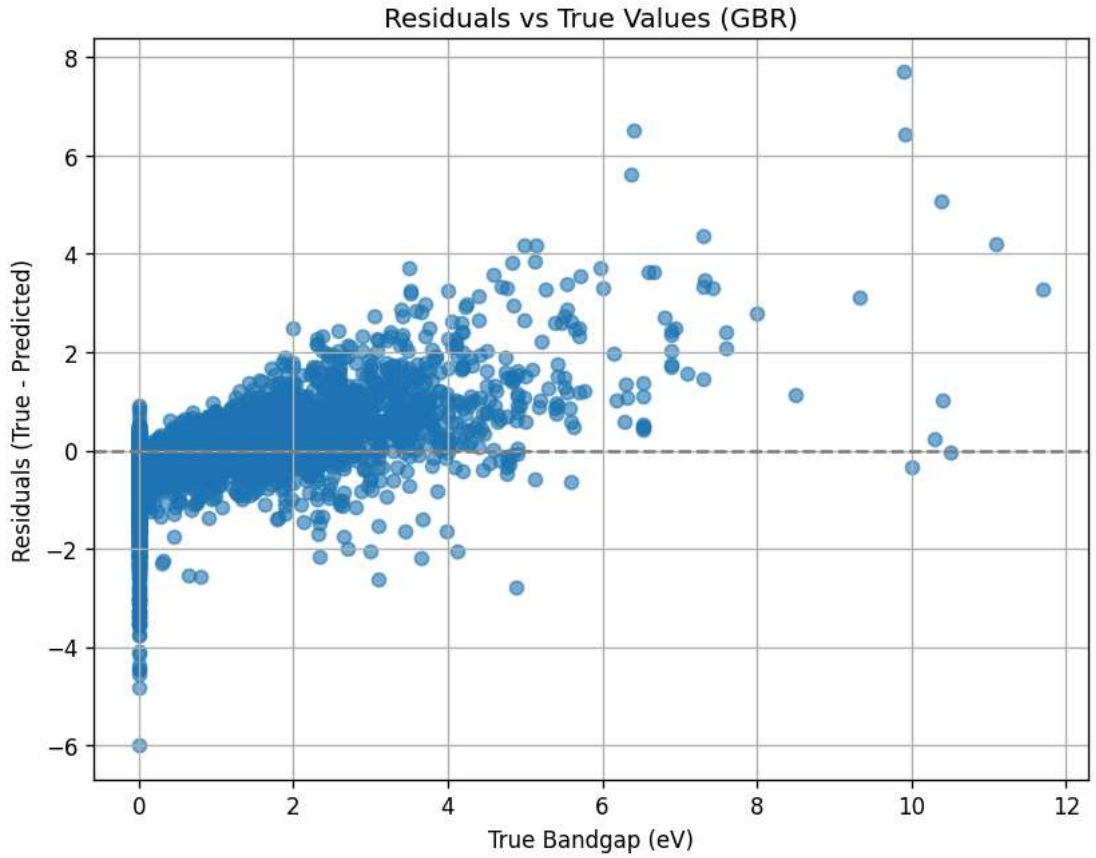


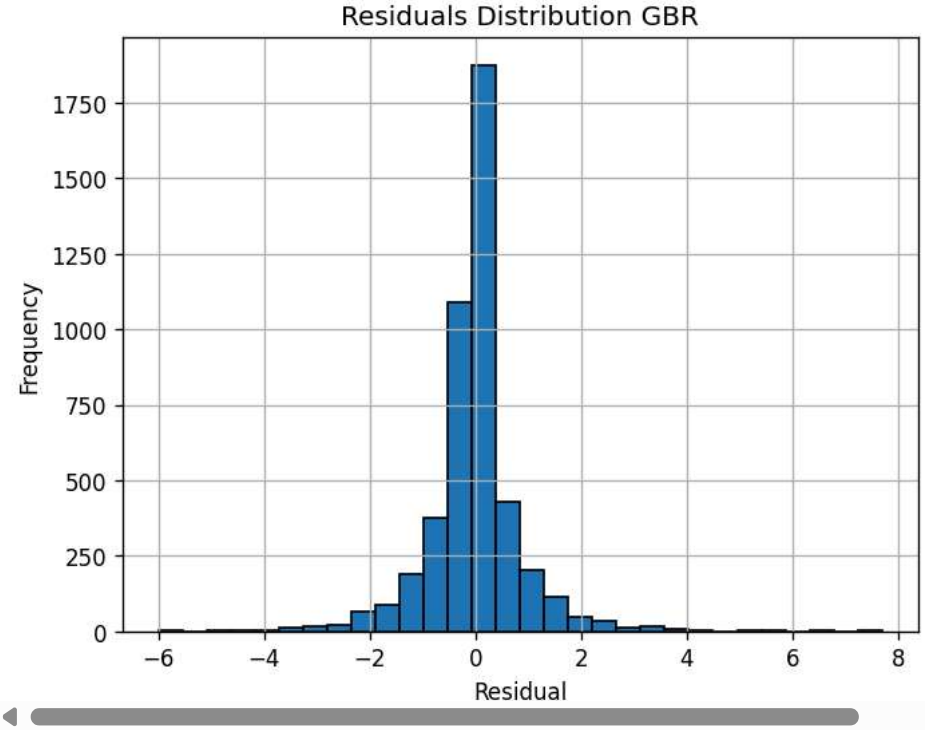

```
# Feature importances for GBR
importances = best_gbr.feature_importances_
indices = np.argsort(importances)[::-1]
top_features = np.array(feature_names)[indices[:10]]
```

```
plt.figure(figsize=(10, 6))
plt.title("Top 10 Feature Importances (Gradient Boosting)")
plt.bar(range(10), importances[indices[:10]], align="center")
plt.xticks(range(10), top_features, rotation=90)
plt.tight_layout()
plt.show()
```

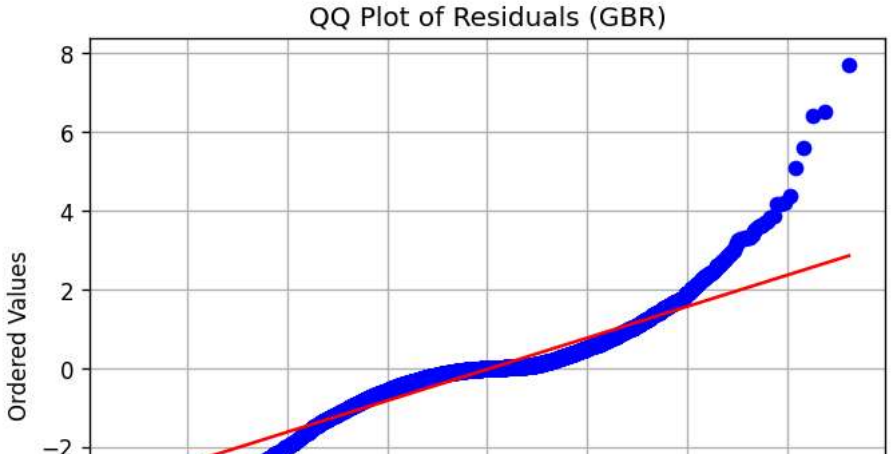



```
# Residual plot for GBR
residuals = y - y_pred

plt.figure(figsize=(8, 6))
plt.scatter(y, residuals, alpha=0.6)
plt.axhline(0, color='gray', linestyle='--')
plt.xlabel('True Bandgap (eV)')
plt.ylabel('Residuals (True - Predicted)')
plt.title('Residuals vs True Values (GBR)')
plt.grid(True)
plt.show()

plt.figure()
plt.hist(residuals, bins=30, edgecolor='k')
plt.title("Residuals Distribution GBR")
plt.xlabel("Residual")
plt.ylabel("Frequency")
plt.grid(True)
plt.show()
```






```
stats.probplot(residuals, dist="norm", plot=plt)
plt.title("QQ Plot of Residuals (GBR)")
plt.grid(True)
plt.show()
```


QQ Plot of Residuals (GBR)
Ordered Values
8
6
4
2
0
−2

```
# featurizing composition
from sklearn.impute import SimpleImputer

featurizer = MultipleFeaturizer([
    ElementProperty.from_preset('magpie'),
])

df['composition_obj'] = df['composition'].apply(str_to_composition)
feature_df = featurizer.featurize_dataframe(df, 'composition_obj', ignore_errors=True)

# keeping numeric features only
X_raw = feature_df.select_dtypes(include=[np.number])
X_raw = X_raw.replace([np.inf, -np.inf], np.nan)
X_raw = X_raw.dropna(axis=1, how='all')  # Remove all-NaN columns

# imputing missing values
imputer = SimpleImputer(strategy='mean')
X = imputer.fit_transform(X_raw)
y = df['gap expt'].values
```

```
/usr/local/lib/python3.11/dist-packages/matminer/utils/data.py:326: UserWarning: MagpieData(impute_nan=False):
In a future release, impute_nan will be set to True by default.
                    This means that features that are missing or are NaNs for elements
                    from the data source will be replaced by the average of that value
                    over the available elements.
                    This avoids NaNs after featurization that are often replaced by
                    dataset-dependent averages.
  warnings.warn(f"{self.__class__.__name__}(impute_nan=False):\n" + IMPUTE_NAN_WARNING)
MultipleFfeaturizer: 100%                                                6354/6354 [00:47<00:00, 105.13it/s]
```

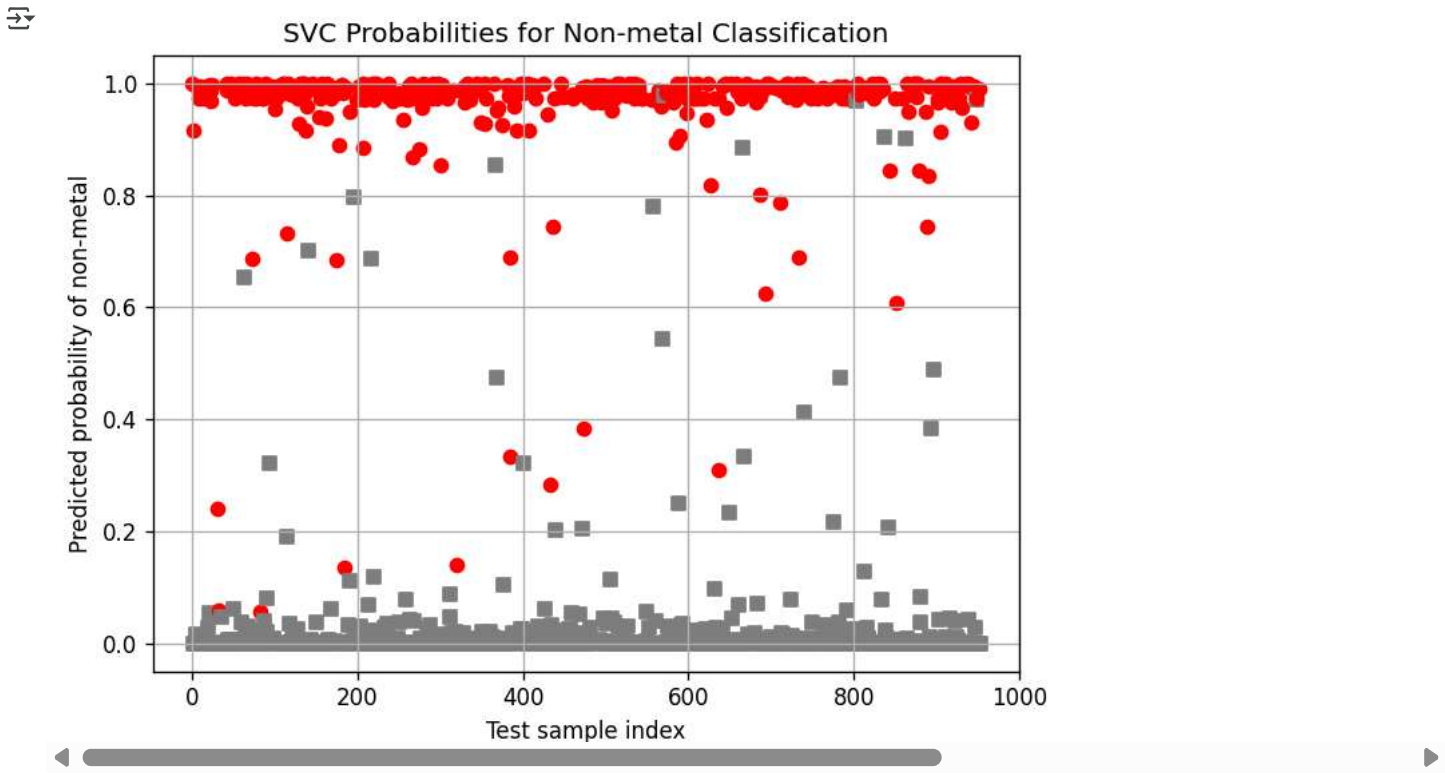
```
print('Feature matrix:', X.shape)
metals = y < 0.1
```

```
Feature matrix: (6354, 133)
```

```
from sklearn.preprocessing import StandardScaler
X_scaled = StandardScaler().fit_transform(X)


# SVC -> SVR
X_train, X_test, y_train, y_test, m_train, m_test = train_test_split(
    X_scaled, y, metals, test_size=0.15, stratify=metals, random_state=SEED)

svc = SVC(class_weight='balanced', kernel='rbf', C=10, gamma=0.01, probability=True, random_state=SEED)
svc.fit(X_train, m_train)

# only training SVR on non-metals
svr = SVR(kernel='rbf', C=100, gamma=0.01, epsilon=0.1)
svr.fit(X_train[~m_train], y_train[~m_train])

m_pred = svc.predict(X_test)
preds = np.where(m_pred, 0.0, svr.predict(X_test))

mae = mean_absolute_error(y_test, preds)
rmse = root_mean_squared_error(y_test, preds)
print(f"SVC→SVR  MAE = {mae:.3f} eV  RMSE = {rmse:.3f} eV")
```

```
SVC→SVR  MAE = 0.055 eV  RMSE = 0.124 eV
```

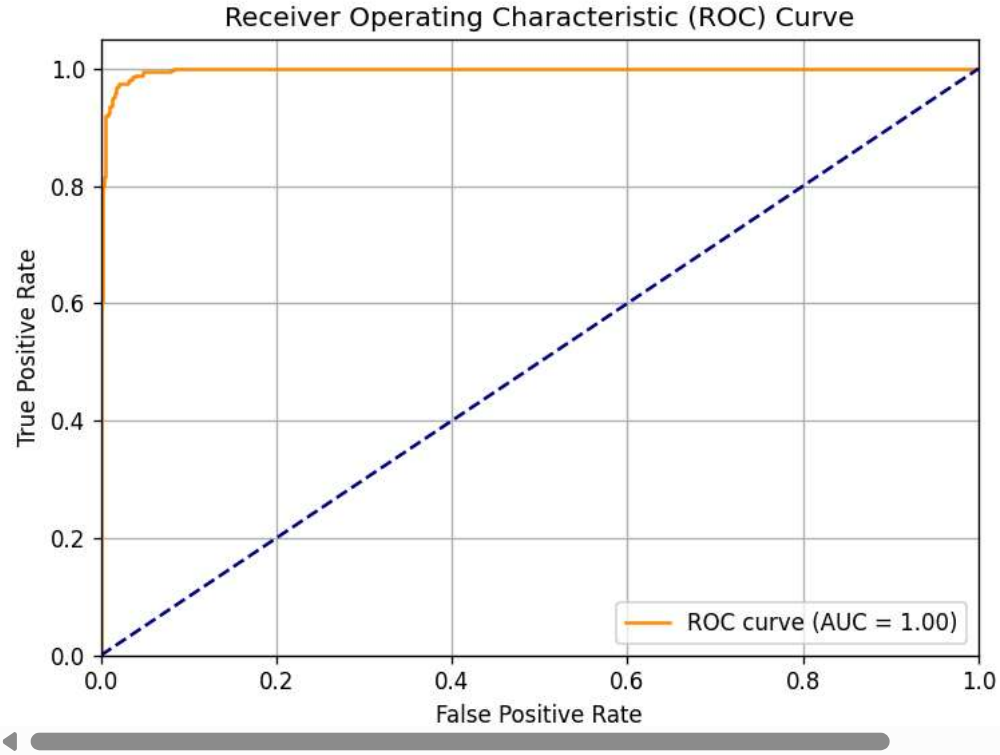
```
probs = svc.predict_proba(X_test)[:, 1]

# Scatter plot: red for non-metals, gray for metals
plt.figure()
for i in range(len(probs)):
    color = 'red' if m_test[i] else 'gray'
    marker = 'o' if m_test[i] else 's'
    plt.scatter(i, probs[i], color=color, marker=marker)
plt.xlabel('Test sample index')
plt.ylabel('Predicted probability of non-metal')
plt.title('SVC Probabilities for Non-metal Classification')
plt.grid(True)
plt.tight_layout()
plt.show()
```



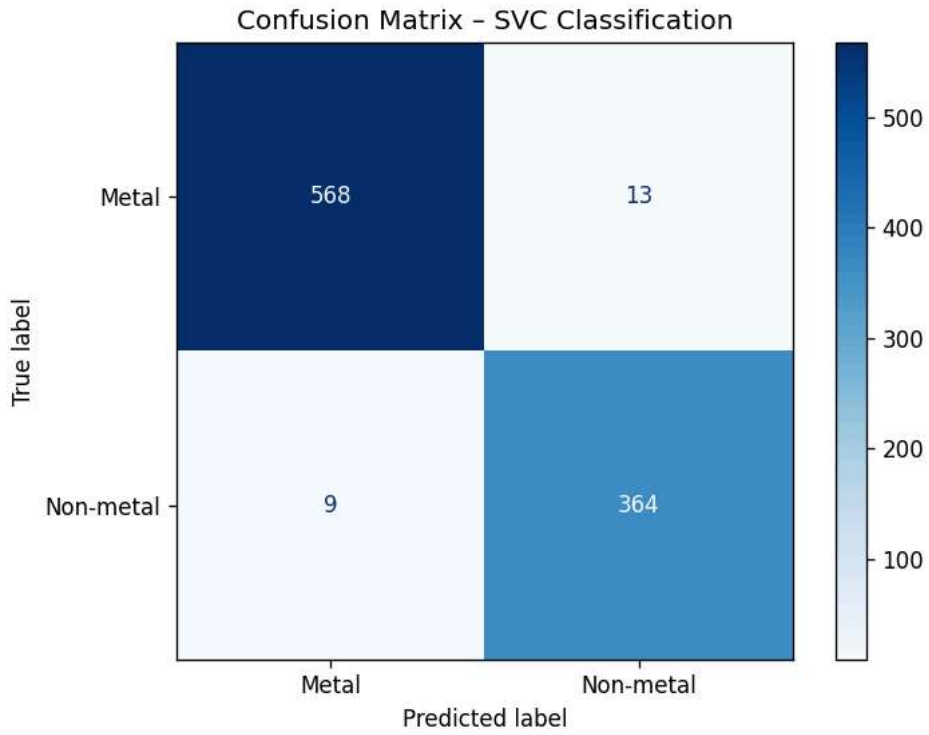

```
accuracy = accuracy_score(m_test, m_pred)
print(f"SVC Accuracy: {accuracy:.2f}")
```

```
SVC Accuracy: 0.98
```

```
from sklearn.metrics import roc_curve, auc

fpr, tpr, _ = roc_curve(m_test, probs)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.grid(True)
plt.tight_layout()
plt.show()
```

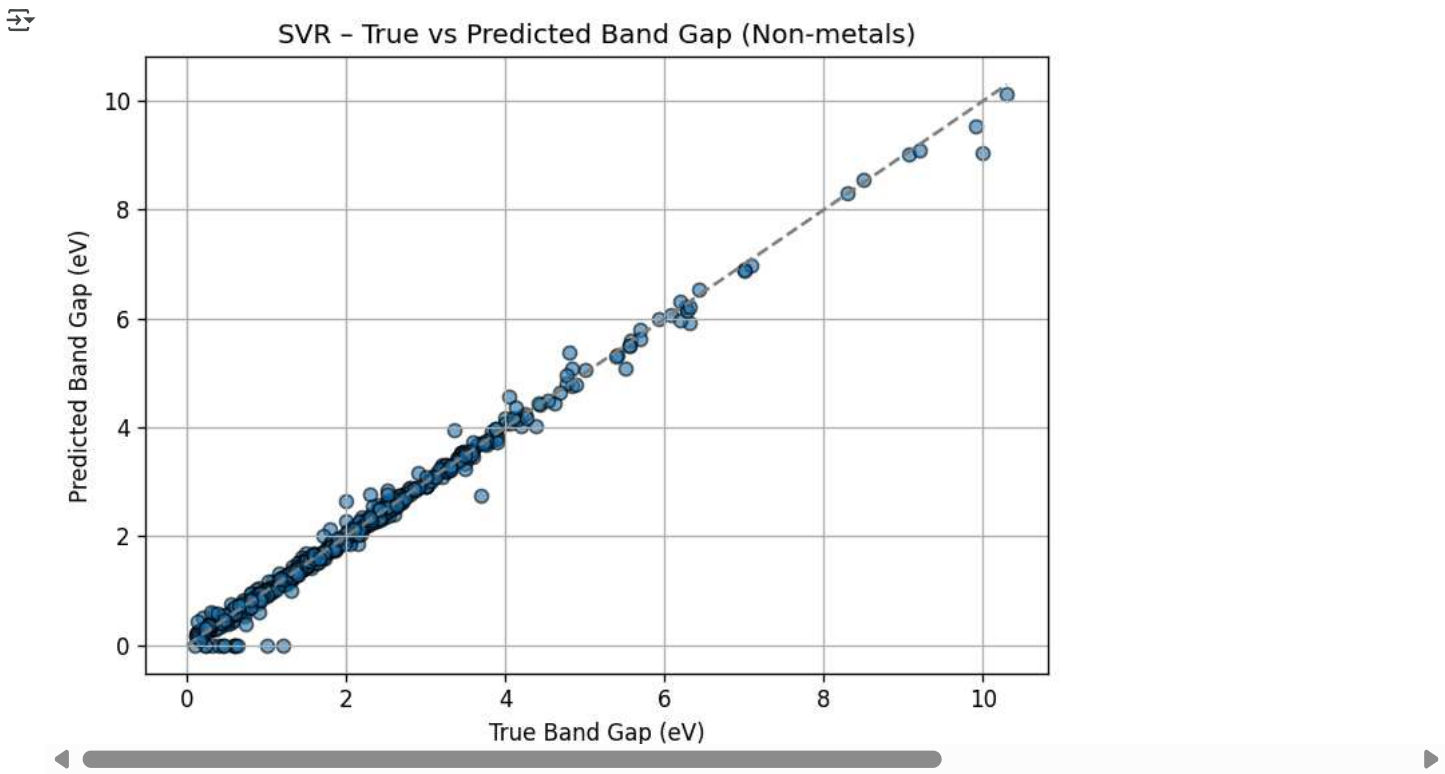



```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

cm = confusion_matrix(m_test, m_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Metal", "Non-metal"])
disp.plot(cmap='Blues')
plt.title("Confusion Matrix – SVC Classification")
plt.grid(False)
plt.tight_layout()
plt.show()
```




```
true_nonmetals = y_test[~m_test]
pred_nonmetals = preds[~m_test]

plt.figure()
```

```
plt.scatter(true_nonmetals, pred_nonmetals, alpha=0.6, edgecolor='k')
plt.plot([0, max(true_nonmetals)], [0, max(true_nonmetals)], '--', color='gray')
plt.xlabel('True Band Gap (eV)')
plt.ylabel('Predicted Band Gap (eV)')
plt.title('SVR – True vs Predicted Band Gap (Non-metals)')
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
residuals = pred_nonmetals - true_nonmetals

plt.figure()
plt.scatter(true_nonmetals, residuals, alpha=0.6, edgecolor='k')
plt.axhline(0, linestyle='--', color='gray')
plt.xlabel('True Band Gap (eV)')
plt.ylabel('Residuals (Predicted - True)')
plt.title('SVR – Residual Plot (Non-metals)')
plt.grid(True)
plt.tight_layout()
plt.show()
```