IC 272: Lab8: Prediction using Linear and Polynomial Regression

You are given with data file "`AirQuality.csv`" that contains the responses of a gas multisensor device deployed on the field in an Italian city. Hourly responses averages are recorded along with gas concentrations references from a certified analyzer. The dataset contains 9358 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. More details are given in [1].
The dataset contains 13 attributes, in which 11 are input variables and one is output variable i.e. concentration of CO.

Attribute Information:
Input variable:

1. PT08.S1(CO): PT08.S1 (tin oxide) hourly averaged sensor response (nominally CO targeted)
2. NMHC: True hourly averaged overall Non Metanic HydroCarbons concentration in microg/m^3 (reference analyzer)
3. C6H6(GT): True hourly averaged Benzene concentration in microg/m^3 (reference analyzer)
4. PT08.S2(NMHC): PT08.S2 (titania) hourly averaged sensor response (nominally NMHC targeted)
5. NOx: True hourly averaged NOx concentration in ppb (reference analyzer)
6. PT08.S3(NOx): PT08.S3 (tungsten oxide) hourly averaged sensor response (nominally NOx targeted)
7. NO2(GT): True hourly averaged NO2 concentration in microg/m^3 (reference analyzer)
8. PT08.S4(NO2): PT08.S4 (tungsten oxide) hourly averaged sensor response (nominally NO2 targeted)
9. PT08.S5(O3): PT08.S5 (indium oxide) hourly averaged sensor response (nominally O3 targeted)
10. T: Temperature in Â°C
11. RH: Relative Humidity (%)
12. AH: Absolute Humidity

Output variable:

13. CO: True hourly averaged concentration CO in mg/m^3 (reference analyzer)

1. Missing value in each attribute is represented as -200. Replace -200 in each attribute by its median.
2. Split the data from winequality-red.csv into **train data** and **test data**. Train data contain **70%** of tuples and test data contain remaining **30%** of tuples. Save the train data as `AirQuality-train.csv` and save the test data as `AirQuality-test.csv`

3. Build the simple linear regression (**straight-line regression**) model to predict CO given PT08.S1(CO).
    a. Plot the best fit line on the training data where x-axis is PT08.S1(CO) value and y-axis is CO.
    b. Find the **prediction accuracy** on the training data using *root mean squared error*.
    c. Find the **prediction accuracy** on the test data using *root mean squared error*.
    d. Plot the scatter plot of *actual temperature* vs *predicted temperature* on the test data. Comment on the scatter plot.
4. Build the simple nonlinear regression model using **polynomial curve fitting** to predict temperature given pressure.
    a. Plot the best fit curve on the training data where x-axis is PT08.S1(CO) value and y-axis is PT08.S1(CO).
    b. Find the **prediction accuracy** on the training data for the different values of degree of polynomial (p = 2, 3, 4, 5) using *root mean squared error (RMSE)*. Plot the bar graph of *RMSE* (y-axis) vs different values of degree of polynomial (x-axis).
    c. Find the **prediction accuracy** on the test data for the different values of degree of polynomial (p = 2, 3, 4, 5) using *root mean squared error (RMSE)*. Plot the bar graph of *RMSE* (y-axis) vs different values of degree of polynomial (x-axis).
    d. Plot the scatter plot of *actual CO* vs *predicted CO* on the test data for the best degree of polynomial (p). Comment on the scatter plot and compare with that of in 2(d).
5. Compute the Pearson correlation coefficient for every attribute with the attribute *CO* (dependent variable) on the training data. Select two attributes that are highly correlated (either positively or negatively correlated) with attribute *CO*.
    a. Build the multiple linear regression model considering only the selected two attributes to predict CO.
        i. Plot the best fit plane on the training data where x and y-axis are the two selected attributes z-axis is CO.
        ii. Find the **prediction accuracy** on the training data using *root mean squared error*.
        iii. Find the **prediction accuracy** on the test data using *root mean squared error*.
        iv. Plot the scatter plot of *actual CO* vs *predicted CO* on the test data. Comment on the scatter plot.
    b. Build the multivariate polynomial regression model considering only the selected two attributes to predict temperature.
        i. Plot the best fit surface on the training data where x and y-axis are the two selected attributes z-axis is CO.
        ii. Find the **prediction accuracy** on the training data for the different values of degree of polynomial (p = 2, 3, 4, 5) using *root mean squared error (RMSE)*. Plot the bar graph of *RMSE* (y-axis) vs different values of degree of polynomial (x-axis).
        iii. Find the **prediction accuracy** on the test data for the different values of degree of polynomial (p = 2, 3, 4, 5) using *root mean squared error (RMSE)*. Plot the bar graph of *RMSE* (y-axis) vs different values of degree of polynomial (x-axis).
        iv. Plot the scatter plot of *actual CO* vs *predicted CO* on the test data. Comment on the scatter plot.
6. Compare each of the regression models (all the cases from questions 2-6) based on *RMSE*.

Note:

A. **Simple and Multiple Linear Regression**:
Import the `LinearRegression` from `sklearn.linear_model`
A code snippet for prediction using linear regression:

```
regressor = LinearRegression()
regressor.fit(x, y)
```

`x` is set of univariate or multivariate training data used for building simple of multiple linear regression. `y` is corresponding dependent variable.

```
y_pred = regressor.predict(x)
```

B. **Polynomial Curve Fitting and Polynomial Regression**:
Import the `PolynomialFeatures` from `sklearn.preprocessing`
A code snippet for prediction using linear regression:

```
polynomial_features= PolynomialFeatures(degree=p)
x_poly = polynomial_features.fit_transform(x)
```

`x` is set of univariate or multivariate training data used for building simple of multiple polynomial regression.

```
regressor = LinearRegression()
regressor.fit(x_poly, y)
```

`x_poly` is set of polynomial expansions (monomials of polynomial up to degree `p`) training data used for building simple of multiple linear regression. `y` is corresponding dependent variable.

```
y_pred = regressor.predict(x)
```

Reference:

[1] S. De Vito, E. Massera, M. Piga, L. Martinotto, G. Di Francia, 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