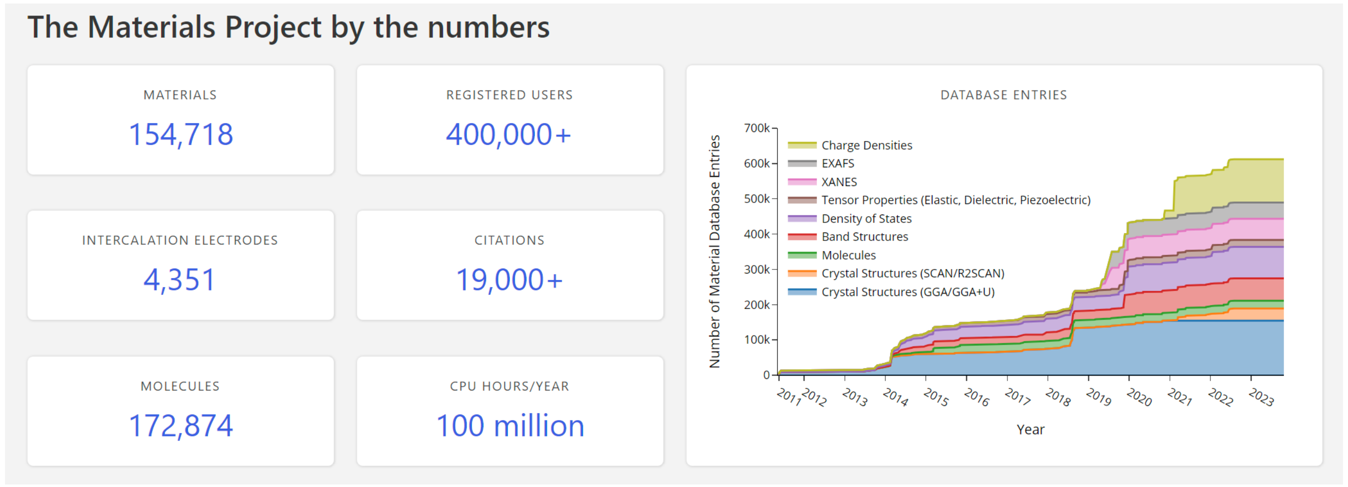
실습 자료 안내

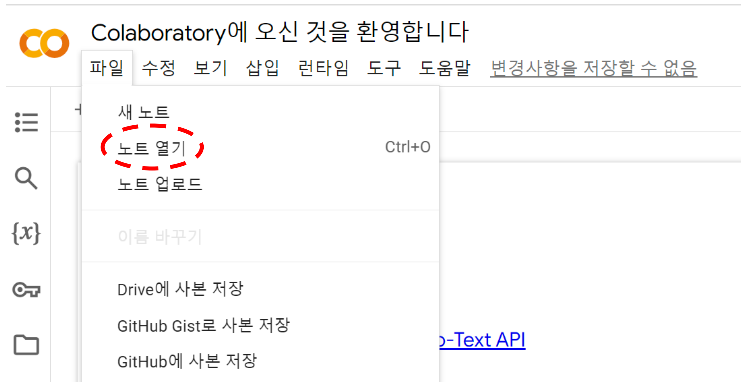


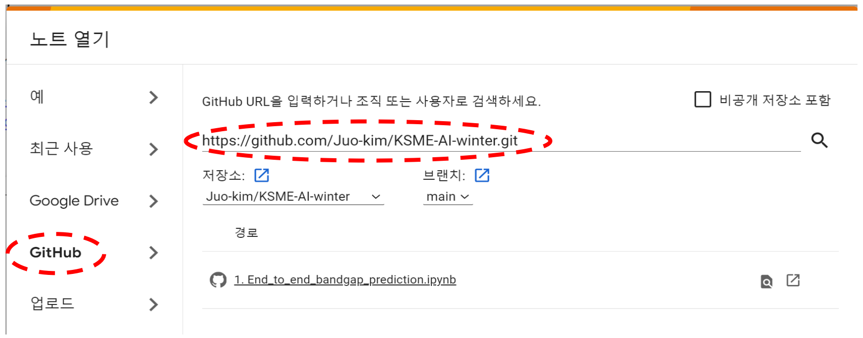
The Materials Project by the numbers

MATERIALS 154,718	REGISTERED USERS 400,000+
INTERCALATION ELECTRODES 4,351	CITATIONS 19,000+
MOLECULES 172,874	CPU HOURS/YEAR 100 million

- 15만여개 Inorganic 물질에 대한 DFT 계산 정보를 포함하고 있는 Materials Project 데이터 베이스 활용
- 물질의 전자 상태를 나타내는 값인 Bandgap 예측하는 모델 생성

1. Colab 접속 (https://colab.google/)
2. 파일 ➔ 노트 열기 ➔ GitHub ➔ URL 검색





GitHub URL을 입력하거나 조직 또는 사용자로 검색하세요.

https://github.com/Juo-kim/KSME-AI-winter.git

저장소: Juo-kim/KSME-AI-winter 브랜치: main

1. End_to_end_bandgap_prediction.ipynb

URL : https://github.com/Juo-kim/KSME-AI-winter.git

Computational Science
Artificial Intelligence

Soongsil University

KSME-AI-winter

Juo Kim

CS-AI Lab
School of Mechanical Engineering
Soongsil University

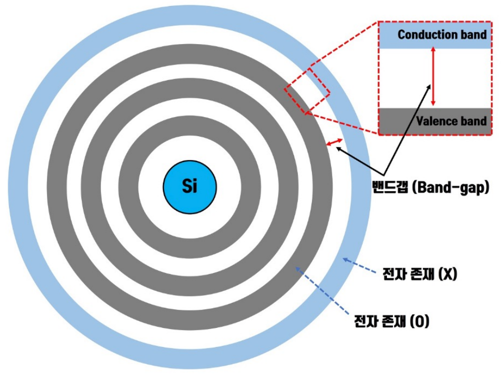
Contact : rlawndh14@gmail.com

Computational Science
Artificial Intelligence

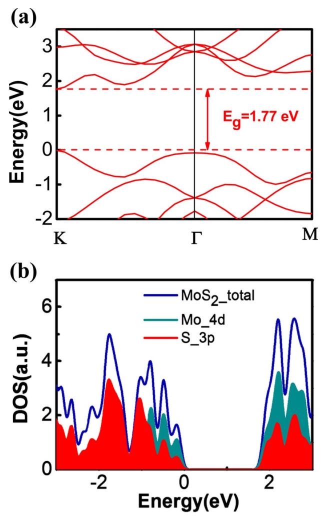
Soongsil University

Bandgap?

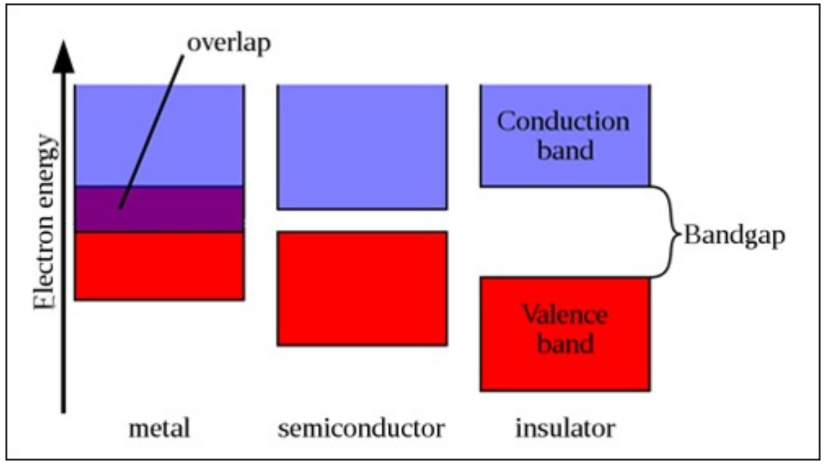
- Energy gap : 전자가 존재하고 있는 가장 높은 에너지 레벨 (Valence Band) 부터 전자가 존재하지 않는 가장 낮은 에너지레벨 (Conduction Band) 사이의 에너지 준위
- 물질이 외부에서 충분한 에너지를 공급받게 된다면, Valence band에 위치한 전자가 Conduction Band로 뛰어넘는 것이 가능함
- Bandgap의 크기에 따라 물질의 전기적 성질을 확인할 수 있음
- Metal = 0 eV, 0 eV < Semiconductor < 4 eV, Insulator ≥ 4 eV
- 전기적 성질을 알아보기 위해서 매우 중요하지만, Bandgap 계산을 위해 필요한 계산적, 실험적 cost가 큼



https://j-science.tistory.com/11



DOI: 10.1186/1556-276X-9-676

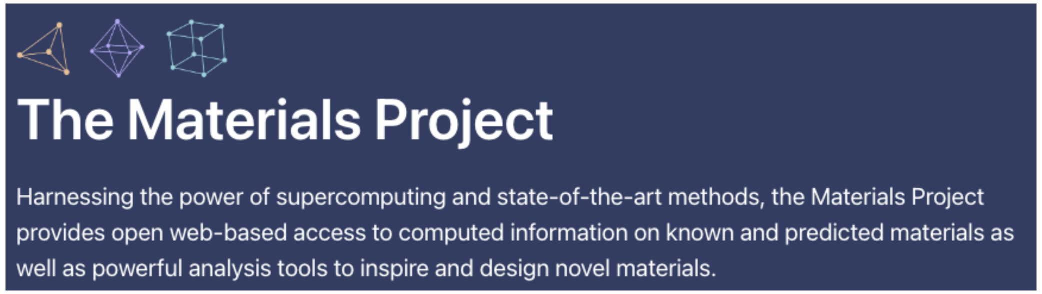


http://solarcellcentral.com/junction_page.html

Materials Project

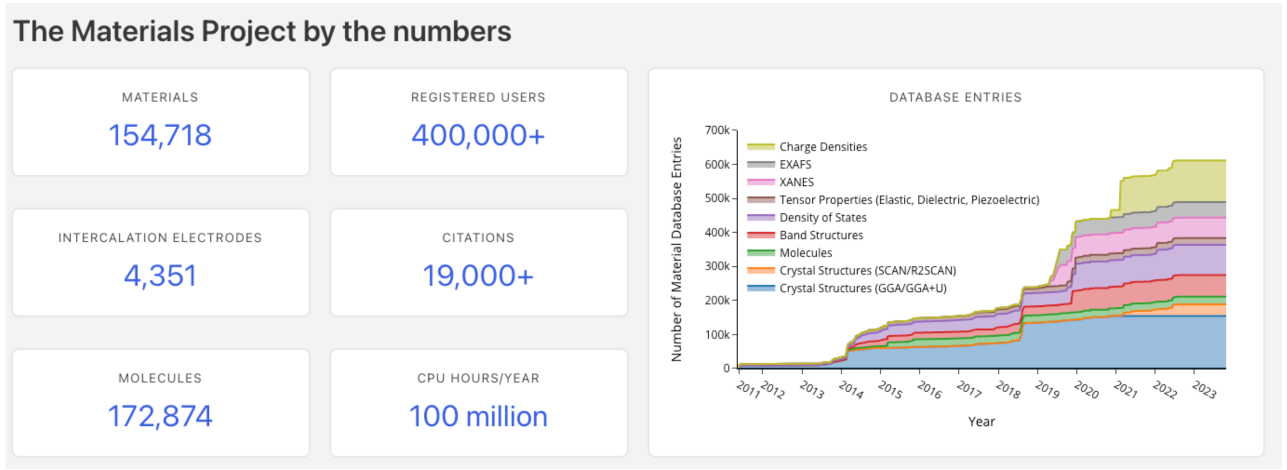
- Open source Inorganic database 중 하나
- 154,718개의 Inorganic 물질에 대한 DFT 계산 기반의 물성 포함
- Bandgap, Energy above hull, Space group number, Crystal structure 등 많은 계산 결과

➔ <u>Materials Project database로 bandgap을 예측할 수 있는 regression model을 만드는 것이 실습의 목표</u>



The Materials Project

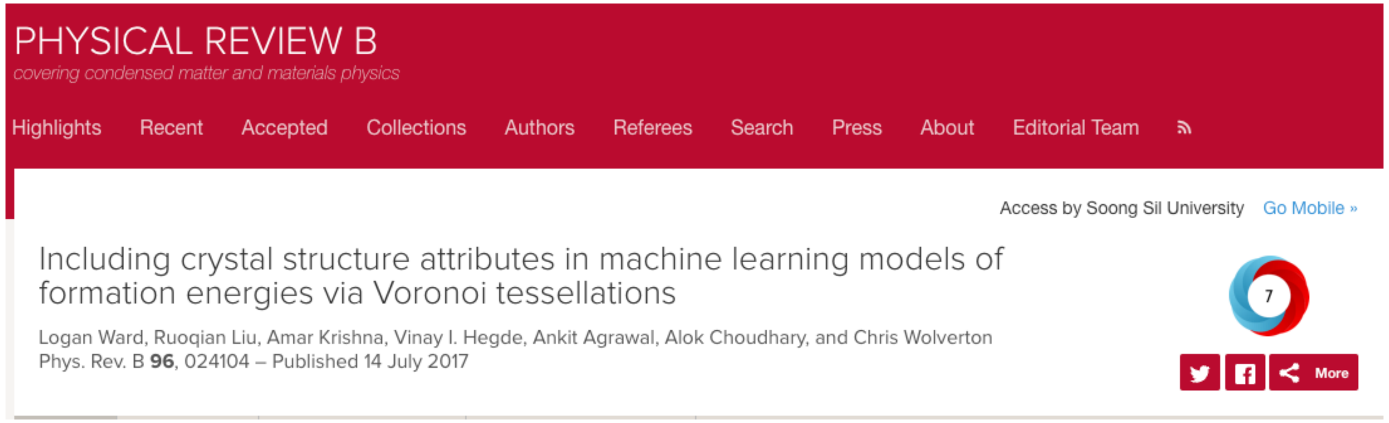
Harnessing the power of supercomputing and state-of-the-art methods, the Materials Project provides open web-based access to computed information on known and predicted materials as well as powerful analysis tools to inspire and design novel materials.



The Materials Project by the numbers

MATERIALS 154,718	REGISTERED USERS 400,000+
INTERCALATION ELECTRODES 4,351	CITATIONS 19,000+
MOLECULES 172,874	CPU HOURS/YEAR 100 million

Chemical Descriptor

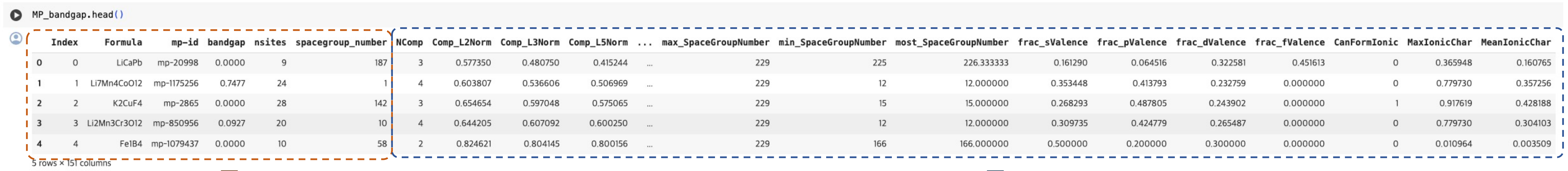


Including crystal structure attributes in machine learning models of formation energies via Voronoi tessellations

Logan Ward, Ruoqian Liu, Amar Krishna, Vinay I. Hegde, Ankit Agrawal, Alok Choudhary, and Chris Wolverton
Phys. Rev. B **96**, 024104 – Published 14 July 2017

Meaning of chemical descriptors (CDs)	Number of attributes		Attributes
Stoichiometric attributes	6		Ncomp, Comp_L2Norm, Comp_L3Norm, Comp_L5Norm, Comp_L7Norm, Comp_L10Norm
Elemental-property-based attributes	132	Mean (22)	Atomic Number, MendeleevNumber, AtomicWeight, MeltingT, Column, Row, CovalentRadius, Electronegativity, NsValence, NpValence, NdValence, NfValence, NValance, NsUnfilled, NpUnfilled, NdUnfilled, NfUnfilled, NUnfilled, GSvolume_pa, GSbandgap, GSmagmom, SpaceGroupNumber
		Range (22)	
		Dev (22)	
		Max (22)	
		Min (22)	
		Most (22)	
Valance orbital occupation attributes	4		frac_sValence, frac_pValence, frac_dValence, frac_fValence
Ionic compound attributes	3		CanFormIonic, MaxIonicChar, MeanIonicChar

Take a quick look at the Data structure



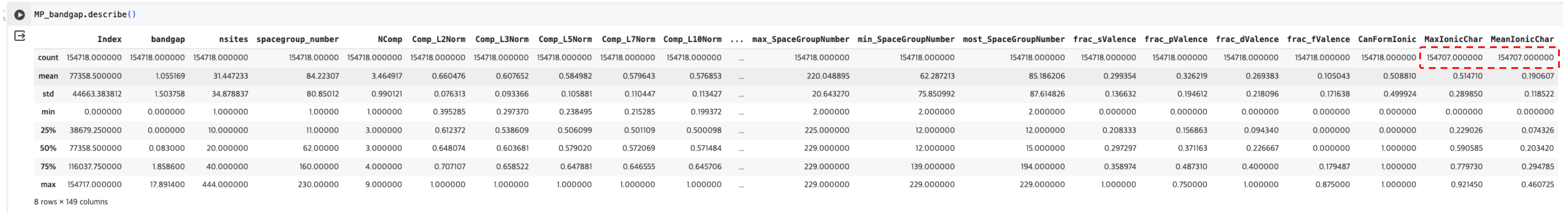
```
MP_bandgap.head()
```

	Index	Formula	mp-id	bandgap	nsites	spacegroup_number	NComp	Comp_L2Norm	Comp_L3Norm	Comp_L5Norm	...	max_SpaceGroupNumber	min_SpaceGroupNumber	most_SpaceGroupNumber	frac_sValence	frac_pValence	frac_dValence	frac_fValence	CanFormIonic	MaxIonicChar	MeanIonicChar
0	0	LiCaPb	mp-20998	0.0000	9	187	3	0.577350	0.480750	0.415244	...	229	225	226.333333	0.161290	0.064516	0.322581	0.451613	0	0.365948	0.160765
1	1	Li7Mn4CoO12	mp-1175256	0.7477	24	1	4	0.603807	0.536606	0.506969	...	229	12	12.000000	0.353448	0.413793	0.232759	0.000000	0	0.779730	0.357256
2	2	K2CuF4	mp-2865	0.0000	28	142	3	0.654654	0.597048	0.575065	...	229	15	15.000000	0.268293	0.487805	0.243902	0.000000	1	0.917619	0.428188
3	3	Li2Mn3Cr3O12	mp-850956	0.0927	20	10	4	0.644205	0.607092	0.600250	...	229	12	12.000000	0.309735	0.424779	0.265487	0.000000	0	0.779730	0.304103
4	4	FeIB4	mp-1079437	0.0000	10	58	2	0.824621	0.804145	0.800156	...	229	166	166.000000	0.500000	0.200000	0.300000	0.000000	0	0.010964	0.003509

5 rows × 151 columns

Properties from Materials Project

features created by Chemical Descriptor



```
MP_bandgap.describe()
```

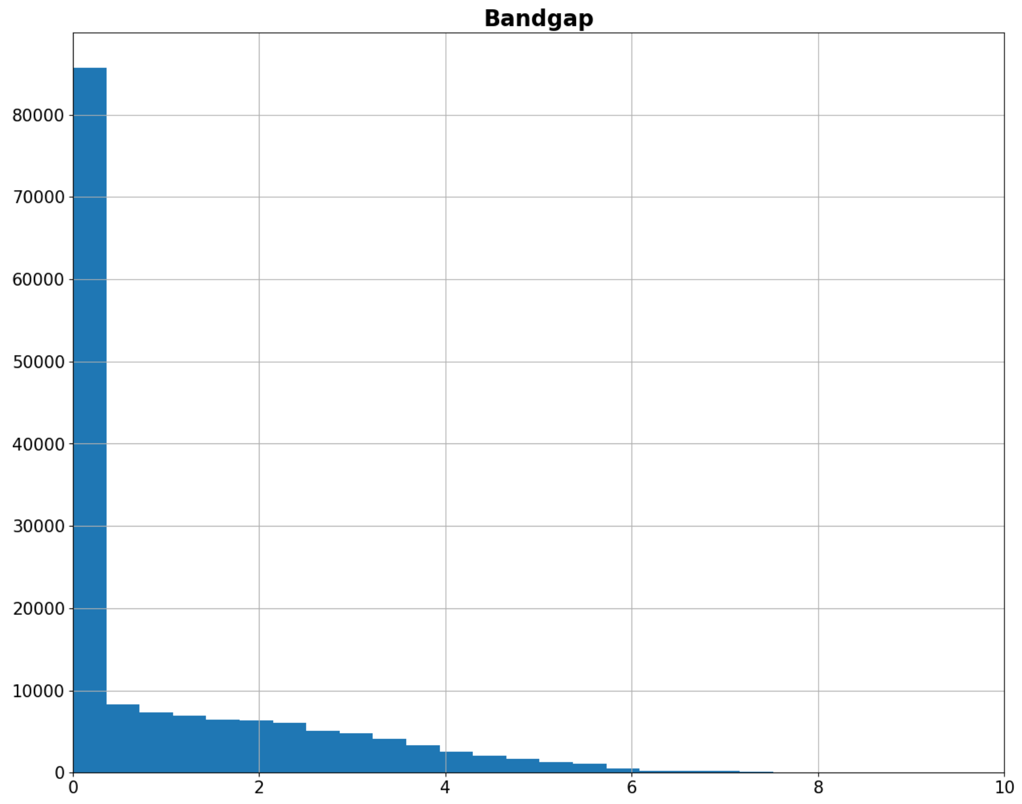
	Index	bandgap	nsites	spacegroup_number	NComp	Comp_L2Norm	Comp_L3Norm	Comp_L5Norm	Comp_L7Norm	Comp_L10Norm	...	max_SpaceGroupNumber	min_SpaceGroupNumber	most_SpaceGroupNumber	frac_sValence	frac_pValence	frac_dValence	frac_fValence	CanFormIonic	MaxIonicChar	MeanIonicChar
count	154718.000000	154718.000000	154718.000000	154718.00000	154718.000000	154718.000000	154718.000000	154718.000000	154718.000000	154718.000000	...	154718.000000	154718.000000	154718.000000	154718.000000	154718.000000	154718.000000	154718.000000	154718.000000	154707.000000	154707.000000
mean	77358.500000	1.055169	31.447233	84.22307	3.464917	0.660476	0.607652	0.584982	0.579643	0.576853	...	220.048895	62.287213	85.186206	0.299354	0.326219	0.269383	0.105043	0.508810	0.514710	0.190607
std	44663.383812	1.503758	34.878837	80.85012	0.990121	0.076313	0.093366	0.105881	0.110447	0.113427	...	20.643270	75.850992	87.614826	0.136632	0.194612	0.218096	0.171638	0.499924	0.289850	0.118522
min	0.000000	0.000000	1.000000	1.00000	1.000000	0.395285	0.297370	0.238495	0.215285	0.199372	...	2.000000	2.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	38679.250000	0.000000	10.000000	11.00000	3.000000	0.612372	0.538609	0.506099	0.501109	0.500098	...	225.000000	12.000000	12.000000	0.208333	0.156863	0.094340	0.000000	0.000000	0.229026	0.074326
50%	77358.500000	0.083000	20.000000	62.00000	3.000000	0.648074	0.603681	0.579020	0.572069	0.571484	...	229.000000	12.000000	15.000000	0.297297	0.371163	0.226667	0.000000	1.000000	0.590585	0.203420
75%	116037.750000	1.858600	40.000000	160.00000	4.000000	0.707107	0.658522	0.647881	0.646555	0.645706	...	229.000000	139.000000	194.000000	0.358974	0.487310	0.400000	0.179487	1.000000	0.779730	0.294785
max	154717.000000	17.891400	444.000000	230.00000	9.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	229.000000	229.000000	229.000000	1.000000	0.750000	1.000000	0.875000	1.000000	0.921450	0.460725

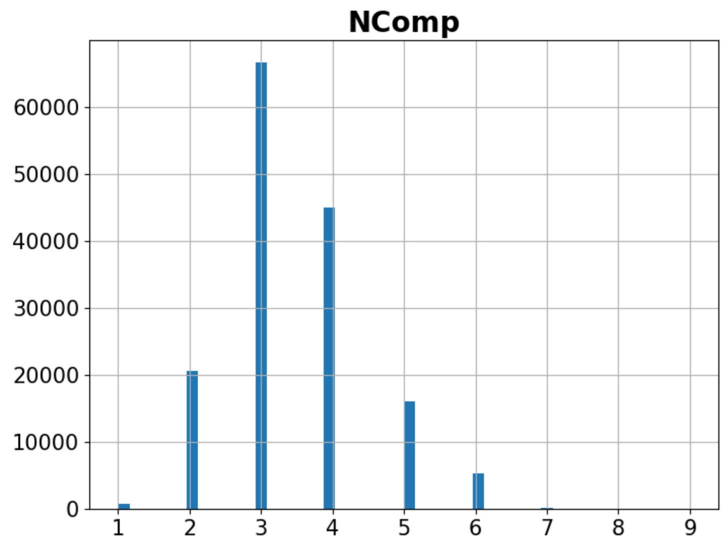
8 rows × 149 columns

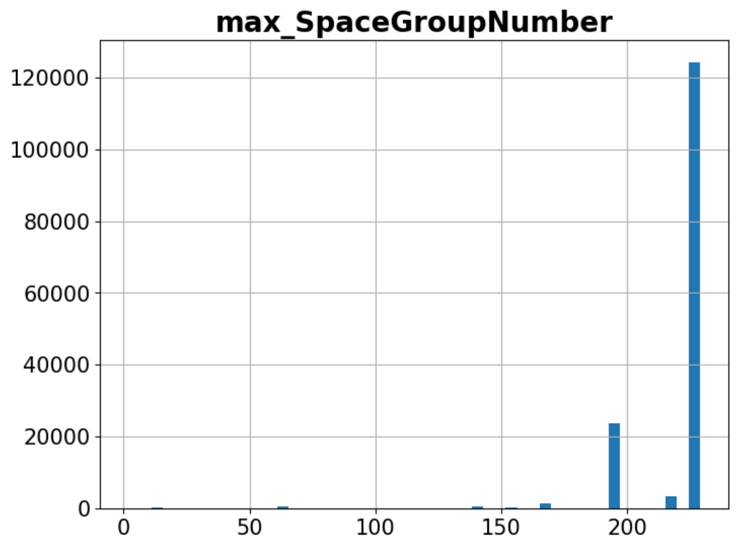
Computational Science
Artificial Intelligence

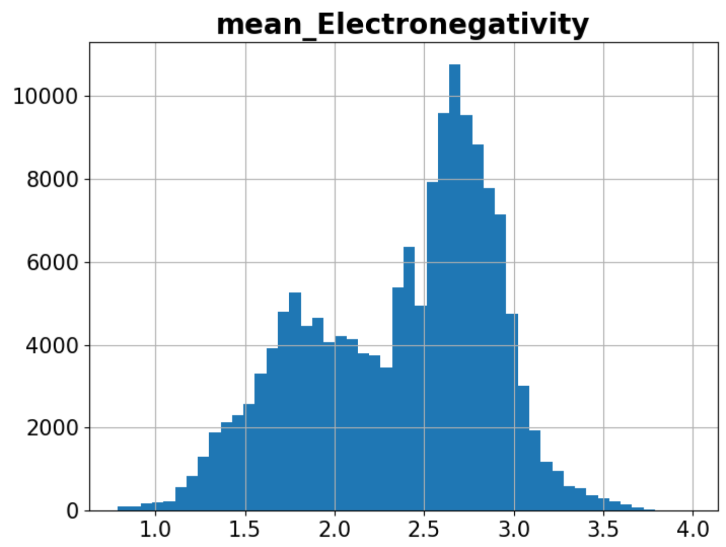
Soongsil University

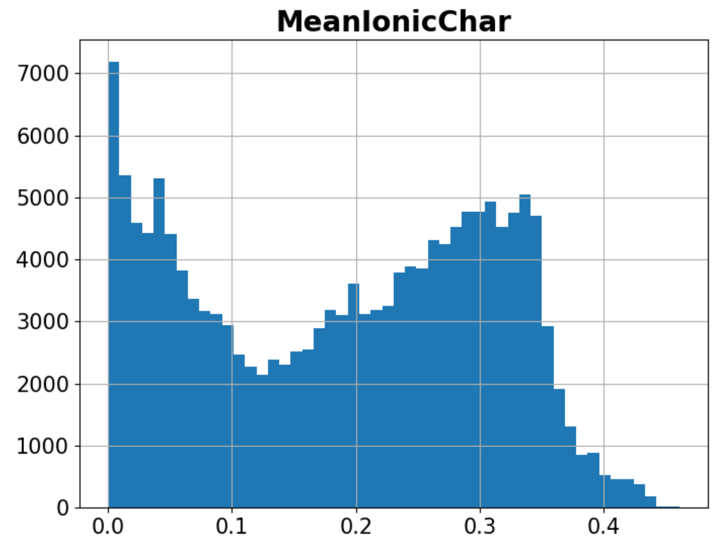
Take a quick look at the Data structure





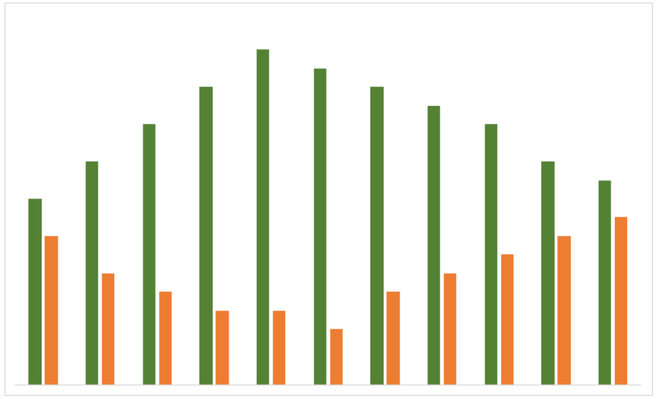


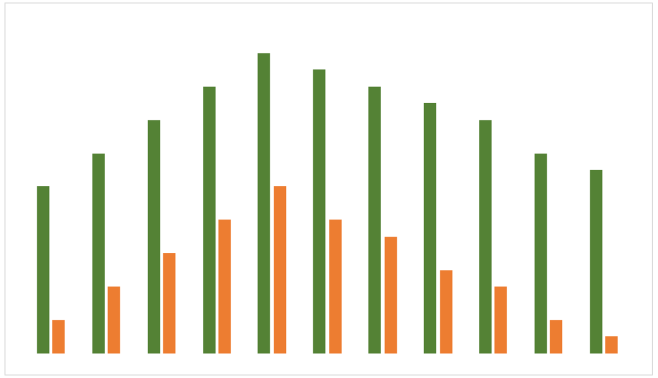


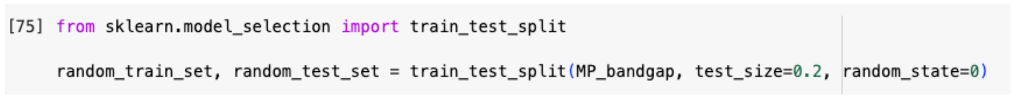


Create a Test set

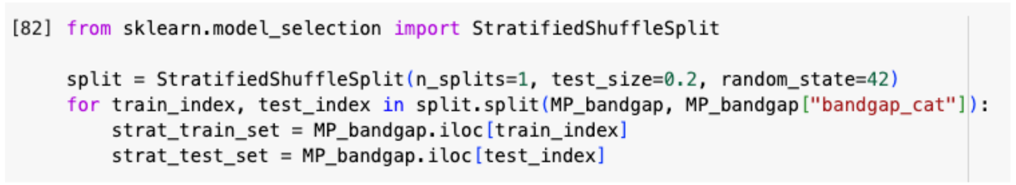
Database
Sampling







```
[75] from sklearn.model_selection import train_test_split

     random_train_set, random_test_set = train_test_split(MP_bandgap, test_size=0.2, random_state=0)
```

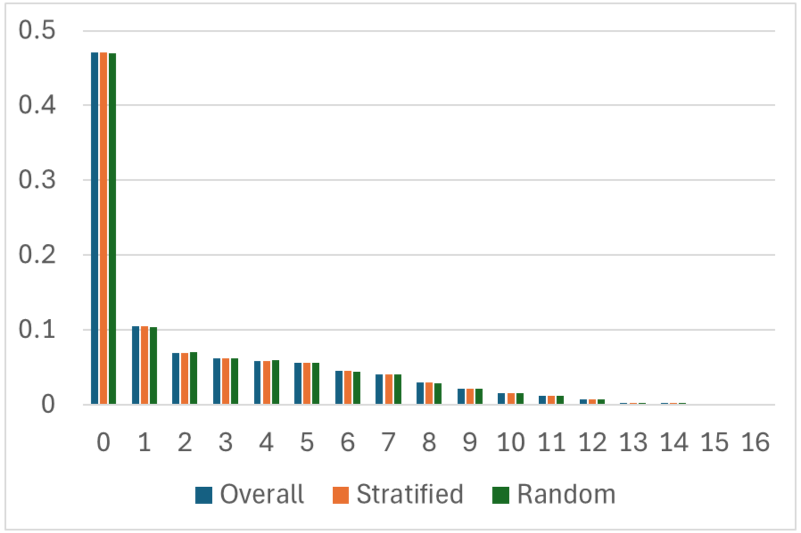


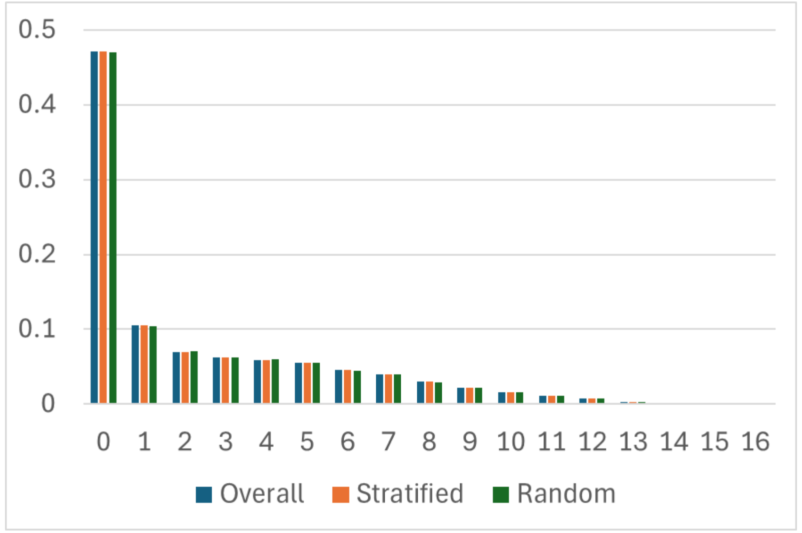
```
[82] from sklearn.model_selection import StratifiedShuffleSplit

     split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
     for train_index, test_index in split.split(MP_bandgap, MP_bandgap["bandgap_cat"]):
         strat_train_set = MP_bandgap.iloc[train_index]
         strat_test_set = MP_bandgap.iloc[test_index]
```

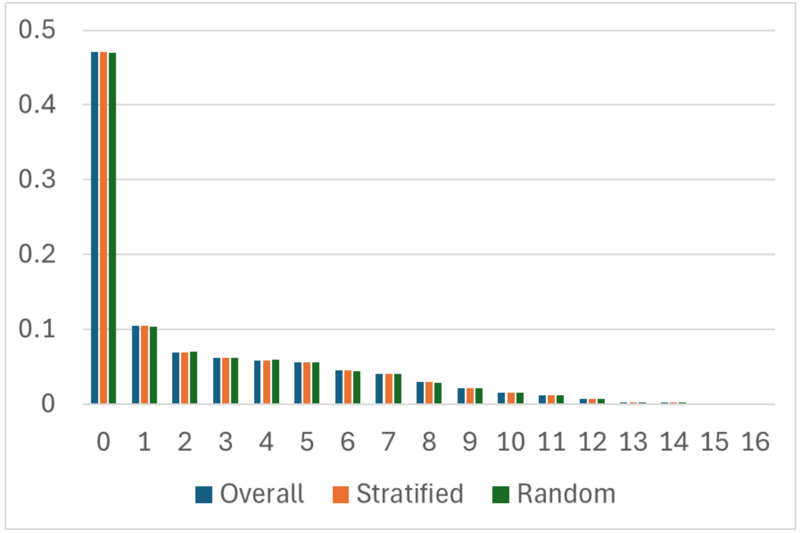
✓ Stratified sampling

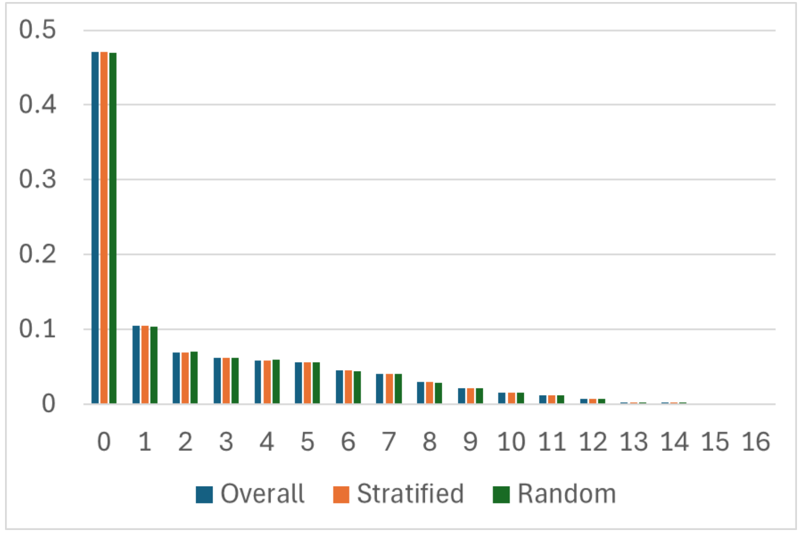
- 모집단을 여러 층으로 분류하고 각 층에서 n개씩 랜덤하게 추출하는 방법
- 데이터 편향을 예방할 수 있음

Random vs Stratified sampling







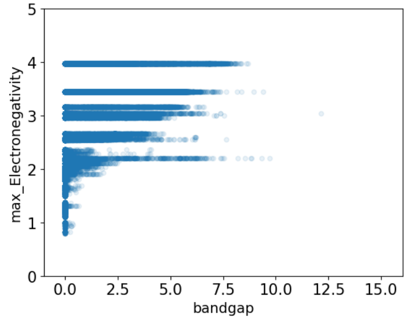


Correlation Coefficient

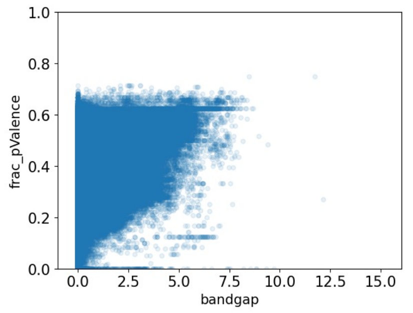
```
[94] corr_matrix = train_set.corr()
```

```
[95] corr_matrix['bandgap'].sort_values(ascending=False)

     bandgap                      1.000000
     CanFormIonic                 0.512913
     frac_pValence                0.494154
     max_Electronegativity        0.420207
     maxdiff_Electronegativity    0.400274
                                    ...
     frac_dValence               -0.403884
     most_CovalentRadius         -0.408413
     mean_Row                    -0.413997
     mean_CovalentRadius         -0.420306
     min_CovalentRadius          -0.421281
     Name: bandgap, Length: 146, dtype: float64
```

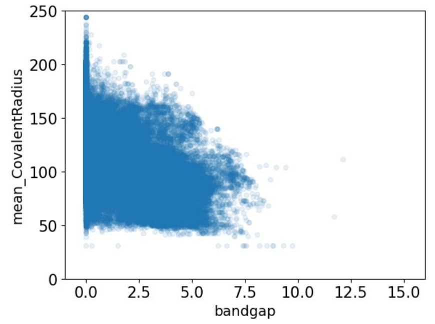
```
train_set.plot(kind="scatter", x="bandgap", y="max_Electronegativity",
               alpha=0.1)
plt.axis([-1, 16, 0, 5])

(-1.0, 16.0, 0.0, 5.0)
```



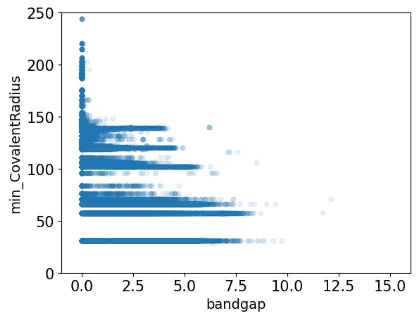
```
train_set.plot(kind="scatter", x="bandgap", y="frac_pValence",
               alpha=0.1)
plt.axis([-1.0, 16, 0, 1.0])

(-1.0, 16.0, 0.0, 1.0)
```



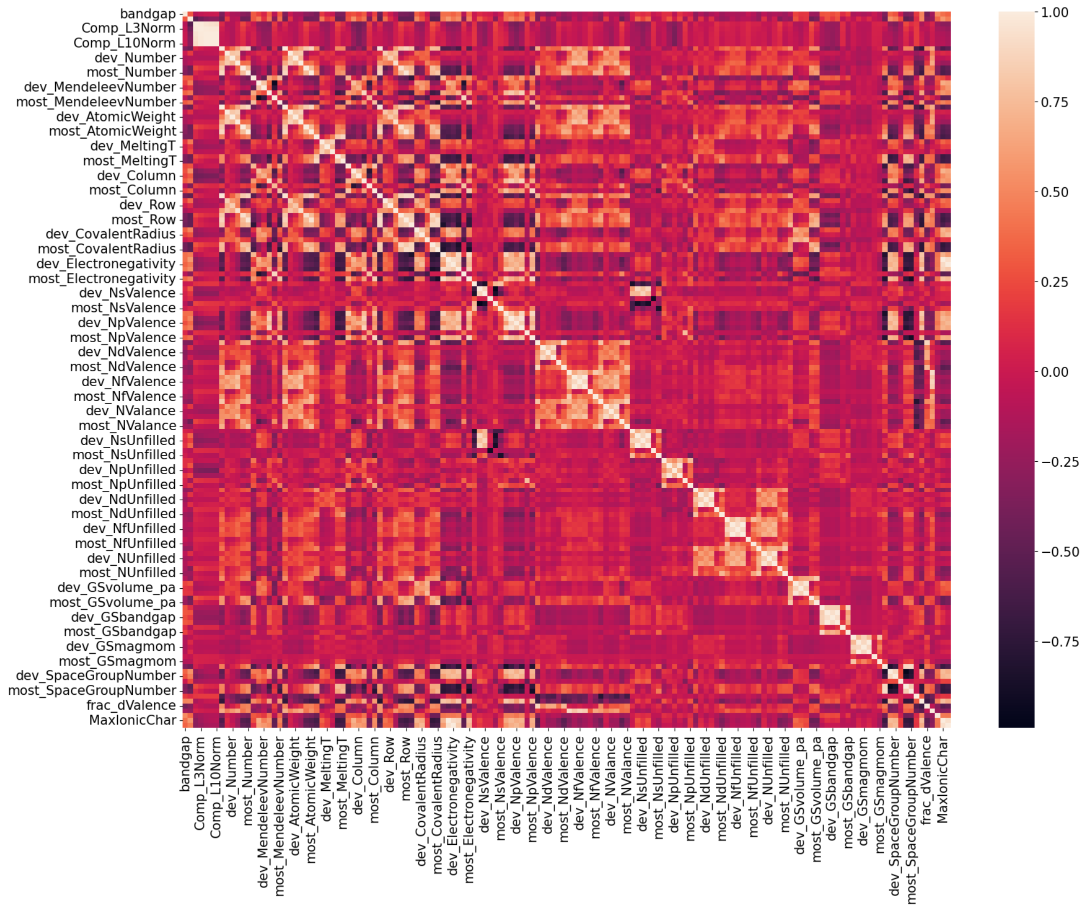
```
[101] train_set.plot(kind="scatter", x="bandgap", y="mean_CovalentRadius",
                     alpha=0.1)
      plt.axis([-1, 16, 0, 250])

      (-1.0, 16.0, 0.0, 250.0)
```



```
train_set.plot(kind="scatter", x="bandgap", y="min_CovalentRadius",
               alpha=0.1)
plt.axis([-1, 16, 0, 250])

(-1.0, 16.0, 0.0, 250.0)
```



```
import seaborn as sns

plt.figure(figsize = (20,15))
sns.heatmap(corr_matrix)
```

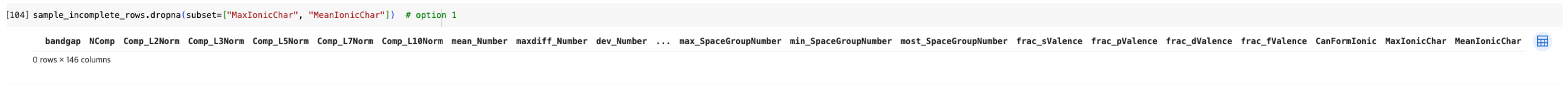


Soongsil University

Prepare the Data for Machine Learning Algorithms

✓ MaxIonicChar, MeanIonicChar의 feature 값들이 없는 부분 존재

- Option 1 : 해당 data 삭제

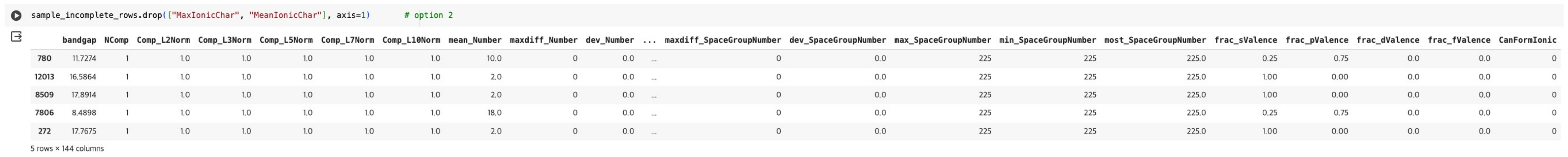


```
[104] sample_incomplete_rows.dropna(subset=["MaxIonicChar", "MeanIonicChar"])  # option 1
```

bandgap	NComp	Comp_L2Norm	Comp_L3Norm	Comp_L5Norm	Comp_L7Norm	Comp_L10Norm	mean_Number	maxdiff_Number	dev_Number	...	max_SpaceGroupNumber	min_SpaceGroupNumber	most_SpaceGroupNumber	frac_sValence	frac_pValence	frac_dValence	frac_fValence	CanFormIonic	MaxIonicChar	MeanIonicChar

0 rows × 146 columns

- Option 2 : 해당 feature 삭제

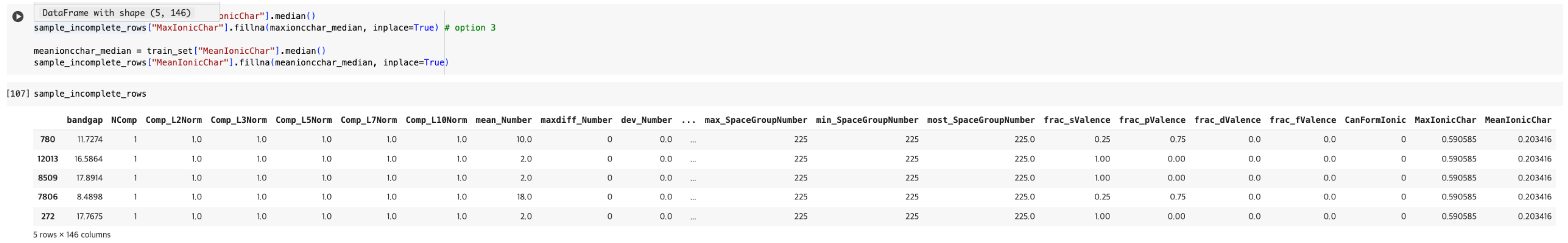


```
sample_incomplete_rows.drop(["MaxIonicChar", "MeanIonicChar"], axis=1)        # option 2
```

	bandgap	NComp	Comp_L2Norm	Comp_L3Norm	Comp_L5Norm	Comp_L7Norm	Comp_L10Norm	mean_Number	maxdiff_Number	dev_Number	...	maxdiff_SpaceGroupNumber	dev_SpaceGroupNumber	max_SpaceGroupNumber	min_SpaceGroupNumber	most_SpaceGroupNumber	frac_sValence	frac_pValence	frac_dValence	frac_fValence	CanFormIonic
780	11.7274	1	1.0	1.0	1.0	1.0	1.0	10.0	0	0.0	...	0	0.0	225	225	225.0	0.25	0.75	0.0	0.0	0
12013	16.5864	1	1.0	1.0	1.0	1.0	1.0	2.0	0	0.0	...	0	0.0	225	225	225.0	1.00	0.00	0.0	0.0	0
8509	17.8914	1	1.0	1.0	1.0	1.0	1.0	2.0	0	0.0	...	0	0.0	225	225	225.0	1.00	0.00	0.0	0.0	0
7806	8.4898	1	1.0	1.0	1.0	1.0	1.0	18.0	0	0.0	...	0	0.0	225	225	225.0	0.25	0.75	0.0	0.0	0
272	17.7675	1	1.0	1.0	1.0	1.0	1.0	2.0	0	0.0	...	0	0.0	225	225	225.0	1.00	0.00	0.0	0.0	0

5 rows × 144 columns

- Option 3 : mean, average 등 대표값을 취함



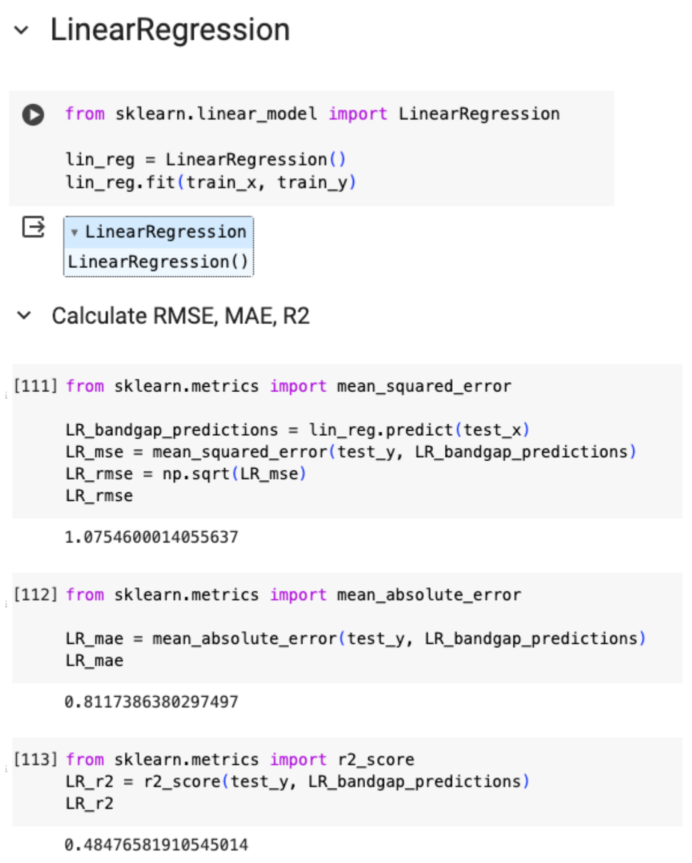
```
 DataFrame with shape (5, 146)  onicChar"].median()
sample_incomplete_rows["MaxIonicChar"].fillna(maxioncchar_median, inplace=True) # option 3

meanioncchar_median = train_set["MeanIonicChar"].median()
sample_incomplete_rows["MeanIonicChar"].fillna(meanioncchar_median, inplace=True)
```

```
[107] sample_incomplete_rows
```

	bandgap	NComp	Comp_L2Norm	Comp_L3Norm	Comp_L5Norm	Comp_L7Norm	Comp_L10Norm	mean_Number	maxdiff_Number	dev_Number	...	max_SpaceGroupNumber	min_SpaceGroupNumber	most_SpaceGroupNumber	frac_sValence	frac_pValence	frac_dValence	frac_fValence	CanFormIonic	MaxIonicChar	MeanIonicChar
780	11.7274	1	1.0	1.0	1.0	1.0	1.0	10.0	0	0.0	...	225	225	225.0	0.25	0.75	0.0	0.0	0	0.590585	0.203416
12013	16.5864	1	1.0	1.0	1.0	1.0	1.0	2.0	0	0.0	...	225	225	225.0	1.00	0.00	0.0	0.0	0	0.590585	0.203416
8509	17.8914	1	1.0	1.0	1.0	1.0	1.0	2.0	0	0.0	...	225	225	225.0	1.00	0.00	0.0	0.0	0	0.590585	0.203416
7806	8.4898	1	1.0	1.0	1.0	1.0	1.0	18.0	0	0.0	...	225	225	225.0	0.25	0.75	0.0	0.0	0	0.590585	0.203416
272	17.7675	1	1.0	1.0	1.0	1.0	1.0	2.0	0	0.0	...	225	225	225.0	1.00	0.00	0.0	0.0	0	0.590585	0.203416

5 rows × 146 columns

Training and Evaluation



LinearRegression

```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(train_x, train_y)
```

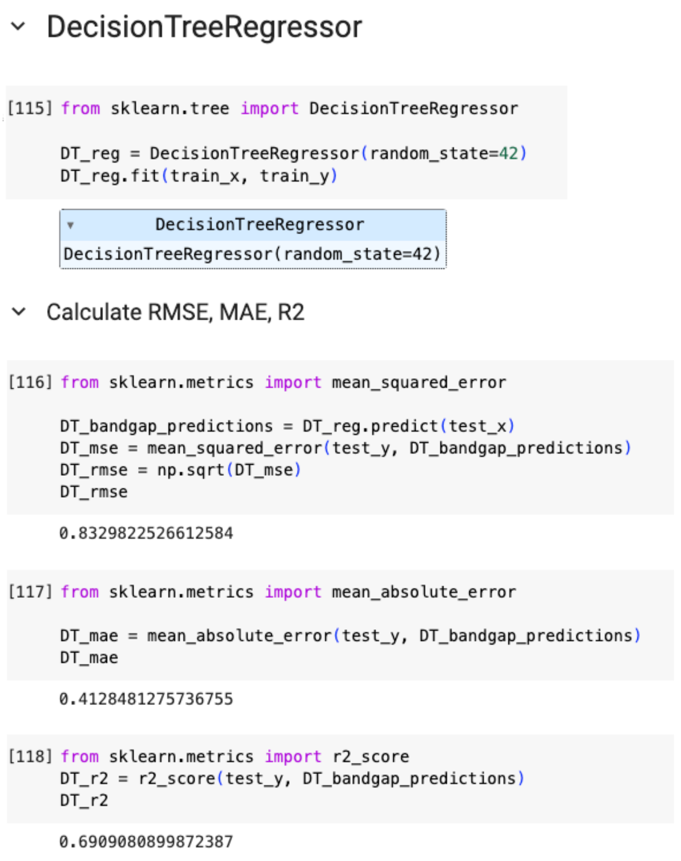
```
▾ LinearRegression
LinearRegression()
```

Calculate RMSE, MAE, R2

```
[111] from sklearn.metrics import mean_squared_error

      LR_bandgap_predictions = lin_reg.predict(test_x)
      LR_mse = mean_squared_error(test_y, LR_bandgap_predictions)
      LR_rmse = np.sqrt(LR_mse)
      LR_rmse

      1.0754600014055637
```

```
[112] from sklearn.metrics import mean_absolute_error

      LR_mae = mean_absolute_error(test_y, LR_bandgap_predictions)
      LR_mae

      0.8117386380297497
```

```
[113] from sklearn.metrics import r2_score
      LR_r2 = r2_score(test_y, LR_bandgap_predictions)
      LR_r2

      0.48476581910545014
```



DecisionTreeRegressor

```
[115] from sklearn.tree import DecisionTreeRegressor

      DT_reg = DecisionTreeRegressor(random_state=42)
      DT_reg.fit(train_x, train_y)
```

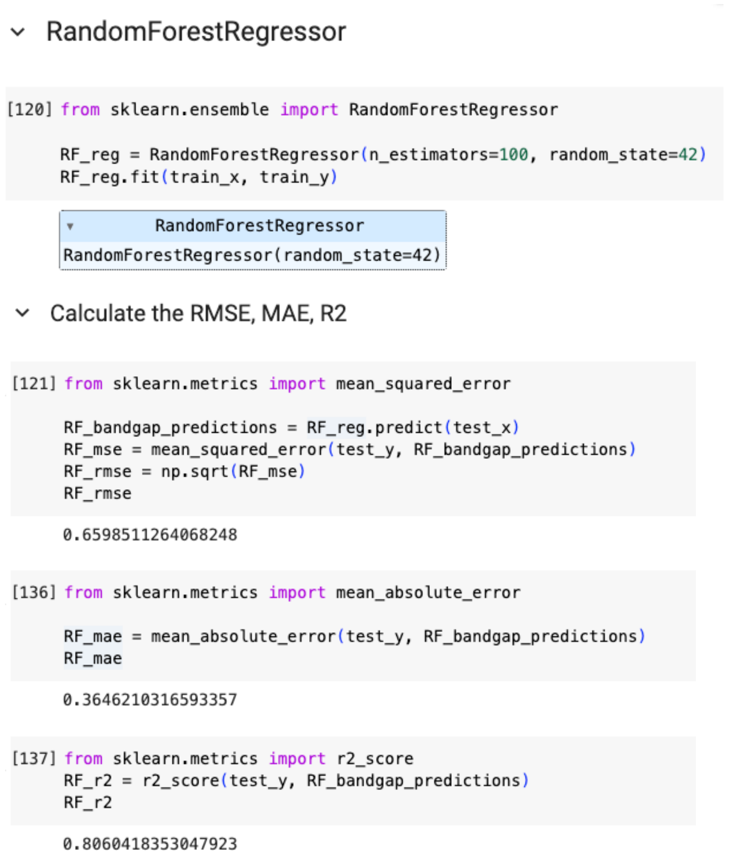
```
▾ DecisionTreeRegressor
DecisionTreeRegressor(random_state=42)
```

Calculate RMSE, MAE, R2

```
[116] from sklearn.metrics import mean_squared_error

      DT_bandgap_predictions = DT_reg.predict(test_x)
      DT_mse = mean_squared_error(test_y, DT_bandgap_predictions)
      DT_rmse = np.sqrt(DT_mse)
      DT_rmse

      0.8329822526612584
```

```
[117] from sklearn.metrics import mean_absolute_error

      DT_mae = mean_absolute_error(test_y, DT_bandgap_predictions)
      DT_mae

      0.4128481275736755
```

```
[118] from sklearn.metrics import r2_score
      DT_r2 = r2_score(test_y, DT_bandgap_predictions)
      DT_r2

      0.6909080899872387
```



RandomForestRegressor

```
[120] from sklearn.ensemble import RandomForestRegressor

      RF_reg = RandomForestRegressor(n_estimators=100, random_state=42)
      RF_reg.fit(train_x, train_y)
```

```
▾ RandomForestRegressor
RandomForestRegressor(random_state=42)
```

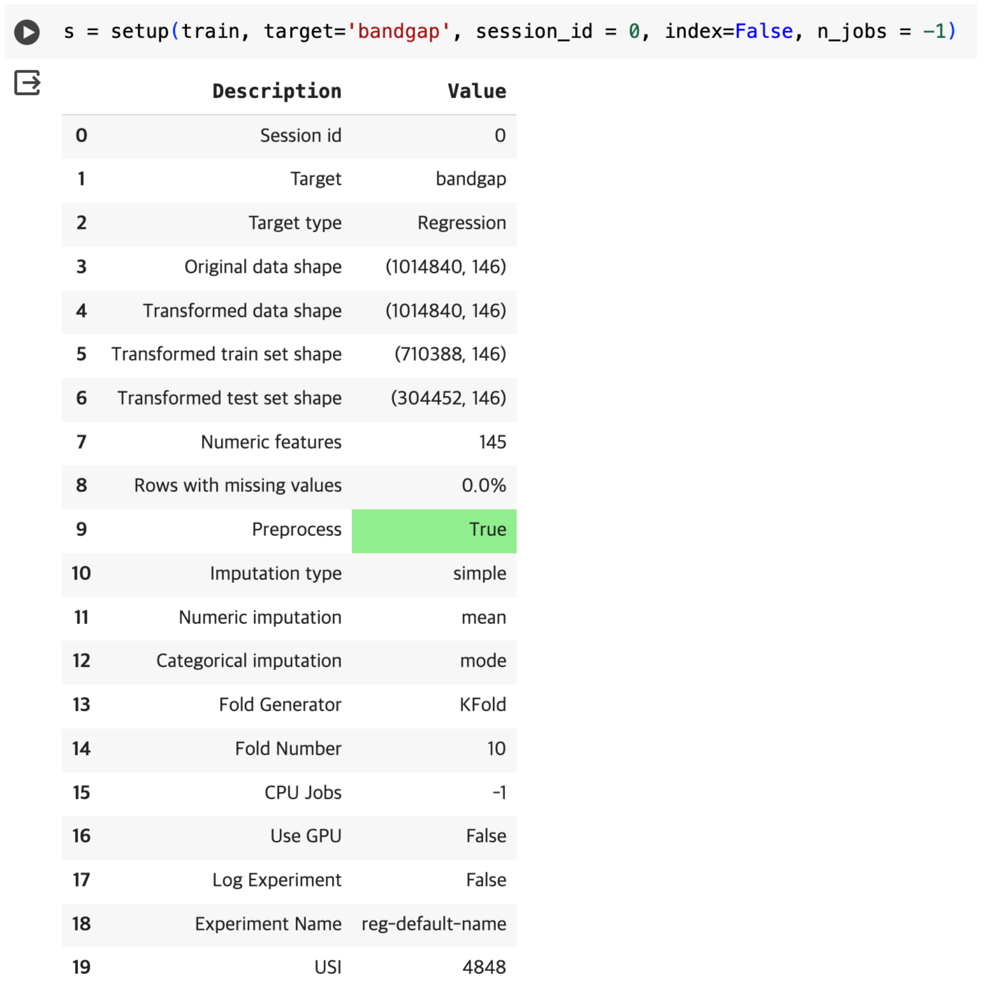
Calculate the RMSE, MAE, R2

```
[121] from sklearn.metrics import mean_squared_error

      RF_bandgap_predictions = RF_reg.predict(test_x)
      RF_mse = mean_squared_error(test_y, RF_bandgap_predictions)
      RF_rmse = np.sqrt(RF_mse)
      RF_rmse

      0.6598511264068248
```

```
[136] from sklearn.metrics import mean_absolute_error

      RF_mae = mean_absolute_error(test_y, RF_bandgap_predictions)
      RF_mae

      0.3646210316593357
```

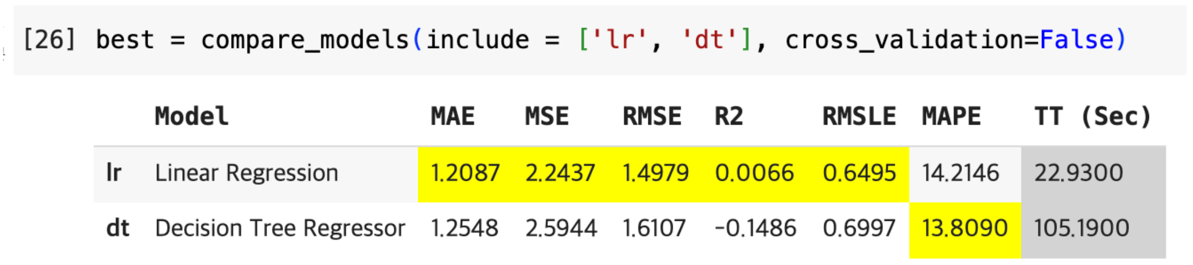
```
[137] from sklearn.metrics import r2_score
      RF_r2 = r2_score(test_y, RF_bandgap_predictions)
      RF_r2

      0.8060418353047923
```

Using AutoML – Pycaret



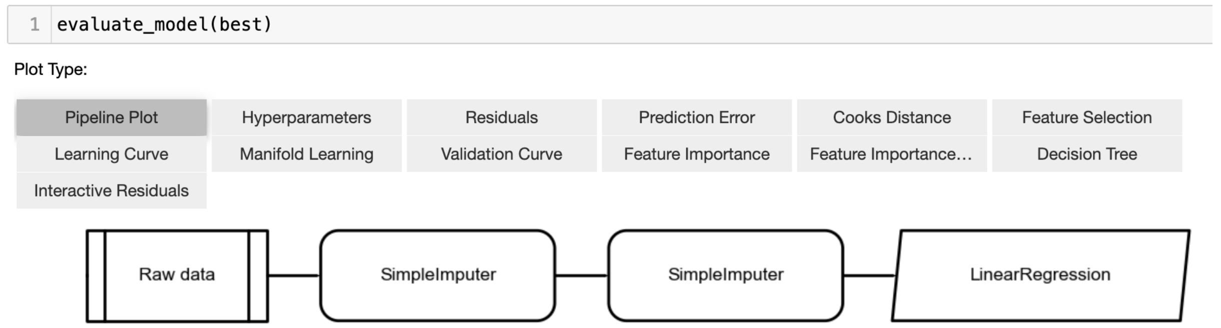
```
s = setup(train, target='bandgap', session_id = 0, index=False, n_jobs = -1)
```

	Description	Value
0	Session id	0
1	Target	bandgap
2	Target type	Regression
3	Original data shape	(1014840, 146)
4	Transformed data shape	(1014840, 146)
5	Transformed train set shape	(710388, 146)
6	Transformed test set shape	(304452, 146)
7	Numeric features	145
8	Rows with missing values	0.0%
9	Preprocess	True
10	Imputation type	simple
11	Numeric imputation	mean
12	Categorical imputation	mode
13	Fold Generator	KFold
14	Fold Number	10
15	CPU Jobs	-1
16	Use GPU	False
17	Log Experiment	False
18	Experiment Name	reg-default-name
19	USI	4848



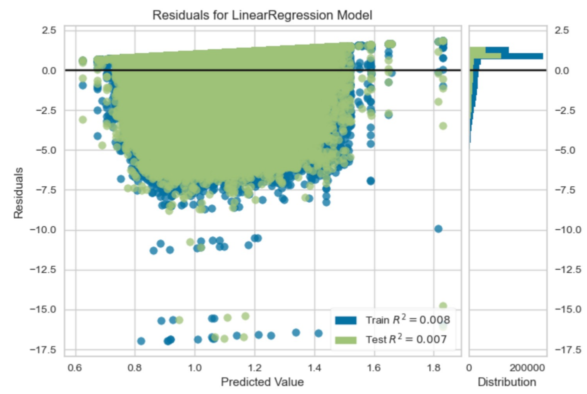
```
[26] best = compare_models(include = ['lr', 'dt'], cross_validation=False)
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
lr	Linear Regression	1.2087	2.2437	1.4979	0.0066	0.6495	14.2146	22.9300
dt	Decision Tree Regressor	1.2548	2.5944	1.6107	-0.1486	0.6997	13.8090	105.1900

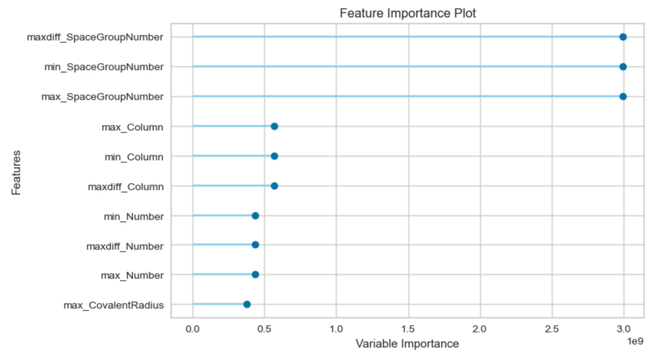


```
evaluate_model(best)
```

```
plot_model(best, plot='residuals')
```



```
plot_model(best, plot = 'feature')
```



Using AutoML – Pycaret

Tune model

```python
tuned_model = tune_model(best, n_iter = 20, optimize = 'MAE')
```

Blending model

```python
blend1 = blend_models(estimator_list = [lr, dt],fold=3)
```

	MAE	MSE	RMSE	R2	RMSLE	MAPE
Fold						
0	1.2298	2.3993	1.5490	-0.0578	0.6692	15.6101
1	1.2268	2.3823	1.5435	-0.0589	0.6685	14.7248
2	1.2282	2.3925	1.5468	-0.0567	0.6680	14.6943
Mean	1.2283	2.3914	1.5464	-0.0578	0.6686	15.0097
Std	0.0012	0.0070	0.0023	0.0009	0.0005	0.4247

Ensemble model

```python
ensembled_model1 = ensemble_model(dt, fold=3) #Bagging
#ensembled_model2 = ensemble_model(dt, method='Boosting', fold=3) #Boosting
```

	MAE	MSE	RMSE	R2	RMSLE	MAPE
Fold						
0	1.2764	2.7582	1.6608	-0.2160	0.7183	15.0171
1	1.2753	2.7545	1.6597	-0.2244	0.7182	14.8193
2	1.2745	2.7481	1.6577	-0.2138	0.7168	14.3727
Mean	1.2754	2.7536	1.6594	-0.2181	0.7178	14.7364
Std	0.0008	0.0042	0.0013	0.0046	0.0007	0.2695

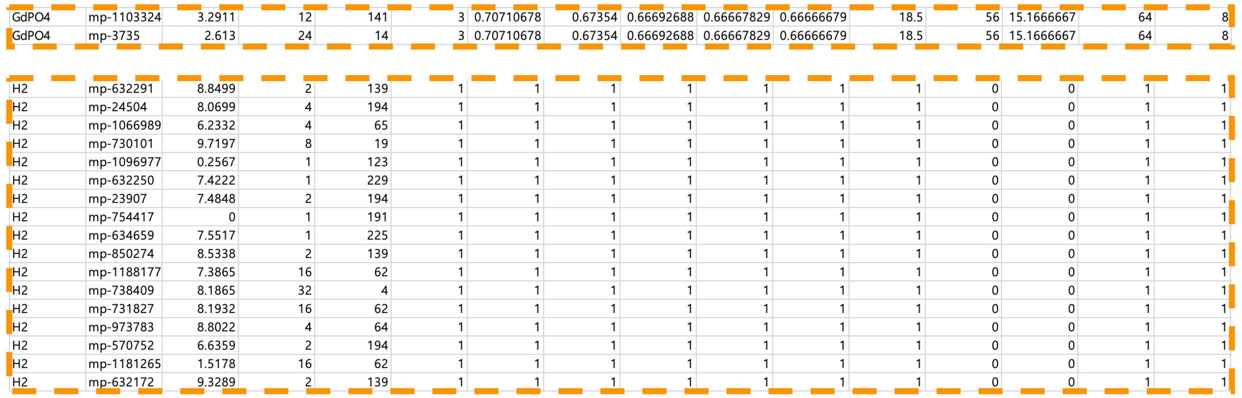
```python
final_model = finalize_model(blend1)
```

```python
prediction = predict_model(final_model, data = test)
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Voting Regressor	1.1888	2.2040	1.4846	0.0273	0.6425	13.4030

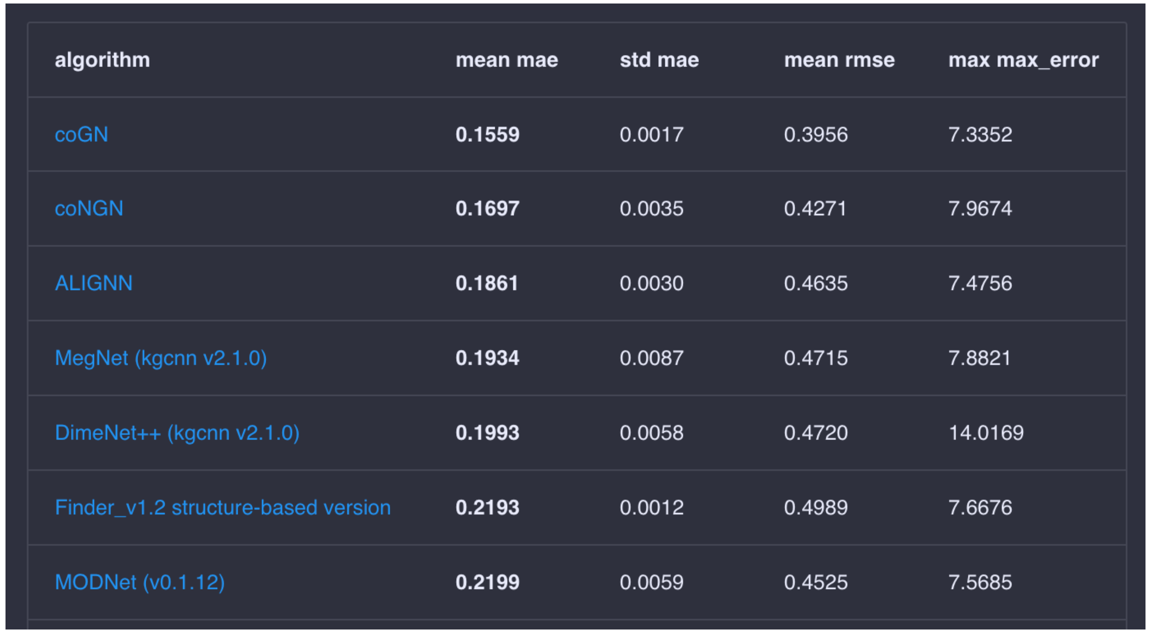
Training and Evaluation

	LR	DT	RF
R^2	0.485	0.691	0.806
MAE	0.812	0.413	0.365
RMSE	1.075	0.833	0.650



GdPO4	mp-1103324	3.2911	12	141	3	0.70710678	0.67354	0.66692688	0.66667829	0.66666679	18.5	56	15.1666667	64	8
GdPO4	mp-3735	2.613	24	14	3	0.70710678	0.67354	0.66692688	0.66667829	0.66666679	18.5	56	15.1666667	64	8

H2	mp-632291	8.8499	2	139	1	1	1	1	1	1	1	0	0	1	1
H2	mp-24504	8.0699	4	194	1	1	1	1	1	1	1	0	0	1	1
H2	mp-1066989	6.2332	4	65	1	1	1	1	1	1	1	0	0	1	1
H2	mp-730101	9.7197	8	19	1	1	1	1	1	1	1	0	0	1	1
H2	mp-1096977	0.2567	1	123	1	1	1	1	1	1	1	0	0	1	1
H2	mp-632250	7.4222	1	229	1	1	1	1	1	1	1	0	0	1	1
H2	mp-23907	7.4848	2	194	1	1	1	1	1	1	1	0	0	1	1
H2	mp-754417	0	1	191	1	1	1	1	1	1	1	0	0	1	1
H2	mp-634659	7.5517	1	225	1	1	1	1	1	1	1	0	0	1	1
H2	mp-850274	8.5338	2	139	1	1	1	1	1	1	1	0	0	1	1
H2	mp-1188177	7.3865	16	62	1	1	1	1	1	1	1	0	0	1	1
H2	mp-738409	8.1865	32	4	1	1	1	1	1	1	1	0	0	1	1
H2	mp-731827	8.1932	16	62	1	1	1	1	1	1	1	0	0	1	1
H2	mp-973783	8.8022	4	64	1	1	1	1	1	1	1	0	0	1	1
H2	mp-570752	6.6359	2	194	1	1	1	1	1	1	1	0	0	1	1
H2	mp-1181265	1.5178	16	62	1	1	1	1	1	1	1	0	0	1	1
H2	mp-632172	9.3289	2	139	1	1	1	1	1	1	1	0	0	1	1



algorithm	mean mae	std mae	mean rmse	max max_error
coGN	**0.1559**	0.0017	0.3956	7.3352
coNGN	**0.1697**	0.0035	0.4271	7.9674
ALIGNN	**0.1861**	0.0030	0.4635	7.4756
MegNet (kgcnn v2.1.0)	**0.1934**	0.0087	0.4715	7.8821
DimeNet++ (kgcnn v2.1.0)	**0.1993**	0.0058	0.4720	14.0169
Finder_v1.2 structure-based version	**0.2193**	0.0012	0.4989	7.6676
MODNet (v0.1.12)	**0.2199**	0.0059	0.4525	7.5685

https://matbench.materialsproject.org/

- Chemical Composition만을 가지고 물질을 대표하는 feature 생성할 수 없음
- 새로운 ML Model, Hyperparameter optimization 등 정확한 예측을 하기 위한 방법이 존재
- AutoML 사용하여 low code로도 더 많은 비교 가능

Thank you