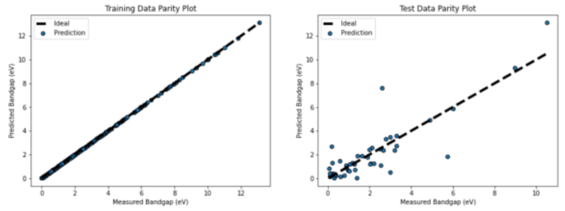
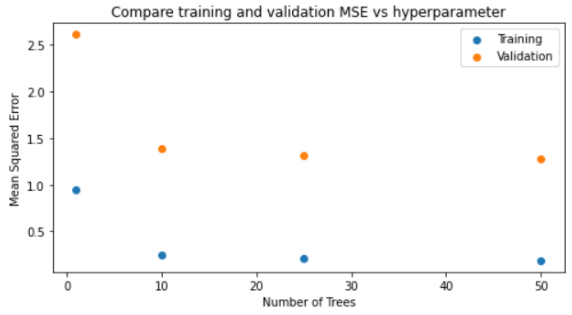
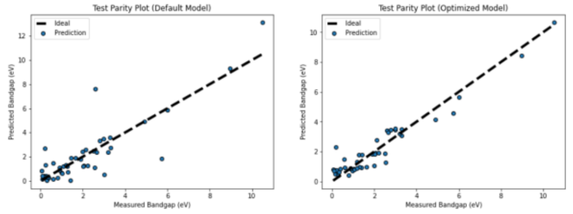
Assessment Figures



	Error Metric	Training Data	Test Data	Note
0	RMSE	0.0003 (eV)	1.2398 (eV)	(0.0 for perfect prediction)
1	RMSE/std	0.0001	0.5771	(0.0 for perfect prediction)
2	MAE	0.0 (eV)	0.723 (eV)	(0.0 for perfect prediction)
3	R2	1	0.6669	(1.0 for perfect prediction)



```
Minimum Mean Squared Error:  1.281
Number of Trees at minimum:  50
```



	Error Metric	Test Set (Default Model)	Test Set (Optimized Model)	Note
0	RMSE	1.2398 (eV)	0.6089 (eV)	(0.0 for perfect prediction)
1	RMSE/std	0.5771	0.2834	(0.0 for perfect prediction)
2	MAE	0.723 (eV)	0.4809 (eV)	(0.0 for perfect prediction)
3	R2	0.6669	0.9196	(1.0 for perfect prediction)

ML4ER
Band Gap Prediction

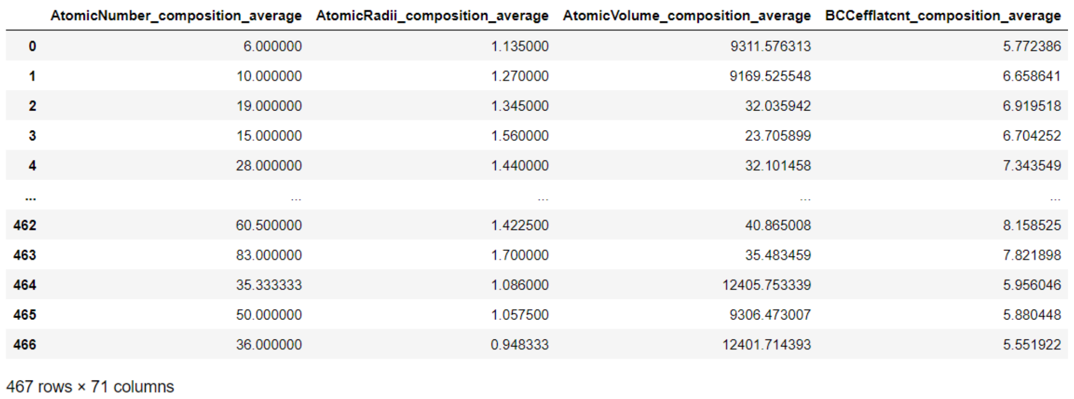
Muhammad Zain Azeem,
Informatics Skunkworks (**non-credits),** Week 2
24/07/2024

Progress

FEATURE NORMALIZATION

- Machine Learning algorithms are quite sensitive to features in terms of shape and size. To resolve such a problem best approach is to make features similar by rescaling. Here, we did rescaling using MinMaxScaler().

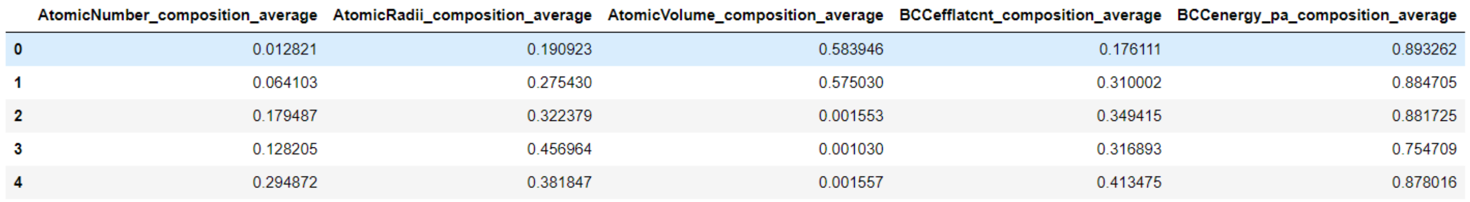
Before scaling



	AtomicNumber_composition_average	AtomicRadii_composition_average	AtomicVolume_composition_average	BCCefflatcnt_composition_average
0	6.000000	1.135000	9311.576313	5.772386
1	10.000000	1.270000	9169.525548	6.658641
2	19.000000	1.345000	32.035942	6.919518
3	15.000000	1.560000	23.705899	6.704252
4	28.000000	1.440000	32.101458	7.343549
...	...	...	...	...
462	60.500000	1.422500	40.865008	8.158525
463	83.000000	1.700000	35.483459	7.821898
464	35.333333	1.086000	12405.753339	5.956046
465	50.000000	1.057500	9306.473007	5.880448
466	36.000000	0.948333	12401.714393	5.551922

467 rows × 71 columns

After scaling



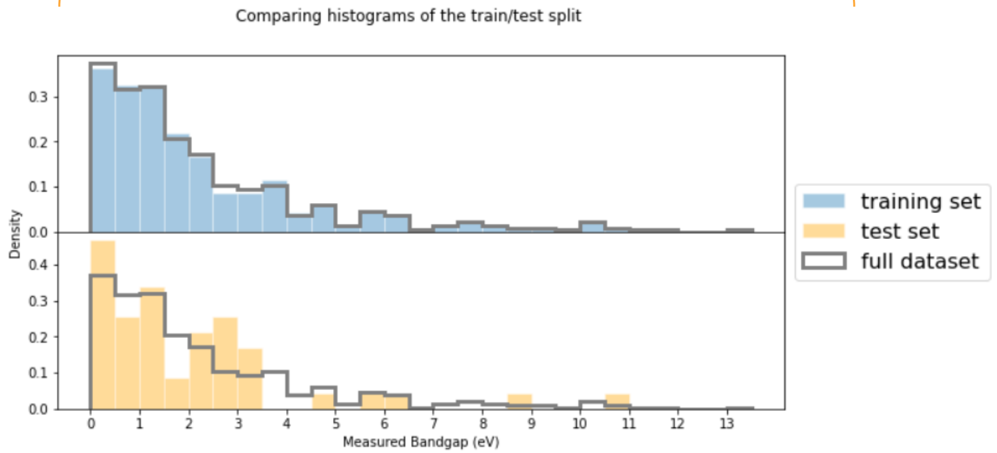
	AtomicNumber_composition_average	AtomicRadii_composition_average	AtomicVolume_composition_average	BCCefflatcnt_composition_average	BCCenergy_pa_composition_average
0	0.012821	0.190923	0.583946	0.176111	0.893262
1	0.064103	0.275430	0.575030	0.310002	0.884705
2	0.179487	0.322379	0.001553	0.349415	0.881725
3	0.128205	0.456964	0.001030	0.316893	0.754709
4	0.294872	0.381847	0.001557	0.413475	0.878016

Progress

SETUP FOR MODEL EVALUATION

- In this part, we are doing an unbiased estimation of model error by employing the cross-validation technique, in which we created training and testing sets.
- In this case, we used testing/training sets 10/90, which means 10% of the dataset will be used as a testing set while the remaining will be considered a training set.

Output

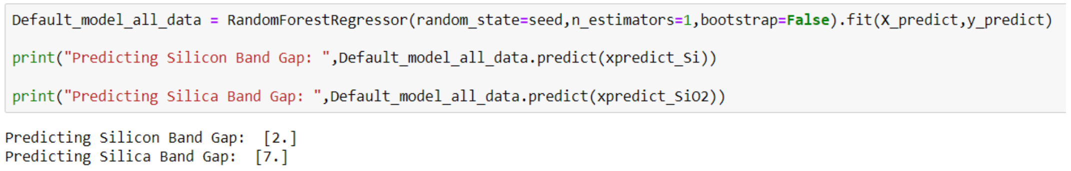


Progress

- Fitting is highly perfect for the training data set

FITTING THE DECISION TREE MODEL

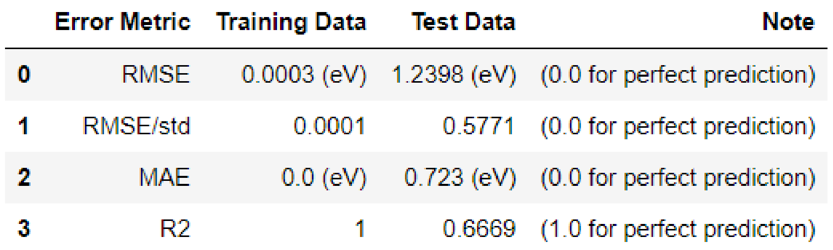
- Applied RandomForestRegressor for determining the band gap



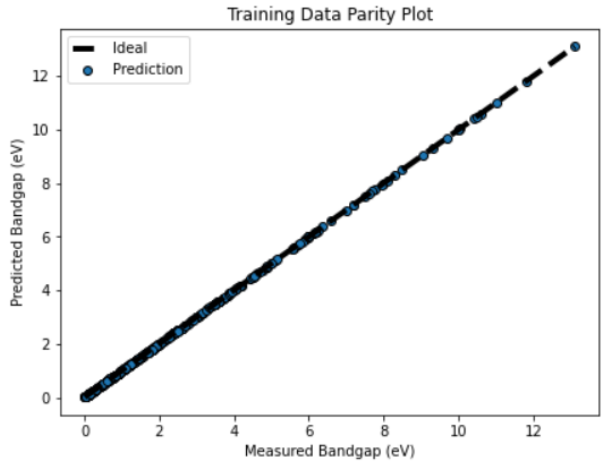
```
Default_model_all_data = RandomForestRegressor(random_state=seed,n_estimators=1,bootstrap=False).fit(X_predict,y_predict)

print("Predicting Silicon Band Gap: ",Default_model_all_data.predict(xpredict_Si))

print("Predicting Silica Band Gap: ",Default_model_all_data.predict(xpredict_SiO2))
```

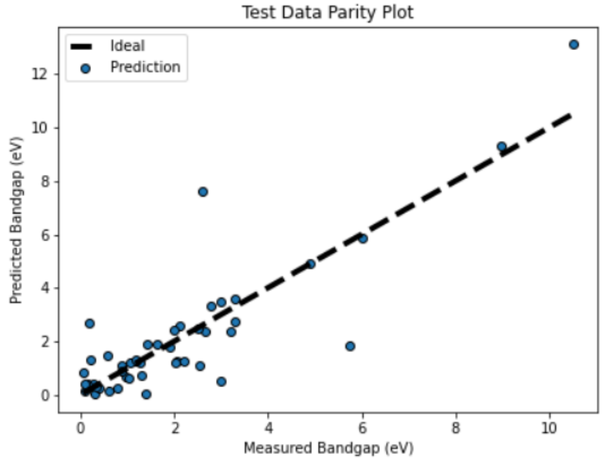
```
Predicting Silicon Band Gap:  [2.]
Predicting Silica Band Gap:  [7.]
```

EVALUATING MODEL PERFORMANCE ON TRAINING AND TEST DATA



	Error Metric	Training Data	Test Data	Note
0	RMSE	0.0003 (eV)	1.2398 (eV)	(0.0 for perfect prediction)
1	RMSE/std	0.0001	0.5771	(0.0 for perfect prediction)
2	MAE	0.0 (eV)	0.723 (eV)	(0.0 for perfect prediction)
3	R2	1	0.6669	(1.0 for perfect prediction)





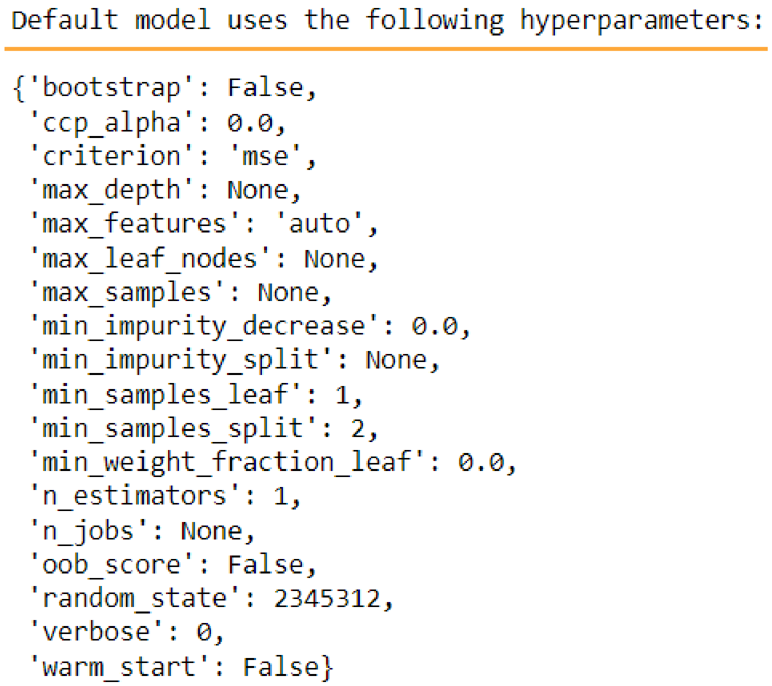
- A point closer to the straight line shows good fitting results however point far away from it indicating poor fitting

Progress

IMPROVING THE MODEL BY OPTIMIZING HYPERPARAMETERS

- Used “**hyperparameters**” to overcome the poor results of the training dataset (previous slides).

- Its **main purpose** is to control the model learning process and how its fitting works.



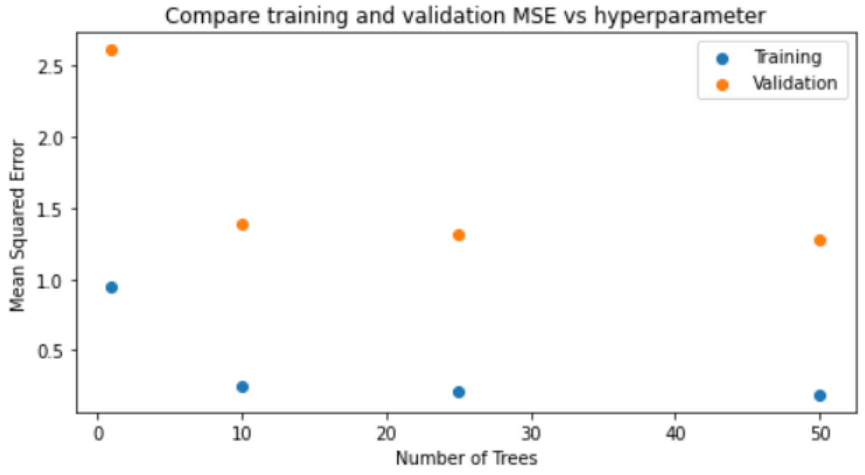
```
Default model uses the following hyperparameters:

{'bootstrap': False,
 'ccp_alpha': 0.0,
 'criterion': 'mse',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 1,
 'n_jobs': None,
 'oob_score': False,
 'random_state': 2345312,
 'verbose': 0,
 'warm_start': False}
```

Progress

CROSS-VALIDATION STRATEGY: REPEATEDKFOLD CROSS-VALIDATION.

Here we are combining cross-validation with a grid of hyperparameters using scikit-learn “Grid Search” approach

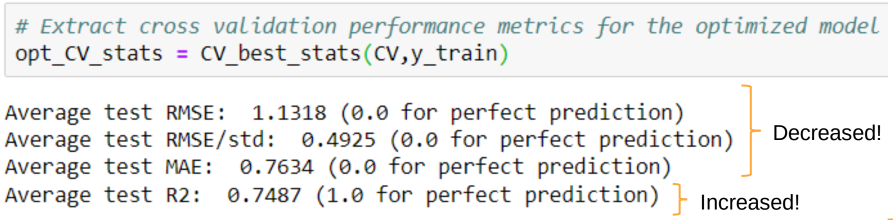
- MSE for both training and validation dropped with the increase of decision trees, indicating better performance



```
Minimum Mean Squared Error:  1.281
Number of Trees at minimum:  50
```

Progress

DEFAULT VS. OPTIMIZED MODEL: CROSS-VALIDATION PERFORMANCE



```
# Extract cross validation performance metrics for the optimized model
opt_CV_stats = CV_best_stats(CV,y_train)
```

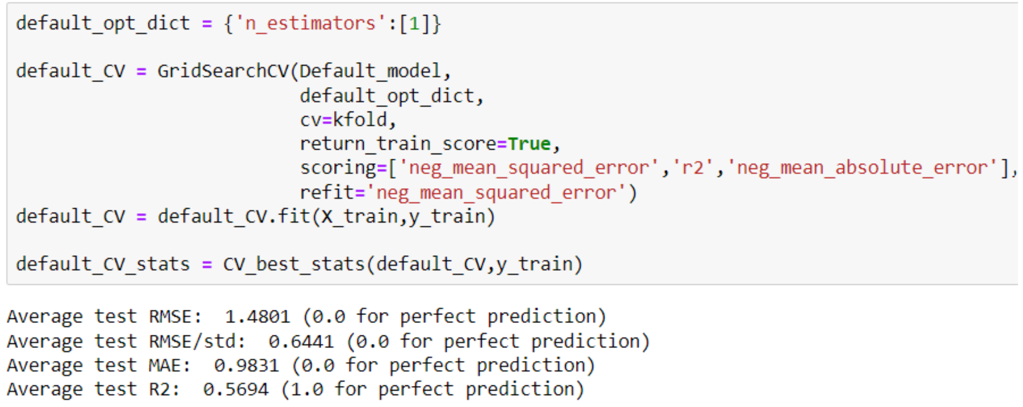
```
Average test RMSE:  1.1318 (0.0 for perfect prediction)
Average test RMSE/std:  0.4925 (0.0 for perfect prediction)
Average test MAE:  0.7634 (0.0 for perfect prediction)
Average test R2:  0.7487 (1.0 for perfect prediction)
```

Decreased!

Increased!

- Results after model optimization where we have 50 decision trees

- Comparing optimized model results with the default model



```
default_opt_dict = {'n_estimators':[1]}

default_CV = GridSearchCV(Default_model,
                          default_opt_dict,
                          cv=kfold,
                          return_train_score=True,
                          scoring=['neg_mean_squared_error','r2','neg_mean_absolute_error'],
                          refit='neg_mean_squared_error')
default_CV = default_CV.fit(X_train,y_train)

default_CV_stats = CV_best_stats(default_CV,y_train)
```

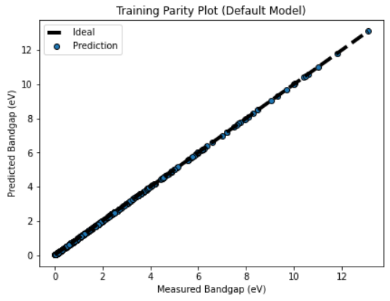
```
Average test RMSE:  1.4801 (0.0 for perfect prediction)
Average test RMSE/std:  0.6441 (0.0 for perfect prediction)
Average test MAE:  0.9831 (0.0 for perfect prediction)
Average test R2:  0.5694 (1.0 for perfect prediction)
```

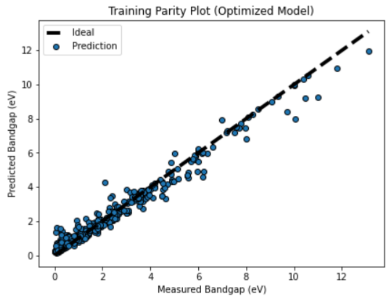
Progress

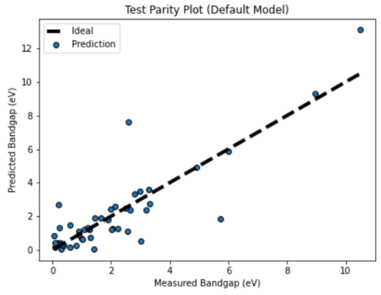
DEFAULT VS. OPTIMIZED MODEL: TRAINING AND TEST DATA PERFORMANCE

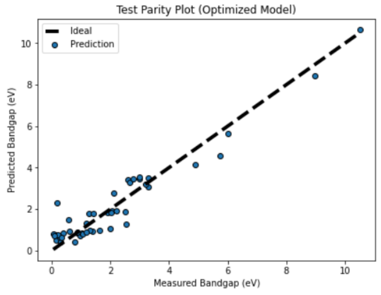
OPTIMIZED MODEL

DEFAULT MODEL









	Error Metric	Training Set (Default Model)	Training Set (Optimized Model)	Note
0	RMSE	0.0003 (eV)	0.4241 (eV)	(0.0 for perfect prediction)
1	RMSE/std	0.0001	0.1845	(0.0 for perfect prediction)
2	MAE	0.0 (eV)	0.2766 (eV)	(0.0 for perfect prediction)
3	R2	1	0.9659	(1.0 for perfect prediction)

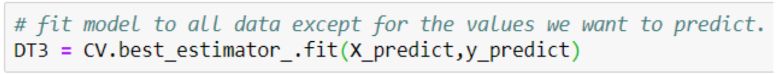
	Error Metric	Test Set (Default Model)	Test Set (Optimized Model)	Note
0	RMSE	1.2398 (eV)	0.6089 (eV)	(0.0 for perfect prediction)
1	RMSE/std	0.5771	0.2834	(0.0 for perfect prediction)
2	MAE	0.723 (eV)	0.4809 (eV)	(0.0 for perfect prediction)
3	R2	0.6669	0.9196	(1.0 for perfect prediction)

- A high accurate prediction in the optimized model in contrast to the default model

Progress

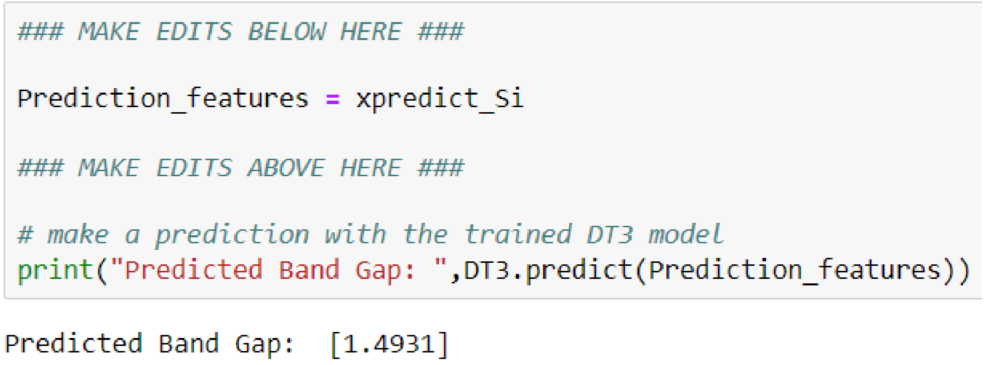
MAKING PREDICTIONS

DT3 model fitting



```
# fit model to all data except for the values we want to predict.
DT3 = CV.best_estimator_.fit(X_predict,y_predict)
```

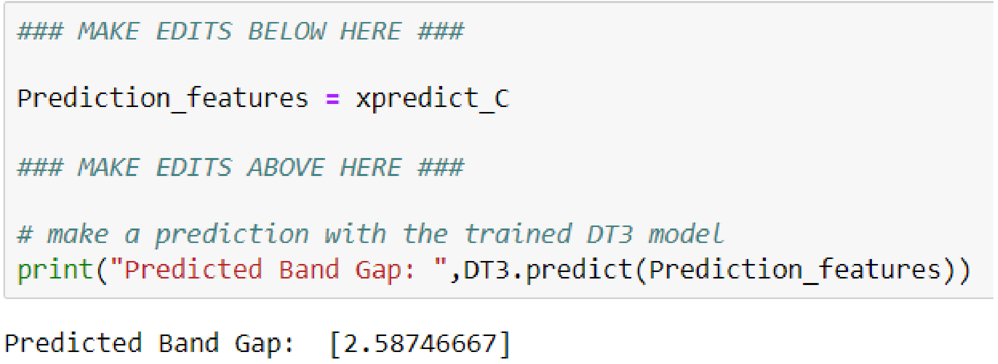
Prediction for Si



```
### MAKE EDITS BELOW HERE ###

Prediction_features = xpredict_Si

### MAKE EDITS ABOVE HERE ###

# make a prediction with the trained DT3 model
print("Predicted Band Gap: ",DT3.predict(Prediction_features))
```

```
Predicted Band Gap:  [1.4931]
```

Prediction for C



```
### MAKE EDITS BELOW HERE ###

Prediction_features = xpredict_C

### MAKE EDITS ABOVE HERE ###

# make a prediction with the trained DT3 model
print("Predicted Band Gap: ",DT3.predict(Prediction_features))
```

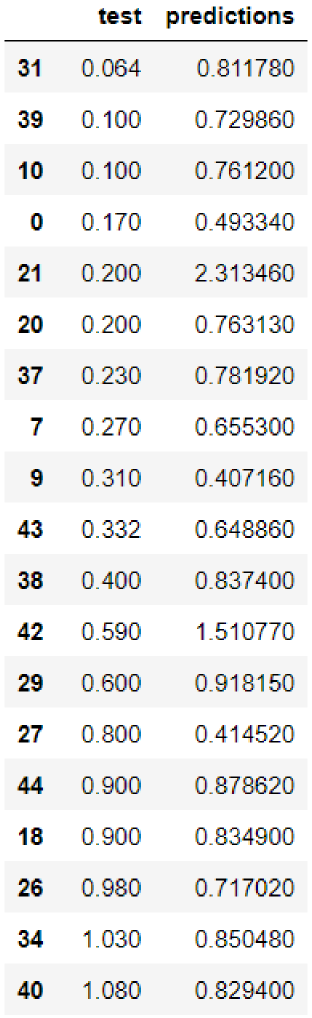
```
Predicted Band Gap:  [2.58746667]
```

Progress

FINAL MODEL PREDICTIONS

```
# combine previous data into one dataframe for visualization
predictions_combined = pd.DataFrame(list(zip(y_test,Test_predictions2)),columns=['test','predictions'])
```

```
# sort on the Test values from low to high
predictions_combined.sort_values("test")
```



	test	predictions
31	0.064	0.811780
39	0.100	0.729860
10	0.100	0.761200
0	0.170	0.493340
21	0.200	2.313460
20	0.200	0.763130
37	0.230	0.781920
7	0.270	0.655300
9	0.310	0.407160
43	0.332	0.648860
38	0.400	0.837400
42	0.590	1.510770
29	0.600	0.918150
27	0.800	0.414520
44	0.900	0.878620
18	0.900	0.834900
26	0.980	0.717020
34	1.030	0.850480
40	1.080	0.829400