วิชา ปัญญาประดิษฐ์ (Artificial Intelligence)
CPE5007

พรภวิษย์ บุญศรีเมือง

Matlab: Generate clean signal (sine wave)

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate clean signal (sine wave)
t = linspace(0,1,1000)';
clean_signal = sin(2*pi*5*t);

% Add Gaussian noise
noisy_signal = clean_signal + 0.3*randn(size(t));

% Create simple autoencoder with 1 hidden layer
autoenc = trainAutoencoder(noisy_signal', 10, ...
    'L2WeightRegularization', 0.001, ...
    'SparsityRegularization', 4, ...
    'SparsityProportion', 0.05, ...
    'DecoderTransferFunction','purelin', ...
    'ShowProgressWindow', false);

% Reconstruct signal
reconstructed_signal = predict(autoenc, noisy_signal');

% Plot
figure;
subplot(3,1,1); plot(t, clean_signal); title('Clean Signal');
subplot(3,1,2); plot(t, noisy_signal); title('Noisy Signal');
subplot(3,1,3); plot(t, reconstructed_signal); title('Denoised Signal by Autoencoder');
```

Matlab: Temp_Regression

LatencyCloud_005.m | Temp_Regression_001.m | KNN_002.m | Linear_Regression_001.m | Random_Forests_002.m | IDAR_ML_001.m

```
%close;
% กำหนดค่าเริ่มต้นเพื่อให้ได้ผลลัพธ์ที่เหมือนเดิมทุกครั้งที่รัน
rng(2);

% --- ข้อมูลสมมติสำหรับอุณหภูมิ ---
% สมมติว่ามีข้อมูลอุณหภูมิที่ระดับความสูงต่างกัน 50 จุด
elevation = (100:20:1080)'; % ระดับความสูง (เมตร) (คุณลักษณะ/ตัวแปรต้น - X)
% สร้างอุณหภูมิที่ลดลงตามระดับความสูง และมี 'noise'
temperature = 30 - 0.015 * elevation + 3 * randn(length(elevation), 1); % อุณหภูมิ (องศาเซลเซียส) (ตัวแปรตาม - Y)

% --- พล็อตข้อมูลอุณหภูมิแบบธรรมดา ---
figure;
plot(elevation, temperature, 'g^', 'MarkerSize', 8, 'MarkerFaceColor', 'g');
title('ข้อมูลอุณหภูมิตามระดับความสูง');
xlabel('ระดับความสูง (เมตร)');
ylabel('อุณหภูมิ (องศาเซลเซียส)');
grid on;
legend('อุณหภูมิจริง');

% --- สร้างและฝึกโมเดล Linear Regression ---
temperature_model = fitlm(elevation, temperature);

% --- ทำการพยากรณ์อุณหภูมิที่ระดับความสูงใหม่ ---
new_elevation = [500; 1500; 2000]; % ระดับความสูงที่ต้องการพยากรณ์
predicted_temperature = predict(temperature_model, new_elevation);

% --- พล็อตผลลัพธ์การพยากรณ์ ---
figure;
plot(elevation, temperature, 'g^', 'MarkerSize', 8, 'MarkerFaceColor', 'g'); % พล็อตข้อมูลจริง
hold on;
plot(temperature_model); % พล็อตเส้น Linear Regression ที่ฟิตกับข้อมูลจริง
plot(new_elevation, predicted_temperature, 'ro', 'MarkerSize', 10, 'MarkerFaceColor', 'r'); % พล็อตจุดพยากรณ์
title('การพยากรณ์อุณหภูมิด้วย Linear Regression');
xlabel('ระดับความสูง (เมตร)');
ylabel('อุณหภูมิ (องศาเซลเซียส)');
legend('อุณหภูมิจริง', 'เส้น Linear Regression (ข้อมูลที่ฝึก)', 'อุณหภูมิที่พยากรณ์');
grid on;
hold off;

% แสดงค่าสัมประสิทธิ์และจุดตัดแกน Y ของโมเดลอุณหภูมิ
fprintf('\n--- โมเดลพยากรณ์อุณหภูมิ ---\n');
fprintf('ค่าสัมประสิทธิ์ (Coefficient/Slope): %.3f\n', temperature_model.Coefficients.Estimate(2));
fprintf('จุดตัดแกน Y (Intercept): %.2f\n', temperature_model.Coefficients.Estimate(1));
```

MatLab: k-Nearst Neighbors (KNN)

KNN_001.m | KNN_002.m | Random_Forests_001.m | Random_Forests_002.m | SVM_001.m | SVM_002.m | TrainASupportVectorMachineClassifierExample.mlx | TrainAKNNClassifierExample.mlx

```
clear all;
clc;
clf;
close;

%==========================================================================

% --- MATLAB: k-Nearest Neighbors (KNN) with Visualization ---

% 1. สร้างข้อมูลจำลอง (Simulate Training Data)
%    Columns: [Signal_Strength, Data_Rate]
%    Labels: 1 = 'Normal', 2 = 'Anomalous'
rng(1); % for reproducibility
training_data = [
    -50, 100; -55, 95; -52, 105; -58, 98;  % Normal signals
    -80, 20; -85, 25; -82, 15; -90, 18    % Anomalous signals
];
training_labels = [1; 1; 1; 1; 2; 2; 2; 2];

% 2. ฝึกโมเดล KNN (Train the KNN Model)
k = 3; % จำนวนเพื่อนบ้านที่ใช้ในการตัดสินใจ
knn_model = fitcknn(training_data, training_labels, 'NumNeighbors', k);

% 3. ทำนายข้อมูลใหม่ (Predict N
% ew Data)
new_signal = [-65, 50];
predicted_label = predict(knn_model, new_signal);

% 4. พล็อตและแสดงผล (Plot and Visualize Results)
figure;
gscatter(training_data(:,1), training_data(:,2), training_labels, 'rb', 'xo', 10);
hold on;
plot(new_signal(1), new_signal(2), 'k*', 'MarkerSize', 15, 'LineWidth', 2);
title(sprintf('KNN Classification (k=%d)', k));
xlabel('Signal Strength');
ylabel('Data Rate');
legend('Normal', 'Anomalous', 'New Data', 'Location', 'best');
grid on;
hold off;

fprintf('--- KNN for Simple Signal Classification with Plot ---\n');
fprintf('New signal with strength %.1f and rate %.1f is predicted as Class: %d\n', new_signal(1), new_signal(2), predicted_label);
fprintf('  (1 = Normal, 2 = Anomalous)\n');
```

Matlab: Random Forests for Device Failure Prediction with plot

KNN_001.m | KNN_002.m | Random_Forests_001.m | Random_Forests_002.m | SVM_001.m | SVM_002.m | TrainASupportVectorMachineClassifierExample.mlx | TrainAKNNClassifierExample.mlx

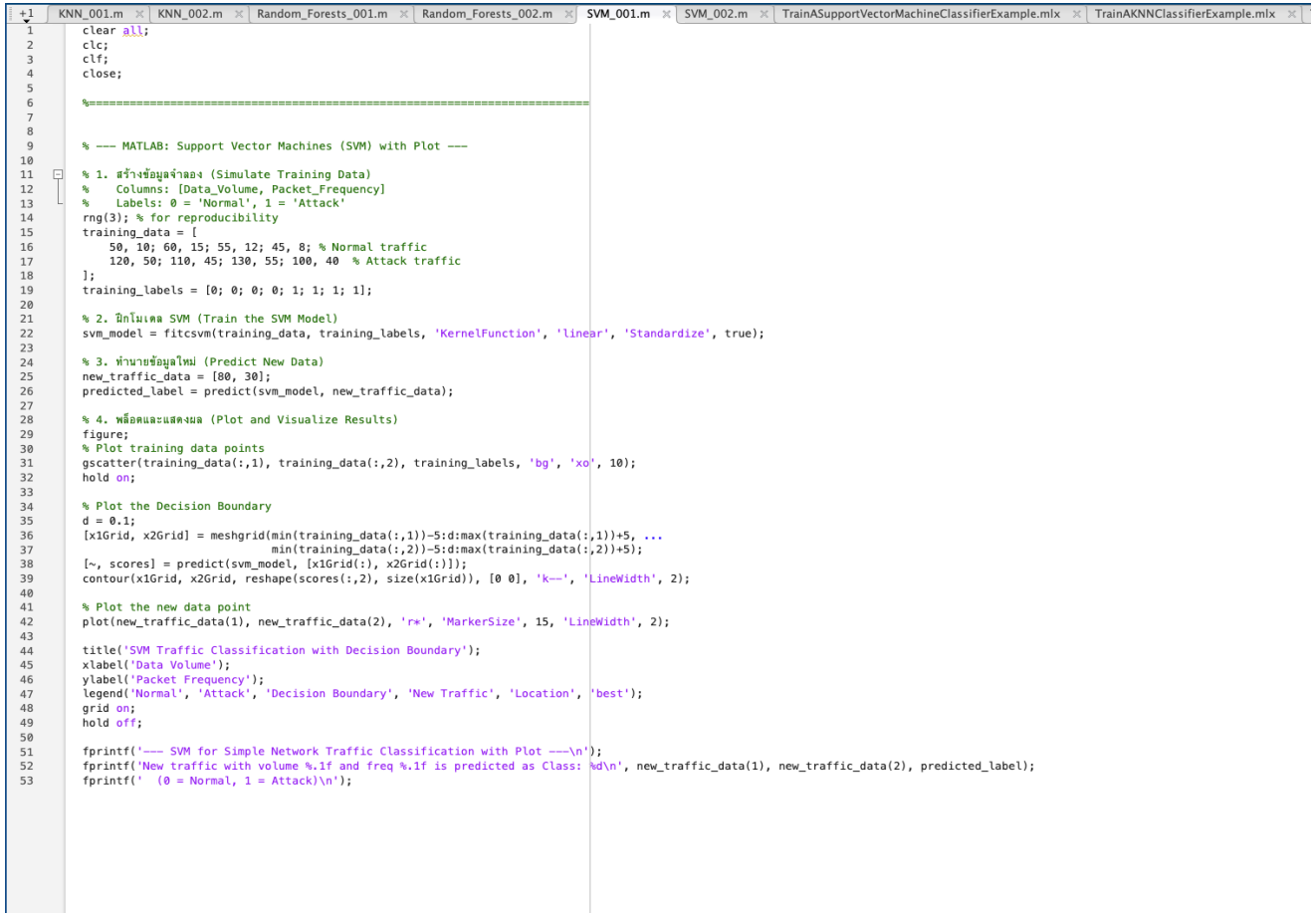
```
clear all;
clc;
clf;
close;

%======================================================================


% --- MATLAB: Random Forests for Device Failure Prediction with Plot ---

% 1. สร้างข้อมูลจำลอง (Simulate Training Data)
%    Columns: [Temperature, CPU_Usage]
%    Labels: 0 = 'Normal', 1 = 'Failure'
rng(2); % for reproducibility
training_data = [
    45, 60; 50, 70; 48, 65; 42, 55; % Normal devices
    70, 95; 75, 90; 68, 88; 80, 98  % Failed devices
];
training_labels = [0; 0; 0; 0; 1; 1; 1; 1];

% 2. ฝึกโมเดล Random Forests (Train the Random Forests Model)
num_trees = 100; % จำนวนต้นไม้ในป่า
rf_model = TreeBagger(num_trees, training_data, training_labels, 'OOBPrediction', 'on');

% 3. ทำนายข้อมูลใหม่ (Predict New Data)
new_device_data = [60, 85];
predicted_label = predict(rf_model, new_device_data);
predicted_label = str2double(predicted_label); % Convert from string to number

% 4. พล็อตและแสดงผล (Plot and Visualize Results)
figure;
gscatter(training_data(:,1), training_data(:,2), training_labels, 'bg', 'xo', 10);
hold on;
plot(new_device_data(1), new_device_data(2), 'r*', 'MarkerSize', 15, 'LineWidth', 2);
title('Random Forests for Device Failure Prediction');
xlabel('Temperature');
ylabel('CPU Usage');
legend('Normal', 'Failure', 'New Device', 'Location', 'best');
grid on;
hold off;

fprintf('--- Random Forests for Device Failure Prediction with Plot ---\n');
fprintf('New device with temp %.1f and CPU usage %.1f is predicted as Class: %d\n', new_device_data(1), new_device_data(2), predicted_label);
fprintf('  (0 = Normal, 1 = Failure)\n');
```

Matlab: Random Forests for channel Performance Prediction with Plot

KNN_001.m | KNN_002.m | Random_Forests_001.m | **Random_Forests_002.m** | SVM_001.m | SVM_002.m | TrainASupportVectorMachineClassifierExample.mlx | TrainAKNNClassifierExample.mlx

```
clear all;
clc;
clf;
close;

%======================================================================



% --- MATLAB: Random Forests for Channel Performance Prediction with Plot ---

% 1. Simulate Training Data for channel performance
%    Features: [Distance, Interference_Level]
%    Labels: 0 = 'Poor', 1 = 'Good'
rng(456); % For reproducibility
poor_performance = [15 + 5*randn(30,1), 10 + 3*randn(30,1)];
good_performance = [5 + 2*randn(30,1), 2 + 1*randn(30,1)];
training_data = [poor_performance; good_performance];
training_labels = [zeros(30,1); ones(30,1)];

% 2. Train the Random Forests Model
num_trees = 50; % Number of decision trees
rf_model = TreeBagger(num_trees, training_data, training_labels);

% 3. Predict the performance of a new channel
new_channel_data = [8, 5]; % New channel with unknown performance
predicted_label = predict(rf_model, new_channel_data);
predicted_label = str2double(predicted_label);

% 4. Plot the results to visualize the prediction
figure;
gscatter(training_data(:,1), training_data(:,2), training_labels, 'rb', 'xo', 10);
hold on;
plot(new_channel_data(1), new_channel_data(2), 'k*', 'MarkerSize', 15, 'LineWidth', 2);
title('Random Forests for Channel Performance Prediction');
xlabel('Distance (m)');
ylabel('Interference Level (units)');
legend('Poor', 'Good', 'New Channel', 'Location', 'best');
grid on;
hold off;

fprintf('--- Random Forests for Channel Performance Prediction with Plot ---\n');
fprintf('New channel with distance %.1f and interference %.1f is predicted as Class: %d\n', new_channel_data(1), new_channel_data(2), predicted_label);
fprintf('  (0 = Poor, 1 = Good)\n');
```

Matlab: Support Vector Machines (SVM) with plot



KNN_001.m | KNN_002.m | Random_Forests_001.m | Random_Forests_002.m | SVM_001.m | SVM_002.m | TrainASupportVectorMachineClassifierExample.mlx | TrainAKNNClassifierExample.mlx

```
clear all;
clc;
clf;
close;

%==========================================================================


% --- MATLAB: Support Vector Machines (SVM) with Plot ---

% 1. สร้างข้อมูลจำลอง (Simulate Training Data)
%    Columns: [Data_Volume, Packet_Frequency]
%    Labels: 0 = 'Normal', 1 = 'Attack'
rng(3); % for reproducibility
training_data = [
    50, 10; 60, 15; 55, 12; 45, 8; % Normal traffic
    120, 50; 110, 45; 130, 55; 100, 40  % Attack traffic
];
training_labels = [0; 0; 0; 0; 1; 1; 1; 1];

% 2. ฝึกโมเดล SVM (Train the SVM Model)
svm_model = fitcsvm(training_data, training_labels, 'KernelFunction', 'linear', 'Standardize', true);

% 3. ทำนายข้อมูลใหม่ (Predict New Data)
new_traffic_data = [80, 30];
predicted_label = predict(svm_model, new_traffic_data);

% 4. พล็อตและแสดงผล (Plot and Visualize Results)
figure;
% Plot training data points
gscatter(training_data(:,1), training_data(:,2), training_labels, 'bg', 'xo', 10);
hold on;

% Plot the Decision Boundary
d = 0.1;
[x1Grid, x2Grid] = meshgrid(min(training_data(:,1))-5:d:max(training_data(:,1))+5, ...
                            min(training_data(:,2))-5:d:max(training_data(:,2))+5);
[~, scores] = predict(svm_model, [x1Grid(:), x2Grid(:)]);
contour(x1Grid, x2Grid, reshape(scores(:,2), size(x1Grid)), [0 0], 'k--', 'LineWidth', 2);

% Plot the new data point
plot(new_traffic_data(1), new_traffic_data(2), 'r*', 'MarkerSize', 15, 'LineWidth', 2);

title('SVM Traffic Classification with Decision Boundary');
xlabel('Data Volume');
ylabel('Packet Frequency');
legend('Normal', 'Attack', 'Decision Boundary', 'New Traffic', 'Location', 'best');
grid on;
hold off;

fprintf('--- SVM for Simple Network Traffic Classification with Plot ---\n');
fprintf('New traffic with volume %.1f and freq %.1f is predicted as Class: %d\n', new_traffic_data(1), new_traffic_data(2), predicted_label);
fprintf('  (0 = Normal, 1 = Attack)\n');
```

Matlab: SVM for Network intrusion Dection with Plot

KNN_001.m | KNN_002.m | Random_Forests_001.m | Random_Forests_002.m | SVM_001.m | SVM_002.m | TrainASupportVectorMachineClassifierExample.mlx | TrainAKNNClassifierExample.mlx

```
clear all;
clc;
clf;
close;

%======================================================================


% --- MATLAB: SVM for Network Intrusion Detection with Plot ---


% 1. Simulate Training Data for network traffic
%    Features: [Data_Sent_Volume, Packet_Rate]
%    Labels: 0 = 'Normal', 1 = 'Intrusion'
rng(789); % For reproducibility
normal_traffic = [100 + 20*randn(40,1), 50 + 10*randn(40,1)];
intrusion_traffic = [300 + 50*randn(40,1), 200 + 40*randn(40,1)];
training_data = [normal_traffic; intrusion_traffic];
training_labels = [zeros(40,1); ones(40,1)];

% 2. Train the SVM Model
svm_model = fitcsvm(training_data, training_labels, 'KernelFunction', 'linear', 'Standardize', true);

% 3. Predict a new traffic type

new_traffic = [150, 80]; % New traffic with unknown type
predicted_label = predict(svm_model, new_traffic);

% 4. Plot the results to visualize the decision boundary
figure;
gscatter(training_data(:,1), training_data(:,2), training_labels, 'rb', 'xo', 10);
hold on;

% Plot the Decision Boundary
d = 5;
[x1Grid, x2Grid] = meshgrid(min(training_data(:,1))-d:d:max(training_data(:,1))+d, ...
                            min(training_data(:,2))-d:d:max(training_data(:,2))+d);
[~, scores] = predict(svm_model, [x1Grid(:), x2Grid(:)]);
contour(x1Grid, x2Grid, reshape(scores(:,2), size(x1Grid)), [0 0], 'k--', 'LineWidth', 2);

% Plot the new traffic point
plot(new_traffic(1), new_traffic(2), 'g*', 'MarkerSize', 15, 'LineWidth', 2);

title('SVM for Network Intrusion Detection');
xlabel('Data Sent Volume (MB)');
ylabel('Packet Rate (packets/sec)');
legend('Normal', 'Intrusion', 'Decision Boundary', 'New Traffic', 'Location', 'best');
grid on;
hold off;

fprintf('--- SVM for Network Intrusion Detection with Plot ---\n');
fprintf('New traffic with volume %.1f and rate %.1f is predicted as Class: %d\n', new_traffic(1), new_traffic(2), predicted_label);
fprintf('  (0 = Normal, 1 = Intrusion)\n');
```

Matlab:KNN for Radio Signal Classification

KNN_001.m | KNN_002.m | Random_Forests_001.m | Random_Forests_002.m | SVM_001.m | SVM_002.m | TrainASupportVectorMachineClassifierExample.mlx | TrainAKNNClassifierExample.mlx | TrainEnsembleOfBaggedClassificationTreesExample.mlx

```
clear all;
clc;
clf;
close;

%==========================================================================


% --- MATLAB: KNN for Radio Signal Classification with Plot ---

% 1. Simulate Training Data for two types of radio signals
%    Features: [RSSI, Frequency_Deviation]
%    Labels: 1 = 'WiFi', 2 = 'Bluetooth'
rng(123); % For reproducibility
wifi_data = [-60 + 5*randn(50,1), 10 + 2*randn(50,1)];
bluetooth_data = [-80 + 8*randn(50,1), 5 + 1*randn(50,1)];
training_data = [wifi_data; bluetooth_data];
training_labels = [ones(50,1); 2*ones(50,1)];

% 2. Train the KNN Model
k = 5; % Number of neighbors to consider
knn_model = fitcknn(training_data, training_labels, 'NumNeighbors', k);

% 3. Predict a new signal
new_signal = [-68, 7.5]; % A new signal with unknown type
predicted_label = predict(knn_model, new_signal);

% 4. Plot the results to visualize the classification
figure;
gscatter(training_data(:,1), training_data(:,2), training_labels, 'rb', 'xo', 10);
hold on;
plot(new_signal(1), new_signal(2), 'k*', 'MarkerSize', 15, 'LineWidth', 2);
title(sprintf('KNN Classification of Radio Signals (k=%d)', k));
xlabel('RSSI (dBm)');
ylabel('Frequency Deviation (kHz)');
legend('WiFi', 'Bluetooth', 'New Signal', 'Location', 'best');
grid on;
hold off;

fprintf('--- KNN for Radio Signal Classification with Plot ---\n');
fprintf('New signal with RSSI %.1f and Freq Dev %.1f is predicted as Class: %d\n', new_signal(1), new_signal(2), predicted_label);
fprintf('  (1 = WiFi, 2 = Bluetooth)\n');
```