Intro. to Machine Learning - Portfolio Pt.II

Exploratory Data Analysis and Regression

Programming, research, and report content produced by
Cameron Scott – csco0012
Monash College (Monash University).

Table of Contents

1.0 Introduction

Presented with a Kaggle dataset summarising air pollutant readings captured by roadside sensors, this paper investigates the trends of air particulate matter and characteristics, for the purposes of simple data science investigation.

Looking into the 2008 paper (De Vito et al.) discussing atmospheric pollutants in urban environments, this report simulates how data exploration and analysis offers insights into academic and commercial applications.

After establishing a body of descriptive statistical insights, we apply prediction techniques to forecast key variables under investigation. This is achieved through a conventional process in following the '*Data Science Roadmap*', beginning with data wrangling - cleaning, filtering, transforming - and iterating through the process drawing inferential insights through **regression techniques**.

Data visualisations are interspersed throughout to help inform and accelerate the insights gathering process, while alternative techniques are used to test the accuracy and veracity of model selection and predictive capacity.

Comparative modelling is undertaken for evaluation of respective models' predictive power. The ranging models are interpreted for context and pedagogical utility.

Concluding the paper, we see convergence of key performance metrics between higher-performing models. Additionally, cross-validation of differing regression techniques suggests confidence within bounds of examined prediction targets.

Importantly, the deployed techniques of manipulating the dataset, specifically through brash outlier removal, are observed as highly effective measures to 'bend the data to match the model'.

While these and similar conventional data wrangling techniques are ubiquitously adopted as pragmatic tools of practice, they are also highly impactful for dramatically reappraising a model's value – specifically, herein raising the R^2 metric by 60% after removing less than 2% of the datapoints.

The results may be interpreted to validate such analysis techniques as necessary and self-evidently valuable. Notwithstanding, such dramatic inferential impacts may also serve to remind data science practitioners of the inherent dangers posed where caution is not exercised when removing outliers and implementing other data cleaning techniques.

2.0 Report Body

This paper has been organised into a 'centre heavy' structure, wherein the main Report Body and Methodology sections house most of the content. This is in part due to the iterative workflow represented by the Data Science Roadmap, as described in section 3.2 Methodology.

2.1 Data / Materials Review

The dataset contains 9,373 instances of hourly averaged responses from an array of five metal oxide chemical sensors embedded in an Air Quality Chemical Multisensor Device. The device was located on the field in a significantly polluted area, at road level, within an Italian city. Data were recorded over a one-year period from March, 2004 to February, 2005, representing the longest freely available recordings of on-field deployed air quality chemical sensor devices responses. Ground Truth hourly averaged concentrations for CO, Non-Methane Hydrocarbons, Benzene, Total Nitrogen Oxides (NOx) and Nitrogen Dioxide (NO2) and were provided by a co-located reference certified analyzer. Evidence of cross-sensitivities as well as both concept and sensor drifts are presented as described in De Vito et al. (2008), eventually affecting sensor concentration estimation capabilities.

Column number	Description
1	Date (DD/MM/YYYY)
2	Time (HH:MM:SS)
3	True hourly average concentration of CO in mg/m^3 (reference analyzer)
4	Tin oxide hourly average sensor response (nominally CO targeted)
5	True hourly averaged overall Non-Methane Hydrocarbons concentration in $\mu g/m^3$ (reference analyzer)
6	True hourly averaged Benzene concentration in $\mu g/m^3$ (reference analyzer)
7	Titania hourly averaged sensor response (nominally NMHC targeted)
8	True hourly averaged NO_x concentration in ppb (reference analyzer)
9	Tungsten oxide hourly averaged sensor response (nominally NO_x targeted)
10	True hourly averaged NO_2 concentration in $\mu g/m^3$ (reference analyzer)
11	Tungsten oxide hourly averaged sensor response (nominally NO_2 targeted)
12	Indium oxide hourly averaged sensor response (nominally O_3 targeted)
13	Temperature in degrees Celsius
14	Relative humidity (%)
15	Absolute humidity

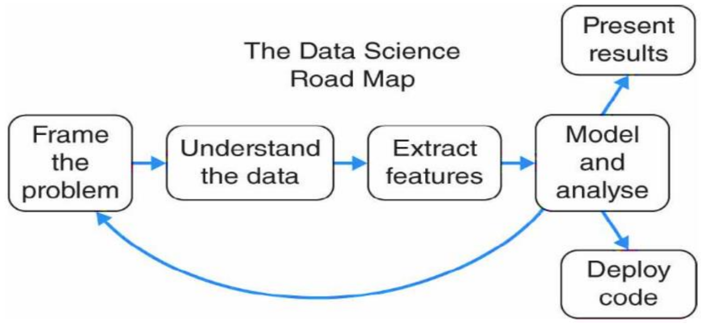
2.2 Methodology

The following process was utilised for iterating from task Problem through to solution.

This process approximately follows the depicted *Data Science Roadmap*. These steps are transcribed with respect to the project Problem as summarised below.

#1 >> Descriptive data profiling: Scatter-density plotting, correlation analysis:

To understand the data, 'eyeballing' the nature of key variables for feature extraction, data structure, shape, trend phenomena, etc.



#2 >> Data wrangling, cleaning, and transformation:

Preliminary data preparation for executing and evaluating 'best fit' linear and non-linear regression analysis. Outlier and erroneous data management techniques for sound evaluation of key trends.

#3 >> Inferential statistical techniques for predictive modelling.

Linear and non-linear regression employed first for correlation optimisation, then for further deployment of regression model selection and predictive extrapolation.

#4 >> Evaluation of models on their comparative predictive performance and selection justification.

Constrained regression model testing and parameter tweaking, nominating model of greatest predictive power for reliable extrapolative prediction.

These four broad stages of iterating through the Data Science Roadmap were executed via MatLab compiled adapted C code.

Application of this methodological framework is presented here on through extracts of the standard Matlab script editor, interwoven with comments and MatLab-executed output graphic images.

2.3 Matlab Code Workflow

```
%%%  <<<  SECTION 1  >>>  %%%

% Load dataset into memory from csv file
Dataset = "Dataset\AirQualityUCI.csv"
AirQual_MSTR = importdata(Dataset,",",1)

% >>> OUTPUT Master Dataset structure >>>
        >> AirQual_MSTR = importdata(Dataset,",",1)
        AirQual_MSTR = struct with fields:

                data: [9373×13 double]
            textdata: {9374×15 cell}

AirQualTT = readtimetable(Dataset);

% Review the data profile:
size(AirQualTT)
summary(AirQualTT)


% >>> OUTPUT TimeTable Data summary >>>
        >> size(AirQualTT)
        ans =      9373          14

        >> summary(AirQualTT)

        RowTimes:

            Date: 9373×1 datetime
                Values:
                    Min         03/10/2004
                    Median      09/21/2004
                    Max         04/04/2005

        Variables:

            Time: 9373×1 duration

                Properties:
                    Description:   Time
                Values:

                    Min        00:00:00
                    Median     11:00:00
                    Max        23:00:00

            CO_GT_: 9373×1 double

                Properties:
                    Description:   CO(GT)
                Values:

                    Min             0.1
                    Median          1.8
                    Max             11.9
                    NumMissing      1683

            PT08_S1_CO_: 9373×1 double

                Properties:
                    Description:   PT08.S1(CO)
                Values:

                    Min             647
                    Median          1064
                    Max             2040
                    NumMissing      366

            NMHC_GT_: 9373×1 double

                Properties:
                    Description:   NMHC(GT)
                Values:

                    Min             7
                    Median          150
                    Max             1189
                    NumMissing      8443

            C6H6_GT_: 9373×1 double

                Properties:
                    Description:   C6H6(GT)
                Values:

                    Min             0.1
                    Median          8.2
                    Max             63.7
                    NumMissing      366
            PT08_S2_NMHC_: 9373×1 double

                Properties:
                    Description:   PT08.S2(NMHC)
                Values:

                    Min             383
                    Median          909
                    Max             2214
                    NumMissing      366

            NOx_GT_: 9373×1 double

                Properties:
                    Description:   NOx(GT)
                Values:

                    Min             2
                    Median          180
                    Max             1479
                    NumMissing      1639

            PT08_S3_NOx_: 9373×1 double

                Properties:
                    Description:   PT08.S3(NOx)
                Values:

                    Min             322
                    Median          806
                    Max             2683
                    NumMissing      366
```

```
NO2_GT_: 9373×1 double

    Properties:
        Description:  NO2(GT)
    Values:

        Min            2
        Median         109
        Max            340
        NumMissing     1642

PT08_S4_NO2_: 9373×1 double

    Properties:
        Description:  PT08.S4(NO2)
    Values:

        Min            551
        Median         1463
        Max            2775
        NumMissing     366

PT08_S5_O3_: 9373×1 double

    Properties:
        Description:  PT08.S5(O3)
    Values:

        Min            221
        Median         964
        Max            2523
```

```
        NumMissing     366
T: 9373×1 double

    Properties:
        Description:  T
    Values:

        Min            -1.9
        Median         17.7
        Max            44.6
        NumMissing     366

RH: 9373×1 double

    Properties:
        Description:  RH
    Values:

        Min            9.2
        Median         49.6
        Max            88.7
        NumMissing     366

AH: 9373×1 double

    Properties:
        Description:  AH
    Values:

        Min       -3.9446
        Median     0.9195
        Max        5.2707
```

```
%%%  <<<  SECTION 2  >>>  %%%
% Eyeball the dataset for missing items via logic array:
Cheq = ismissing(AirQualTT)

% >>> OUTPUT Logic array >>>
        (not printed in report)

% Remove missing rows containing missing data:
AirQual = rmmissing(AirQualTT);

% Remove duplicated rows
AirQual = unique(AirQual);


% Re-CHECK the data structure - to see result of cleaning:
size(AirQual)

% >>> OUTPUT CLEANED Data structure >>>
        >> size(AirQual)

        ans =    840    14
```

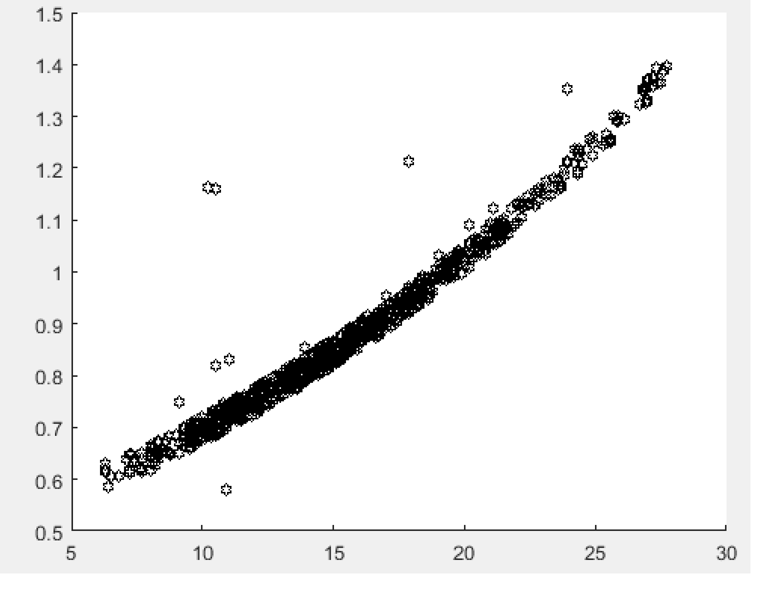
The following sections were programmed asynchronous to the indicative report Section structure. This allowed for necessary variable allocation consistent with MatLab compiler dependencies.

Therefore, Section 3 begins next, with the remainder of Section 2 to follow.

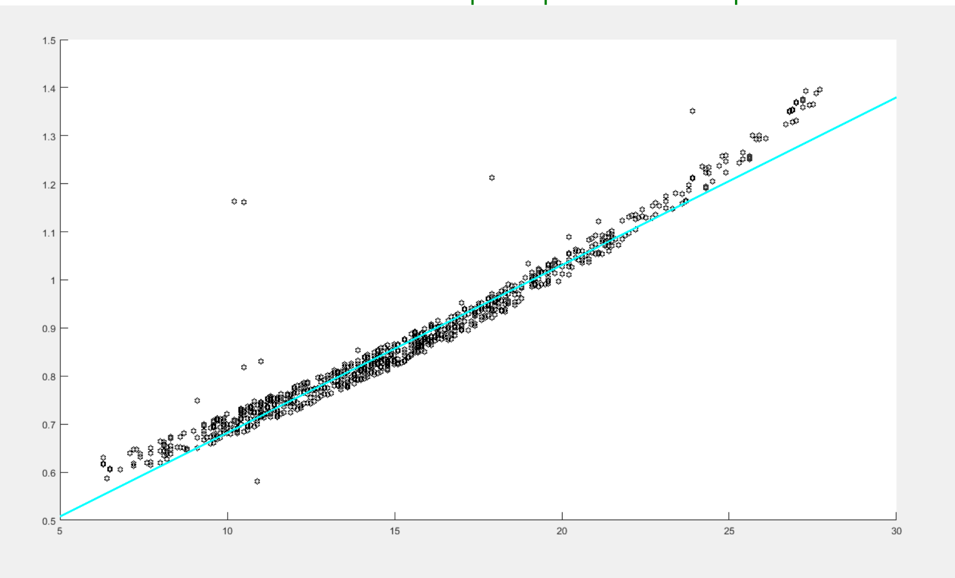
```
%%%  <<<  SECTION 3  >>>  %%%

%  Convert 'TIMETABLE' back into standard table format
TmprHmdty = array2table([AirQual.T(:), AirQual.AH(:)]);

% Rename relevant Table variables
TmprHmdty.Properties.VariableNames([1 2]) = {'Tmpr' 'Hmdty'};
Column_Headers = TmprHmdty.Properties.VariableNames  % <-- check

% >>> OUTPUT relabelled column headers >>>
      >> Column_Headers = TmprHmdty.Properties.VariableNames

      Column_Headers =  1×2 cell array
            {'Tmpr'}    {'Hmdty'}

% Extract and assign variables from table
Tmpr = TmprHmdty.Tmpr(:) ;
Hmdty = TmprHmdty.Hmdty(:) ;
```

As discussed above, Section 2 follows asynchronously as follows.

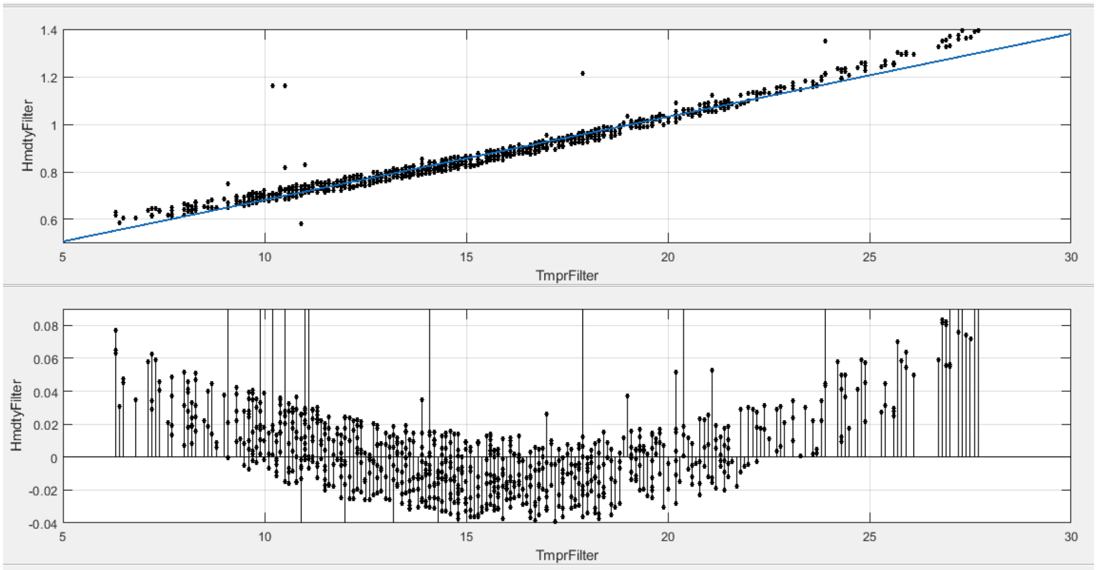
```
%%% (( SECTION 2 – Data removal LOOP )) %%%

% Filter out data > 28degrees via Loop algorithm
 k = 1;    % index for clean arrays

% Iterate through the original data arrays
for i = 1:length(Tmpr)
    if Tmpr(i) < 28
        Tmpr_crop(k) = Tmpr(i);
        Hmdty_crop(k) = Hmdty(i);
        k = k + 1;
    end
end

% 'Eyeball' the data via basic scatterplot
axis([5 30 0.5 1.5]);
hold on
plot(Tmpr_crop,Hmdty_crop,'kh');

%       >>>>> OUTPUT FIG.1: raw SCATTERPLOT  >>>>
```



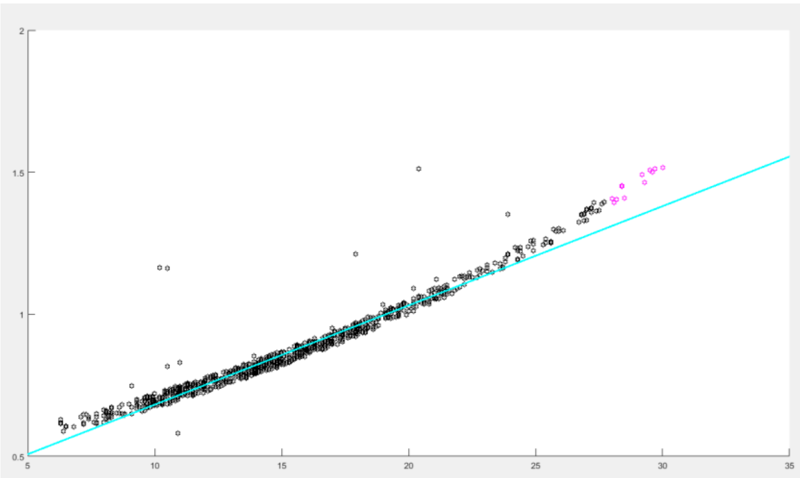
```
%%% <<< SECTION 4(A) >>> %%%

% Manually code Linear Regression metrics, model:

% Parameters generated from Polyfit function
p = polyfit(Tmpr_crop,Hmdty_crop,1)

% >>> OUTPUT [1x2] array:  p = [0.0349,0.3331]
p1 = p(1);  p2 = p(2);    % <--- assign as variables


% Create 'Line of best fit' model function:
x = 0:0.1:40;
fn_Lin = p1.*x + p2;

% Plot Linear Regression Model OVER scatterplot (ie. 'hold on' active)
plot(x,fn_Lin,'-c','LineWidth',2)

%  >>>> OUTPUT FIG.2: BestFit superimposed Scatterplot >>>>
```



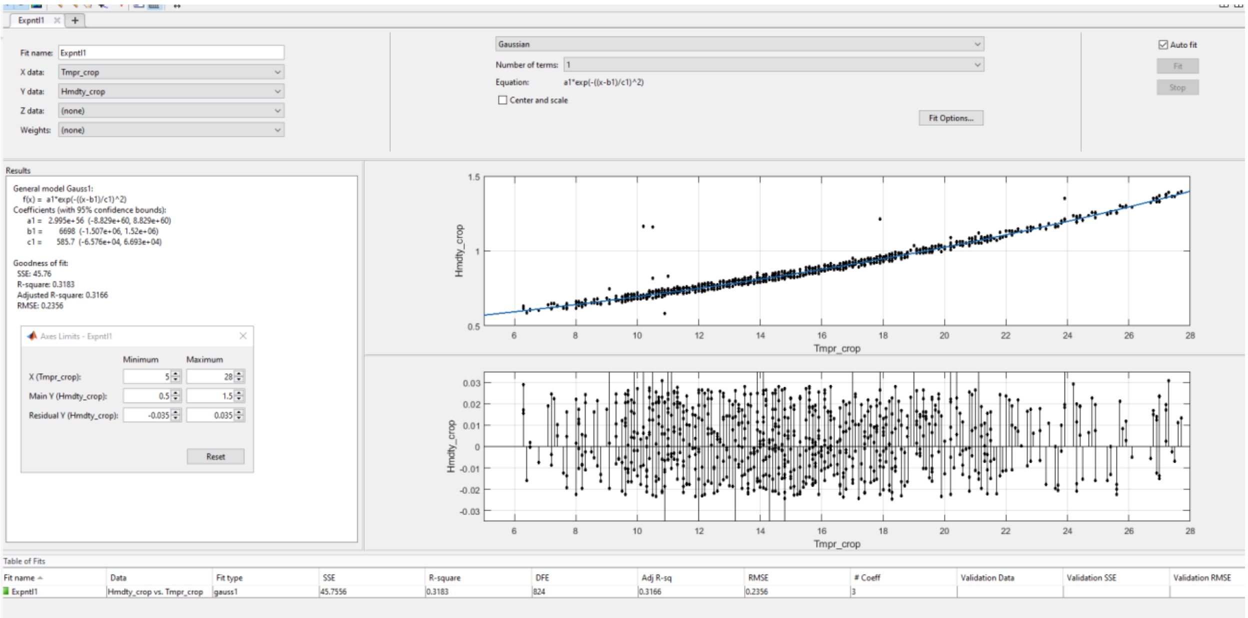
Repeating the previous step utilising in-built '*Curve Fitting Toolkit*' GUI:



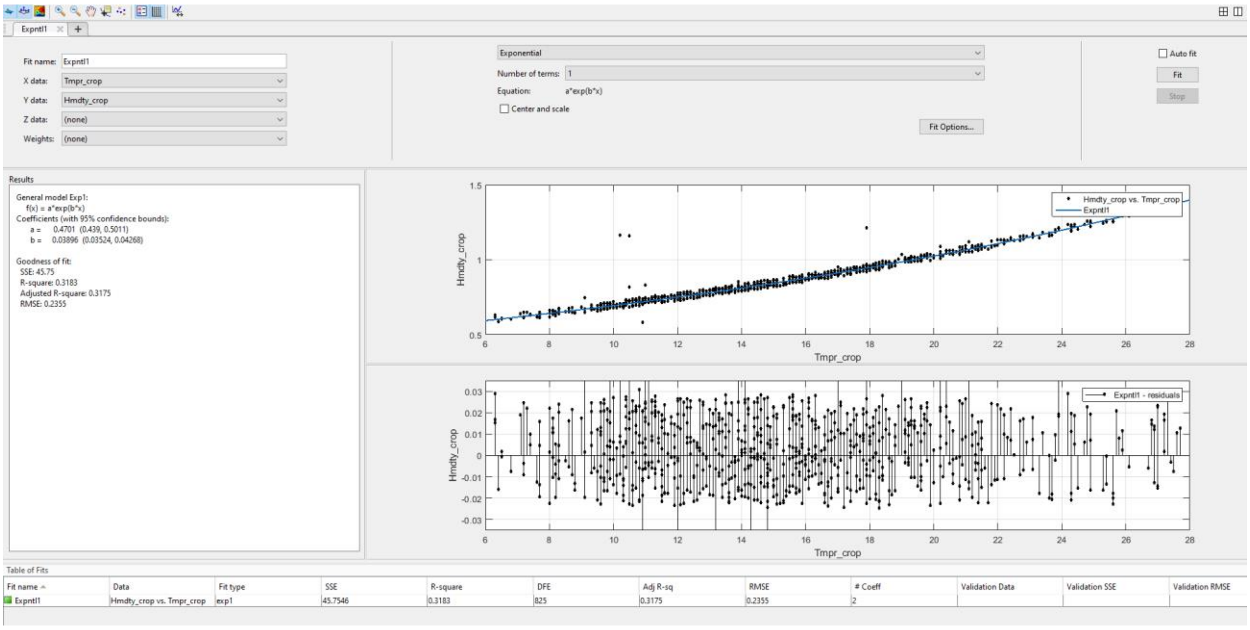
```
% Test predictive power of Linear Regression model:

Tmpr35 = 35;    % set 35degrees variable for model input

% Create LINEAR REGRESSION function @35 degs:
%  Input stored "p" variables as coefficients into 'Polyval' algorithm,
%  to predict Humidity at 35 degrees
fn_Lin35 = polyval(p,Tmpr35);

%  >>> OUTPUT distinctive Linear variable @35:
Hmdty_at35_Lin = fn_Lin35

      >> Hmdty_at35_Lin =  polyval(p,Tmpr35)

      Hmdty_at35_Lin =     1.5549


%%%% (( SUPPLEMENTARY VISUALISATION - for assessing data trend & model fit )) %%%
%        -> Adding extracted T>=28 data range, for guided analysis

% Extract separate variables
Tmpr_cln_fullngth = AirQual.T(:) ;
Hmdty_cln_fullngth = AirQual.AH(:) ;

%  Reset k index for 28+deg datapoint loop iteration
k = 1;

% Iterate through the original data arrays
for i = 1:length(Tmpr_cln_fullngth)
    if Tmpr_cln_fullngth(i) >= 28
        Tmpr_28plus(k) = Tmpr_cln_fullngth(i);
        Hmdty_28plus(k) = Hmdty_cln_fullngth(i);
        k = k + 1;
    end
end

% Check filter loop result
size([Tmpr_28plus Hmdty_28plus])

%  >>> OUTPUT check of total # re-implanted datapoints
      >> size([Tmpr_28plus Hmdty_28plus])

      ans =    1    26

% VISUALISE >28degs datapoints, overlaid onto <28degs data
hold off
axis([5 35 0.5 2]);
hold on
plot(Tmpr_crop,Hmdty_crop,'kh');
plot(Tmpr_28plus,Hmdty_28plus,'hm')
plot(x,fn_Lin,'-c','LineWidth',2)


%    >>>>  OUTPUT FIG.3:
Superimposed ‘BestFit’ Regsn Line +
Scatterplot + deleted items >>>>
```



```
%%%  <<<  SECTION 5(B)  >>>  %%%

% Using the GUI 'Curve Fitting Tool'-
% Extract equations and parameters of key models for PREDICTION TESTING;
```

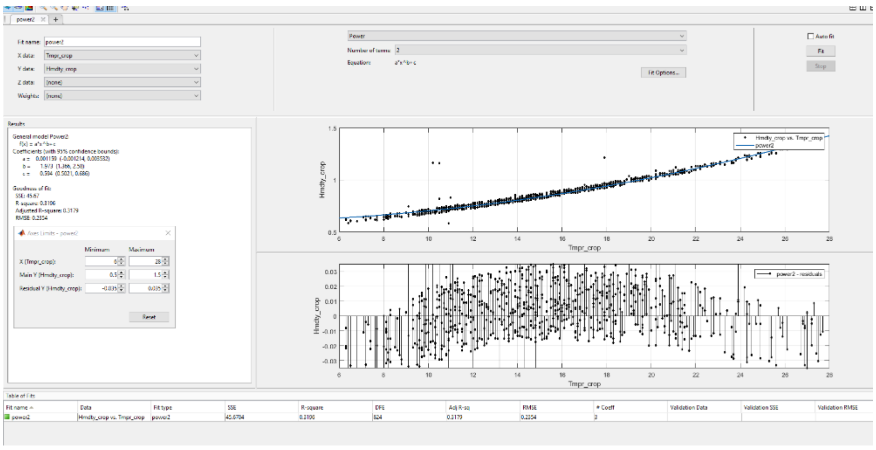


```
% General model of "GAUSSIAN 1-term":
%  f(x) =  a1*exp(-((x-b1)/c1)^2)

%  Model coefficients (with 95% confidence bounds):
   a1 =    2.995e+56; % (-8.829e+60, 8.829e+60)
   b1 =    6698e+56 ;     % (-1.507e+06, 1.52e+06)
   c1 =    585.7e+56 ;    % (-6.576e+04, 6.693e+04)

  % Generate function, test model
  fn_Gauss1= @(x) a1.*exp(-((x-b1)/c1).^2)

  %  >>> OUTPUT Gaussian equation
      >> fn_Gauss1= @(x) a1.*exp(-((x-b1)/c1).^2)

      fn_Gauss1 =  function_handle with value:

         @(x)a1.*exp(-((x-b1)/c1).^2)

  % Check Gauss-1 model output @Temp=35:
  fn_Gauss1_Tmpr35 = fn_Gauss1(Tmpr35);

  Hmdty_at35_Gauss1 = fn_Gauss1_Tmpr35
  %  >>> OUTPUT Gaussian variable @35:
      >> Hmdty_at35_Gauss1 = fn_Gauss1_Tmpr35

      Hmdty_at35_Gauss1 =     1.8689
```

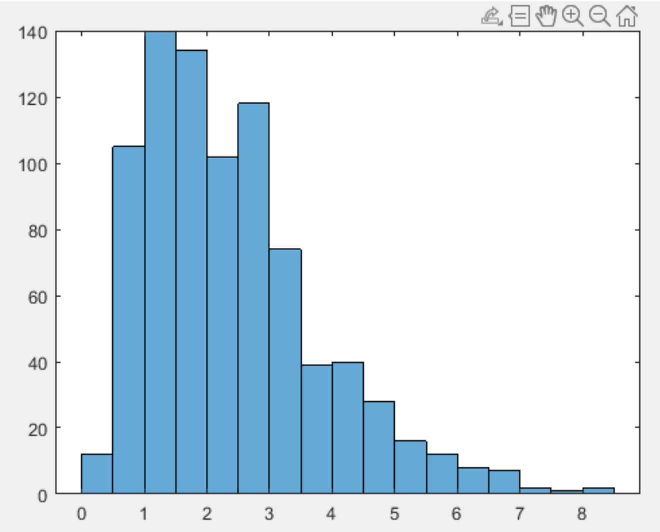


```
% General model for “EXPONENTIAL 1-term”:

%       f(x) = a*exp(b*x)
%       where x is normalized by mean 15.34 and std 4.549

%  Coefficients (with 95% confidence bounds):
   a2 =    0.4701;  %(0.439, 0.5011)
   b2 =    0.03896; % (0.03524, 0.04268)

  % Generate function, test model
  fn_Exp1 = @(x) a2.*exp(b2.*x)

  % Expntl-1 model output @Temp=35:
  fn_Exp1_Tmpr35 = fn_Exp1(Tmpr35);

  % Create distinctive variable:
  Hmdty_at35_Exp1 = fn_Exp1_Tmpr35    % <== 1.8382


  %  >>> OUTPUT Exponential variable @35:

       >> Hmdty_at35_Exp1 = fn_Exp1_Tmpr35

       Hmdty_at35_Exp1 =     1.8382
```

2-Term Power model below for comparison. Residual ‘curve’ demonstrates non-linearity.

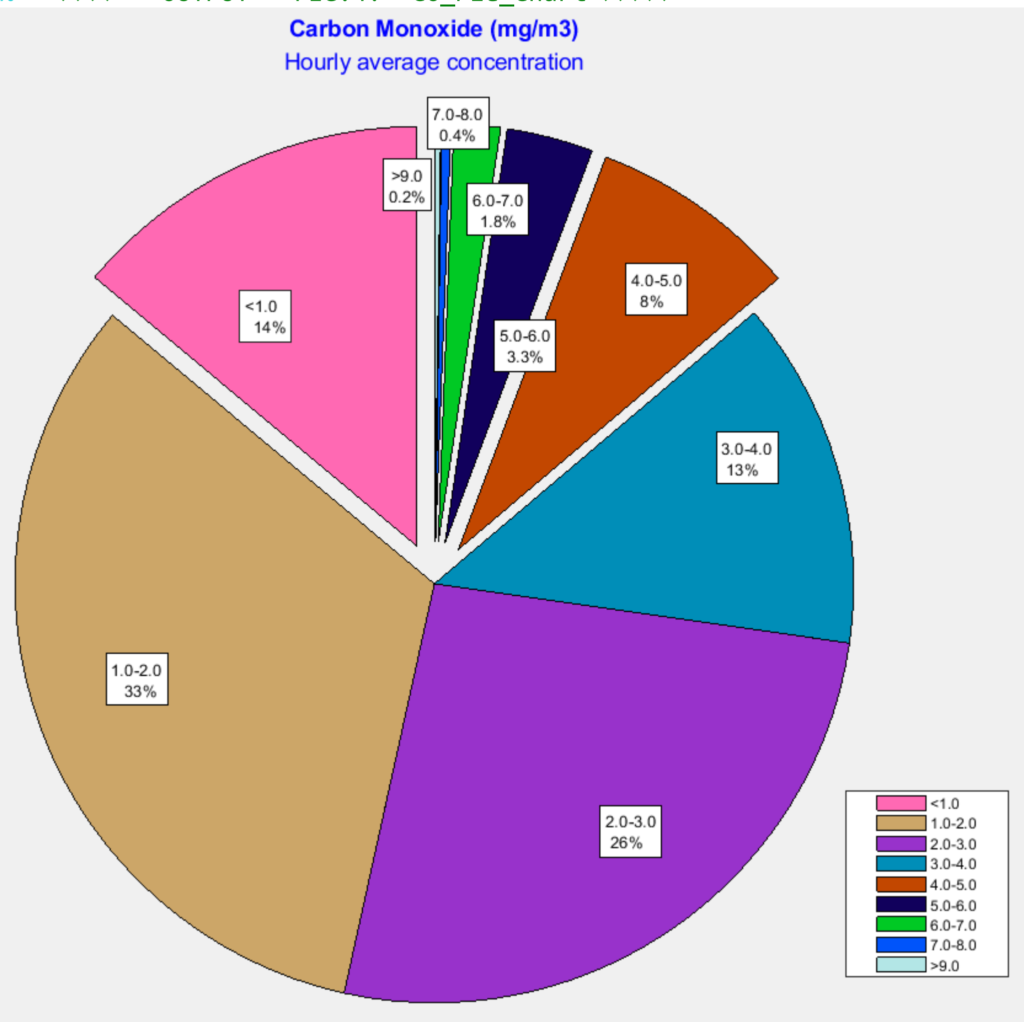


```
%%%  <<<  SECTION 6  >>>  %%%

% Pie Chart --- firstly, look at the original Data features to grab Variable name
AirQual.Properties.VariableNames{2}    % <== #2 = 'CO_GT_'

% Create SORTED CO array (ascending)
CO = sort(AirQual.CO_GT_(:));

% 'Eyeball' CO distribution via Histogram
hold off
histogram(CO)
%    >>>>  OUTPUT FIG.4:  raw CO HISTOGRAM >>>>
```

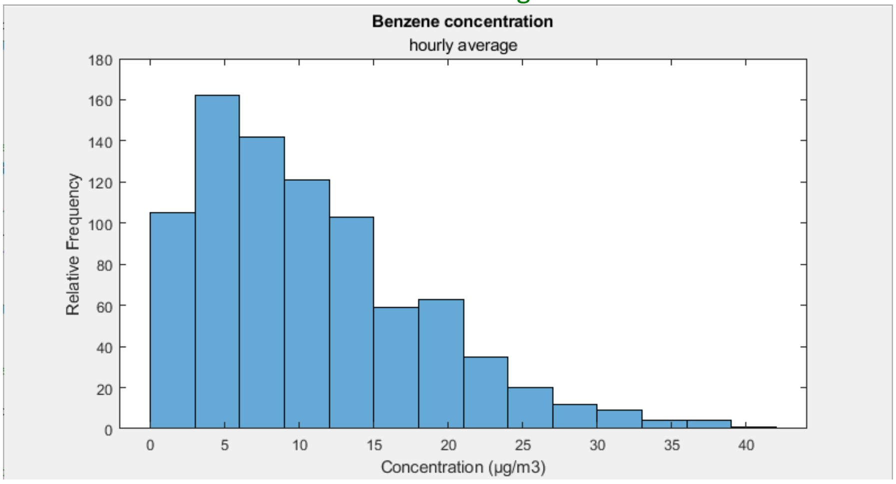



```
% Yummy, Pie time!

% Crude creation of Categorical bins
CO_bin1 = length(CO(1:117));   %0-1
CO_bin2 = length(CO(118:391)); %1-2
CO_bin3 = length(CO(392:611)); %2-3
CO_bin4 = length(CO(612:724)); %3-4
CO_bin5 = length(CO(725:792)); %4-5
CO_bin6 = length(CO(793:820)); %5-6
CO_bin7 = length(CO(821:835)); %6-7
CO_bin8 = length(CO(836:838)); %7-8
CO_bin9 = length(CO(839:840)); % >8

% Manage CO Pie bin data -->> Array for functional tallying
CO_bin_array = [CO_bin1 CO_bin2 CO_bin3 CO_bin4 CO_bin5 CO_bin6 CO_bin7 CO_bin8 CO_bin9]

% Calc & check all datapoints captured = original array length
CO_total = sum(CO_bin_array)     % <===  #840, affirmative.  Good.

% Calc CO percentage for Pie labelling
CO_prcntg = (CO_bin_array/CO_total)*100

%  >>> OUTPUT CO Percentage ("%") array:

>> CO_prcntg = (CO_bin_array/CO_total)*100

CO_prcntg =   13.9286   32.6190   26.1905   13.4524    8.0952    3.3333    1.7857    0.3571    0.2381


% Pie 'explode' select pieces: formatting for readability
explode = [1 0 0 0 1 1 1 1 1];
```

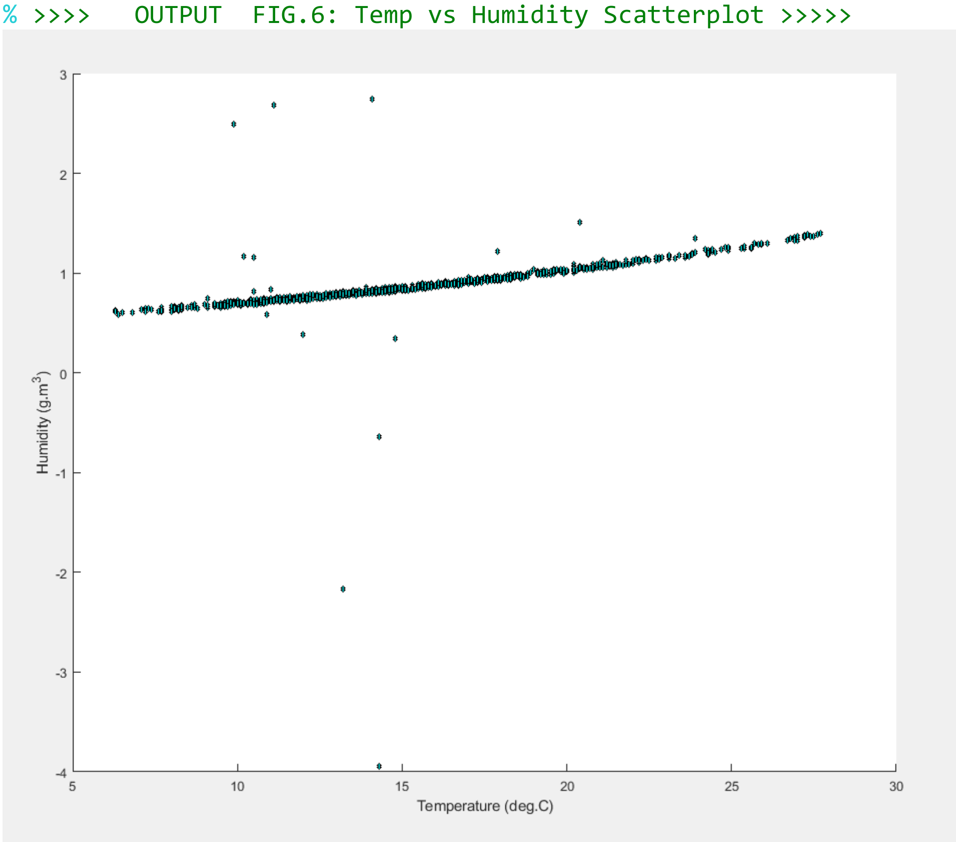
```
% Create Pie; define category bins
CO_Pie = pie(CO_bin_array,explode,{'<1.0','1.0-2.0','2.0-3.0','3.0-4.0','4.0-
5.0','5.0-6.0','6.0-7.0','7.0-8.0','>9.0'});
title('Carbon Monoxide (mg/m3)', 'Hourly average concentration','FontSize',
14,'Color','b');

% Define unique colour wheel
newColors = [...
    1,        0.41016, 0.70313; % Hot pink
    0.80,      0.65,    0.409;  % Spring green
    0.59766, 0.19531, 0.79688; % Dark orchid
    0,        0.556,   0.723;   % etc
    0.759,    0.278,   0;
    0.065,    0,       0.361;
    0,        0.788,   0.140;
    0,        0.329,   0.980;
    0.7,      0.9,     0.9 ];

% COLOURING - Override CO Pie default with new >>>

% Isolate the patch handles  [Modified algorithm, sampled from UI patch tool]
% h = CO_Pie();  h=pie() output is a vector of alternating patch and text handles.
patchHand = findobj(CO_Pie, 'Type', 'Patch');

% Colour override with vector UI input
% Set the color of all patches using the nx3 "newColors" matrix
   %  [disabled; function modified to substitute above Unique Colour wheel]
set(patchHand, {'FaceColor'}, mat2cell(newColors, ones(size(newColors,1),1), 3))

%   >>>>   OUTPUT   FIG.4:  CO_Pie_Chart >>>>>
```



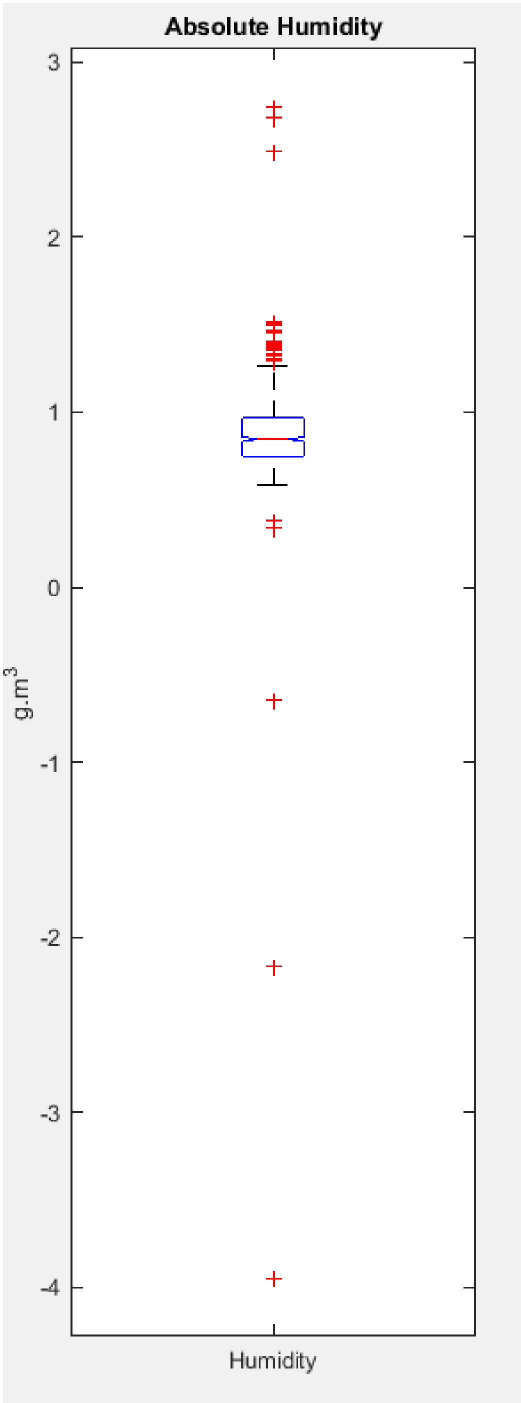
```
%  >>> HISTOGRAM >>>

% Look at the original Data features, grab Variable name
AirQual.Properties.VariableNames{5}    % <== #6: 'C6H6_GT_'

% Create Benzene HISTOGRAM from housed array
BenzHisto = histogram(AirQual. C6H6_GT_(:));
title('Benzene concentration', 'hourly average')
xlabel('Concentration (µg/m3)')
ylabel('Relative Frequency')

% >>>>  OUTPUT FIG.5: Benzene Histogram >>>>>
```

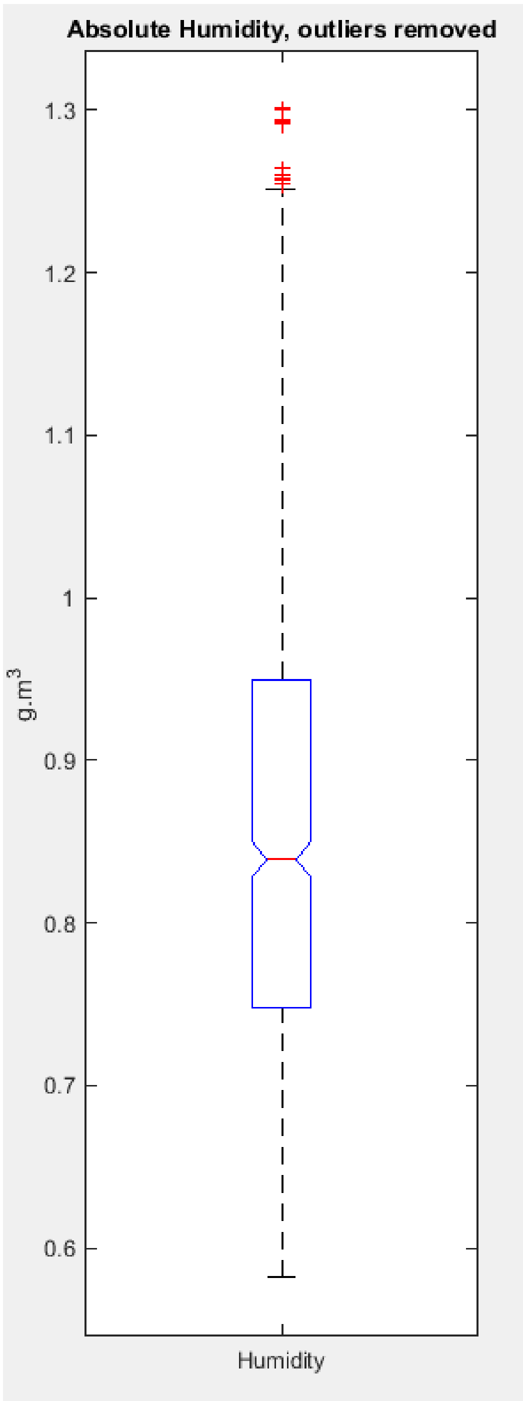



```
% SCATTERPLOT -- widened perspective; inc. outliers
hold off
sz=20; scatter(Tmpr_crop,Hmdty_crop,sz,'kh','MarkerFaceColor','c')
xlabel('Temperature (deg.C)')
ylabel('Humidity (g.m^3)')

% >>>>   OUTPUT  FIG.6: Temp vs Humidity Scatterplot >>>>>
```

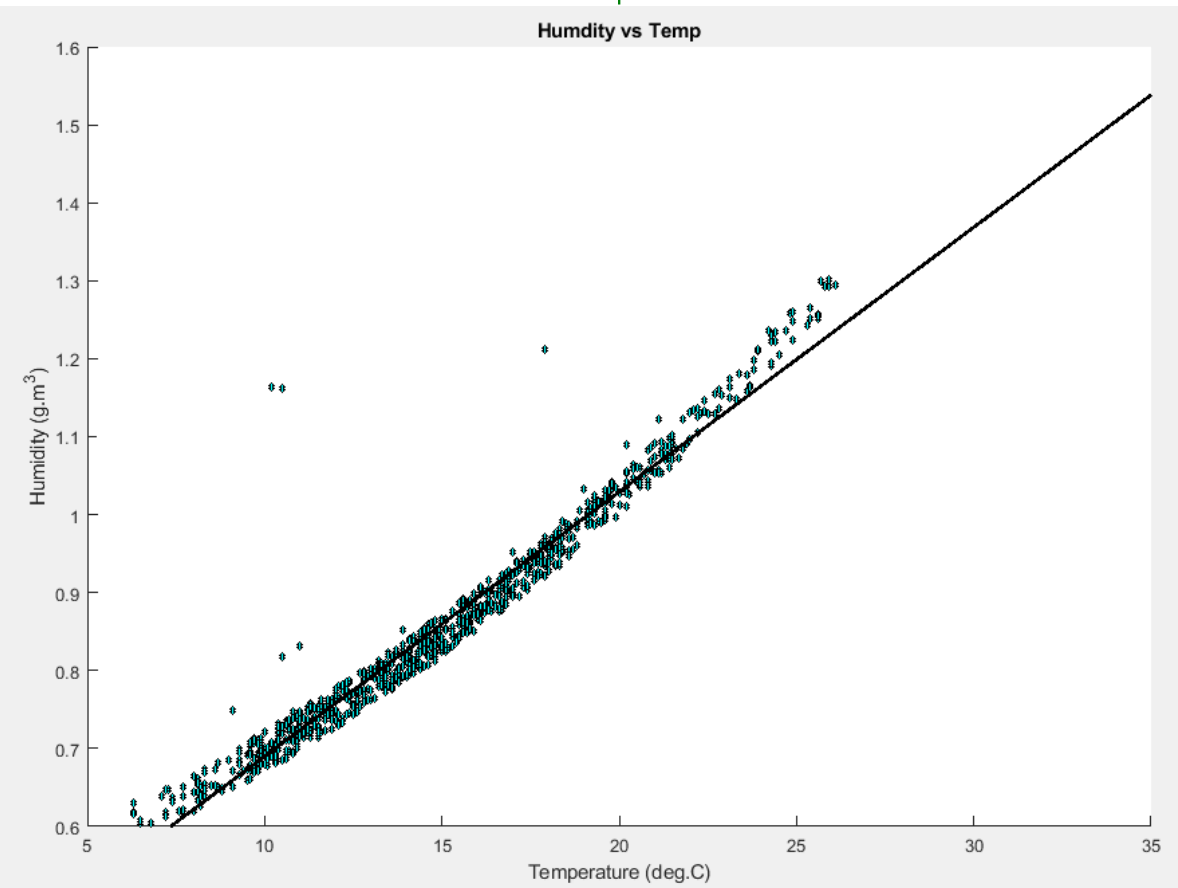


```
% BOXPLOT of Humidity
boxplot(TmprHmdty.Hmdty,'Notch','on','Labels',{'Humidity'});
title('Absolute Humidity'); ylabel('g.m^3')
%   >>>>>  OUTPUT  FIG.7:  Humidity BOXPLOT >>>>>

% Remove outliers (across whole array, for later use)
TmprHmdy_Outd = rmoutliers(TmprHmdty)

% Repeat boxplot post-removal
hold off
boxplot(TmprHmdy_Outd.Hmdty,'Notch','on','Labels',{'Humidity'})
title('Absolute Humidity, outliers removed'); ylabel('g.m^3')

%    >>>>>  OUTPUT   FIG.8:  'Inlier data' Hmdty BOXPLOT >>>>>
```





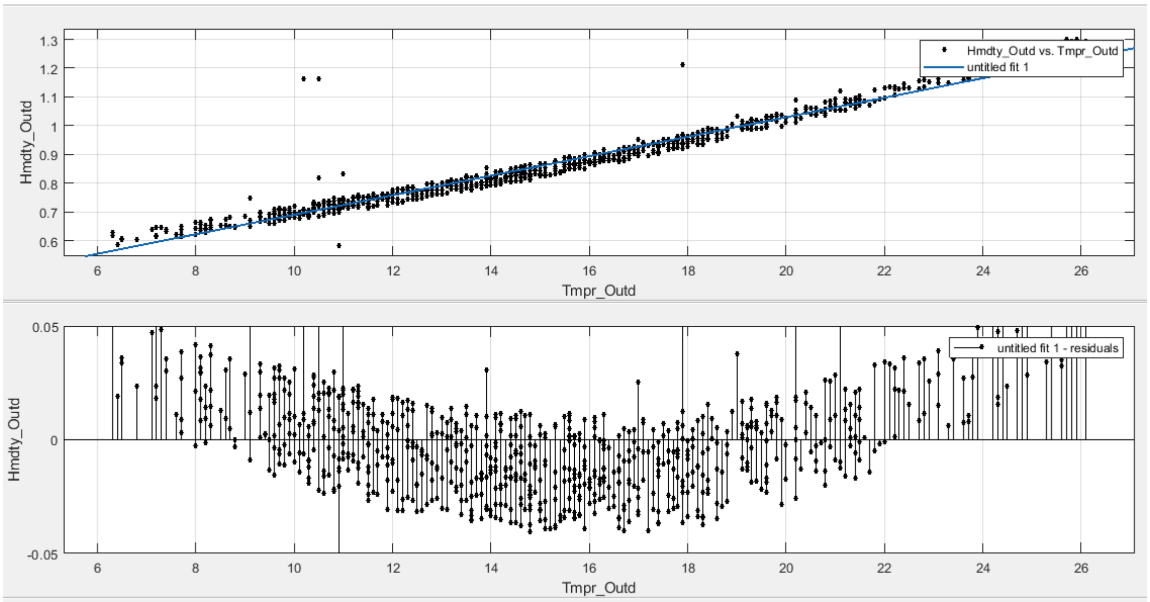
```
%%%  <<<  SECTION 7  >>>  %%%

% New 'Inlier' variables creation
Tmpr_Outd = TmprHmdy_Outd.Tmpr;
Hmdty_Outd = TmprHmdy_Outd.Hmdty;

% Manual scatterplot of constricted data
axis([5 35 0.6 1.6]);
hold on; sz =20;
scatter(Tmpr_Outd,Hmdty_Outd,sz,'kh','MarkerFaceColor','c');
xlabel('Temperature (deg.C)')
ylabel('Humidity (g.m^3)')
title('Humdity vs Temp')


% Manual 'Line of Best Fit' over variables>>

% Parameters generated from Polyfit function
p = polyfit(Tmpr_Outd,Hmdty_Outd,1)    % <-- ('1'=Linear)
% --> [1x2] array ouptut:  [0.0339,0.3507]
p1 = p(1);  p2 = p(2);   % <--- assign as variables

% Create 'Line of best fit' model function:
x = 0:0.1:40;
fn_Lin = p1.*x + p2;

% Plot Linear Regression Model over scatterplot
plot(x,fn_Lin,'-k','LineWidth',2)

%    >>>>>  OUTPUT  FIG.8:  Manual Scatterplot + Line BestFit  >>>>
```



Repeating the previous step utilising the '*Curve Fitting Toolkit*' GUI:

```
% >>> Using 'Curve-Fitting Tool' >> Run LINEAR REGRESSION over variables>>
```

```
Compared to results prior to Outlier removal, we see marked improvement on observed
fit and performance metrics, particularly R-sqrd correlation coefficients.
The linear model apparently fails the linearity test; 2nd Order polynomial better.
```



Results

Linear model Poly1:
  f(x) = p1*x + p2
Coefficients (with 95% confidence bounds):
  p1 = 0.03393 (0.03338, 0.03448)
  p2 = 0.3507 (0.3421, 0.3593)

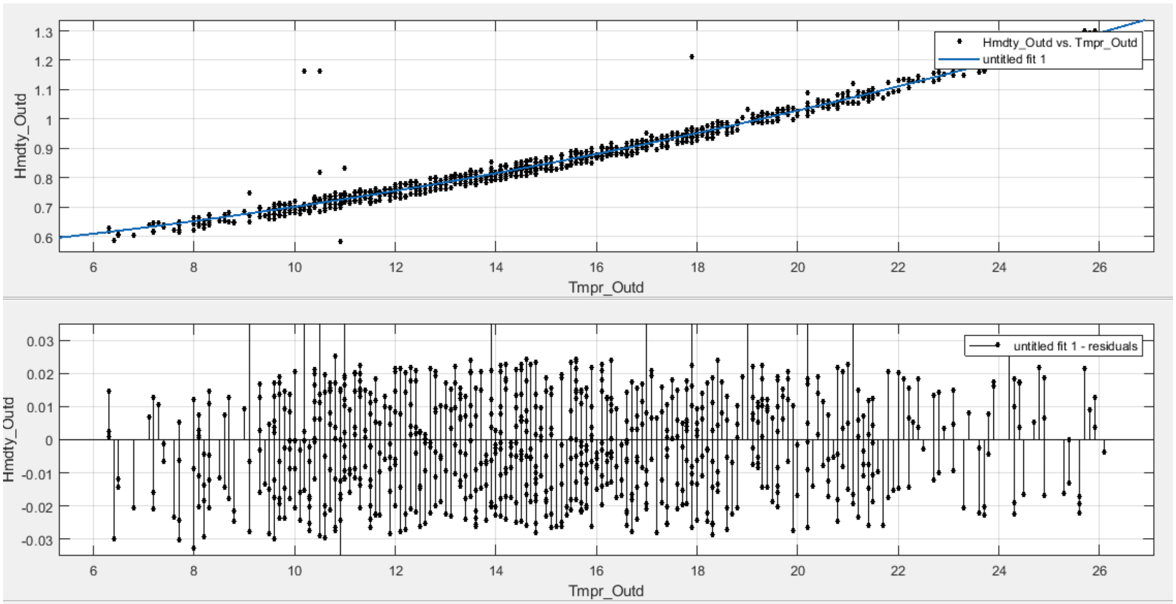
Goodness of fit:
  SSE: 0.8982
  R-square: 0.9485
  Adjusted R-square: 0.9485
  RMSE: 0.03357

Results

Linear model Poly2:
  f(x) = p1*x^2 + p2*x + p3
Coefficients (with 95% confidence bounds):
  p1 = 0.0007044 (0.0006095, 0.0007993)
  p2 = 0.01166 (0.00862, 0.0147)
  p3 = 0.5137 (0.4905, 0.537)

Goodness of fit:
  SSE: 0.709
  R-square: 0.9594
  Adjusted R-square: 0.9593
  RMSE: 0.02985

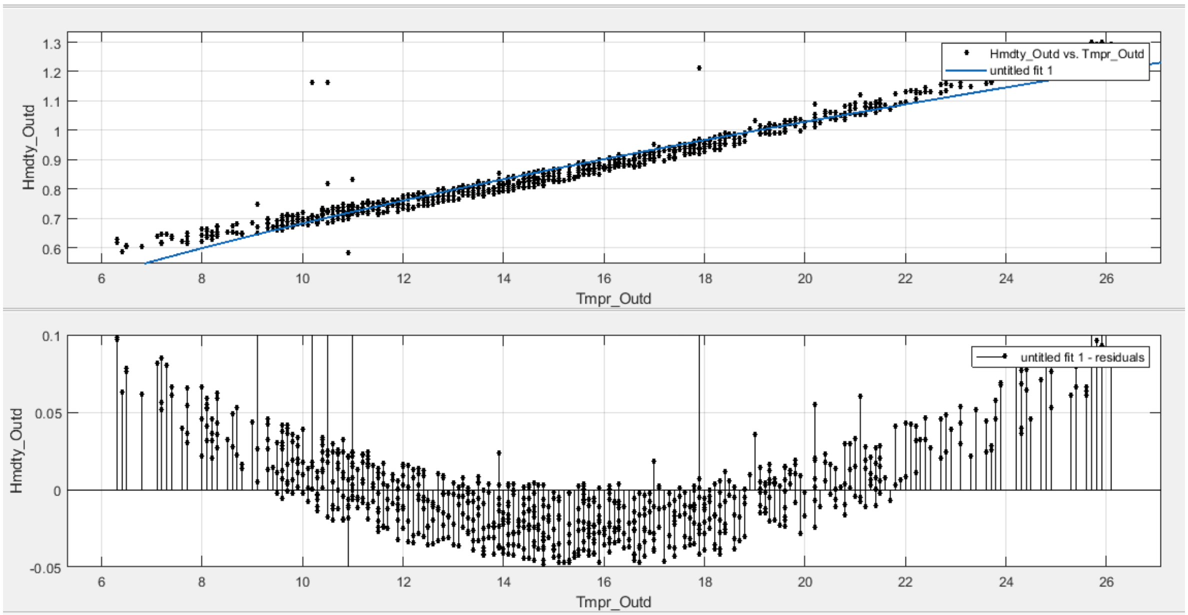
← Linear | 2nd Order →



```
%%%  <<<  SECTION 8  >>>  %%%

% >>>  Open 'Curve-Fitting Tool'  >>  Run NON-LINEAR MODELS over variables>>
```

'Power Model 1-Term': Residual plot demonstrates failure of linearity. The R-squared value is inferior to Linear Regression nonetheless.

'2-Term Power Model' performs notably better; and ostensibly passes Linearity test.



Results

General model Power1:
f(x) = a*x^b
Coefficients (with 95% confidence bounds):
a = 0.1747 (0.1691, 0.1802)
b = 0.5919 (0.5805, 0.6033)

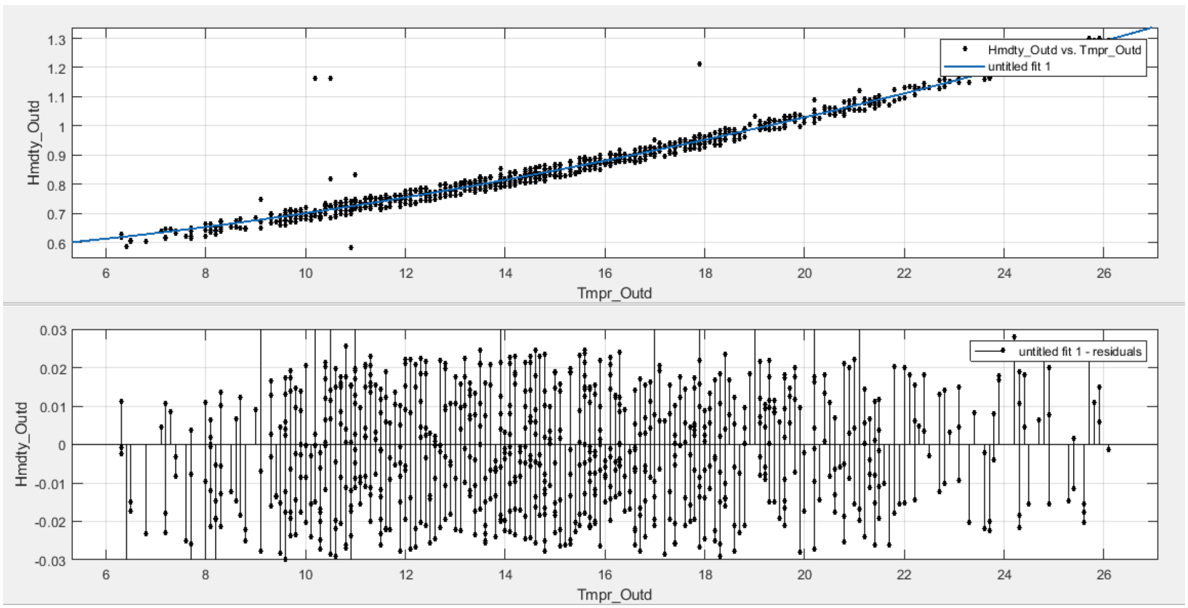
Goodness of fit:
SSE: 1.226
R-square: 0.9297
Adjusted R-square: 0.9296
RMSE: 0.03922

1-term ←Power→ 2-term

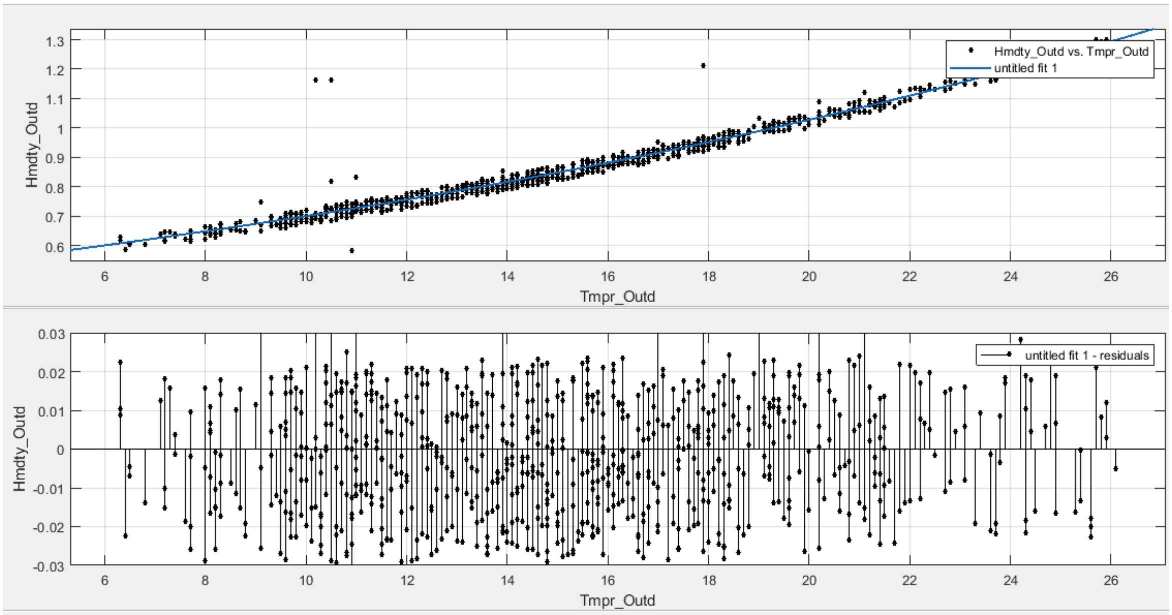
Results

General model Power2:
f(x) = a*x^b+c
Coefficients (with 95% confidence bounds):
a = 0.003429 (0.002382, 0.004476)
b = 1.651 (1.561, 1.741)
c = 0.547 (0.5295, 0.5645)

Goodness of fit:
SSE: 0.7104
R-square: 0.9593
Adjusted R-square: 0.9592
RMSE: 0.02987



'Exponential Model 1-term': Marginal difference in performance between 1 & 2 Term models. Equal best R-squared. Residuals demonstrate consistent fitting of the data.



Results

General model Exp1:
f(x) = a*exp(b*x)
Coefficients (with 95% confidence bounds):
a = 0.4768 (0.4724, 0.4812)
b = 0.0384 (0.03787, 0.03894)

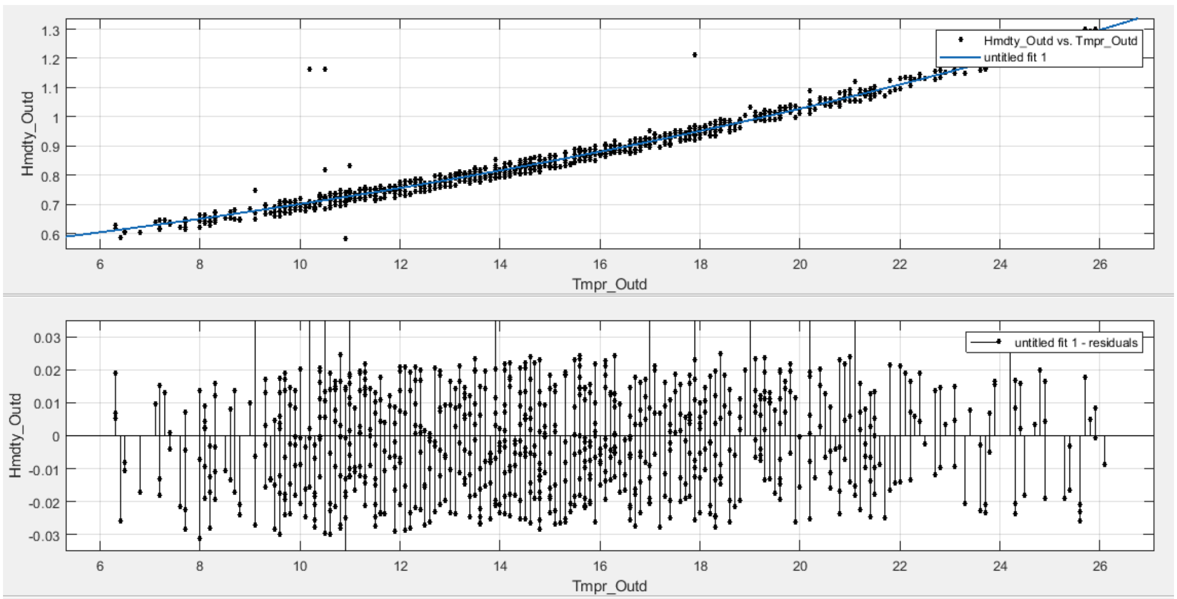
Goodness of fit:
SSE: 0.7095
R-square: 0.9593
Adjusted R-square: 0.9593
RMSE: 0.02984

1-term ←Exp→2-term

Results

General model Exp2:
f(x) = a*exp(b*x) + c*exp(d*x)
Coefficients (with 95% confidence bounds):
a = 0.4643 (0.2687, 0.6599)
b = 0.03939 (0.02765, 0.05114)
c = 0.02367 (-0.1006, 0.1479)
d = -0.06416 (-1.255, 1.127)

Goodness of fit:
SSE: 0.7087
R-square: 0.9594
Adjusted R-square: 0.9592
RMSE: 0.02986



```
%%%  <<<  SECTION 9  >>>  %%%

% Manual test of various Regression models' predictive power

% Use earlier stored 'Tmpr35' variable for model input

% Predictive power of LINEAR REGRESSION:
%  Input stored "p" variables as coefficients into 'Polyval' algorithm,
%  to predict Humidity at 35 degrees -- Linear model:
Hmdty_at35_Linr = polyval(p,Tmpr35)

%  >>> OUTPUT Humidity prediction when Temp = 35degrees:

      >> Hmdty_at35_Linr = polyval(p,Tmpr35)

      Hmdty_at35_Linr =     1.5383

% Prediction of NON-LINEAR REGRESSION models:
%   < Extract equations and parameters of key models from GUI 'Curve Fitting Tool >

% "GAUSSIAN 1-term" model:  f(x) =  a1*exp(-((x-b1)/c1)^2)

%  Model coefficients (with 95% confidence bounds):
   a1 =   2.995e+56; % (-8.829e+60, 8.829e+60)
   b1 =   6698e+56 ;    % (-1.507e+06, 1.52e+06)
   c1 =   585.7e+56 ;   % (-6.576e+04, 6.693e+04)

  % Generate function, test model
  fn_Gauss1= @(x) a1.*exp(-((x-b1)/c1).^2)

  %  >>> OUTPUT Gaussian equation
      >> fn_Gauss1= @(x) a1.*exp(-((x-b1)/c1).^2)

      fn_Gauss1 =  function_handle with value:

         @(x)a1.*exp(-((x-b1)/c1).^2)

  % Check Gauss-1 model output @Temp=35:
  Hmdty_at35_Gauss1 = fn_Gauss1(Tmpr35);

  %  >>> OUTPUT Gaussian variable @35:
      >> Hmdty_at35_Gauss1 = fn_Gauss1_Tmpr35

      Hmdty_at35_Gauss1 =     1.8759


% "EXPONENTIAL 1-term" model:   f(x) = a*exp(b*x)
%      where x is normalized by mean 15.34 and std 4.549

%  Coefficients (with 95% confidence bounds):
   a2 =   0.4768;  %(0.439, 0.5011)
   b2 =   0.03884; %(0.03524, 0.04268)

  % Generate function, test model
  fn_Exp1 = @(x) a2.*exp(b2.*x)

  % Expntl-1 model output @Temp=35:
  Hmdty_at35_Exp1 = fn_Exp1(Tmpr35);


  %  >>> OUTPUT Exponential variable @35:

      >> Hmdty_at35_Exp1 = fn_Exp1_Tmpr35

      Hmdty_at35_Exp1 =     1.8282
```

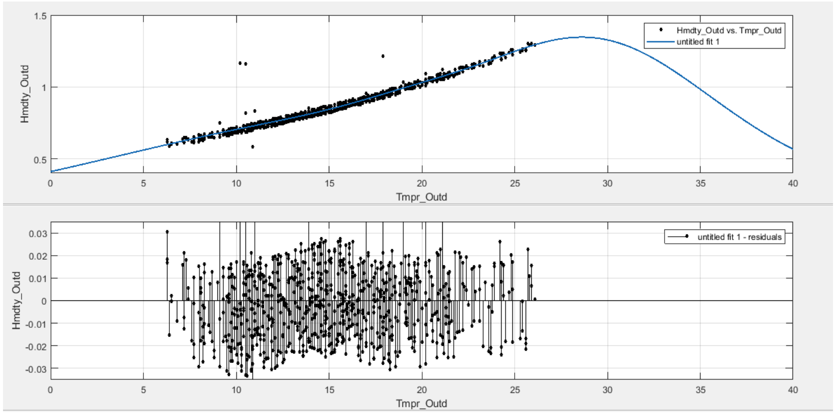
2.4 Evaluation

We see a limiting at 96% correlation coefficients on the best performing models.

Gaussian model performs well, reaching the best metrics obtained by the single-term Exponential. This is not surprising given that mathematically the Gaussian model is, at core, simply a 'messy' exponential equation.

However, higher order models lack appropriateness for selection due 'peaking': the 3-Term Gaussian depicted below peaks around the temperature of 28 degrees Celsius, and residuals appear non-linear.

The peaking predictions is not commensurate with recognised properties of atmospheric thermodynamics.



Other models, such as the Fourier series, similarly demonstrate incompatibility with real-world conditions by theoretically inferring negative humidity. Again, inconsistent with laws of nature.

The single-term Gaussian regression approximates the performance metrics of the best Exponential and Power models.

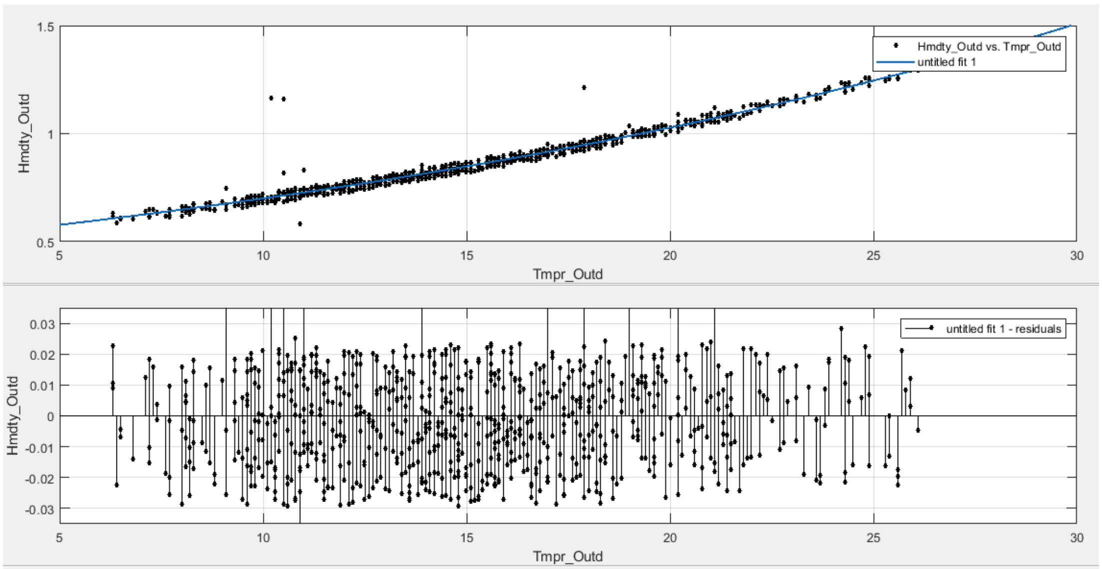
It is a more sophisticated model very similar in nature to its exponential equivalent, sharing near equal performance metrics.

Either of these models should be adopted for inferential data analysis around these Temperature & Humidity variables.

Results

General model Gauss1:
f(x) = a1*exp(-((x-b1)/c1)^2)
Coefficients (with 95% confidence bounds):
a1 = 1.18e+59 (-6.227e+62, 6.229e+62)
b1 = 7106 (-2.678e+05, 2.82e+05)
c1 = 607.7 (-1.118e+04, 1.239e+04)

Goodness of fit:
SSE: 0.7096
R-square: 0.9593
Adjusted R-square: 0.9592
RMSE: 0.02986



3.0 Conclusion

Investigating the data gathered behind the De Vito et al. paper (2008), this paper serves as a learning device for recognising how data science, through data exploration, analysis and predictive modelling, offers unlimited insights into all disciplines and professions.

Having first established descriptive statistical insights, prediction techniques were applied to key variables under investigation. The 'Data Science Roadmap' was employed to build structure, starting with data wrangling, and iterating through the process, culminating in predictive regression techniques.

Various models were comparatively evaluated for their predictive power. A convergence of key model performance metrics became apparent, while confidence can be drawn from **cross-validation of differing regression techniques.**

Outlier removal data manipulation are demonstratively effective for bending the data. Notably, herein the coefficient of correlation **(R^2) increased by 60% after removing less than 2% of the datapoints.**

These results offer a warning to data practitioners of the **risks exposed through careless data handling and manipulation,** in particular when removing outliers and implementing other data cleaning techniques.

By proxy, investigation of the data has brought deeper insights to the studies from which it originates. It has highlighted that, like all research, there are limits to the insights that can be drawn. Extrapolating prediction from trend risks overreaching epistemological boundaries.

It should not be assumed that exogenous forces can be ignored. Namely, dynamics of localised traffic and industrial activity surrounding these sensor modules, as well as variation in vehicle efficiencies for example, and myriad other pollutant factors, all contribute to significant 'outside shocks' to forecast trends of chemical composition and related thermodynamics.

To dampen the effects of these exogenous factors, a far longer accumulation of data should be considered for analysis. Further, a larger geographical matrix of sensor locations would extend the range of reliable insights drawn from the data collection.

Overall, the results and methodological process practised in this paper have served to reaffirm the value of data skills, integrity and analytic capacity for resolving real-world challenges and extending the scope of scientific enquiry.

4.0 Appendix

4.1 Data

The data can be downloaded from this link. This data set has been slightly modified from the one by De Vito et.al (2008), which is stored in the UCI repository (Dua & Graff, 2019).

The dataset contains 9357 instances of hourly averaged responses from an array of 5 metal oxide chemical sensors embedded in an Air Quality Chemical Multisensor Device. The device was located on the field in a significantly polluted area, at road level,within an Italian city. Data were recorded from March 2004 to February 2005 (one year) representing the longest freely available recordings of on field deployed air quality chemical sensor devices responses. Ground Truth hourly averaged concentrations for CO, Non Metanic Hydrocarbons, Benzene, Total Nitrogen Oxides (NOx) and Nitrogen Dioxide (NO2) and were provided by a co-located reference certified analyzer. Evidences of cross-sensitivities as well as both concept and sensor drifts are present as described in De Vito et al., Sens. And Act. B, Vol. 129,2,2008 (citation required) eventually affecting sensors concentration estimation capabilities. Missing values are tagged with -200 value. (S. De Vito et al, 2008)

Information about attributes (fields/columns) tabled below.

Column number	Description
1	Date (DD/MM/YYYY)
2	Time (HH:MM:SS)
3	True hourly average concentration of CO in mg/m^3 (reference analyzer)
4	Tin oxide hourly average sensor response (nominally CO targeted)
5	True hourly averaged overall Non-Methane Hydrocarbons concentration in $\mu g/m^3$ (reference analyzer)
6	True hourly averaged Benzene concentration in $\mu g/m^3$ (reference analyzer)
7	Titania hourly averaged sensor response (nominally NMHC targeted)
8	True hourly averaged NO_x concentration in ppb (reference analyzer)
9	Tungsten oxide hourly averaged sensor response (nominally NO_x targeted)
10	True hourly averaged NO_2 concentration in $\mu g/m^3$ (reference analyzer)
11	Tungsten oxide hourly averaged sensor response (nominally NO_2 targeted)
12	Indium oxide hourly averaged sensor response (nominally O_3 targeted)
13	Temperature in degrees Celsius
14	Relative humidity (%)
15	Absolute humidity

4.2 References

Daniel, J. 2022, GetGoogleSpreadsheet, MATLAB Central File Exchange. Retrieved May 10, 2022. https://www.mathworks.com/matlabcentral/fileexchange/39915-getgooglespreadsheet

Dua, D. and Graff, C. 2019, UCI Machine Learning Repository. School of Information and Computer Science, University of California, Irvine, CA.

Misc., 2022, MathWork Help Centre, 'Imread: Read image from graphics file' - MATLAB imread, MathWorks Australia, https://au.mathworks.com/help/matlab/ref/imread.html

Monash College (Monash University), 2022, "Introduction to Machine Learning - 2022 JAN", Learning Management System, course content.

S. De Vito, E. Massera, M. Piga, L. Martinotto, G. Di Francia, 2008, 'On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario', Sensors and Actuators B: Chemical, Volume 129, Issue 2, 22 February 2008, Pages 750-757, ISSN 0925-4005.