Indian Institute of Technology Dharwad
CS214: Artificial Intelligence Laboratory
Lab 9 – Regression with Multi-Input Variables

Dataset Description

You are given the dataset `AirQuality_data.csv`, which contains various air-pollution sensor readings. Your task is to predict the **CO concentration (mg/m^3)**.

- **Input Variables (12 total):**
  - PT08.S1(CO): Tin oxide sensor response (nominally CO targeted)
  - NMHC: Non-Methanic HydroCarbons concentration (microg/m^3)
  - C6H6(GT): Benzene concentration (microg/m^3)
  - PT08.S2(NMHC): Titania sensor response (nominally NMHC targeted)
  - NOx: NOx concentration (ppb)
  - PT08.S3(NOx): Tungsten oxide sensor response (nominally NOx targeted)
  - NO2(GT): NO2 concentration (microg/m^3)
  - PT08.S4(NO2): Tungsten oxide sensor response (nominally NO2 targeted)
  - PT08.S5(O3): Indium oxide sensor response (nominally O3 targeted)
  - T: Temperature (°C)
  - RH: Relative Humidity
  - AH: Absolute Humidity
- **Output Variable:** CO concentration (mg/m^3).

Data Preprocessing

1. Split the data into **60% training**, **20% validation**, and **20% test** data.
2. Compute the **Pearson correlation** coefficient for every attribute with the attribute **CO concentration (mg/m^3)** (dependent variable) on the training data. Select two attributes that are highly correlated with **CO concentration (mg/m^3)**.

Problem Statement 1 (Top-2 Features)

Using the top 2 attributes(the ones that have the highest absolute correlation with **CO concentration** on the training set), do the following:

1. Multiple Linear Regression (Top 2 Features)

1. Build a multiple linear regression model **using only the two selected attributes** to predict CO concentration (mg/m^3).
2. **Plot:** Best-fit plane considering the training data and predicted data
   - x-axis: Feature 1
   - y-axis: Feature 2
   - z-axis: CO concentration
3. Compute **RMSE** on the training, validation, and test data.
4. **Scatter Plot:** Actual vs. Predicted CO concentration on the test data.

2. Polynomial Regression (Top 2 Features)

1. Build a multivariate polynomial regression model **only with those two attributes**. Use degrees $p = 2, 3, 4, 5$.
2. Compute **RMSE** on the training and validation datasets.
3. **Plot:** Best-fit surface considering the training data and predicted data.
   - x-axis: Feature 1
   - y-axis: Feature 2
   - z-axis: CO concentration

   for each polynomial degree.
4. Select the polynomial degree with the **lowest validation RMSE** and evaluate this model on the test data.
5. Compute **RMSE** for the test data.
6. **Bar Graph:** RMSE of validation data vs. Degree of Polynomial.
7. **Scatter Plot:** Actual vs. Predicted CO concentration on the test data for the best polynomial degree.

3. Neural Network (Top 2 Features)

1. Use **Top-2** input features as the independent variables to predict the CO concentration.
2. Build a simple neural network using PyTorch:
   - **Input layer:** 2 neurons (one for each selected feature).
   - **Hidden layer:** 1 layer with Sigmoid activation function. Experiment with different numbers of neurons (e.g. 8, 16, 32, etc.).

- **Output layer:** 1 neuron (regression output).
- **Loss function:** Mean Squared Error (MSE).
- **Optimizer:** Stochastic Gradient Descent (SGD).

3. Train the model with varying hidden neurons.
4. Compute **RMSE** on training and validation datasets.
5. **Plot:** Best-fit surface considering the training data and predicted data
   - x-axis: Feature 1
   - y-axis: Feature 2
   - z-axis: CO concentration

   for each architecture.
6. Pick the network configuration with the **lowest validation RMSE** and test it on the test dataset.
7. Compute **RMSE** on the test dataset.
8. **Plot:** Training loss vs. epochs for the best network.
9. **Scatter Plot:** Actual vs. Predicted CO concentration on the test data for the best network.

Problem Statement 2 (All 12 Features)

Now, instead of using only the top-2 features, repeat each of the three models below using all 12 features(the entire input space).

1. Multiple Linear Regression (All 12 Features)

1. Build a multiple linear regression model **using all 12 attributes** to predict CO concentration.
2. Compute **RMSE** on training, validation, and test data.
3. **Scatter Plot:** Actual vs. Predicted CO concentration on the test data.

2. Polynomial Regression (All 12 Features)

1. Build a multivariate polynomial regression model **using all 12 features** for degrees $p = 2, 3, 4, 5$.
2. Compute **RMSE** on the training, validation data set for each polynomial degree.
3. Select the polynomial degree with the **lowest validation RMSE** and evaluate this model on the test data.
4. Compute **RMSE** for the test data.

5. **Bar Graph:** RMSE of validation data vs. Degree of Polynomial.

6. **Scatter Plot:** Actual vs. Predicted CO concentration on the test data for the best polynomial degree.

3. Neural Network (All 12 Features)

1. Use **all 12** input features as the independent variables to predict CO concentration.

2. Build a simple neural network using PyTorch:

   - **Input layer:** 12 neurons (one per feature).
   - **Hidden layer:** 1 layer with Sigmoid activation function. Experiment with different numbers of neurons (e.g. 8, 16, 32, etc.).
   - **Output layer:** 1 neuron (regression output).
   - **Loss function:** Mean Squared Error (MSE).
   - **Optimizer:** Stochastic Gradient Descent (SGD) or Adam.

3. Train the model with varying hidden neurons and compare the RMSE on training and validation.

4. Pick the network configuration with the **lowest validation RMSE** and evaluate it on the test data.

5. Compute **RMSE** on the test dataset.

6. **Scatter Plot:** Actual vs. Predicted CO concentration on the test data for the best network.

Note

A. Multiple Linear Regression

Use the scikit-learn `LinearRegression` model:

```
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()
regressor.fit(X_train, y_train)

y_pred = regressor.predict(X_test)
```

When using **top-2 features**, `X_train` and `X_test` each have only those two columns. When using **all 12 features**, they have all columns except the target.

B. Polynomial Regression

For top-2 or all-12 features, you can expand your input with `PolynomialFeatures`:

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

poly = PolynomialFeatures(degree=d)
X_train_poly = poly.fit_transform(X_train)
model = LinearRegression()
model.fit(X_train_poly, y_train)

X_val_poly = poly.transform(X_val)
y_val_pred = model.predict(X_val_poly)
```

Remember to call the same `poly.transform(...)` for validation and test splits.

C. Neural Network Implementation (PyTorch Example)

Adjust the input/output dimensions to match your features:

```
import torch
import torch.nn as nn
import torch.optim as optim

class NeuralNet(nn.Module):
    def __init__(self, input_size=12, hidden_dim=32):
        super().__init__()
        self.network = nn.Sequential(
            nn.Linear(input_size, hidden_dim),
            nn.Sigmoid(),
            nn.Linear(hidden_dim, 1)
        )
    def forward(self, x):
        return self.network(x)

model = NeuralNet(input_size=12, hidden_dim=32)
criterion = nn.MSELoss()
optimizer = optim.SGD(model.parameters(), lr=0.01)

# Typical training loop ...
```

For top-2 features, use `input_size=2`; for all-12, use `input_size=12`.

References

[1] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford, UK: Oxford University Press, 1995.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*, New York, NY: Springer, 2006.

[3] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, *Extreme Learning Machine for Regression and Multiclass Classification*, 2012.