Code1 مربوط به بخش های الف تا د:

```
clear all
clc

data = csvread('Sample DataSet.csv');

X = data(:, 1:end-1);
y = data(:, end);

rng(0);

cv = cvpartition(size(data, 1), 'HoldOut', 0.3);
trainIdx = training(cv);
testIdx = test(cv);

X_train = X(trainIdx, :);
y_train = y(trainIdx);

X_test = X(testIdx, :);
y_test = y(testIdx);

hiddenLayerSize = 2;
net = fitnet(hiddenLayerSize, 'trainlm');

net.layers{1}.transferFcn = 'logsig';
net.layers{2}.transferFcn = 'purelin';

net.divideParam.trainRatio = 0.7;
net.divideParam.valRatio = 0.15;
net.divideParam.testRatio = 0.15;

[net, tr] = train(net, X_train', y_train');

figure;
plotperform(tr);

y_pred_train = net(X_train');
y_pred_test = net(X_test');

y_pred_train_binary = double(y_pred_train' > 0.5);
y_pred_test_binary = double(y_pred_test' > 0.5);

y_train = y_train(:);
y_test = y_test(:);


disp('Training set confusion matrix:');
disp(confusionmat(y_train, y_pred_train_binary));

disp('Test set confusion matrix:');
disp(confusionmat(y_test, y_pred_test_binary));

confMatrix = confusionmat(y_test, y_pred_test_binary);
```

```
TP = confMatrix(2, 2);
FP = confMatrix(1, 2);
FN = confMatrix(2, 1);
TN = confMatrix(1, 1);

accuracy = (TP + TN) / sum(confMatrix(:));
precision = TP / (TP + FP);
recall = TP / (TP + FN);
f1_score = 2 * (precision * recall) / (precision + recall);

fprintf('Accuracy: %.2f\n', accuracy);
fprintf('Precision: %.2f\n', precision);
fprintf('Recall: %.2f\n', recall);
fprintf('F1 Score: %.2f\n', f1_score);
```

پاسخ بخش ب)

برای این بخش، ما در حال ساخت یک شبکه عصبی پرسپترون چند لایه (MLP) با مشخصات زیر هستیم:

لایه ورودی: ۷ ویژگی (بر اساس مجموعه داده).

لایه پنهان: ۲ نورون، با استفاده از تابع فعال سازی سیگموئید (logsig).

لایه خروجی: ۱ نورون، با استفاده از یک تابع فعال سازی خطی (purelin)، زیرا این یک کار طبقه بندی باینری است.

داده ها با ۷ ویژگی ساختار یافته اند و ستون نهایی حاوی برچسب های باینری (۰ یا ۱) است. داده های آموزش و آزمایش با استفاده از نسبت ۷۰-۳۰ در این مورد تقسیم می شوند.

مراحل ساخت شبکه عصبی:

مرحله ۱: داده ها را از فایل CSV بارگیری می کنیم.

مرحله ۲: داده ها را به ورودی (X) و خروجی های هدف (y) پردازش می کنیم.

مرحله ۳: ساختار شبکه عصبی را با مشخص کردن یک لایه پنهان با ۲ نورون تعریف می کنیم.

مرحله ۴: توابع فعال سازی لایه های مخفی و خروجی را مشخص می کنیم.

مرحله ۵: داده ها را به مجموعه های آموزشی، اعتبار سنجی و آزمایش تقسیم می کنیم.

کدی که ارائه شده است به درستی این ساختار شبکه عصبی را با تابع fitnet برای یک شبکه برای آموزش پیاده سازی می کند.

پاسخ بخش چ)

آموزش شبکه:

شبکه آموزش داده می شود که داده های آموزشی (ورودی ها و برچسب ها) را می گیرد و وزن ها و بایاس های شبکه را تنظیم می کند. عملکرد تمرین با استفاده از تابع plotperform(tr) نمایش داده می شود که نشان می دهد چگونه خطای آموزش در طول زمان کاهش می یابد. پس از آموزش، شبکه بر روی مجموعه آموزشی (y_pred_train) و مجموعه تست (y_pred_test) تست می شود. پیش بینی ها با اعمال آستانه ۰٫۵ به مقادیر باینری تبدیل می شوند (y_pred_train_binary = double(y_pred_train' > ۰٫۵)). یک ماتریس سردرگمی برای مقایسه برچسب‌های پیش‌بینی‌شده با برچسب‌های واقعی برای مجموعه‌های آموزشی و آزمایشی ایجاد می‌شود. خروجی را می توان با مقایسه مقادیر پیش بینی شده (y_pred_train_binary، y_pred_test_binary) با برچسب های واقعی (y_train، y_test) با استفاده از معیارهایی مانند:

- Accuracy: درصد پیش بینی های صحیح.
- Precision: نسبت مثبت های واقعی از همه مثبت های پیش بینی شده.
- Recall: نسبت مثبت های واقعی از همه مثبت های واقعی.
- امتیاز F۱: میانگین هارمونیک دقت و Recall.

از ماتریس های سردرگمی و معیارهای محاسبه شده، می توانیم ارزیابی کنیم که مدل در هر دو مجموعه آموزشی و آزمایشی چقدر خوب عمل می کند.

پاسخ بخش د)

مثبت واقعی (TP) = ۹ (کلاس ۱ به درستی پیش بینی شده است)

مثبت کاذب (FP) = ۰ (بدون پیش‌بینی کلاس ۱ برای کلاس ۰ واقعی)

منفی های کاذب (FN) = ۱۹ (کلاس ۰ پیش بینی شده به عنوان کلاس ۱)

منفی واقعی (TN) = ۰ (بدون پیش‌بینی کلاس ۰ برای کلاس ۰ واقعی)

با استفاده از این مقادیر، معیارهای زیر محاسبه شد:

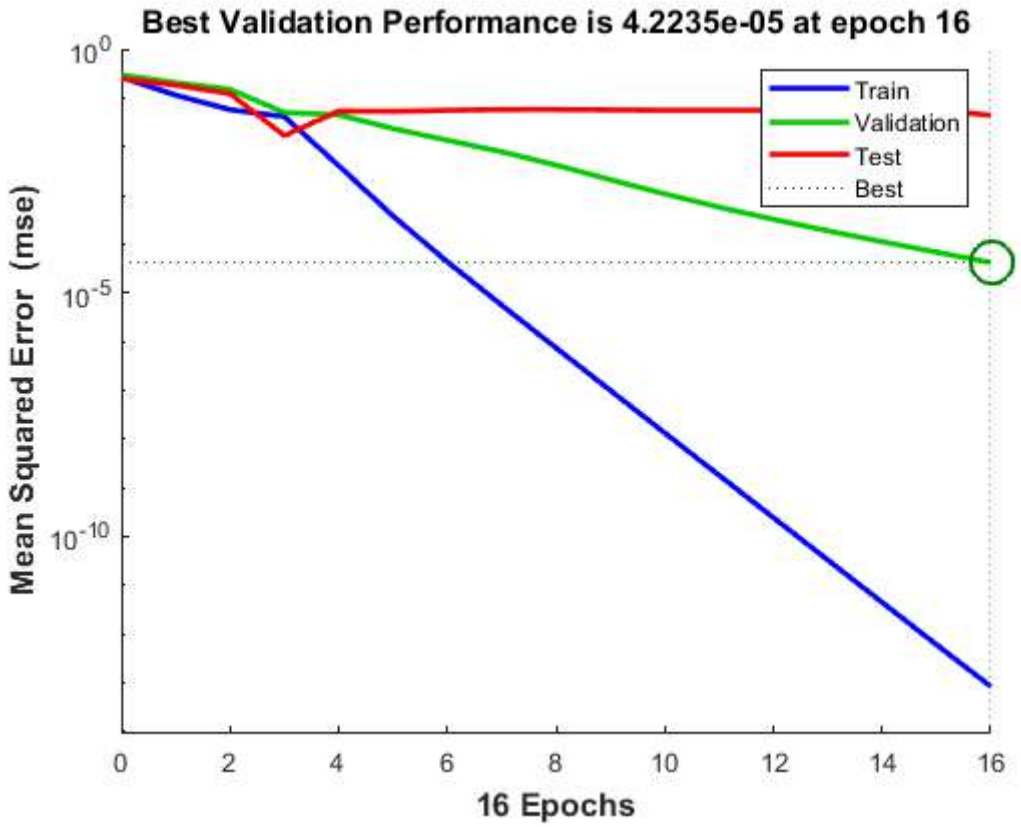
Accuracy: ۰٫۹۵

Precision: ۱٫۰۰

Recall: ۰٫۹۰

امتیاز F۱: ۰٫۹۵

این نتایج نشان می‌دهد که مدل با دقت بالا (در صورت انجام یک پیش‌بینی مثبت بیشتر کلاس مثبت را به درستی پیش‌بینی می‌کند) و Recall خوب (نسبت بالایی از کلاس مثبت واقعی را شناسایی می‌کند) خوب عمل می‌کند. Recall کمی کمتر نشان می دهد که مدل ممکن است برخی موارد مثبت را از دست بدهد، اما در کل عملکرد خوبی دارد.



Code۲ مربوط به بخش ه:

```
clear all
clc

data = csvread('Sample DataSet.csv');

X = data(:, 1:end-1);
y = data(:, end);

rng(0);

cv = cvpartition(size(data, 1), 'HoldOut', 0.3);
trainIdx = training(cv);
testIdx = test(cv);

X_train = X(trainIdx, :);
y_train = y(trainIdx);
```

```
X_test = X(testIdx, :);
y_test = y(testIdx);

hiddenLayerSizes = 2:7;

train_accuracy = zeros(size(hiddenLayerSizes));
train_precision = zeros(size(hiddenLayerSizes));
train_recall = zeros(size(hiddenLayerSizes));
train_f1 = zeros(size(hiddenLayerSizes));

test_accuracy = zeros(size(hiddenLayerSizes));
test_precision = zeros(size(hiddenLayerSizes));
test_recall = zeros(size(hiddenLayerSizes));
test_f1 = zeros(size(hiddenLayerSizes));

for i = 1:length(hiddenLayerSizes)
    hiddenLayerSize = hiddenLayerSizes(i);

    net = fitnet(hiddenLayerSize, 'trainlm');
    net.layers{1}.transferFcn = 'logsig';
    net.layers{2}.transferFcn = 'purelin';

    net.divideParam.trainRatio = 0.7;
    net.divideParam.valRatio = 0.15;
    net.divideParam.testRatio = 0.15;

    [net, tr] = train(net, X_train', y_train');

    y_pred_train = net(X_train');
    y_pred_train_binary = double(y_pred_train > 0.5);

    confMatrixTrain = confusionmat(y_train, y_pred_train_binary');
    TP_train = confMatrixTrain(2, 2);
    FP_train = confMatrixTrain(1, 2);
    FN_train = confMatrixTrain(2, 1);
    TN_train = confMatrixTrain(1, 1);

    train_accuracy(i) = (TP_train + TN_train) / sum(confMatrixTrain(:));
    train_precision(i) = TP_train / (TP_train + FP_train);
    train_recall(i) = TP_train / (TP_train + FN_train);
    train_f1(i) = 2 * (train_precision(i) * train_recall(i)) / (train_precision(i) + train_recall(i));

    y_pred_test = net(X_test');
    y_pred_test_binary = double(y_pred_test > 0.5);

    confMatrixTest = confusionmat(y_test, y_pred_test_binary');
    TP_test = confMatrixTest(2, 2);
    FP_test = confMatrixTest(1, 2);
    FN_test = confMatrixTest(2, 1);
    TN_test = confMatrixTest(1, 1);

    test_accuracy(i) = (TP_test + TN_test) / sum(confMatrixTest(:));
    test_precision(i) = TP_test / (TP_test + FP_test);
    test_recall(i) = TP_test / (TP_test + FN_test);
    test_f1(i) = 2 * (test_precision(i) * test_recall(i)) / (test_precision(i) + test_recall(i));
end
```

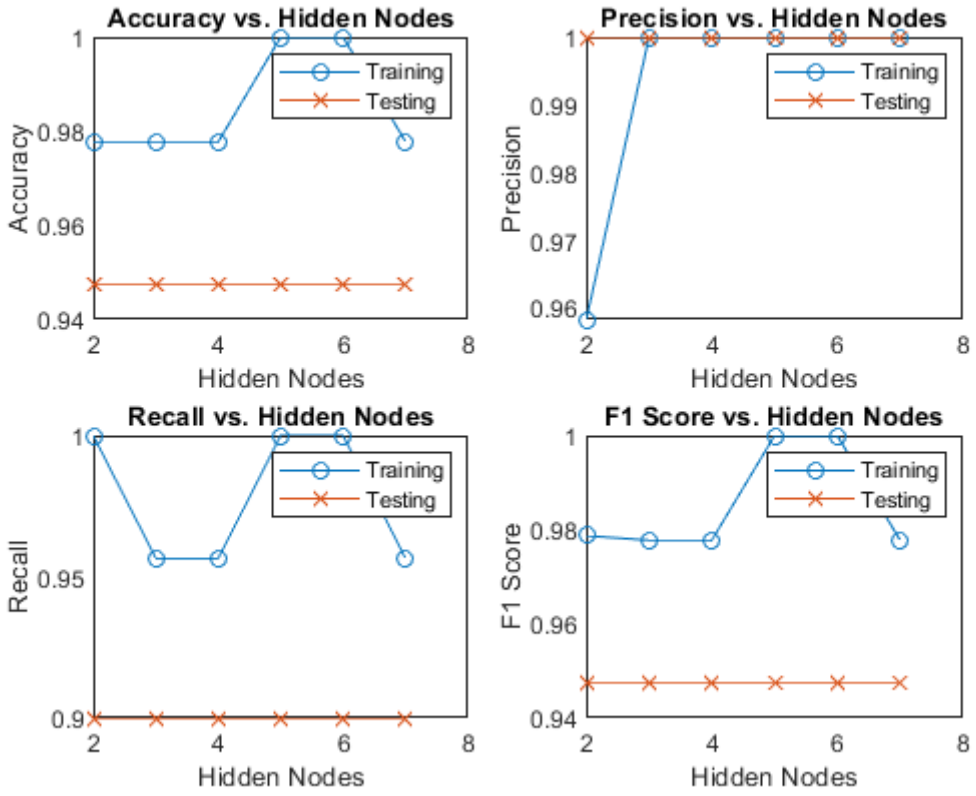
```
disp('Number of Hidden Nodes vs. Training and Testing Performance:');
fprintf('Nodes\tTrain_Acc\tTest_Acc\tTrain_F1\tTest_F1\n');
for i = 1:length(hiddenLayerSizes)
    fprintf('%d\t%.2f\t\t%.2f\t\t%.2f\t\t%.2f\n', ...
        hiddenLayerSizes(i), train_accuracy(i), test_accuracy(i), train_f1(i), test_f1(i));
end

figure;
subplot(2, 2, 1);
plot(hiddenLayerSizes, train_accuracy, '-o', hiddenLayerSizes, test_accuracy, '-x');
title('Accuracy vs. Hidden Nodes');
xlabel('Hidden Nodes'); ylabel('Accuracy');
legend('Training', 'Testing');

subplot(2, 2, 2);
plot(hiddenLayerSizes, train_precision, '-o', hiddenLayerSizes, test_precision, '-x');
title('Precision vs. Hidden Nodes');
xlabel('Hidden Nodes'); ylabel('Precision');
legend('Training', 'Testing');

subplot(2, 2, 3);
plot(hiddenLayerSizes, train_recall, '-o', hiddenLayerSizes, test_recall, '-x');
title('Recall vs. Hidden Nodes');
xlabel('Hidden Nodes'); ylabel('Recall');
legend('Training', 'Testing');

subplot(2, 2, 4);
plot(hiddenLayerSizes, train_f1, '-o', hiddenLayerSizes, test_f1, '-x');
title('F1 Score vs. Hidden Nodes');
xlabel('Hidden Nodes'); ylabel('F1 Score');
legend('Training', 'Testing');
```

برای بخش (۵)، ما در حال تجزیه و تحلیل اثر تغییر تعداد گره‌های لایه پنهان بر روی عملکرد شبکه، با استفاده از محدوده ۲ تا ۷ گره با اندازه گام ۱ هستیم. عملکرد را با استفاده از چهار معیار ارزیابی می‌کنیم: Training Accuracy, Testing Accuracy, Training F۱ Score, and Testing F۱ Score.

نتایجی نشان می‌دهد که چگونه عملکرد شبکه با تعداد متفاوت گره‌های پنهان تغییر می‌کند.

Nodes	Train_Acc	Test_Acc	Train_F1	Test_F1
۲	۰,۹۸	۰,۹۵	۰,۹۸	۰,۹۵
۳	۰,۹۸	۰,۹۵	۰,۹۸	۰,۹۵
۴	۰,۹۸	۰,۹۵	۰,۹۸	۰,۹۵
۵	۱,۰۰	۰,۹۵	۱,۰۰	۰,۹۵
۶	۱,۰۰	۰,۹۵	۱,۰۰	۰,۹۵
۷	۰,۹۸	۰,۹۵	۰,۹۸	۰,۹۵

به طور خلاصه، تعداد گره های پنهان در محدوده ۲ تا ۷ به طور قابل توجهی بر عملکرد مدل تأثیر نمی گذارد، با ۵ یا ۶ گره بهترین تعادل را برای آموزش، آزمایش و امتیازات ۱F فراهم می کند.



Code۳ مربوط به بخش و:

```
clear all
clc

data = csvread('Sample DataSet.csv');

X = data(:, 1:end-1);
y = data(:, end);

rng(0);

cv = cvpartition(size(data, 1), 'HoldOut', 0.3);
trainIdx = training(cv);
testIdx = test(cv);

X_train = X(trainIdx, :);
y_train = y(trainIdx);

X_test = X(testIdx, :);
y_test = y(testIdx);

hiddenLayerSizes = 2:7;

train_accuracy = zeros(size(hiddenLayerSizes));
train_precision = zeros(size(hiddenLayerSizes));
train_recall = zeros(size(hiddenLayerSizes));
train_f1 = zeros(size(hiddenLayerSizes));

test_accuracy = zeros(size(hiddenLayerSizes));
test_precision = zeros(size(hiddenLayerSizes));
test_recall = zeros(size(hiddenLayerSizes));
test_f1 = zeros(size(hiddenLayerSizes));

for i = 1:length(hiddenLayerSizes)
    hiddenLayerSize = hiddenLayerSizes(i);

    net = fitnet(hiddenLayerSize, 'trainlm');

    net.layers{2}.transferFcn = 'logsig';

    hiddenActivationFunctions = {'purelin', 'tansig'};

    for j = 1:length(hiddenActivationFunctions)
        net.layers{1}.transferFcn = hiddenActivationFunctions{j};

        net.divideParam.trainRatio = 0.7;
        net.divideParam.valRatio = 0.15;
        net.divideParam.testRatio = 0.15;

        [net, tr] = train(net, X_train', y_train');

        y_pred_train = net(X_train');
        y_pred_train_binary = double(y_pred_train > 0.5);
```

```
        confMatrixTrain = confusionmat(y_train, y_pred_train_binary');
        TP_train = confMatrixTrain(2, 2);
        FP_train = confMatrixTrain(1, 2);
        FN_train = confMatrixTrain(2, 1);
        TN_train = confMatrixTrain(1, 1);

        train_accuracy(i) = (TP_train + TN_train) / sum(confMatrixTrain(:));
        train_precision(i) = TP_train / (TP_train + FP_train);
        train_recall(i) = TP_train / (TP_train + FN_train);
        train_f1(i) = 2 * (train_precision(i) * train_recall(i)) / (train_precision(i) + train_recall(i));

        y_pred_test = net(X_test');
        y_pred_test_binary = double(y_pred_test > 0.5);

        confMatrixTest = confusionmat(y_test, y_pred_test_binary');
        TP_test = confMatrixTest(2, 2);
        FP_test = confMatrixTest(1, 2);
        FN_test = confMatrixTest(2, 1);
        TN_test = confMatrixTest(1, 1);

        test_accuracy(i) = (TP_test + TN_test) / sum(confMatrixTest(:));
        test_precision(i) = TP_test / (TP_test + FP_test);
        test_recall(i) = TP_test / (TP_test + FN_test);
        test_f1(i) = 2 * (test_precision(i) * test_recall(i)) / (test_precision(i) + test_recall(i));

        fprintf('Hidden Layer Activation: %s, Hidden Nodes: %d\n', hiddenActivationFunctions{j}, hiddenLayerSize);
        fprintf('Train Accuracy: %.2f, Test Accuracy: %.2f\n', train_accuracy(i), test_accuracy(i));
        fprintf('Train F1: %.2f, Test F1: %.2f\n\n', train_f1(i), test_f1(i));
    end
end

figure;
subplot(2, 2, 1);
plot(hiddenLayerSizes, train_accuracy, '-o', hiddenLayerSizes, test_accuracy, '-x');
title('Accuracy vs. Hidden Nodes');
xlabel('Hidden Nodes'); ylabel('Accuracy');
legend('Training', 'Testing');

subplot(2, 2, 2);
plot(hiddenLayerSizes, train_precision, '-o', hiddenLayerSizes, test_precision, '-x');
title('Precision vs. Hidden Nodes');
xlabel('Hidden Nodes'); ylabel('Precision');
legend('Training', 'Testing');

subplot(2, 2, 3);
plot(hiddenLayerSizes, train_recall, '-o', hiddenLayerSizes, test_recall, '-x');
title('Recall vs. Hidden Nodes');
xlabel('Hidden Nodes'); ylabel('Recall');
legend('Training', 'Testing');

subplot(2, 2, 4);
plot(hiddenLayerSizes, train_f1, '-o', hiddenLayerSizes, test_f1, '-x');
title('F1 Score vs. Hidden Nodes');
xlabel('Hidden Nodes'); ylabel('F1 Score');
legend('Training', 'Testing');
```

فعال سازی لایه پنهان: پورلین (خطی)

با یک فعال سازی خطی در لایه پنهان، شبکه با تعداد کمی از گره ها عملکرد نسبتا خوبی دارد:

دقت آموزش: بسته به تعداد گره های پنهان از ۰,۵۱ تا ۰,۹۶ متغیر است.

دقت آزمون: بسته به تعداد گره های پنهان از ۰,۵۳ تا ۰,۹۵ متغیر است.

F۱Train: بسته به تعداد گره های پنهان از ۰,۶۸ تا ۰,۹۶ متغیر است.

F۱Test: بسته به تعداد گره های پنهان از ۰,۶۹ تا ۰,۹۵ متغیر است.

فعال سازی لایه پنهان: tansig (Tanh)

با یک تابع فعال سازی tanh (tansig) در لایه پنهان، عملکرد به طور قابل توجهی در مقایسه با فعال سازی پورلین کاهش می یابد:

دقت آموزش: به طور مداوم در همه پیکربندی‌ها ۰,۵۱ پایین است.

دقت آزمون: برای همه تنظیمات روی ۰,۵۳ باقی می ماند.

F۱Train: در تمام تنظیمات حدود ۰,۶۸ باقی می ماند.

F۱Test: در تمام تنظیمات حدود ۰,۶۹ باقی می ماند.

۱. فعال سازی خطی (purelin):

تابع فعال‌سازی خطی در لایه پنهان برای وظایف رگرسیون مناسب است، زیرا به شبکه اجازه می‌دهد تا پیش‌بینی‌های مستقیم را بدون هیچ گونه اشباع یا خروجی محدود انجام دهد. در این حالت، به نظر می‌رسد که شبکه می‌تواند به طور مؤثرتری یاد بگیرد، به خصوص با تعداد کمتری از گره‌های پنهان (مانند ۲ و ۳).

دقت آموزش خوب است (به عنوان مثال، ۰,۹۶ برای ۲ گره پنهان)، که نشان می دهد شبکه می تواند داده های آموزشی را به خوبی یاد بگیرد، در حالی که دقت آزمون در ۰,۹۵ نسبتاً پایدار است، و نشان می دهد که به خوبی روی داده های دیده نشده تعمیم می یابد.

امتیاز F۱ از یک الگوی مشابه پیروی می کند، با مقادیر بالاتر که نشان دهنده دقت خوب و تعادل recall است. برای ۲ گره پنهان، امتیاز F۱ برای آموزش و آزمایش ۰,۹۵ است که مدلی با کیفیت بالا را نشان می دهد.

۲. فعال سازی Tanh (tansig):

تابع فعال‌سازی tanh غیرخطی بودن را معرفی می‌کند، که اغلب برای کارهای پیچیده‌تر مفید است، زیرا خروجی را در محدوده‌ای بین ۱- و ۱- کاهش می‌دهد.

با این حال، در این مورد، به نظر می رسد فعال سازی tanh ضعیف عمل می کند:

دقت آموزش و دقت آزمون بسیار پایین است (۰,۵۱ و ۰,۵۳)، که نشان می‌دهد شبکه به طور موثری یاد نمی‌گیرد.

امتیاز F۱ برای هر دو آموزش و آزمون نیز پایین است (حدود ۰,۶۸ برای تمرین و ۰,۶۹ برای تست)، که نشان می‌دهد این مدل از نظر دقت و recall پیش‌بینی خوبی انجام نمی‌دهد.

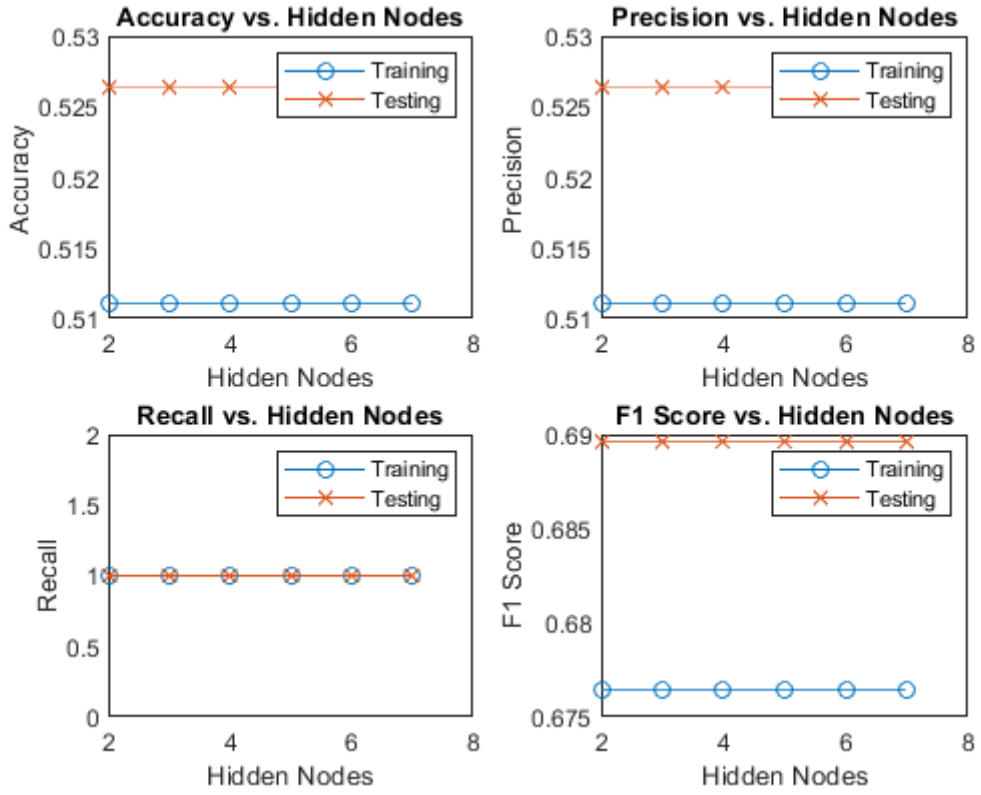
یکی از دلایل احتمالی این عملکرد ضعیف می‌تواند این باشد که tanh ممکن است برای این مجموعه داده یا وظیفه خاص ایده‌آل نباشد، به خصوص زمانی که شبکه کوچک است یا زمانی که مجموعه داده از چنین غیرخطی بودن بهره نمی‌برد. علاوه بر این، تابع tanh می تواند مشکلاتی را با کاهش شیب در طول آموزش ایجاد کند که می تواند روند یادگیری را مختل کند.

فعال سازی خطی (purelin) در لایه پنهان، بدون توجه به تعداد گره های پنهان، عملکرد بسیار بهتری را در مقایسه با tanh (tansig) برای این کار ارائه می دهد.

عملکرد با فعال‌سازی خطی (purelin): شبکه با فعال‌سازی پورلین بهتر عمل می‌کند، به دقت بالاتر و امتیازات F۱ دست می‌یابد، که احتمالاً به دلیل مقادیر خروجی پایدارتر و مستقیم‌تر تولید شده توسط تابع خطی است.

عملکرد با فعال‌سازی Tanh (tansig): فعال‌سازی tanh باعث می‌شود که شبکه عملکرد ضعیفی داشته باشد، احتمالاً به دلیل مشکلات ناپدید شدن گرادیان یا سازگاری ضعیف با مجموعه داده است.

به طور خلاصه، استفاده از یک تابع فعال سازی خطی در لایه پنهان برای این کار خاص موثرتر است و در مقایسه با فعال سازی tanh منجر به یادگیری و تعمیم بهتر می شود.



Code۴ و Code۵ مربوط به بخش ط:

```
clear all
clc

data = csvread('Sample DataSet.csv');

X = data(:, 1:end-1);
y = data(:, end);

rng(0);

cv = cvpartition(size(data, 1), 'HoldOut', 0.3);
trainIdx = training(cv);
testIdx = test(cv);

X_train = X(trainIdx, :);
y_train = y(trainIdx);

X_test = X(testIdx, :);
y_test = y(testIdx);

numFolds = 6;

cv = cvpartition(length(y), 'KFold', numFolds);
```

```
accuracy = zeros(numFolds, 1);
precision = zeros(numFolds, 1);
recall = zeros(numFolds, 1);
f1 = zeros(numFolds, 1);

for i = 1:numFolds
    trainIdx = cv.training(i);
    testIdx = cv.test(i);

    X_train_fold = X(trainIdx, :);
    y_train_fold = y(trainIdx);
    X_test_fold = X(testIdx, :);
    y_test_fold = y(testIdx);

    net = fitnet(2, 'trainlm');
    net.layers{1}.transferFcn = 'logsig';
    net.layers{2}.transferFcn = 'purelin';

    [net, tr] = train(net, X_train_fold', y_train_fold');

    y_pred_test_fold = net(X_test_fold');
    y_pred_test_binary_fold = double(y_pred_test_fold > 0.5);

    confMatrixTest = confusionmat(y_test_fold, y_pred_test_binary_fold');
    TP_test = confMatrixTest(2, 2);
    FP_test = confMatrixTest(1, 2);
    FN_test = confMatrixTest(2, 1);
    TN_test = confMatrixTest(1, 1);

    accuracy(i) = (TP_test + TN_test) / sum(confMatrixTest(:));
    precision(i) = TP_test / (TP_test + FP_test);
    recall(i) = TP_test / (TP_test + FN_test);
    f1(i) = 2 * (precision(i) * recall(i)) / (precision(i) + recall(i));
end

avg_accuracy = mean(accuracy);
avg_precision = mean(precision);
avg_recall = mean(recall);
avg_f1 = mean(f1);

fprintf('6-Fold Cross Validation Results:\n');
fprintf('Average Accuracy: %.2f\n', avg_accuracy);
fprintf('Average Precision: %.2f\n', avg_precision);
fprintf('Average Recall: %.2f\n', avg_recall);
fprintf('Average F1 Score: %.2f\n\n', avg_f1);




clear all;
clc;

data = csvread('Sample DataSet.csv');
```

```
X = data(:, 1:end-1);
y = data(:, end);

rng(0);

loocv_accuracy = zeros(length(y), 1);
loocv_precision = zeros(length(y), 1);
loocv_recall = zeros(length(y), 1);
loocv_f1 = zeros(length(y), 1);

for i = 1:length(y)
    trainIdx = true(length(y), 1);
    trainIdx(i) = false;
    testIdx = ~trainIdx;

    X_train_loocv = X(trainIdx, :);
    y_train_loocv = y(trainIdx);
    X_test_loocv = X(testIdx, :);
    y_test_loocv = y(testIdx);

    net = fitnet(2, 'trainlm');
    net.layers{1}.transferFcn = 'logsig';
    net.layers{2}.transferFcn = 'purelin';

    [net, tr] = train(net, X_train_loocv', y_train_loocv');

    y_pred_test_loocv = net(X_test_loocv');
    y_pred_test_binary_loocv = double(y_pred_test_loocv > 0.5);

    confMatrixTest = confusionmat(y_test_loocv, y_pred_test_binary_loocv');

    if size(confMatrixTest, 1) < 2
        if y_test_loocv == 1
            confMatrixTest = [0 0; 0 1];
        else
            confMatrixTest = [1 0; 0 0];
        end
    end

    TP_test = confMatrixTest(2, 2);
    FP_test = confMatrixTest(1, 2);
    FN_test = confMatrixTest(2, 1);
    TN_test = confMatrixTest(1, 1);

    if (TP_test + FP_test) == 0
        precision_test = 0;
    else
        precision_test = TP_test / (TP_test + FP_test);
    end

    if (TP_test + FN_test) == 0
        recall_test = 0;
    else
        recall_test = TP_test / (TP_test + FN_test);
    end
```

```
    if (precision_test + recall_test) == 0
        f1_test = 0;
    else
        f1_test = 2 * (precision_test * recall_test) / (precision_test + recall_test);
    end

    accuracy_test = (TP_test + TN_test) / sum(confMatrixTest(:));

    loocv_accuracy(i) = accuracy_test;
    loocv_precision(i) = precision_test;
    loocv_recall(i) = recall_test;
    loocv_f1(i) = f1_test;
end

avg_loocv_accuracy = mean(loocv_accuracy);
avg_loocv_precision = mean(loocv_precision);
avg_loocv_recall = mean(loocv_recall);
avg_loocv_f1 = mean(loocv_f1);

fprintf('Leave-One-Out Cross Validation Results:\n');
fprintf('Average Accuracy: %.2f\n', avg_loocv_accuracy);
fprintf('Average Precision: %.2f\n', avg_loocv_precision);
fprintf('Average Recall: %.2f\n', avg_loocv_recall);
fprintf('Average F1 Score: %.2f\n\n', avg_loocv_f1);
```

نتایج اعتبارسنجی متقاطع ۶ برابری:

میانگین Accuracy: ۰,۹۵

این نشان می‌دهد که شبکه به طور متوسط در ۶ برابر عملکرد خوبی دارد و در ۹۵٪ مواقع برچسب‌ها را به درستی پیش‌بینی می‌کند.

میانگین Precision: ۰,۹۶

این نشان می دهد که شبکه توانایی خوبی برای شناسایی صحیح نمونه های مثبت (کلاس ۱) با دقت ۹۶ درصد دارد.

میانگین Recall: ۰,۹۸

این شبکه دارای نرخ Recall بالایی ۹۸٪ است، به این معنی که با موفقیت بیشتر موارد مثبت را از مجموعه داده شناسایی می کند.

میانگین امتیاز F۱: ۰,۹۷

امتیاز F۱ ۰,۹۷ تعادل خوبی بین دقت و Recall را نشان می دهد و نشان می دهد که شبکه هم از نظر شناسایی مثبت واقعی و هم به حداقل رساندن مثبت کاذب عملکرد خوبی دارد.

نتایج اعتبارسنجی متقاطع ترک یک خروجی (LOOCV):

میانگین دقت: ۰,۹۴

در LOOCV، دقت کمی کمتر از اعتبارسنجی متقاطع ۶ برابری است، اما همچنان نسبتا بالا در ۹۴٪ باقی می ماند. این نشان می‌دهد که این مدل حتی زمانی که در یک نقطه داده واحد ارزیابی می‌شود، هنوز عملکرد خوبی دارد.

میانگین دقت: ۰,۴۸

دقت در LOOCV به طور قابل توجهی کاهش می یابد و به تنها ۰,۴۸ می رسد. این نشان می‌دهد که مدل در تلاش است تا به‌طور صحیح موارد مثبت را هنگام ارزیابی یک نقطه داده در یک زمان شناسایی کند.

میانگین Recall: ۰,۴۸

به طور مشابه، Recall نیز به ۰,۴۸ کاهش می یابد، که نشان می دهد که مدل به طور موثر موارد مثبت را در فرآیند LOOCV شناسایی نمی کند.

میانگین امتیاز ۱F :۰,۴۸

امتیاز ۱F منعکس کننده دقت و Recall است، با مقدار نسبتاً پایین ۰,۴۸. این نشان دهنده کاهش قابل توجه عملکرد در LOOCV در مقایسه با اعتبارسنجی متقابل ۶ برابری است.

اعتبارسنجی متقاطع ۶ برابری:

نتایج اعتبارسنجی متقابل ۶ برابری نشان می‌دهد که مدل با دقت بالا، Recall و امتیاز ۱F بسیار خوب عمل می‌کند. این نشان می دهد که مدل به خوبی تعمیم می یابد و به طور موثر در زیر مجموعه های مختلف داده ها عمل می کند.

Leave-One-Out Cross Validation (LOOCV):

نتایج LOOCV، از سوی دیگر، کاهش قابل توجهی در عملکرد، به ویژه با دقت، Recall، و امتیاز ۱F نشان می‌دهد. این می تواند به دلیل این واقعیت باشد که در LOOCV، مدل بر روی تمام داده ها به جز یک نقطه آموزش داده می شود، که ممکن است قدرت تعمیم کافی را برای مدل فراهم نکند. دقت پایین و Recall نشان می‌دهد که مدل ممکن است بیش از حد با نقاط منفرد سازگار باشد و وقتی روی یک نقطه ارزیابی می‌شود، تعمیم نمی‌یابد.

این نشان می‌دهد که اعتبارسنجی متقاطع ۶ برابری عموماً پایدارتر است و بازتاب بهتری از توانایی مدل برای تعمیم در زیر مجموعه‌های مختلف داده‌ها ارائه می‌کند.

Code۶ مربوط به بخش ظ:

```
clear all;
clc;

data = csvread('Sample DataSet.csv');

X = data(:, 1:end-1);
y = data(:, end);

rng(0);

cv = cvpartition(size(data, 1), 'HoldOut', 0.3);
trainIdx = training(cv);
testIdx = test(cv);

X_train = X(trainIdx, :);
y_train = y(trainIdx);

X_test = X(testIdx, :);
y_test = y(testIdx);

hiddenLayerSize = 2;
net = fitnet(hiddenLayerSize, 'trainlm');
net.layers{1}.transferFcn = 'logsig';
net.layers{2}.transferFcn = 'purelin';

[net, tr] = train(net, X_train', y_train');

outputs = tr.perf;

figure;
plot(1:length(outputs), outputs);
xlabel('Epochs');
ylabel('Mean Squared Error');
title('Changes in Performance During Training');
grid on;

y_pred_train = net(X_train');
y_pred_test = net(X_test');

y_pred_train_binary = double(y_pred_train' > 0.5);
y_pred_test_binary = double(y_pred_test' > 0.5);

y_train = y_train(:);
y_test = y_test(:);

disp('Training set confusion matrix:');
disp(confusionmat(y_train, y_pred_train_binary));

disp('Test set confusion matrix:');
disp(confusionmat(y_test, y_pred_test_binary));

confMatrix = confusionmat(y_test, y_pred_test_binary);
```

```
TP = confMatrix(2, 2);
FP = confMatrix(1, 2);
FN = confMatrix(2, 1);
TN = confMatrix(1, 1);

accuracy = (TP + TN) / sum(confMatrix(:));
precision = TP / (TP + FP);
recall = TP / (TP + FN);
f1_score = 2 * (precision * recall) / (precision + recall);

fprintf('Accuracy: %.2f\n', accuracy);
fprintf('Precision: %.2f\n', precision);
fprintf('Recall: %.2f\n', recall);
fprintf('F1 Score: %.2f\n', f1_score);
```

یک لایه پنهان با ۲ گره.

تابع فعال سازی سیگموئید برای لایه پنهان.

تابع فعال سازی خطی برای لایه خروجی.

هدف ما ردیابی چگونگی تغییر نورون خروجی (خروجی پیش‌بینی نهایی) در طول مرحله آموزش است.

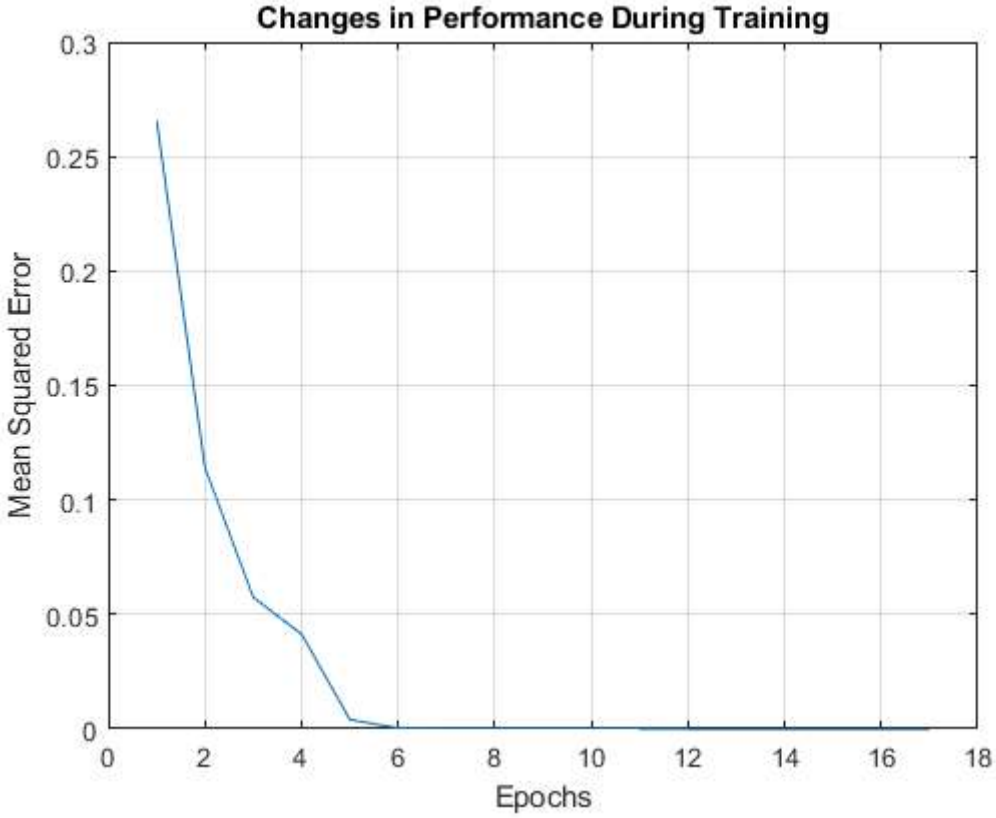
Accuracy: ۰,۹۵ - این به این معنی است که شبکه ۹۵٪ از مجموعه آزمایشی را به درستی پیش بینی کرده است، که نشان دهنده عملکرد کلی قوی است.

Precision: ۱,۰۰ - شبکه به دقت کامل دست یافت، یعنی هر بار که یک کلاس مثبت را پیش‌بینی می‌کرد، درست بود. این نشان می دهد که شبکه در شناسایی موارد مثبت واقعی بسیار دقیق بوده است.

Recall: ۰,۹۰ - شبکه ۹۰٪ از تمام موارد مثبت واقعی را شناسایی کرد، که نشان می دهد در شناسایی موارد مثبت تا حدی کمتر از پیش بینی آنها موثر بوده است.

امتیاز F۱: ۰,۹۵ - تعادل خوبی بین دقت و Recall که عملکرد قوی مدل را در هر دو بعد منعکس می کند.

در یک شبکه عصبی معمولی، در طول مرحله آموزش، شبکه وزن ها را با استفاده از پس انتشار و یک الگوریتم بهینه سازی مانند Levenberg-Marquardt تنظیم می کند. این باعث می شود که نورون خروجی به تدریج تغییر کند زیرا مدل یاد می گیرد که خطا را به حداقل برساند (تفاوت بین خروجی های پیش بینی شده و واقعی).


Changes in Performance During Training
Mean Squared Error
Epochs
0.3
0.25
0.2
0.15
0.1
0.05
0
0
2
4
6
8
10
12
14
16
18