```
# Import libraries which are useful for this task.

import numpy as np #Library with many mathematical functions.
import pandas as pd #Library to manage data.
import sklearn as sk #Machine learning library.
import matplotlib.pyplot as plt #Plotting library

#Import the CFD dataset with pandas - This is the Data original without normalization - Later in the code we will normalize it
dataset = pd.read_csv('data/Data_original_all46cases.csv')
dataset = dataset.sample(frac=1).reset_index(drop=True)

# Data interested
dataset_pred = pd.read_csv('data/Data_original_INTERESTED_KINS.csv')
dataset_pred = dataset_pred.sample(frac=1).reset_index(drop=True)

# Clean dataset deleting all missing values and show some samples
datasetClean = dataset.dropna()
datasetClean_pred = dataset_pred.dropna()


# Splits data into inputs and targets using sklearn.
import sklearn
from sklearn.model_selection  import train_test_split

# Drop deletes variables that you don't want as inputs (efficiency. We will try using all other variables as the input for our
X_all = datasetClean.drop(['Power coefficient','Efficiency'], axis = 1)
y_PC  = datasetClean['Power coefficient']
y_Eff = datasetClean['Efficiency']


# Putting each feature in an array shape and reshaping it
y_PC  = np.asarray(y_PC)
y_PC  = np.reshape(y_PC, (-1, 1))
y_Eff = np.asarray(y_Eff)
y_Eff = np.reshape(y_Eff,(-1, 1))


# This cell is where I normalized the Data_original.csv and create the Data_normalize.csv. I'm using these built-in functions
from sklearn import preprocessing

scalerAll = preprocessing.StandardScaler().fit(X_all)
YscaterPCAll  = preprocessing.StandardScaler().fit(y_PC)
YscaterEffAll = preprocessing.StandardScaler().fit(y_Eff)
X_scaledALL = scalerAll.transform(X_all)
y_scaledPCAll = YscaterPCAll.transform(y_PC)
y_scaledEffAll = YscaterEffAll.transform(y_Eff)


# This is perhaps the most important cell of the code. We setup our regression model here. The "GridSearchCV" is a tool that t
from sklearn import linear_model
from sklearn.svm import SVR
from sklearn import svm
from sklearn.model_selection import GridSearchCV

# RBF KERNEL
def svc_param_selection(X, y, nfolds):
    Cs = [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000]
    gammas = [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000]
    param_grid = {'C': Cs, 'gamma' : gammas}
    grid_search = GridSearchCV(SVR(kernel='rbf'), param_grid, iid=True, cv=nfolds)
    grid_search.fit(X, y)
    grid_search.best_params_
    return grid_search.best_params_


# Checking which combination of parameters provides highest score using 5-fold cross-validation (That's the 5 at the end of se
# This is done for each response variable (Cp and eta) and each model (RBF, Linear). Below I'm showing for eta and RBF kernel.
print('Parameters RBF for Efficiency prediction')
RBF_Eff = svc_param_selection(X_scaledALL, np.ravel(y_scaledEffAll, order='C'), 5)
print('RBF_Eff is', RBF_Eff)
print('Parameters RBF for Power prediction')
RBF_PC = svc_param_selection(X_scaledALL, np.ravel(y_scaledPCAll, order='C'), 5)
print('RBF_PC is', RBF_PC)
```

```
Parameters RBF for Efficiency prediction
/data/bernardo/anaconda3/envs/tensorFlow-gpu/lib/python3.7/site-packages/sklearn/model_selection/_search.py:823: FutureWa
  "removed in 0.24.", FutureWarning
RBF_Eff is {'C': 100000, 'gamma': 0.001}
Parameters RBF for Power prediction
RBF_PC is {'C': 10000, 'gamma': 0.01}
/data/bernardo/anaconda3/envs/tensorFlow-gpu/lib/python3.7/site-packages/sklearn/model_selection/_search.py:823: FutureWa
  "removed in 0.24.", FutureWarning
```

```
# Import the values above and put them as parameters for the models below.
svr_rbf_Eff = SVR(kernel='rbf', C=RBF_Eff['C'], gamma=RBF_Eff['gamma'])
svr_rbf_PC  = SVR(kernel='rbf', C=RBF_PC['C'], gamma=RBF_PC['gamma'])


#Code for setting up the 5-fold cross-validation.
from sklearn.model_selection import KFold

scalerPC  = preprocessing.StandardScaler().fit(datasetClean_pred) ##changed
scalerEff = preprocessing.StandardScaler().fit(datasetClean_pred) ##changed

y_PC_not_scaled = pd.DataFrame(y_PC)
y_Eff_not_scaled = pd.DataFrame(y_Eff)
X_not_scaled = pd.DataFrame(X_all)
X    = pd.DataFrame(X_scaledALL)
y_PC = pd.DataFrame(y_scaledPCAll)
y_Eff = pd.DataFrame(y_scaledEffAll)

X_test_scaledEff_NEW = scalerEff.transform(datasetClean_pred)
X_test_scaledPC_NEW = scalerPC.transform(datasetClean_pred)

kf = KFold(n_splits=5) # Define the split - into 5 folds
kf.get_n_splits(X) # returns the number of splitting iterations in the cross-validator
```

```
    5
```

```
# Score for each fold
import csv
i = 0
count = 0
predict_eta_all = [0,0,0,0,0]
print('RBF Efficiency KFold')
for train_index, test_index in kf.split(X):
    i = i+1
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y_Eff.iloc[train_index], y_Eff.iloc[test_index]
    svr_rbf_Eff.fit(X_train, np.ravel(y_train))
    score = svr_rbf_Eff.score(X_test, y_test)
    count = count + score


    predict_eta_NOTscaled = svr_rbf_Eff.predict(X_test_scaledEff_NEW)
    predict_eta_folds = YscaterEffAll.inverse_transform(predict_eta_NOTscaled)    ##changed
    predict_eta_all = predict_eta_all + predict_eta_folds


    print('Fold:', i, 'Score:', svr_rbf_Eff.score(X_test, y_test))
    print('Predicted Eff:', predict_eta_folds)
print('Average Score:',count/5)
print('Mean predicted efficiency:', predict_eta_all/5)
print(datasetClean_pred)

with open('predicted_Eff_RBF_testdata_AlexisJulia.csv', 'w') as o:
    write = csv.writer(o)
    write.writerow(predict_eta_all/5)
datasetClean_pred.to_csv(path_or_buf='./X_testdata_Eff.csv',index=False)
```

```
    RBF Efficiency KFold
    Fold: 1 Score: 0.9874570607502415
    Predicted Eff: [0.23588251 0.31081699 0.22826429 0.13026965 0.20220122]
    Fold: 2 Score: 0.9889167979693078
    Predicted Eff: [0.21692794 0.30697559 0.26025195 0.10699508 0.19137404]
    Fold: 3 Score: 0.8874901416728731
    Predicted Eff: [0.22531108 0.2913457  0.23693887 0.12341879 0.19482979]
    Fold: 4 Score: 0.9826172673932861
    Predicted Eff: [0.24360205 0.32425237 0.22752229 0.14306151 0.186465  ]
    Fold: 5 Score: 0.9636737296362647
    Predicted Eff: [0.23243075 0.30653323 0.22449175 0.13255851 0.19413135]
    Average Score: 0.9620309994843946
    Mean predicted efficiency: [0.23083087 0.30798478 0.23549383 0.12726071 0.19380028]
       Heave amplitude  Pitch amplitude  Frequency  α_{T/4}  α_rate_{max}      Yp
    0             1.00               65       0.15   0.3787        0.3223  2.2747
    1             0.75               65       0.15   0.5191        0.4147  1.8377
    2             0.75               65       0.10   0.6941        0.4167  1.8377
    3             1.25               65       0.15   0.2675        0.2949  2.7303
    4             1.25               65       0.10   0.4687        0.2480  2.7303
```

```
# Score for each fold
import csv
i = 0
count = 0
predict_PC_all = [0,0,0,0,0]
print('RBF Power Coefficient KFold')
```

```
for train_index, test_index in kf.split(X):
    i = i+1
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y_PC.iloc[train_index], y_PC.iloc[test_index]
    svr_rbf_PC.fit(X_train, np.ravel(y_train))
    score = svr_rbf_PC.score(X_test, y_test)
    count = count + score


    predict_PC_NOTscaled = svr_rbf_PC.predict(X_test_scaledPC_NEW)
    predict_PC_folds = YscaterPCAll.inverse_transform(predict_PC_NOTscaled)    ##changed
    predict_PC_all = predict_PC_all + predict_PC_folds


    print('Fold:', i, 'Score:', svr_rbf_PC.score(X_test, y_test))
    print('Predicted Eff:', predict_PC_folds)
print('Average Score:',count/5)
print('Mean predicted power coefficient:', predict_PC_all/5)
print(datasetClean_pred)

with open('predicted_PC_RBF_testdata_AlexisJulia.csv', 'w') as o:
    write = csv.writer(o)
    write.writerow(predict_PC_all/5)
datasetClean_pred.to_csv(path_or_buf='./X_testdata_PC.csv',index=False)
```

```
RBF Power Coefficient KFold
Fold: 1 Score: 0.9930890847631009
Predicted Eff: [0.47793828 0.47194519 0.32770408 0.33429577 0.56480493]
Fold: 2 Score: 0.9833287042413719
Predicted Eff: [0.47879782 0.47470006 0.32574282 0.33638702 0.57320042]
Fold: 3 Score: 0.8534773282486624
Predicted Eff: [0.4816932  0.44151156 0.30816416 0.34311442 0.5370895 ]
Fold: 4 Score: 0.9940822573237588
Predicted Eff: [0.47467726 0.46912075 0.33688421 0.33082598 0.56686715]
Fold: 5 Score: 0.9140633465650669
Predicted Eff: [0.45542843 0.46481571 0.3145118  0.3149773  0.59033283]
Average Score: 0.9476081442283922
Mean predicted power coefficient: [0.473707   0.46441865 0.32260141 0.3319201  0.56645897]
   Heave amplitude  Pitch amplitude  Frequency  α_{T/4}  α_rate_{max}      Yp
0             1.00               65       0.15   0.3787        0.3223  2.2747
1             0.75               65       0.15   0.5191        0.4147  1.8377
2             0.75               65       0.10   0.6941        0.4167  1.8377
3             1.25               65       0.15   0.2675        0.2949  2.7303
4             1.25               65       0.10   0.4687        0.2480  2.7303
```