In [2]:

```
#1
import numpy as np
my_NIA = 100419401
np.random.seed(my_NIA)
#2
from numpy.random import randint
import pandas as pd
train = pd.read_csv('C:/Users/15096/Downloads/python/train.csv')
test = pd.read_csv('C:/Users/15096/Downloads/python/test.csv')
#5.select the first blue point 75 columns
X_trainfirst= train.iloc[:,0:75]
y_train= train.energy.values
X_testfirst= test.iloc[:,0:75]
y_test =  test.energy.values
#3.Normalize using MinMaxScaler
from sklearn import preprocessing
min_max_scaler = preprocessing.MinMaxScaler()
X_train_minmax = min_max_scaler.fit_transform(X_trainfirst)
X_test_minmax = min_max_scaler.transform(X_testfirst)
#6(a)knn
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
knn= KNeighborsRegressor()
knn.fit(X_train_minmax, y_train)
y_test_knn_pred = knn.predict(X_test_minmax)
knnmsesqrt=np.sqrt(mean_squared_error(y_test_knn_pred, y_test))
knnR2=r2_score(y_test,y_test_knn_pred )
print('KNN Square root of Mean squared error: '+str(knnmsesqrt))
print('KNN R_squared: '+str(knnR2))
#6(a)decisiontree
from sklearn import tree
np.random.seed(my_NIA)
clf = tree.DecisionTreeRegressor()
clf.fit(X_train_minmax, y_train)
y_test_tree_pred = clf.predict(X_test_minmax)
treemsesqrt=np.sqrt(mean_squared_error(y_test_tree_pred, y_test))
treeR2=r2_score(y_test,y_test_tree_pred )
print('Regression Tree Square root of Mean squared error: '+str(treemsesqrt))
print('Regression Tree R_squared: '+str(treeR2))
#6(a)svm
from sklearn.preprocessing import StandardScaler
#from sklearn import svm
from sklearn.svm import SVR
np.random.seed(my_NIA)
svr = SVR(gamma = 'auto')
scaler=StandardScaler()
ytrain=scaler.fit_transform(y_train.reshape(-1, 1))
ytest=scaler.transform(y_test.reshape(-1, 1))
svr.fit(X_train_minmax, ytrain)
y_test_svm_pred = svr.predict(X_test_minmax)
svmmsesqrt=np.sqrt(mean_squared_error(scaler.inverse_transform(y_test_svm_pred),scaler.
inverse_transform(ytest)))
svmR2=r2_score(ytest,y_test_svm_pred )
print('SVM Square root of Mean squared error: '+str(svmmsesqrt))
print('SVM R_squared: '+str(svmR2))
```

```
KNN Square root of Mean squared error: 3526915.724816481
KNN R_squared: 0.7809776040385839
Regression Tree Square root of Mean squared error: 4600124.03386241
Regression Tree R_squared: 0.6274046381817606
```

```
C:\Users\15096\Anaconda3\lib\site-packages\sklearn\utils\validation.py:72
4: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

```
SVM Square root of Mean squared error: 3136212.3113783826
SVM R_squared: 0.8268153997679144
```

In [3]:

```
#4.validation using PredefinedSplit
from sklearn.model_selection import PredefinedSplit
validation_indices = np.zeros(X_train_minmax.shape[0])
validation_indices[:round(10/12*X_train_minmax.shape[0])] = -1
tr_val_partition = PredefinedSplit(validation_indices)
#6(b)knn
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import RandomizedSearchCV
import numpy as np
from sklearn import metrics
knn = KNeighborsRegressor()
#parameter number of neighbors is integer
param_grid = {'n_neighbors': range(1,733,1)}
knn_grid = RandomizedSearchCV(knn,param_grid,n_iter=20,cv=tr_val_partition,n_jobs=-1, v
erbose=1,random_state = my_NIA)
knn_grid.fit(X_train_minmax, y_train)
y_test_knnb_pred = knn_grid.predict(X_test_minmax)
knnbbest=knn_grid.best_params_
knnbmsesqrt=np.sqrt(mean_squared_error(y_test_knnb_pred, y_test))
knnbR2=r2_score(y_test,y_test_knnb_pred )
print('KNN Best parameters: '+str(knnbbest))
print('KNN Square root of Mean squared error: '+str(knnbmsesqrt))
print('KNN R_squared: '+str(knnbR2))
```

```
Fitting 1 folds for each of 20 candidates, totalling 20 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent worker
s.
[Parallel(n_jobs=-1)]: Done  20 out of  20 | elapsed:   36.3s finished
```

```
KNN Best parameters: {'n_neighbors': 85}
KNN Square root of Mean squared error: 3281007.5023953407
KNN R_squared: 0.8104547926855329
```

In [4]:

```
#6b.tree
from sklearn import tree
np.random.seed(my_NIA)
clf = tree.DecisionTreeRegressor()
param_grid = {'max_depth': range(2,100,2),'min_samples_split': range(2,100,2)}
clf_grid = RandomizedSearchCV(clf,param_grid,n_iter=20,cv=tr_val_partition,n_jobs=-1, v
erbose=1,random_state = my_NIA)
clf_grid.fit(X_train_minmax, y_train)
y_test_treeb_pred = clf_grid.predict(X_test_minmax)
treebbest=clf_grid.best_params_
treebmsesqrt=np.sqrt(mean_squared_error(y_test_treeb_pred, y_test))
treebR2=r2_score(y_test,y_test_treeb_pred )
print('Regression Tree Best parameters: '+str(treebbest))
print('Regression Tree Square root of Mean squared error: '+str(treebmsesqrt))
print('Regression Tree R_squared: '+str(treebR2))
```

```
Fitting 1 folds for each of 20 candidates, totalling 20 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent worker
s.
[Parallel(n_jobs=-1)]: Done  20 out of  20 | elapsed:    4.2s finished

Regression Tree Best parameters: {'min_samples_split': 96, 'max_depth': 8
4}
Regression Tree Square root of Mean squared error: 3685109.584086934
Regression Tree R_squared: 0.7608892097470532
```

In [14]:

```
#6b.svm
from sklearn.svm import SVR
np.random.seed(my_NIA)
svr = SVR(gamma = 'auto')
param_grid = {'C': range(1,10000,10),'degree':range(2,20,1),'epsilon': np.arange(0.001,
1,0.01)}
svm_grid = RandomizedSearchCV(svr,param_grid,n_iter=20,cv=tr_val_partition,n_jobs=-1, v
erbose=1,random_state = my_NIA)
svm_grid.fit(X_train_minmax, ytrain)
y_test_svmb_pred = svm_grid.predict(X_test_minmax)
svmbbest=svm_grid.best_params_
svmbmsesqrt=np.sqrt(mean_squared_error(scaler.inverse_transform(y_test_svmb_pred),scale
r.inverse_transform(ytest)))
svmbR2=r2_score(ytest,y_test_svmb_pred )
print('SVM Best parameters: '+str(svmbbest))
print('SVM Square root of Mean squared error: '+str(svmbmsesqrt))
print('SVM R_squared: '+str(svmbR2))
```

```
Fitting 1 folds for each of 20 candidates, totalling 20 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent worker
s.
[Parallel(n_jobs=-1)]: Done  20 out of  20 | elapsed:  4.9min finished
C:\Users\15096\Anaconda3\lib\site-packages\sklearn\utils\validation.py:72
4: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)

SVM Best parameters: {'epsilon': 0.32099999999999995, 'degree': 9, 'C': 13
81}
SVM Square root of Mean squared error: 2993328.661320041
SVM R_squared: 0.8422362688166181
```

In [ ]:

```

```

In these 6 methods, the results of SVM Hyper-parameters tuning has the smallest squart root of MSE and the largest R-squared, so it is the best.