Example 1: A quick look at 5 deterministic machine-learning models

Huan Tran

Some deterministic (non-probabilistic) ML models supported by matsML are introduced here. They models are

1. Support Vector Regression
2. Random Forest Regression
3. Kernel Ridge Regression
4. Gaussian Process Regression
5. Fully-Connected Neural Net

A simple dataset will be obtained from www.matsml.org for this example.

Load data

This is a *fingerprinted* dataset, being ready for machine learning. It contains 192 compositions of hybrid organic-inorganic perovskites, each of them is represented by a fingerprint vector and the averaged band gap of multiple atomic structures predicted for this composition. The raw data leading to this dataset is available at https://www.nature.com/articles/sdata201757.

In [1]:
```
from matsml.data import Datasets
import pandas as pd

# obtain data
data=Datasets(S1='fp_hoips_S1_1dest')
data.load_dataset()

# Have a look at the data fields. You will see "ID" is for the identification of the data points,
# 'Ymean' is the target (the averaged band gap mentioned above), and the others are the components
# of the fingerprint vector
fp_data = pd.read_csv('fp_hoips_S1_1dest.csv.gz')
print (fp_data.shape)
print (fp_data.columns)
```

```
  matsML, v1.0.1
  *****
  Load requested dataset(s)
  Data saved in fp_hoips_S1_1dest.csv.gz
(192, 34)
Index(['Unnamed: 0', 'ID', 'Ymean', 'MagpieData avg_dev GSvolume_pa',
       'MatscholarElementData mean embedding 54',
       'MatscholarElementData std_dev embedding 116',
       'MatscholarElementData std_dev embedding 155',
       'MatscholarElementData mean embedding 4',
       'PymatgenData mean mendeleev_no',
       'MatscholarElementData std_dev embedding 136',
       'MatscholarElementData std_dev embedding 153',
       'MatscholarElementData mean embedding 140',
       'MatscholarElementData mean embedding 170', 'H1N4H1', 'H1N3H1',
       'H1N3C3', 'N3C3N3', 'N3C3H1', 'H1C3C3', 'C3C3N3', 'C3N3C3', 'H1C4H1',
       'H1C4C4', 'C4C4C4', 'C4C4N4', 'H1C4N4', 'C4N4H1', 'N4N3H1', 'H1N4N3',
       'C4N4C4', 'H1N4O2', 'N4O2H1', 'C3C4H1', 'C4C3N3'],
      dtype='object')
```

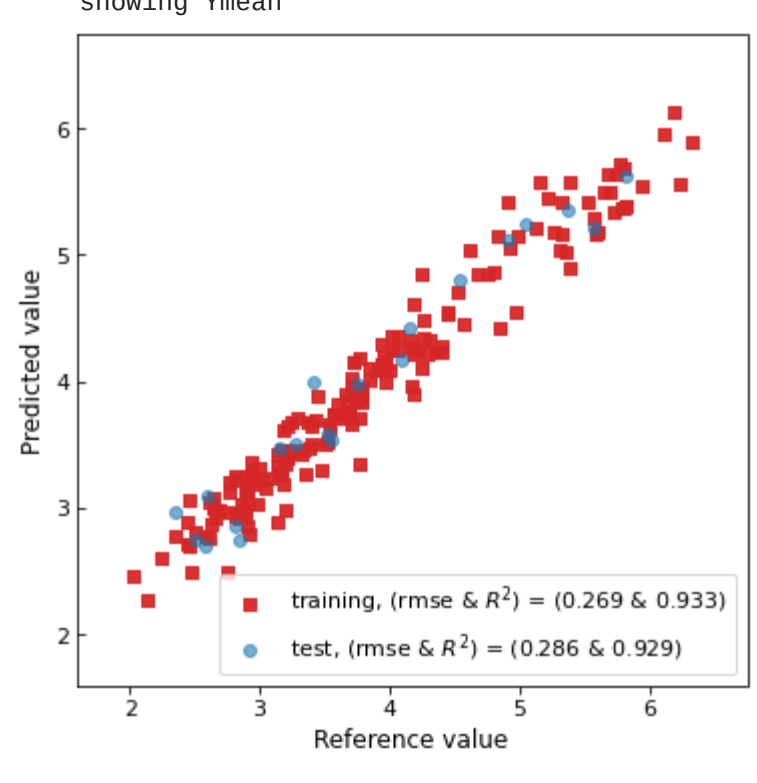
Essential parameters of the obtained dataset, given as a dict, and needed for ML models

In [2]:
```
# data parameters
data_file ='fp_hoips_S1_1dest.csv.gz'
id_col=['ID']
y_cols=['Ymean']
comment_cols=[]
n_trains=0.9
sampling='random'
x_scaling='minmax'
y_scaling='minmax'

data_params={'data_file':data_file,'id_col':id_col,'y_cols':y_cols, 'comment_cols':comment_cols,
    'y_scaling':y_scaling,'x_scaling':x_scaling,'sampling':sampling, 'n_trains':n_trains}
```

Model 1: Support Vector Regression

In [3]:
```
from matsml.models import SVecR

# Model parameters
nfold_cv=5
model_file='model_svr.pkl'
verbosity=0
rmse_cv=False
regular_param=2
kernel='rbf'
max_iter=-1

model_params={'kernel':kernel,'nfold_cv':nfold_cv,'regular_param':regular_param, 'max_iter':max_iter,
    'model_file':model_file,'verbosity':verbosity,'rmse_cv':rmse_cv}

model=SVecR(data_params=data_params,model_params=model_params)
model.train()
model.plot(pdf_output=False)
```

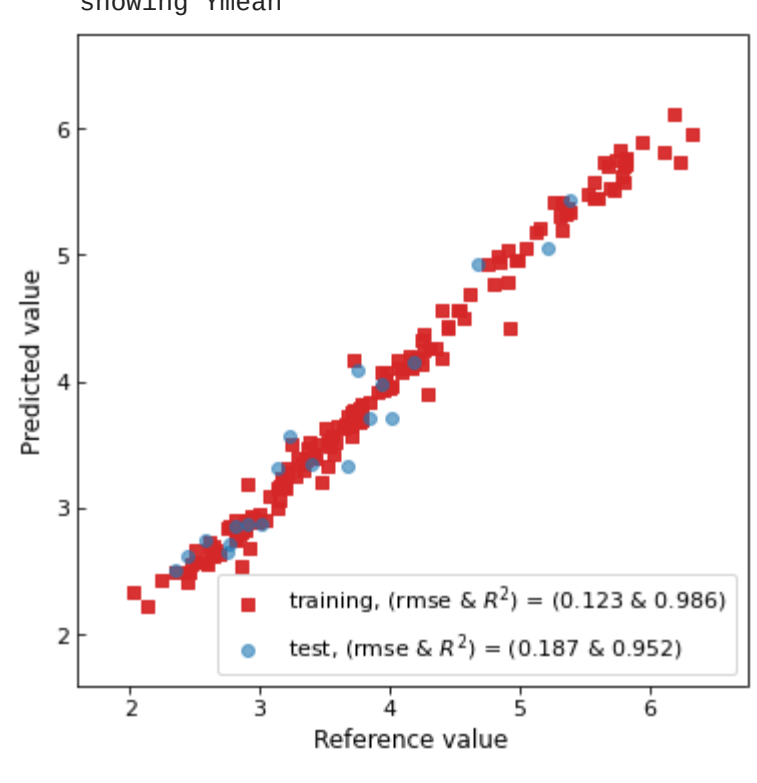
```
Checking parameters
  all passed                    True

Learning fingerprinted/featured data
  algorithm                     support vector regression w/ scikit-learn
  kernel                        rbf
  regular_param                 2
  max_iter                      -1
  nfold_cv                      5
Read data
  data file                     fp_hoips_S1_1dest.csv.gz
  data size                     192
  training size                 172 (89.6 %)
  test size                     20 (10.4 %)
  x dimensionality              32
  y dimensionality              1
  y label(s)                    ['Ymean']
Scaling x                       minmax
  xscaler saved in              xscaler.pkl
Scaling y                       minmax
Prepare train/test sets         random
Training model w/ cross validation
  cv,rmse_train,rmse_test,rmse_opt: 0 0.062320 0.078931 0.078931
  cv,rmse_train,rmse_test,rmse_opt: 1 0.067054 0.072070 0.072070
  cv,rmse_train,rmse_test,rmse_opt: 2 0.066908 0.073685 0.072070
  cv,rmse_train,rmse_test,rmse_opt: 3 0.062854 0.077470 0.072070
  cv,rmse_train,rmse_test,rmse_opt: 4 0.062729 0.061967 0.061967
RFR model trained and saved in "model_svr.pkl"
Now make predictions & invert scaling
  unscaling y: minmax
     rmse training     Ymean            0.269206
  unscaling y: minmax
     rmse test         Ymean            0.285963
Predictions made & saved in "training.csv" & "test.csv"
Plot results in "training.csv" & "test.csv"
  training, (rmse & R2) = (0.269 & 0.933)
  test, (rmse & R2) = (0.286 & 0.929)
  showing Ymean
```



Model 2: Random Forest Regression

In [4]:
```
from matsml.models import RFR

# Model parameters
nfold_cv=5
model_file='model_rfr.pkl'
verbosity=0
rmse_cv=False
n_estimators=20
random_state=11
criterion='mse'
max_depth=8
get_feature_importances=True

model_params={'nfold_cv':nfold_cv,'n_estimators':n_estimators,'random_state':random_state,
    'criterion':criterion,'max_depth':max_depth,'get_feature_importances':get_feature_importances,
    'model_file':model_file,'verbosity':verbosity,'rmse_cv':rmse_cv}

model=RFR(data_params=data_params,model_params=model_params)
model.train()
model.plot(pdf_output=False)
```

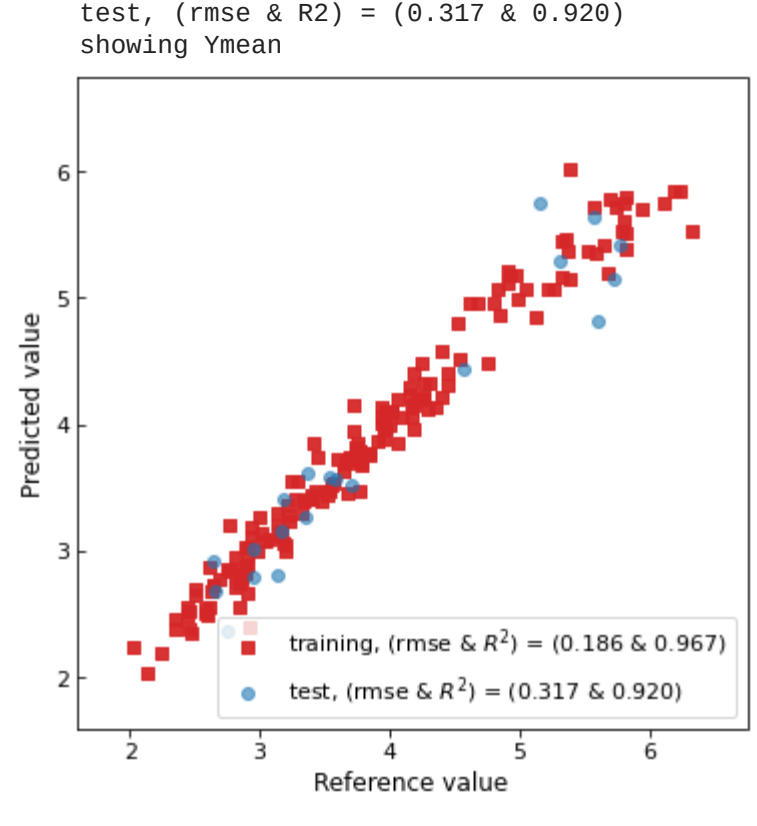
```
Checking parameters
  all passed                    True

Learning fingerprinted/featured data
  algorithm                     random forest regression w/ scikit-learn
  nfold_cv                      5
  n_estimators                  20
  max_depth                     8
  criterion                     mse
  get_feature_importances       True
  random_state                  11
Read data
  data file                     fp_hoips_S1_1dest.csv.gz
  data size                     192
  training size                 172 (89.6 %)
  test size                     20 (10.4 %)
  x dimensionality              32
  y dimensionality              1
  y label(s)                    ['Ymean']
Scaling x                       minmax
  xscaler saved in              xscaler.pkl
Scaling y                       minmax
Prepare train/test sets         random
Training model w/ cross validation
  cv,rmse_train,rmse_test,rmse_opt: 0 0.022399 0.046180 0.046180
  cv,rmse_train,rmse_test,rmse_opt: 1 0.022101 0.046171 0.046171
  cv,rmse_train,rmse_test,rmse_opt: 2 0.022216 0.062874 0.046171
  cv,rmse_train,rmse_test,rmse_opt: 3 0.019279 0.061148 0.046171
  cv,rmse_train,rmse_test,rmse_opt: 4 0.023358 0.054302 0.046171
RFR model trained and saved in "model_rfr.pkl"
Top 10 features by importance
  MatscholarElementData std_dev embedding 116        importance: 0.426
  MagpieData avg_dev GSvolume_pa                     importance: 0.214
  MatscholarElementData std_dev embedding 136        importance: 0.095
  MatscholarElementData mean embedding 4             importance: 0.079
  MatscholarElementData mean embedding 54            importance: 0.063
  MatscholarElementData std_dev embedding 155        importance: 0.042
  MatscholarElementData mean embedding 170           importance: 0.025
  MatscholarElementData std_dev embedding 153        importance: 0.022
  MatscholarElementData mean embedding 140           importance: 0.017
  PymatgenData mean mendeleev_no                     importance: 0.004
Now make predictions & invert scaling
  unscaling y: minmax
     rmse training     Ymean            0.123401
  unscaling y: minmax
     rmse test         Ymean            0.186584
Predictions made & saved in "training.csv" & "test.csv"
Plot results in "training.csv" & "test.csv"
  training, (rmse & R2) = (0.123 & 0.986)
  test, (rmse & R2) = (0.187 & 0.952)
  showing Ymean
```



Model 3: Kernel Ridge Regression (KRR)

In [5]:
```
from matsml.models import KRR

# Model parameters
nfold_cv = 5
model_file = 'model_krr.pkl'
alpha = [-2,5]
gamma = [-2,5]
n_grids = 10
kernel = 'rbf'

model_params={'kernel':kernel,'nfold_cv':nfold_cv,'model_file':model_file,'alpha':alpha,
    'gamma':gamma,'n_grids':n_grids}

model = KRR(data_params=data_params,model_params=model_params)
model.train()
model.plot(pdf_output=False)
```

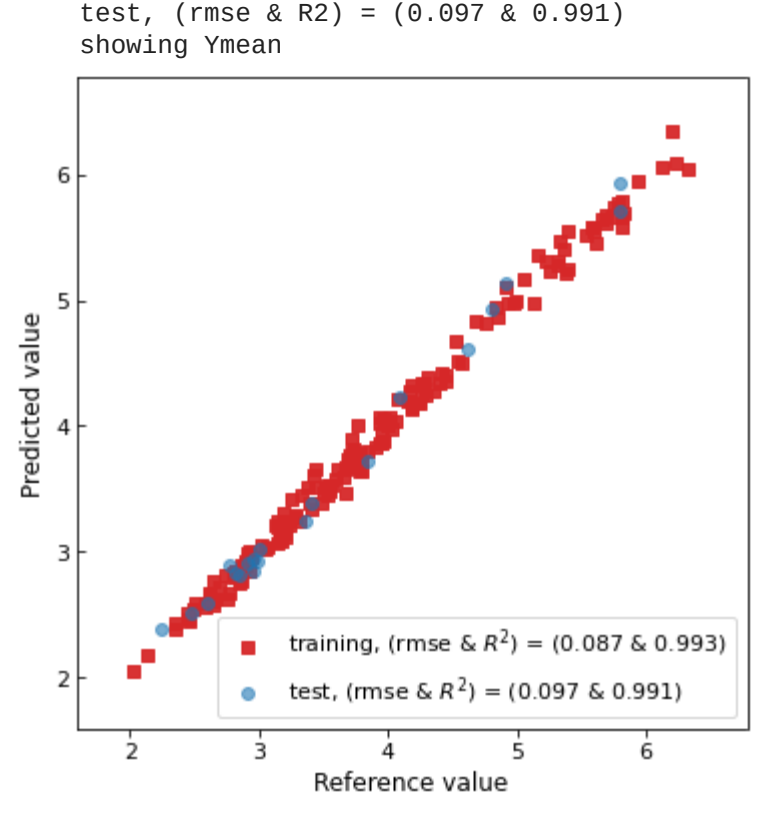
```
Checking parameters
  all passed                    True

Learning fingerprinted/featured data
  algorithm                     kernel ridge regression w/ scikit-learn
  kernel                        rbf
  nfold_cv                      5
  alpha                         [-2, 5]
  gamma                         [-2, 5]
  number of alpha/gamma grids   10
Read data
  data file                     fp_hoips_S1_1dest.csv.gz
  data size                     192
  training size                 172 (89.6 %)
  test size                     20 (10.4 %)
  x dimensionality              32
  y dimensionality              1
  y label(s)                    ['Ymean']
Scaling x                       minmax
  xscaler saved in              xscaler.pkl
Scaling y                       minmax
Prepare train/test sets         random
Building model                  KRR
Training model w/ cross validation
KRR model trained, now make predictions & invert scaling
  unscaling y: minmax
     rmse training     Ymean            0.186155
  unscaling y: minmax
     rmse test         Ymean            0.317096
Predictions made & saved in "training.csv" & "test.csv"
Plot results in "training.csv" & "test.csv"
  training, (rmse & R2) = (0.186 & 0.967)
  test, (rmse & R2) = (0.317 & 0.920)
  showing Ymean
```



Model 4: Gaussian Process Regression

In [6]:
```
from matsml.models import GPR

# Model parameters
nfold_cv=5
model_file='model_gpr.pkl'
verbosity=0
rmse_cv=True
n_restarts_optimizer=100

model_params={'nfold_cv':nfold_cv,'n_restarts_optimizer':n_restarts_optimizer,'model_file':model_file,
    'verbosity':verbosity,'rmse_cv':rmse_cv}

model=GPR(data_params=data_params,model_params=model_params)
model.train()
model.plot(pdf_output=False)
```

```
Checking parameters
  all passed                    True

Learning fingerprinted/featured data
  algorithm                     gaussian process regression w/ scikit-learn
  nfold_cv                      5
  optimizer                     fmin_l_bfgs_b
  n_restarts_optimizer          100
  rmse_cv                       True
Read data
  data file                     fp_hoips_S1_1dest.csv.gz
  data size                     192
  training size                 172 (89.6 %)
  test size                     20 (10.4 %)
  x dimensionality              32
  y dimensionality              1
  y label(s)                    ['Ymean']
Scaling x                       minmax
  xscaler saved in              xscaler.pkl
Scaling y                       minmax
Prepare train/test sets         random
Training model w/ cross validation
  cv,rmse_train,rmse_test,rmse_opt: 0 0.020118 0.034120 0.034120
  unscaling y: minmax
     rmse cv_test      Ymean            0.146779
  cv,rmse_train,rmse_test,rmse_opt: 1 0.016859 0.040843 0.034120
  unscaling y: minmax
     rmse cv_test      Ymean            0.175702
  cv,rmse_train,rmse_test,rmse_opt: 2 0.019410 0.038437 0.034120
  unscaling y: minmax
     rmse cv_test      Ymean            0.165348
  cv,rmse_train,rmse_test,rmse_opt: 3 0.018702 0.031402 0.031402
  unscaling y: minmax
     rmse cv_test      Ymean            0.135085
  cv,rmse_train,rmse_test,rmse_opt: 4 0.020163 0.032842 0.031402
  unscaling y: minmax
     rmse cv_test      Ymean            0.141281
GPR model trained, now make predictions & invert scaling
  unscaling y: minmax
     rmse training     Ymean            0.087482
  unscaling y: minmax
     rmse test         Ymean            0.09733
Predictions made & saved in "training.csv" & "test.csv"
Plot results in "training.csv" & "test.csv"
  training, (rmse & R2) = (0.087 & 0.993)
  test, (rmse & R2) = (0.097 & 0.991)
  showing Ymean
```



Model 5: Fully-Connected Neural Net

In [7]:
```
from matsml.models import FCNN

# model parameters
layers=[5,5]
epochs=300
nfold_cv=5
use_bias=True
model_file='model_fcnn.pkl'
loss='mse'
verbosity=0
batch_size=32
activ_funct='elu'
optimizer='nadam'

model_params={'layers':layers,'activ_funct':activ_funct,'epochs':epochs,'nfold_cv':nfold_cv,
    'optimizer':optimizer,'use_bias':use_bias,'model_file':model_file,'loss':loss,
    'batch_size':batch_size,'verbosity':verbosity,'rmse_cv':False}

model=FCNN(data_params=data_params,model_params=model_params)
model.train()
model.plot(pdf_output=False)
```

```
Checking parameters
  all passed                    True

Learning fingerprinted/featured data
  algorithm                     fully connected NeuralNet w/ TensorFlow
  layers                        [5, 5]
  activ_funct                   elu
  epochs                        300
  optimizer                     nadam
  nfold_cv                      5
Read data
  data file                     fp_hoips_S1_1dest.csv.gz
  data size                     192
  training size                 172 (89.6 %)
  test size                     20 (10.4 %)
  x dimensionality              32
  y dimensionality              1
  y label(s)                    ['Ymean']
Scaling x                       minmax
  xscaler saved in              xscaler.pkl
Scaling y                       minmax
Prepare train/test sets         random
Building model                  FCNN
Training model w/ cross validation
  cv,rmse_train,rmse_test,rmse_opt: 0 0.046565 0.058864 0.058864
  cv,rmse_train,rmse_test,rmse_opt: 1 0.032680 0.040572 0.040572
  cv,rmse_train,rmse_test,rmse_opt: 2 0.028597 0.041473 0.040572
  cv,rmse_train,rmse_test,rmse_opt: 3 0.027411 0.041591 0.040572
  cv,rmse_train,rmse_test,rmse_opt: 4 0.026858 0.033028 0.033028
  Optimal ncv:  4 ; optimal NET saved
FCNN trained, now make predictions & invert scaling
  unscaling y: minmax
     rmse training     Ymean            0.121247
  unscaling y: minmax
     rmse test         Ymean            0.127479
Predictions made & saved in "training.csv" & "test.csv"
Plot results in "training.csv" & "test.csv"
  training, (rmse & R2) = (0.121 & 0.987)
  test, (rmse & R2) = (0.127 & 0.981)
  showing Ymean
```