# V2A3_regression_airfoilnoise

December 14, 2018

```
In [19]: #!/usr/bin/env python
         # V2A3_regression_airfoilnoise.py
         # Programmgeruest zu Versuch 2, Aufgabe 3
         # to log outputs start with: python V2A3_regression_airfoilnoise.py >V2A3_regression_

         import numpy as np
         import pandas as pd

         from V2A2_Regression import *


         # ***** MAIN PROGRAM ********
         # (I) Hyper-Parameters
         S=3;                  # S-fold cross-validation
         lmbda=1;              # regularization parameter (lambda>0 avoids also singularities)
         K=13;                  # K for K-Nearest Neighbors
         flagKLinReg = 1;      # if flag==1 and K>=D then do a linear regression of the KNNs to m
         deg=5;                # degree of basis function polynomials
         flagSTD=1;            # if >0 then standardize data before training (i.e., scale X to me
         N_pred=5;             # number of predictions on the training set for testing
         x_test_1 = [1250,11,0.2,69.2,0.0051];   # REPLACE dummy code: define test vector 1
         x_test_2 = [1305,8,0.1,57.7,0.0048];   # REPLACE dummy code: define test vector 2

In [2]: # (II) Load data
        fname='./AirfoilSelfNoise/airfoil_self_noise.xls'
        airfoil_data = pd.read_excel(fname,0); # load data as pandas data frame
        T = airfoil_data.values[:,5]           # target values = noise load (= column 5 of dat
        X = airfoil_data.values[:,:5]          # feature vectors (= column 0-4 of data table)
        N,D=X.shape                            # size and dimensionality of data set
        idx_perm = np.random.permutation(N)    # get random permutation for selection of test
        print("Data set ",fname," has size N=", N, " and dimensionality D=",D)
        print("X=",X)
        print("T=",T)
        print("x_test_1=",x_test_1)
        print("x_test_2=",x_test_2)
        print("number of basis functions M=", len(phi_polynomial(X[1],deg)))
```

```
Data set  ./AirfoilSelfNoise/airfoil_self_noise.xls  has size N= 1502  and dimensionality D= 5
```

```
X= [[1.00000e+03 0.00000e+00 3.04800e-01 7.13000e+01 2.66337e-03]
 [1.25000e+03 0.00000e+00 3.04800e-01 7.13000e+01 2.66337e-03]
 [1.60000e+03 0.00000e+00 3.04800e-01 7.13000e+01 2.66337e-03]
 ...
 [4.00000e+03 1.56000e+01 1.01600e-01 3.96000e+01 5.28487e-02]
 [5.00000e+03 1.56000e+01 1.01600e-01 3.96000e+01 5.28487e-02]
 [6.30000e+03 1.56000e+01 1.01600e-01 3.96000e+01 5.28487e-02]]
T= [125.201 125.951 127.591 ... 106.604 106.224 104.204]
x_test_1= [1250, 11, 0.2, 69.2, 0.0051]
x_test_2= [1305, 8, 0.1, 57.7, 0.0048]
number of basis functions M= 252
```

```
In [3]: # (III) Do least-squares regression with regularization
        print("\n#### Least Squares Regression with regularization lambda=", lmbda, " ####")
        lsr = LSRRegressifier(lmbda=lmbda,phi=lambda x: phi_polynomial(x,deg),flagSTD=flagSTD)
        lsr.fit(X,T)
        print("lsr.W_LSR=",lsr.W_LSR)    # REPLACE dummy code: print weight vector for least s
        print("III.1) Some predictions on the training data:")
        for i in range(N_pred):
            n=idx_perm[i]
            print("Prediction for X[",n,"]=",X[n]," is y=",lsr.predict(X[n]),", whereas true va
        print("III.2) Some predicitions for new test vectors:")
        print("Prediction for x_test_1 is y=", lsr.predict(x_test_1))    # REPLACE dummy code:
        print("Prediction for x_test_2 is y=", lsr.predict(x_test_2))    # REPLACE dummy code:
        print("III.3) S=",S,"fold Cross Validation:")
        err_abs,err_rel = lsr.crossvalidate(S,X,T)                  # REPLACE dummy code: do c
        print("absolute errors (E,sd,min,max)=", err_abs, "\nrelative errors (E,sd,min,max)=",
```

```
#### Least Squares Regression with regularization lambda= 1  ####
lsr.W_LSR= [-3.94646520e-01 -1.80155527e+00 -3.96662974e-01 -8.64281233e-01
  3.26438812e-01 -6.21770966e-01  6.48905815e-01 -7.71134930e-01
  2.95779376e-01 -8.67132337e-02  3.61560493e-01 -2.81989365e-02
 -3.64074945e-02  1.43998983e-01  4.77503076e-02 -3.63380016e-02
 -9.73944497e-02 -3.71505203e-02 -1.53013579e-01 -2.47759041e-01
  1.29242314e-01  3.13172340e-01  1.50772922e-01  8.40072758e-01
 -8.44083280e-02  1.32532606e+00  2.52907215e-01  2.08504016e-01
 -2.03801432e-01  5.74149325e-01  3.06551563e-01 -2.34531355e-01
  4.66271349e-01 -2.43145107e-01  1.06213081e-02  2.84851888e-01
 -8.48806231e-02 -2.00727798e-01 -6.93365225e-02 -5.88853578e-02
  7.29728355e-02 -1.36528243e-01 -2.26202702e-01 -1.28035094e-01
 -2.24863325e-01 -1.13338956e-01  4.19411521e-01 -1.68848150e-02
  9.78093460e-02 -9.27714500e-02 -1.60258656e-02 -3.22486764e-01
  1.01134491e-01  7.97108235e-03 -1.81313744e-01 -3.69616798e-01
 -2.47146762e-01  2.02432625e-01 -4.38061153e-01  3.64617492e-02
 -7.59308984e-01  1.95668256e-01  6.70564131e-01 -1.47283279e-01
 -2.77532541e-01 -3.08022282e-01  1.19842567e-02 -7.66976903e-02
```

```
  2.18511621e-02  1.72857687e-01 -6.25981195e-01  1.36043453e-01
  6.68313146e-02 -1.65130385e-02 -5.46617990e-02  5.48724195e-02
 -5.19208304e-02 -7.63188408e-02  3.78906795e-02  1.94056398e-01
 -2.78668584e-01 -7.05574660e-01  8.01753250e-05 -1.97576222e-01
  5.53327040e-02 -2.03772994e-02 -1.47505902e-01  5.91246025e-02
 -1.79737621e-02 -7.12236378e-02 -2.34113997e-01 -4.01567558e-02
  2.90612583e-02 -7.81051692e-02  2.06254670e-05  5.56183281e-02
  4.80864131e-03  2.42168243e-02 -1.15039749e-01  1.75335472e-02
  2.94441955e-02 -2.98387467e-01  2.24608964e-03 -5.67996318e-02
  3.60431186e-03  3.50473096e-02 -9.11562735e-02 -2.58145605e-02
  3.76595743e-02  1.50183200e-01 -3.75077570e-02  1.27138401e-02
  1.16133919e-01 -2.18208523e-01  1.41690489e-01  1.29547055e-01
  1.02847461e-01  7.46744531e-03 -5.50688491e-02 -1.09489486e-01
  4.33124764e-02  3.52862834e-02  1.02885915e-01  1.58937930e-01
  3.22368270e-02  1.33278437e-01  1.54200823e-02  1.03812366e-02
  2.68478094e-02 -4.69455614e-03 -1.03924197e-01  3.63370895e-01
  1.95941884e-01  8.55311399e-03 -6.75202423e-01  6.17886875e-02
 -5.57216536e-03 -6.85068836e-01  9.73106644e-04  8.93303864e-04
  3.32687011e-01  1.03800671e-01  2.21151780e-01 -1.24168329e-01
  5.08255859e-02 -6.41651342e-02 -3.57687689e-02  1.84616236e-01
 -5.38713351e-02  2.16353496e-01 -4.49026953e-02  5.07394826e-02
 -1.58277599e-03 -3.52975947e-02 -2.32615544e-02  1.11961868e-01
 -3.17638241e-01  3.66642352e-03  1.01689005e-01 -9.14031375e-02
  1.07835193e-01 -5.52562467e-02 -1.69661381e-01  1.65517660e-02
 -1.62822814e-01 -3.68756103e-01  8.04196574e-02 -8.18975854e-02
 -1.12357085e-01  3.69551139e-02 -5.07037615e-02 -1.69293973e-01
  1.78704521e-01  2.25341988e-01 -8.85750241e-02  5.14764795e-02
  1.97391087e-01  5.44295165e-02  2.08205031e-01  8.63666478e-03
  2.50397522e-01  2.12918833e-01  3.01972410e-02 -4.05213862e-03
 -1.89041169e-02 -1.59662947e-01  2.21658174e-01  2.82466967e-02
  8.76627247e-02  4.79010683e-02  9.35769244e-02  9.70424870e-02
 -2.11928518e-02 -2.45208682e-02 -2.29960413e-02 -1.97186002e-02
  1.40300710e-01  2.80593592e-01  1.01183417e-01  7.86300171e-03
 -1.36568507e-01  1.93395520e-01  1.49290958e-02  9.97525775e-02
 -5.93850011e-02 -3.88941947e-02 -2.24188695e-01  1.14703293e-01
 -8.61448667e-02 -3.94278300e-02 -3.86775078e-02 -1.38123752e-01
  3.28920405e-02 -1.46146334e-01 -8.05652042e-02 -9.70461716e-02
  2.15697923e-01  3.68525314e-02  2.31497755e-01 -2.69752137e-02
  1.55073486e-02  1.84839607e-01  8.53004635e-03  5.65295263e-02
 -2.07214477e-01 -2.92021485e-02  4.13387816e-03  3.20040301e-02
 -6.48052853e-02 -9.35343805e-02 -1.63259658e-02 -1.15971880e-01
 -5.33499384e-02 -1.74711442e-01 -7.62523429e-02 -3.15365640e-03
 -3.13599999e-02  1.47577802e-02  8.98079738e-02  5.76990605e-02
  3.07022562e-02  6.10550828e-02  4.30451202e-02  2.50332155e-02
 -1.23965738e-01 -1.06664213e-01 -3.90944224e-02  3.18752489e-02
 -5.83388127e-02  6.63287979e-03  8.53339681e-02  2.11106178e-02]
III.1) Some predictions on the training data:
Prediction for X[ 829 ]= [4.00000e+02 8.40000e+00 5.08000e-02 3.17000e+01 5.80776e-03]  is y=
Prediction for X[ 416 ]= [2.50000e+03 7.30000e+00 2.28600e-01 5.55000e+01 1.11706e-02]  is y=
Prediction for X[ 1390 ]= [5.00000e+03 8.90000e+00 1.01600e-01 7.13000e+01 1.03088e-02]  is y=
Prediction for X[ 483 ]= [5.00000e+02 0.00000e+00 1.52400e-01 3.96000e+01 1.93287e-03]  is y=
Prediction for X[ 188 ]= [1.60000e+03 0.00000e+00 2.28600e-01 7.13000e+01 2.14345e-03]  is y=
III.2) Some predicitions for new test vectors:
Prediction for x_test_1 is y= 130.45406757371268
Prediction for x_test_2 is y= 133.1060885540002
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.125811178469721, 2.690278675182672, 0.0007087477896305927,
relative errors (E,sd,min,max)= (0.01717171824559571, 0.02247069082835731, 5.604299921959377e-0
```

```
In [13]: lmbdaRange = [1,2,5,10,30,60,100]
         degRange = list(range(1,8))
         LSRErrors = []

         for lmb in lmbdaRange:
             for d in degRange:
                 # (III) Do least-squares regression with regularization
                 print("\n#### Least Squares Regression with regularization lambda=", lmb, " de
                 lsr = LSRRegressifier(lmbda=lmb,phi=lambda x: phi_polynomial(x,d),flagSTD=fla
                 lsr.fit(X,T)
                 print("lsr.W_LSR=",lsr.W_LSR)    # REPLACE dummy code: print weight vector fo
                 #print("III.1) Some predictions on the training data:")
                 #for i in range(N_pred):
                     #n=idx_perm[i]
                     #print("Prediction for X[",n,"]=",X[n]," is y=",lsr.predict(X[n]),", wher
                 #print("III.2) Some predicitions for new test vectors:")
                 #print("Prediction for x_test_1 is y=", lsr.predict(x_test_1))    # REPLACE d
                 #print("Prediction for x_test_2 is y=", lsr.predict(x_test_2))    # REPLACE d
                 print("III.3) S=",S,"fold Cross Validation:")
                 err_abs,err_rel = lsr.crossvalidate(S,X,T)                  # REPLACE dummy c
                 LSRErrors.append((lmb, d, err_abs, err_rel))
                 print("absolute errors (E,sd,min,max)=", err_abs, "\nrelative errors (E,sd,mi
```

```
#### Least Squares Regression with regularization lambda= 1  deg= 1  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.7426904512740404, 3.032703726901288, 0.0029708583213334805,
relative errors (E,sd,min,max)= (0.030060493569663435, 0.02449174579917344, 2.3060478008317074

#### Least Squares Regression with regularization lambda= 1  deg= 2  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.22587019951892, 2.6693604535048245, 0.0026685517388500557,
relative errors (E,sd,min,max)= (0.025897430202553428, 0.021811447774288523, 2.068548547238156

#### Least Squares Regression with regularization lambda= 1  deg= 3  ####
III.3) S= 3 fold Cross Validation:
```

```
absolute errors (E,sd,min,max)= (2.7347807102442734, 2.3893427037029857, 0.0002965685705191845
relative errors (E,sd,min,max)= (0.02196847951570637, 0.01947415135850052, 2.310893914514236e-0

#### Least Squares Regression with regularization lambda= 1  deg= 4  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.2456495781657986, 2.1540798635988536, 1.297882903372738e-06
relative errors (E,sd,min,max)= (0.0180589043841237, 0.017579753437655145, 9.941044619040871e-0

#### Least Squares Regression with regularization lambda= 1  deg= 5  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.079394989462073, 2.6766875476072753, 4.29290338388455e-07, 5
relative errors (E,sd,min,max)= (0.016801803898304456, 0.0226686473647146, 3.4147901076916435e-

#### Least Squares Regression with regularization lambda= 1  deg= 6  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.2957782795547192, 5.122638764911872, 0.0009009670520754298,
relative errors (E,sd,min,max)= (0.018755926460175557, 0.04376694648989689, 7.375967482954668e

#### Least Squares Regression with regularization lambda= 1  deg= 7  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.114929218803049, 14.410874829608249, 0.004234547474098349, 3
relative errors (E,sd,min,max)= (0.02568383146690318, 0.12075851945814975, 3.5880522243203146e-

#### Least Squares Regression with regularization lambda= 2  deg= 1  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.7569078931827056, 3.0317572426077755, 0.003470868555695006,
relative errors (E,sd,min,max)= (0.030179780340022746, 0.02447823687099956, 2.7053599143348243e

#### Least Squares Regression with regularization lambda= 2  deg= 2  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.2284375950967044, 2.672695427737961, 0.0014469068371028015,
relative errors (E,sd,min,max)= (0.025914114620162222, 0.021840776468178086, 1.182142403084063

#### Least Squares Regression with regularization lambda= 2  deg= 3  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.723022378845087, 2.33037313640339, 0.005364161043928561, 17
relative errors (E,sd,min,max)= (0.021889347041410483, 0.019155411861625473, 4.122662468242127e

#### Least Squares Regression with regularization lambda= 2  deg= 4  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.317308938966723, 2.405452925551814, 0.00015155235057306982,
relative errors (E,sd,min,max)= (0.01863658107558814, 0.01991601544274848, 1.2594727048372792e-

#### Least Squares Regression with regularization lambda= 2  deg= 5  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.2140329576941595, 3.4731868862390223, 0.0001087470957656933
relative errors (E,sd,min,max)= (0.017863992272434386, 0.028731687944487178, 8.026208263760669e
#### Least Squares Regression with regularization lambda= 2  deg= 6  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.4477248913966725, 7.652512481362627, 0.00111133510961281, 19
relative errors (E,sd,min,max)= (0.019958046149505533, 0.06601593432000205, 8.655190455002763e-

#### Least Squares Regression with regularization lambda= 2  deg= 7  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.810629787854275, 24.85341735958863, 0.002117378118825286, 60
relative errors (E,sd,min,max)= (0.03162034370223333, 0.2101699733344223, 1.7699094881179666e-0

#### Least Squares Regression with regularization lambda= 5  deg= 1  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.7506024924723578, 3.032830214576294, 0.00037176279994355355
relative errors (E,sd,min,max)= (0.03012803600660093, 0.024495176574837716, 2.8456170964112668e

#### Least Squares Regression with regularization lambda= 5  deg= 2  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.2279765367112643, 2.6775661041333856, 0.001175387957005114,
relative errors (E,sd,min,max)= (0.025918357466509896, 0.021869430842587925, 9.399794928226176e

#### Least Squares Regression with regularization lambda= 5  deg= 3  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.7302481706619557, 2.3531895256067523, 0.0004029707441475239
relative errors (E,sd,min,max)= (0.021911654213841985, 0.01910965928499763, 3.3873335138993636e

#### Least Squares Regression with regularization lambda= 5  deg= 4  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.311375992596948, 2.1700971336011885, 0.0018465395963147557,
relative errors (E,sd,min,max)= (0.018507324981580833, 0.017439732931967857, 1.459288268502300

#### Least Squares Regression with regularization lambda= 5  deg= 5  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.411585043918261, 4.31765592466851, 0.0021945167101051766, 7
relative errors (E,sd,min,max)= (0.01949651009966399, 0.03599614203439813, 1.8158862649917473e-

#### Least Squares Regression with regularization lambda= 5  deg= 6  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.477518643827081, 7.581707045240742, 0.0015872592767038896, 
relative errors (E,sd,min,max)= (0.020103807525183353, 0.06337298471038427, 1.2501451386228514e

#### Least Squares Regression with regularization lambda= 5  deg= 7  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.1845893933877925, 16.082465601489815, 0.0005608191692090259
relative errors (E,sd,min,max)= (0.026402687727984334, 0.1405787988071728, 4.170341388249571e-0

#### Least Squares Regression with regularization lambda= 10  deg= 1  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.742155012101423, 3.0263009930574807, 0.006927864542689122, 
relative errors (E,sd,min,max)= (0.030063424440043188, 0.024441415447091767, 5.748644994887789e

#### Least Squares Regression with regularization lambda= 10  deg= 2  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.2379216280230656, 2.649471203388498, 0.004210543135044986, 1
relative errors (E,sd,min,max)= (0.025999412971781204, 0.021663991894828333, 3.568377856067143

#### Least Squares Regression with regularization lambda= 10  deg= 3  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.7340349643516424, 2.350083066960976, 0.0015759836836792829,
relative errors (E,sd,min,max)= (0.021930042587339042, 0.019126577203682136, 1.353054434973112

#### Least Squares Regression with regularization lambda= 10  deg= 4  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.409388698101218, 2.432416949136901, 0.001532779223296643, 40
relative errors (E,sd,min,max)= (0.019325388017867682, 0.01982834252873825, 1.398278786794846e

#### Least Squares Regression with regularization lambda= 10  deg= 5  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.234195989248932, 2.121534010953418, 0.0025382810222112084, 2
relative errors (E,sd,min,max)= (0.017956140469290147, 0.01725996586805743, 2.244081886845732e

#### Least Squares Regression with regularization lambda= 10  deg= 6  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.5602552756167625, 9.00328476368852, 0.002633425072346540, 2
relative errors (E,sd,min,max)= (0.020751031714465722, 0.07495572547465935, 2.138909253042999e-

#### Least Squares Regression with regularization lambda= 10  deg= 7  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.103051588404497, 13.51087387237831, 0.0020731792111945424, 3
relative errors (E,sd,min,max)= (0.02550898529474507, 0.11480268635477847, 1.724875168432889e-0

#### Least Squares Regression with regularization lambda= 30  deg= 1  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.7777550094122505, 3.0244330098017578, 0.000419805951125340
relative errors (E,sd,min,max)= (0.030365586989831397, 0.024451667317409918, 3.400752976024468

#### Least Squares Regression with regularization lambda= 30  deg= 2  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.2809239249011712, 2.642898794982149, 0.001646122559293417, 1
relative errors (E,sd,min,max)= (0.02635681278700867, 0.021554702116273885, 1.306436107089163e

#### Least Squares Regression with regularization lambda= 30  deg= 3  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.8382511077000037, 2.412407125817957, 1.3588696987199e-05, 19
relative errors (E,sd,min,max)= (0.022742983562023363, 0.019529522912631964, 1.098147515572643e

#### Least Squares Regression with regularization lambda= 30  deg= 4  ####
```

```
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.5890345636142227, 2.273643562159436, 0.012023518690440937, 
relative errors (E,sd,min,max)= (0.02074436751510817, 0.0182647582299099, 9.464431151410935e-05

#### Least Squares Regression with regularization lambda= 30  deg= 5  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.412559346328815, 2.237932988840751, 0.003451936031240166, 3
relative errors (E,sd,min,max)= (0.019386828911170204, 0.018254456542067057, 2.65668418677187

#### Least Squares Regression with regularization lambda= 30  deg= 6  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.5058579497215034, 5.186451328739485, 0.0007290385572389368,
relative errors (E,sd,min,max)= (0.020390232663006793, 0.04625963441738052, 5.537450304117828e

#### Least Squares Regression with regularization lambda= 30  deg= 7  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.9985107098346377, 11.845681164296396, 0.0008234550130765683
relative errors (E,sd,min,max)= (0.024523676736373554, 0.10085170307596095, 6.49740415569821e-

#### Least Squares Regression with regularization lambda= 60  deg= 1  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.8109090124664777, 3.0116547710172616, 0.0001756231887384274
relative errors (E,sd,min,max)= (0.03065971381958029, 0.024400032033097173, 1.3497639665095794

#### Least Squares Regression with regularization lambda= 60  deg= 2  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.3282684035419687, 2.6585966362451803, 0.00193337538621563, 
relative errors (E,sd,min,max)= (0.026733721429068966, 0.0216523422574755, 1.5579047600064706e

#### Least Squares Regression with regularization lambda= 60  deg= 3  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.9532643180504476, 2.4534510335926627, 0.000441089811985762,
relative errors (E,sd,min,max)= (0.023683424387214463, 0.019894615890664918, 3.477447017066468

#### Least Squares Regression with regularization lambda= 60  deg= 4  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.765244625944936, 2.364397090653817, 0.0028601658374896033, 
relative errors (E,sd,min,max)= (0.022188193078656163, 0.019060760838449316, 2.22868907498371

#### Least Squares Regression with regularization lambda= 60  deg= 5  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.6451006847152647, 2.4178997267887268, 0.0009083847589153038
relative errors (E,sd,min,max)= (0.02124472381469571, 0.01963778986070437, 7.32060634491646e-0

#### Least Squares Regression with regularization lambda= 60  deg= 6  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.5410885475503795, 2.876190865147863, 0.0004001764245913364,
relative errors (E,sd,min,max)= (0.020482275891270836, 0.023827882649049083, 3.07891965709290
#### Least Squares Regression with regularization lambda= 60  deg= 7  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.8188246673388804, 7.625955641049158, 0.00020830192210041787
relative errors (E,sd,min,max)= (0.022935438584442813, 0.06513041537890986, 1.6403534413275311

#### Least Squares Regression with regularization lambda= 100  deg= 1  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.8462157593707826, 2.9855651208444205, 0.0007201157916654211
relative errors (E,sd,min,max)= (0.030971720791043118, 0.024240638723650634, 6.178069592187895

#### Least Squares Regression with regularization lambda= 100  deg= 2  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.378695578225125, 2.6798710380810338, 0.00145583518953174, 1
relative errors (E,sd,min,max)= (0.027176009732771445, 0.021937703134095463, 1.162779797234683

#### Least Squares Regression with regularization lambda= 100  deg= 3  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.119905413687999, 2.5589411646946645, 0.0021414888370543395,
relative errors (E,sd,min,max)= (0.025028135401136315, 0.02073456800559623, 1.815482622527141e

#### Least Squares Regression with regularization lambda= 100  deg= 4  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.8660910169225926, 2.413350209168368, 0.0011655085791915099,
relative errors (E,sd,min,max)= (0.02299226727390611, 0.019439980305185255, 8.716885272958858e

#### Least Squares Regression with regularization lambda= 100  deg= 5  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.7274445916032586, 2.3707656183144374, 0.003232786052549841,
relative errors (E,sd,min,max)= (0.021907205814727434, 0.019213121216694294, 2.860012078268344

#### Least Squares Regression with regularization lambda= 100  deg= 6  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (2.636027593825774, 2.5132068344622502, 0.0010726537573333417,
relative errors (E,sd,min,max)= (0.021200250155637607, 0.020675549043432703, 8.927324577902872

#### Least Squares Regression with regularization lambda= 100  deg= 7  ####
III.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.1931982699978474, 12.275334135532917, 0.0001720281927362066
relative errors (E,sd,min,max)= (0.02614840592942353, 0.10900138362891708, 1.3364734748536074e
```

```
In [36]: temp = []
         for i,e in enumerate(LSRErrors):
             temp.append((e[2][3],i))
         temp.sort()
         print(temp)
```

```
        smallest_mean= LSRErrors[4]
        smallest_sd= LSRErrors[25]
        smallest_min= LSRErrors[4]
        smallest_max= LSRErrors[37]
        print(smallest_mean)
        print(smallest_sd)
        print(smallest_min)
        print(smallest_max)
```

```
[(17.002552206577917, 22), (17.17767182006628, 7), (17.183366839571775, 29), (17.28912745095228
(1, 5, (2.079394989462073, 2.6766875476072753, 4.29290338388455e-07, 53.06619440818116), (0.016
(10, 5, (2.234195989248932, 2.121534010953418, 0.0025382810222112084, 25.637192620876164), (0.0
(1, 5, (2.079394989462073, 2.6766875476072753, 4.29290338388455e-07, 53.06619440818116), (0.016
(60, 3, (2.9532643180504476, 2.4534510335926627, 0.000441089811985762, 18.064386599763182), (0
```

```
In [20]: # (IV) Do KNN regression
         print("\n#### KNN regression with flagKLinReg=", flagKLinReg, " ####")
         knnr = KNNRegressifier(K,flagKLinReg)                                          # REPLACE dumm
         knnr.fit(X,T)
         print("IV.1) Some predictions on the training data:")
         for i in range(N_pred):
             n=idx_perm[i]
             print("Prediction for X[",n,"]=",X[n]," is y=",knnr.predict(X[n]),", whereas true
         print("IV.2) Some predicitions for new test vectors:")
         print("Prediction for x_test_1 is y=", knnr.predict(x_test_1))    # REPLACE dummy cod
         print("Prediction for x_test_2 is y=", knnr.predict(x_test_2))    # REPLACE dummy cod
         print("IV.3) S=",S,"fold Cross Validation:")
         err_abs,err_rel = knnr.crossvalidate(S,X,T)                      # REPLACE dummy code: d
         print("absolute errors (E,sd,min,max)=", err_abs, "\nrelative errors (E,sd,min,max)="
```

```
#### KNN regression with flagKLinReg= 1  ####
IV.1) Some predictions on the training data:
Prediction for X[ 829 ]= [4.00000e+02 8.40000e+00 5.08000e-02 3.17000e+01 5.80776e-03]  is y= 
Prediction for X[ 416 ]= [2.50000e+03 7.30000e+00 2.28600e-01 5.55000e+01 1.11706e-02]  is y= 
Prediction for X[ 1390 ]= [5.00000e+03 8.90000e+00 1.01600e-01 7.13000e+01 1.03088e-02]  is y=
Prediction for X[ 483 ]= [5.00000e+02 0.00000e+00 1.52400e-01 3.96000e+01 1.93287e-03]  is y= 
Prediction for X[ 188 ]= [1.60000e+03 0.00000e+00 2.28600e-01 7.13000e+01 2.14345e-03]  is y= 
IV.2) Some predicitions for new test vectors:
Prediction for x_test_1 is y= 126.56192024990972
Prediction for x_test_2 is y= 132.35085577388358
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.0858901947930324, 3.0296772825440974, 0.006016325705218151, 
relative errors (E,sd,min,max)= (0.02482629194750422, 0.024838560765987942, 4.864074982591945e
```

```
In [5]: flagRange = [0,1]
        kRange = list(range(1,15))
```

```
    KNNErrors = []

    for f in flagRange:
        for k in kRange:
            # (IV) Do KNN regression
            print("\n#### KNN regression with flagKLinReg=", f," K=",k, " ####")
            knnr = KNNRegressifier(k,f)                                   # REPLACE dummy
            knnr.fit(X,T)
            #print("IV.1) Some predictions on the training data:")
            #for i in range(N_pred):
             #   n=idx_perm[i]
             #   print("Prediction for X[",n,"]=",X[n]," is y=",knnr.predict(X[n]),", wher
            #print("IV.2) Some predicitions for new test vectors:")
            #print("Prediction for x_test_1 is y=", knnr.predict(x_test_1))    # REPLACE d
            #print("Prediction for x_test_2 is y=", knnr.predict(x_test_2))    # REPLACE d
            print("IV.3) S=",S,"fold Cross Validation:")
            err_abs,err_rel = knnr.crossvalidate(S,X,T)                   # REPLACE dummy
            KNNErrors.append((f, k, err_abs, err_rel))
            print("absolute errors (E,sd,min,max)=", err_abs, "\nrelative errors (E,sd,min
```

```
#### KNN regression with flagKLinReg= 0  K= 1  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (6.0853868175765635, 4.99436767804508, 0.001999999999981128, 2
relative errors (E,sd,min,max)= (0.04912818981049262, 0.040765285335784385, 1.5524213892472526

#### KNN regression with flagKLinReg= 0  K= 2  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (5.401201731025304, 4.437756334591083, 0.004999999999954525,
relative errors (E,sd,min,max)= (0.04362866562469845, 0.03622014995869743, 3.8867252784803315e

#### KNN regression with flagKLinReg= 0  K= 3  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (5.074835996449179, 3.9655100986458507, 0.0010000000000189857,
relative errors (E,sd,min,max)= (0.04098097175540358, 0.032219640944628085, 8.253208434935712e

#### KNN regression with flagKLinReg= 0  K= 4  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (4.883228861517983, 3.750245281393827, 0.0004999999999881766,
relative errors (E,sd,min,max)= (0.03941390524524453, 0.03051554822483716, 3.888538920294102e-

#### KNN regression with flagKLinReg= 0  K= 5  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (4.847787749667112, 3.714858480290325, 0.0018000000000031332, 2
relative errors (E,sd,min,max)= (0.03913347456447976, 0.03026196375103344, 1.377326150856492e-

#### KNN regression with flagKLinReg= 0  K= 6  ####
IV.3) S= 3 fold Cross Validation:
```

```
absolute errors (E,sd,min,max)= (4.89022170439414, 3.7467723260804333, 0.0015000000000071623, 2
relative errors (E,sd,min,max)= (0.03946505572906337, 0.030365017889787457, 1.2067190114615477e

#### KNN regression with flagKLinReg= 0  K= 7  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (4.844357808636098, 3.6938115808069454, 0.009571428571447882, 2
relative errors (E,sd,min,max)= (0.03910101455378332, 0.029957539645611396, 8.285516422652252e-

#### KNN regression with flagKLinReg= 0  K= 8  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (4.771372003994679, 3.5984948639421726, 0.002125000000006594, 1
relative errors (E,sd,min,max)= (0.03854479152090372, 0.02933092400407393, 1.6186653057232912e-

#### KNN regression with flagKLinReg= 0  K= 9  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (4.774563470927656, 3.5035274445073377, 0.004222222222225014, 1
relative errors (E,sd,min,max)= (0.038557961254384515, 0.028503638217019116, 3.3180267520294644

#### KNN regression with flagKLinReg= 0  K= 10  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (4.772091344873507, 3.6162643948215316, 0.0021000000000128694,
relative errors (E,sd,min,max)= (0.038556342121063944, 0.029431227594854188, 1.635539494394671e

#### KNN regression with flagKLinReg= 0  K= 11  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (4.734131400556837, 3.5549829535219537, 0.003181818181829499, 1
relative errors (E,sd,min,max)= (0.038267916335408846, 0.02908225147523832, 2.5518852964105538e

#### KNN regression with flagKLinReg= 0  K= 12  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (4.692815301819794, 3.4911144936223484, 0.006000000000000227, 1
relative errors (E,sd,min,max)= (0.03791187355673365, 0.028425770832824944, 4.764135587298995e-

#### KNN regression with flagKLinReg= 0  K= 13  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (4.756177097203728, 3.501078687348059, 0.017769230769204114, 18
relative errors (E,sd,min,max)= (0.03841421498874818, 0.028513890419747126, 0.000138042391563

#### KNN regression with flagKLinReg= 0  K= 14  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (4.726039756515124, 3.5184082358647784, 0.006785714285712174, 1
relative errors (E,sd,min,max)= (0.03817416444978584, 0.028724311462901435, 5.39117821646037e-0

#### KNN regression with flagKLinReg= 1  K= 1  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (6.060707057256993, 4.964199241964238, 0.010000000000005116, 25
relative errors (E,sd,min,max)= (0.048975214440824325, 0.04069129870337774, 7.925814377431336e-
#### KNN regression with flagKLinReg= 1  K= 2  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (12.525296493965014, 156.19994550924343, 0.007475974974482824,
relative errors (E,sd,min,max)= (0.10314319397674757, 1.3310995272082466, 5.801760846894483e-05

#### KNN regression with flagKLinReg= 1  K= 3  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (6.396450394461716, 14.243951188465537, 0.0005895413019914031,
relative errors (E,sd,min,max)= (0.05186724617747563, 0.11822191402821945, 4.629081488044561e-0

#### KNN regression with flagKLinReg= 1  K= 4  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (9.873142623128897, 29.582040218670087, 0.00048132142815404677
relative errors (E,sd,min,max)= (0.07976598558822078, 0.23506083147472215, 3.903502924893936e-0

#### KNN regression with flagKLinReg= 1  K= 5  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (6.394062215372286, 15.508139185342301, 0.003241006002610902, 1
relative errors (E,sd,min,max)= (0.0517349031397424, 0.12678270331444233, 2.6226540513685906e-0

#### KNN regression with flagKLinReg= 1  K= 6  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.9962579268823135, 5.153183065634384, 0.00028379955287505254
relative errors (E,sd,min,max)= (0.032236385567851994, 0.042056333578962844, 2.129412293849249e

#### KNN regression with flagKLinReg= 1  K= 7  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (4.066692010265598, 8.097136184863551, 0.0006482311390385576, 2
relative errors (E,sd,min,max)= (0.03296856316242199, 0.06974656414658123, 5.383130062851855e-0

#### KNN regression with flagKLinReg= 1  K= 8  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.403333527833385, 4.444956296758516, 0.002397379831705848, 8
relative errors (E,sd,min,max)= (0.02735963406595401, 0.03662814027623785, 1.9089090857526123e-

#### KNN regression with flagKLinReg= 1  K= 9  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.3604407091253656, 3.942000455561722, 0.005157777803233898, 7
relative errors (E,sd,min,max)= (0.02699942230586269, 0.032925383326237546, 4.3464297599449704e

#### KNN regression with flagKLinReg= 1  K= 10  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.2726824107736907, 3.2918649128033817, 0.002522439643087182,
relative errors (E,sd,min,max)= (0.026316080671558897, 0.02701570445005841, 1.9539406197662048e

#### KNN regression with flagKLinReg= 1  K= 11  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.095589333044935, 2.9358769628321757, 0.0040085245723275875,
relative errors (E,sd,min,max)= (0.024799742612328545, 0.02361783214154919, 3.298437045230389e

#### KNN regression with flagKLinReg= 1  K= 12  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.0239845610090543, 3.132087313659863, 0.0009492376890705145,
relative errors (E,sd,min,max)= (0.02427795940437536, 0.02571620093770124, 8.337031118327341e-0

#### KNN regression with flagKLinReg= 1  K= 13  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.055287704727373, 2.9020350466057616, 0.0012752504544266685,
relative errors (E,sd,min,max)= (0.02449401087971545, 0.023395966015677703, 1.0811146896128832e

#### KNN regression with flagKLinReg= 1  K= 14  ####
IV.3) S= 3 fold Cross Validation:
absolute errors (E,sd,min,max)= (3.0374429577788185, 3.599797758879916, 0.0018030588973516615,
relative errors (E,sd,min,max)= (0.024406729588508428, 0.030013356281360655, 1.524643582301332
```

```
In [18]: temp = []
         for i,e in enumerate(KNNErrors):
             temp.append((e[2][3],i))
         temp.sort()
         #print(temp)

         smallest_mean= KNNErrors[25][3]
         smallest_sd= KNNErrors[26][3]
         smallest_min= KNNErrors[19][3]
         smallest_max= KNNErrors[12][3]
         print(smallest_mean)
         print(smallest_sd)
         print(smallest_min)
         print(smallest_max)
         print(KNNErrors[26])
```

```
(0.02427795940437536, 0.02571620093770124, 8.337031118327341e-06, 0.43620988401224)
(0.02449401087971545, 0.023395966015677703, 1.0811146896128832e-05, 0.19243824726314146)
(0.032236385567851994, 0.042056333578962844, 2.129412293849249e-06, 0.5470062541566458)
(0.03841421498874818, 0.028513890419747126, 0.00013804239156331124, 0.13978792564378723)
(1, 13, (3.055287704727373, 2.9020350466057616, 0.0012752504544266685, 22.32668544746967), (0.
```

a) Vervollständigen Sie das Programmgerüst V2A3_regression_airfoilnoise.py um eine Least-Squares-Regression auf den Daten zu berechnen. Optimieren Sie die HyperParameter um bei einer S = 3-fachen Kreuzvalidierung möglichst kleine Fehlerwerte zu erhalten.

Welche Bedeutung haben jeweils die Hyper-Parameter lmbda, deg, flagSTD?

- lmbda: Regularisierungs parameter
- deg: Ist der Grad er polynomiellen Basisfunktion
- flagSTD: gibt an ob die Daten standartisiert werden sollen

Was passiert ohne Skalierung der Daten (flagSTD=0) bei höheren Polynomgraden (achten Sie auf die Werte von maxZ)?

- EXCEPTION DUE TO BAD CONDITION:flagOK= 0 maxZ= 195590361182.93994 eps= 1e-06

- der Fehlerwert beim invertieren explodiert ohne die Skalierung

Geben Sie Ihre optimalen Hyper-Parameter sowie die resultierenden Fehler-Werte an.

- lambda=1, deg=5, flagSTD=1
- absolute errors (E,sd,min,max)= (2.2241976966993438, 3.6504292455503355, 0.0023069096170331704, 83.63604473632992)
- relative errors (E,sd,min,max)= (0.018030635646137383, 0.03132168189811296, 1.746150761488692e-05, 0.762838110293237)

Welche Prognosen ergibt Ihr Modell für die neuen Datenvektoren x_test_1=[1250,11,0.2,69.2,0.0051] bzw. x_test_2=[1305,8,0.1,57.7,0.0048] ?

- Prediction for x_test_1 is y= 130.45406757371268
- Prediction for x_test_2 is y= 133.1060885540002

Welchen Polynomgrad und wieviele Basisfunktionen verwendet Ihr Modell?

- Polynomgrad ist 5
- mit 6 Basisfunktionen

b) Vervollständigen Sie das Programmgerüst V2A3_regression_airfoilnoise.py um eine KNN-Regression auf den Daten zu berechnen. Optimieren Sie die Hyper-Parameter um bei einer S = 3-fachen Kreuzvalidierung möglichst kleine Fehlerwerte zu erhalten.
Welche Bedeutung haben jeweils die Hyper-Parameter K und flagKLinReg?

- K: anzahl der NN die zum Ergebniss beitragen
- flagKLinReg: gibt an ob die KNN noch in einen LSRegressifier gegeben werden oder ob nur der Durchschnitt der KNN berechnet wird.

Geben Sie Ihre optimalen Hyper-Parameter sowie die resultierenden Fehler-Werte an.

- flagKLinReg=1, K=13
- absolute errors (E,sd,min,max)= (3.0858901947930324, 3.0296772825440974, 0.006016325705218151, 32.13059666400561)
- relative errors (E,sd,min,max)= (0.02482629194750422, 0.024838560765987942, 4.864074982591945e-05, 0.2851617187841634)

Welche Prognosen ergibt Ihr Modell für die neuen Datenvektoren x_test_1=[1250,11,0.2,69.2,0.0051] bzw. x_test_2=[1305,8,0.1,57.7,0.0048] ?

- Prediction for x_test_1 is y= 126.56192024990972
- Prediction for x_test_2 is y= 132.35085577388358

c) Vergleichen Sie die beiden Modelle. Welches liefert die besseren Ergebnisse?
LSR: absolute errors (E,sd,min,max)= (2.2241976966993438, 3.6504292455503355, 0.0023069096170331704, 83.63604473632992)
KNN: absolute errors (E,sd,min,max)= (3.0858901947930324, 3.0296772825440974, 0.006016325705218151, 32.13059666400561)
absolute differences: (0.861692498, 0.620751963, 0.003709416, 51.505448072)
LSR: relative errors (E,sd,min,max)= (0.018030635646137383, 0.03132168189811296, 1.746150761488692e-05, 0.762838110293237)
KNN: relative errors (E,sd,min,max)= (0.02482629194750422, 0.024838560765987942, 4.864074982591945e-05, 0.2851617187841634) relative differences: (0.006795656, 0.006483121, 0.000031179, 0.477676392

- KNN scheint die bessereren Ergebnisse zu liefern, verwendet intern aber auch LSR