**Computer Assignment 3 - Signals & Systems - Dr Akhavan**

**Amirali Dehghani - 810102443**

## Question 1

در این بخش هدف، پنهان‌کردن یک متن در یک تصویر است. این کار به روش‌های مختلفی قابل انجام است که در اینجا با پیدا کردن شلوغ‌ترین نقاط عکس و قرار دادن پیام در LSB (Least significant bit) یا کم‌ارزش‌ترین بیت انجام شده است. در ادامه توضیحات بیشتری درباره‌ی این داده خواهد شد.

clc, clearvars, close all;

### 1 - 1) Creating Mapset

در ابتدا ما باید مپ ست مورد نظر خودمان را بسازیم. در اینجا یک آرایه‌ی دو بعدی از حروف الفبای انگلیسی و چند کاراکتر داریم که دومین عضو هر کدام، باینری شده‌ی آن کاراکتر تا 5 بیت است.

Nch = 32;

mapset = cell(2,Nch);

Alphabet ='abcdefghijklmnopqrstuvwxyz .,!";';

for i = 1:Nch

mapset{1,i}=Alphabet(i);

mapset{2,i}=dec2bin(i-1,5);

end

### 1 - 2) Encoding a picture

در این بخش تصویری که می‌خواهیم آن را رمز‌گذاری کنیم را بارگذاری می‌کنیم و آن را سیاه سفید می‌کنیم. علت آن این است که عکس‌های سیاه سفید، در واقع در هر پیکسل مقدار R، G و B آن‌ها با یکدیگر برابر است و در نتیجه در هنگام مواجه با هر پیکسل، فقط با یک عدد کار می‌کنیم. سپس وارد تابع coding میشویم که توضیحات آن در ادامه داده خواهد شد.

[file, path] = uigetfile({'\*.jpg;\*.png'}, "Choose your picture: ");

picture = imread([path, file]);

picture = rgb2gray(picture);

message = 'signal;';

codedPicture = coding(picture, message, mapset);

تابع message2binary، پیام حاوی متن را که به صورت یک رشته است، دریافت کرده و تا زمانی که رشته تمام نشده یا به کاراکتر ; نرسیده، ادامه می‌دهد و در نهایت خروجی آن، یک باینری است.

function binarizedMessage = message2binary(message, mapset)

binarizedMessage = '';

for charInMessage = message

found = false;

for charInMapSet = mapset

if charInMessage == charInMapSet{1};

binarizedMessage = [binarizedMessage, charInMapSet{2}];

found = true;

break;

end

end

if ~found

fprintf('Character "%c" not found in mapset. Skipping...\n', charInMessage);

end

end

end

تابع sortBlocks، بلاک‌هایی که می‌توان در آن‌ها کلمه را پنهان کرد خروجی می‌دهد. اساس کار آن به این شکل است که بدون در نظر گرفتن بیت آخر هر پیکسل یا همان LSB، کنتراست هر بلاک blockSize \* blockSize را به صورت تفاضل بیشترین و کمترین پیکسل آن بلاک محاسبه می‌کند و اگر از threshold ورودی داده شده بیشتر باشد، آن بلاک را به عنوان بلاکی که می‌توان در آن پیام را مخفی کرد ذخیره کرده و در آخر تمامی آن‌ها را خروجی می‌دهد.

function availableBlocks = sortBlocks(picture, roundedSize, blockSize, threshold)

blocks = {}; availableBlocks = {};

for i = 1:blockSize:roundedSize(1)

for j = 1:blockSize:roundedSize(2)

pixels = picture(i:i + blockSize-1, j:j + blockSize-1);

blockWithoutLSB = bitand(pixels, 254);

contrast = max(blockWithoutLSB(:)) - min(blockWithoutLSB(:));

block = containers.Map();

block('pixels') = pixels;

block('contrast') = contrast;

block('position') = [i, j];

if (contrast > threshold)

availableBlocks{end+1} = block;

end

blocks{end+1} = block;

end

end

contrasts = cellfun(@(block) block('contrast'), availableBlocks);

[~, sortedIndices] = sort(contrasts, 'descend');

availableBlocks = availableBlocks(sortedIndices);

end

تابع coding اصل کار ما را انجام می‌دهد. با گرفتن پبام باینری شده و بلاک‌های مطلوب از توابعی که آن‌ها را تعریف کردیم، هر بیت پیام را در بیت آخر هر پیکسل می‌نویسد.

function codedPicture = coding(picture, message, mapset)

blockSize = 20;

threshold = 70;

codedPicture = picture;

[height, width] = size(picture);

roundedSize = [floor(height / blockSize) \* blockSize, floor(width / blockSize) \* blockSize];

messageLength = length(message);

binarizedMessage = message2binary(message, mapset);

availableBlocks = sortBlocks(picture, roundedSize, blockSize, threshold);

numOfAvailableBlocks = length(availableBlocks);

if numOfAvailableBlocks < messageLength

fprintf("You can only fit %d characters in this picture but your message has %d characters.\n", numOfAvailableBlocks, messageLength);

codedPicture = 1;

return

end

visualizeBlocks(picture, blockSize, threshold);

messageIndex = 1;

for i = 1:numOfAvailableBlocks

block = availableBlocks{i};

pixels = block('pixels');

position = block('position'); x = position(1); y = position(2);

for j = 1:blockSize

for k = 1:blockSize

if messageIndex > length(binarizedMessage)

break

end

pixel = pixels(j, k);

binarizedPixel = dec2bin(pixel, 8);

binarizedPixel(end) = binarizedMessage(messageIndex);

pixels(j, k) = bin2dec(binarizedPixel);

messageIndex = messageIndex + 1;

end

if messageIndex > length(binarizedMessage)

break

end

end

codedPicture(x : x + blockSize-1, y : y + blockSize-1) = pixels;

end

end

### 

### Additional) Showing available blocks in photo

این بخش به صورت اضافه، بلاک‌هایی که در آن می‌توان پیام را مخفی کرد را در عکس دورشان را مستطیل قرمز کشیده و نشان می‌دهد.

function visualizeBlocks(picture, blockSize, threshold)

[height, width] = size(picture);

roundedSize = [floor(height / blockSize) \* blockSize, floor(width / blockSize) \* blockSize];

availableBlocks = sortBlocks(picture, roundedSize, blockSize, threshold);

figure, imshow(picture), title('Available Blocks');

hold on;

for i = 1:length(availableBlocks)

block = availableBlocks{i};

pos = block('position');

x = pos(2);

y = pos(1);

rectangle('Position', [x, y, blockSize, blockSize], 'EdgeColor', 'r', 'LineWidth', 1.5);

end

hold off;

end

A person standing in a field of flowers

AI-generated content may be incorrect.

### 1 - 3) Showing original picture and coded picture

figure('Position', [0 0 1000 400]);

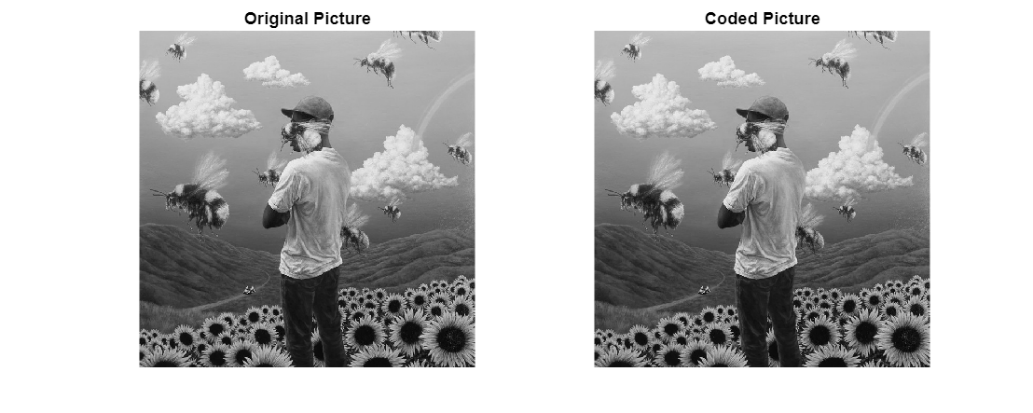
plot1 = subplot(1,2,1);

imshow(picture);

title('Original Picture');

plot2 = subplot(1,2,2);

imshow(codedPicture);

title('Coded Picture');

### 1 - 4) Decoding the picture

در این بخش تصویری که می‌خواهیم آن را رمزگشایی کنیم را از بخش قبلی گرفته و با blockSize و threshold ای که به آن می‌دهیم، پیام را از تابع decoding که در ادامه توضیح آن داده می‌شود، خروجی می‌گیریم و نمایش می‌دهیم.

blockSize = 20;

threshold = 100;

message = decoding(codedPicture, mapset, blockSize, threshold);

fprintf('Hidden message is : "%s"', message);

Hidden message is : "signal;"

این تابع با گرفتن باینری هر کاراکتر، آن را در مپ‌ست پیدا می‌کند و کاراکتر آن را خروجی می‌دهد.

function character = getCharacter(binary, mapset)

for char = mapset

if char{2} == binary

character = char{1};

break;

end

end

end

تابع decoding با گرفتن تصویر، سایز بلاک‌ها، ترشهولد و مپ‌ست، پیام پنهان‌شده در عکس را خروجی می‌دهد. الگوریتم آن مشابه تابع coding هست. یعنی دوباره با استفاده از تابع sortBlocks، بلاک‌هایی که می‌توانیم در آن پیام مخفی کنیم را پیدا می‌کند و حالا این سری بیت آخر هر پیکسل را می‌خواند و آن را ذخیره می‌کند. هر 5 پیکسل را با توجه به مپستی که در بخش اول داشتیم، یک کاراکتر در نظر می‌گیرد و درنهایت تا جایی که به ; نرسیده است ادامه‌ می‌دهد و در نهایت پیام رمزگشایی‌شده را خروجی می‌دهد.

function message = decoding(codedPicture, mapset, blockSize, threshold)

[height, width] = size(codedPicture);

roundedSize = [floor(height / blockSize) \* blockSize, floor(width / blockSize) \* blockSize];

codedBlocks = sortBlocks(codedPicture, roundedSize, blockSize, threshold);

numOfcodedBlocks = length(codedBlocks);

message = '';

binarizedChar = '';

character = '';

for i = 1:numOfcodedBlocks

block = codedBlocks{i};

pixels = block('pixels');

for j = 1:blockSize

for k = 1:blockSize

pixel = pixels(j, k);

codedPixel = dec2bin(bitand(pixel, 1));

binarizedChar = [binarizedChar, codedPixel];

if length(binarizedChar) >= 5

character = getCharacter(binarizedChar, mapset);

message = [message, character];

binarizedChar = '';

if character == ';'

break

end

end

end

if character == ';'

break

end

end

if character == ';'

break

end

end

end

### 1 - 5 ) If noise is unintentionally added to the image, can we still decode it?

بستگی دارد که این نویز به کدام بخش عکس اضافه می‌شود. اگر به بلاک‌هایی که با توجه به ترشهولد و اندازه‌ی مورد نظر ما انتخاب شده بودند، اضافه شوند می‌توانند پیام را تغییر بدهند و آن را خراب کنند اما اگر این نویز به جاهایی که انتخاب نشده بودند اضافه شود تاثیری ندارد.

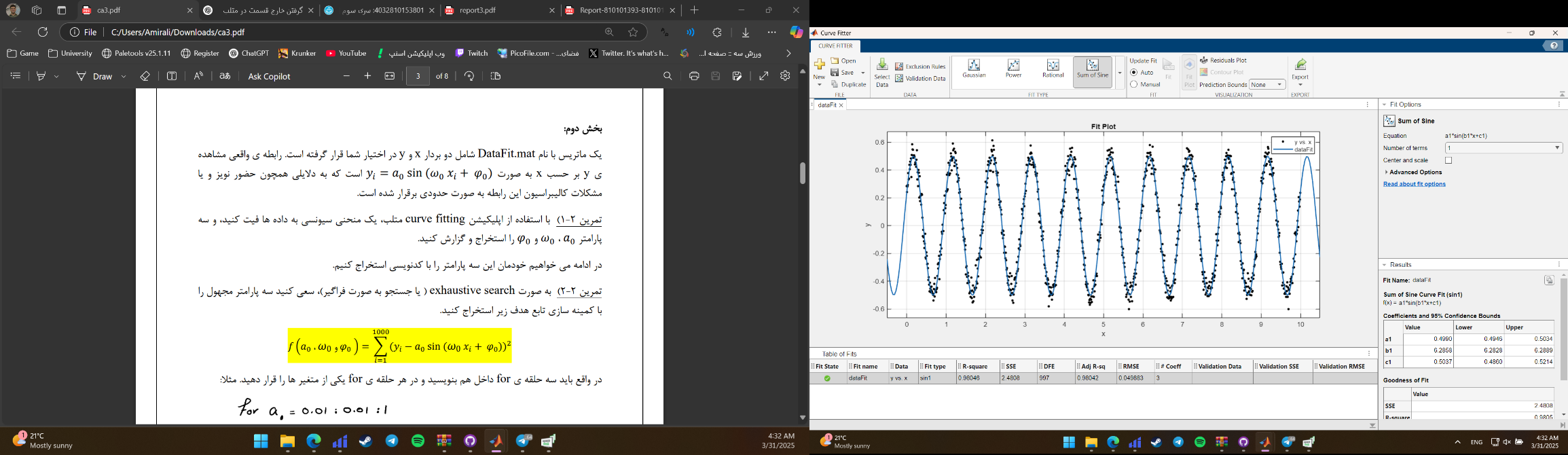
## Question 2

در این سوال هدف اصلی این است که بتوانیم رابطه‌ی واقعی دیتا‌هایی که داریم را به دست بیاوریم که به صورت (درواقع متغیر‌های آن) را به دست آوریم.

clc, clearvars, close all;

### 2 - 1)

همان‌طور که در تصویر مشخص است، مقادیر مورد نظر با استفاده از اپلیکیشن curve fitter به دست آمده است.



### 2 - 2)

در این بخش، با استفاده از 3 حلقه‌ای که در صورت پروژه‌ هم آمده بود، مقادیر ، و به دست می‌‌‌آید. پاسخ به دست آمده برای هرکدام تقریبا مشابه و نزدیک همان مقادیری است که در نرم افزار به دست آمده اما مشکل این روش این است که زمان بسیار زیادی طول می‌کشد تا انجام بشود چون که حلقه تودرتوهم داریم، 2 میلیون بار حلقه اجرا می‌شود که عدد زیادی است و از نظر زمانی مناسب نیست.

load("DataFit.mat");

minError = inf;

bestParameters = containers.Map;

tic;

for a0 = 0.01:0.01:1

for w0 = 0:pi/10:10\*pi

for phi0 = 0:pi/100:2\*pi

y\_pred = a0 \* sin(w0 \* x + phi0);

error = sum((y - y\_pred).^2);

if error < minError

minError = error;

bestParameters('a0') = a0;

bestParameters('w0') = w0;

bestParameters('phi0') = phi0;

end

end

end

end

toc;

Elapsed time is 23.148005 seconds.

fprintf('Best parameters: a0=%.4f, ω0=%.4f, φ0=%.4f\n', bestParameters('a0'), bestParameters('w0'), bestParameters('phi0'));

Best parameters: a0=0.5000, ω0=6.2832, φ0=0.5027

### 2 - 3)

تفاوت این بخش با بخش قبلی این است که به جای اینکه با step‌های کوچک در بازه‌ی مورد نظر بگردیم که از نظر زمانی خیلی طول می‌کشد، ابتدا با step بزرگ‌تر بازه مورد نظر را بگردیم ویک جواب کلی به دست بیاوریم، سپس در یک بازه‌ای به دور جواب کلی به دست آمده با step کوچک‌تر دنبال جواب دقیق‌تر بگردیم. این روش خیلی از نظر زمانی مناسب‌تر است و جواب‌هایی که می‌دهد هم مطلوب و مناسب است.

minError = inf;

bestParameters = containers.Map;

tic;

for a0 = 0.1:0.1:1

for w0 = 0:pi:10\*pi

for phi0 = 0:pi/10:2\*pi

y\_pred = a0 \* sin(w0 \* x + phi0);

error = sum((y - y\_pred).^2);

if error < minError

minError = error;

bestParameters('a0') = a0;

bestParameters('w0') = w0;

bestParameters('phi0') = phi0;

end

end

end

end

minError = inf;

finalParameters = containers.Map;

range\_a0 = max(bestParameters('a0')-0.05,0.01):0.01:min(bestParameters('a0')+0.05,1);

range\_w0 = max(bestParameters('w0')-pi/2,0):pi/10:min(bestParameters('w0')+pi/2,10\*pi);

range\_phi0 = max(bestParameters('phi0')-pi/2,0):pi/100:min(bestParameters('phi0')+pi/2,2\*pi);

for a0 = range\_a0

for w0 = range\_w0

for phi0 = range\_phi0

y\_pred = a0 \* sin(w0 \* x + phi0);

error = sum((y - y\_pred).^2);

if error < minError

minError = error;

finalParameters('a0') = a0;

finalParameters('w0') = w0;

finalParameters('phi0') = phi0;

end

end

end

end

toc;

Elapsed time is 0.069641 seconds.

fprintf('Best parameters: a0=%.4f, ω0=%.4f, φ0=%.4f\n', finalParameters('a0'), finalParameters('w0'), finalParameters('phi0'));

Best parameters: a0=0.5000, ω0=6.2832, φ0=0.5027

### 2 - 4)

در اینجا از روش گرادیان کاهشی مسئله را پیش می‌بریم که در واقع از فرمول به دست می‌ اید که هر سری با به دست آوردن گرادیان تابع و کم کردن آن از مقادیر، به مقادیر جدیدی می‌رسیم که در نهایت اگر اختلاف مقادیر جدید با مقادیر قبلی نزدیک به صفر شود، حلقه متوقف می‌شود و جواب نمایش داده می‌شود. این روش از نظر زمانی مناسب است اما فقط به ازای مقادیر اولیه مشخصی همگرا می شود.

tic

mu = 1e-6;

tol = 1e-5;

z = [0.5; 6; 0.47];

while true

a = z(1);

omega = z(2);

phi = z(3);

grad\_a = -2 \* sum((y - a.\*sin(omega.\*x + phi)) .\* sin(omega.\*x + phi));

grad\_omega = -2 \* sum((y - a.\*sin(omega.\*x + phi)) .\* a.\*x.\*cos(omega.\*x + phi));

grad\_phi = -2 \* sum((y - a.\*sin(omega.\*x + phi)) .\* a.\*cos(omega.\*x + phi));

gradient = [grad\_a; grad\_omega; grad\_phi];

z\_new = z - mu \* gradient;

if norm(z\_new - z) < tol || norm(gradient) < tol

break

end

z = z\_new;

end

toc;

Elapsed time is 0.118924 seconds.

fprintf('Best parameters: a0=%.4f, ω0=%.4f, φ0=%.4f\n', z(1), z(2), z(3));

Best parameters: a0=0.4890, ω0=6.2841, φ0=0.5149

### 2 - 5)

در اینجا 3 بار مقادیر تصادفی به تابع دادیم و همانطور که مشخص است، همگرایی نداریم و تابع هدف دارای مینیمم‌های محلی می‌باشد.

mu = 1e-6;

tol = 1e-5;

Zs = [ [0.5; 2\*pi; pi/4], [0.1; pi; pi], [1; 2; 1] ];

for z = Zs

fprintf('Initial Value: a0=%.4f, ω0=%.4f, φ0=%.4f\n', z(1), z(2), z(3));

while true

a = z(1);

omega = z(2);

phi = z(3);

grad\_a = -2 \* sum((y - a.\*sin(omega.\*x + phi)) .\* sin(omega.\*x + phi));

grad\_omega = -2 \* sum((y - a.\*sin(omega.\*x + phi)) .\* a.\*x.\*cos(omega.\*x + phi));

grad\_phi = -2 \* sum((y - a.\*sin(omega.\*x + phi)) .\* a.\*cos(omega.\*x + phi));

gradient = [grad\_a; grad\_omega; grad\_phi];

z\_new = z - mu \* gradient;

if norm(z\_new - z) < tol || norm(gradient) < tol

break

end

z = z\_new;

end

fprintf('Best parameters: a0=%.4f, ω0=%.4f, φ0=%.4f\n', z(1), z(2), z(3));

fprintf('----------------------------------------------------------------------------------------------------\n')

end

Initial Value: a0=0.5000, ω0=6.2832, φ0=0.7854

Best parameters: a0=0.4965, ω0=6.2605, φ0=0.6720

----------------------------------------------------------------------------------------------------

Initial Value: a0=0.1000, ω0=3.1416, φ0=3.1416

Best parameters: a0=0.0118, ω0=3.1603, φ0=3.1418

----------------------------------------------------------------------------------------------------

Initial Value: a0=1.0000, ω0=2.0000, φ0=1.0000

Best parameters: a0=0.0089, ω0=1.9680, φ0=1.0131

----------------------------------------------------------------------------------------------------

## 

## Question 3

clc, clearvars, close all;

### 3 - 1)

دقت‌ها در عکس مشخص است.

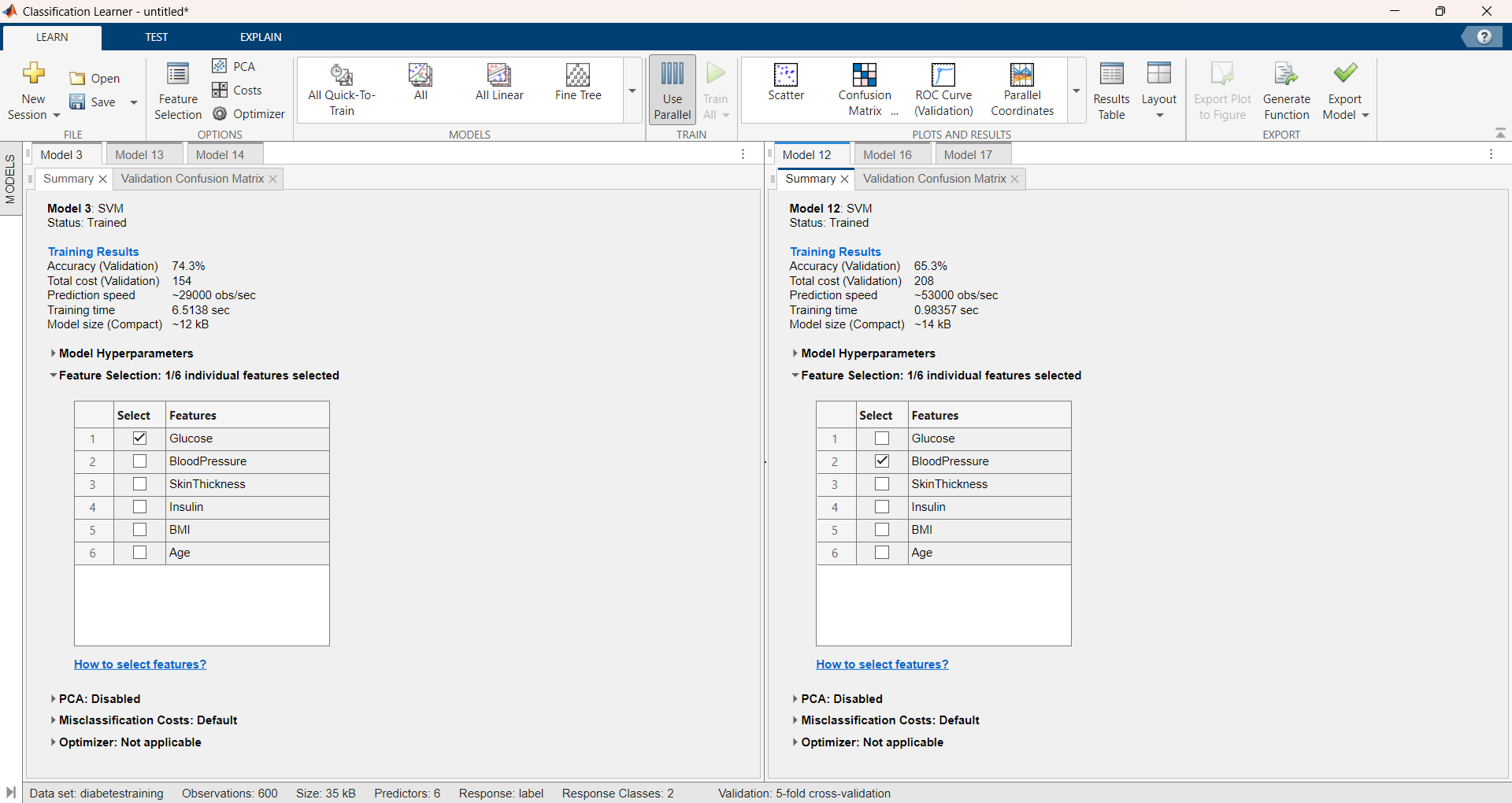
A screenshot of a computer

AI-generated content may be incorrect.

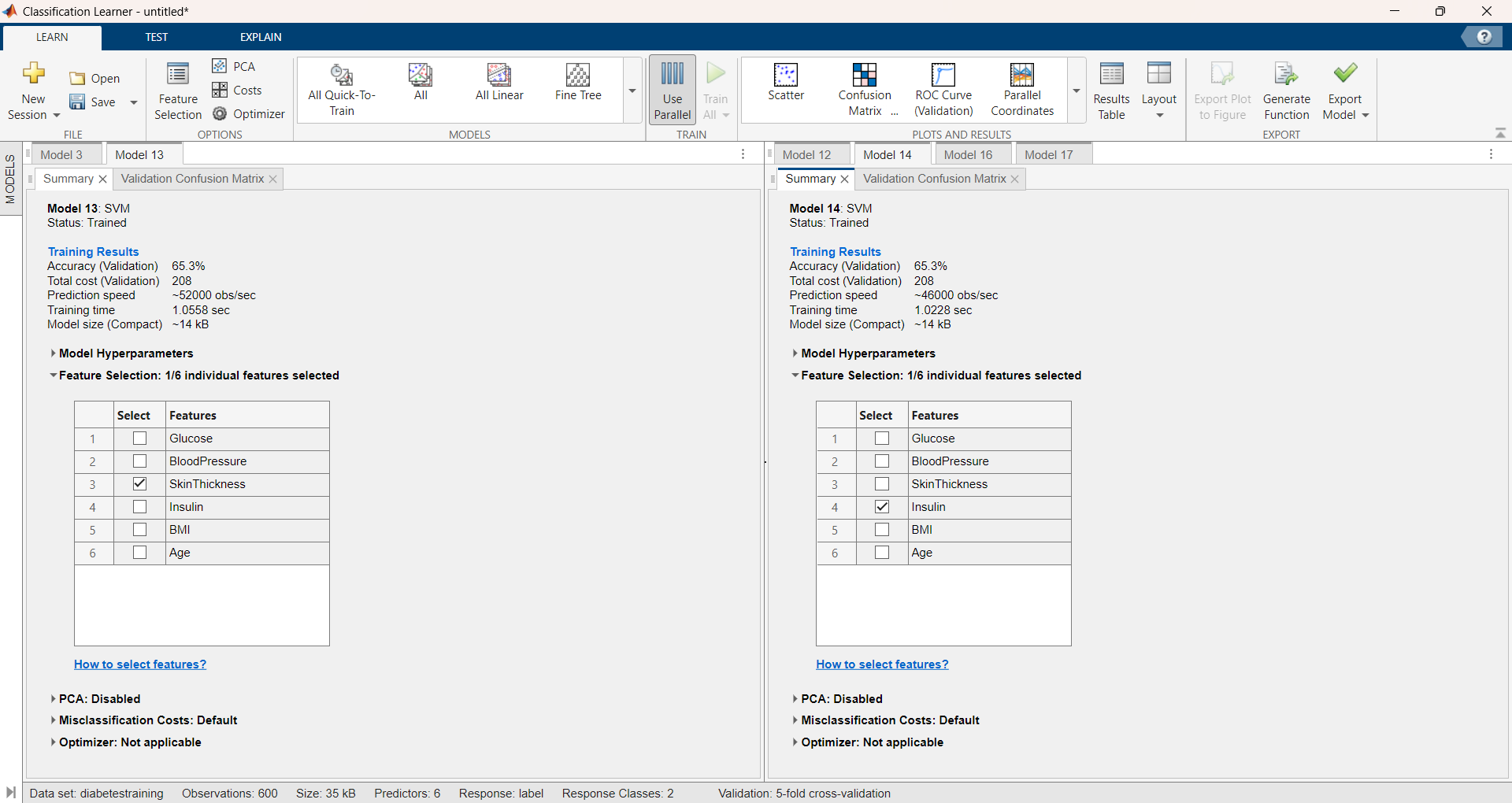


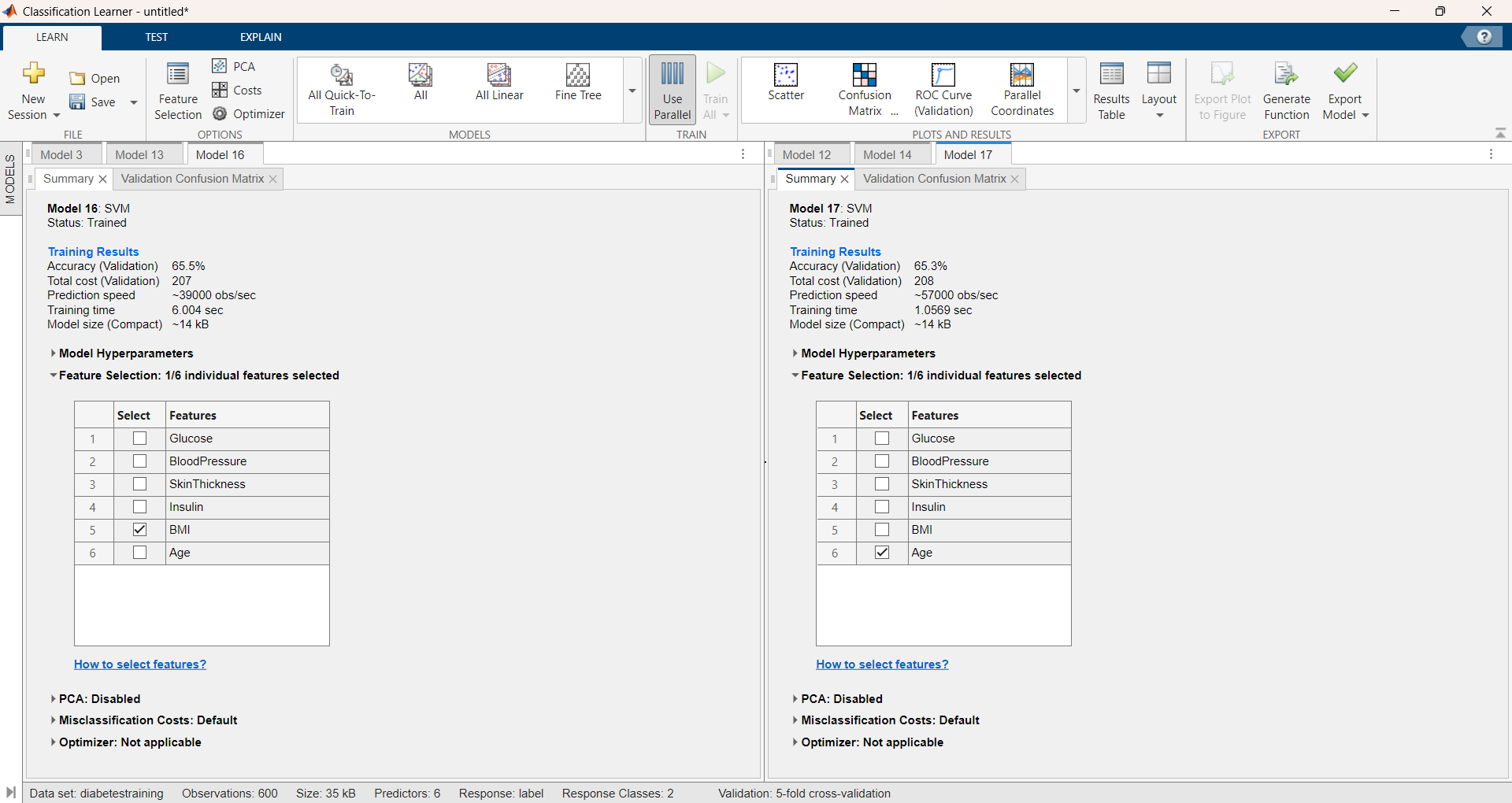
### 3 - 2)

همان‌طور که در تصاویر مشخص است بیشترین تاثیر را گلوکز و بعد از آن BMI دارد.











### 3 - 3)

load('TrainedModel.mat');

dataset = readtable('diabetes-training.csv');

labels = dataset(:, end);

features = dataset(:, 1 : end-1);

predictions = TrainedModel.predictFcn(features);

accuracy = mean(predictions == labels).\*100;

disp(accuracy);

**label**

**\_\_\_\_\_**

77.5

### 3 - 4)

dataset = readtable('diabetes-validation.csv');

labels = dataset(:, end);

features = dataset(:, 1 : end-1);

predictions = TrainedModel.predictFcn(features);

accuracy = mean(predictions == labels).\*100;

disp(accuracy);

**label**

**\_\_\_\_\_**

78

## Part 4

در این بخش، کانولوشن سیگنال‌های مورد نظر محاسبه می‌شود و با شکل‌های داخل صورت پروژه مقایسه می‌شود.

clc, clearvars, close all;

t = linspace(-10, 10, 1000);

a = (t >= 0 & t < 1) .\* t;

b = (t >= 0 & t < 1) .\* (1 - t);

c = (t >= 0 & t <= 1);

signals = {'a', 'b', 'c'};

data = {a, b, c};

figure('Position',[0, 0 1000, 200]);

for z = 1:3

subplot(1, 3, z);

xlim([-1, 3]);

plot(t, data{z}, 'LineWidth', 1.5);

title(signals{z});

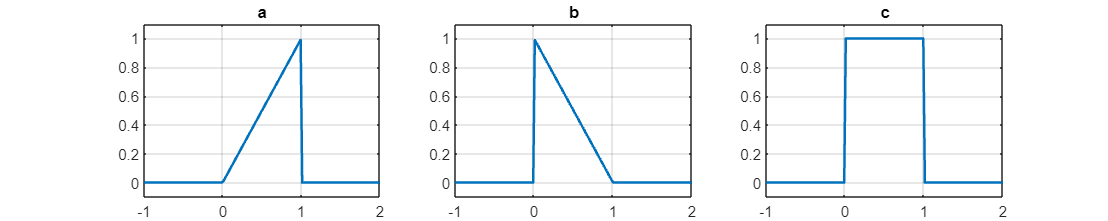
grid on;

xlim([-1, 2]);

ylim([-0.1,1.1])

end

نمودار هر 3 تابع:



figure('Position',[0, 0, 800, 400]);

idx = 1;

for z = 1:3

for j = 1:3

conv\_result = conv(data{z}, data{j}, 'same') \* (t(2) - t(1));

subplot(3, 3, idx);

plot(t, conv\_result, 'LineWidth', 1.5);

title([signals{z}, ' \* ', signals{j}]);

xlim([-0.5,3]);

ylim([-0.1, 1.1]);

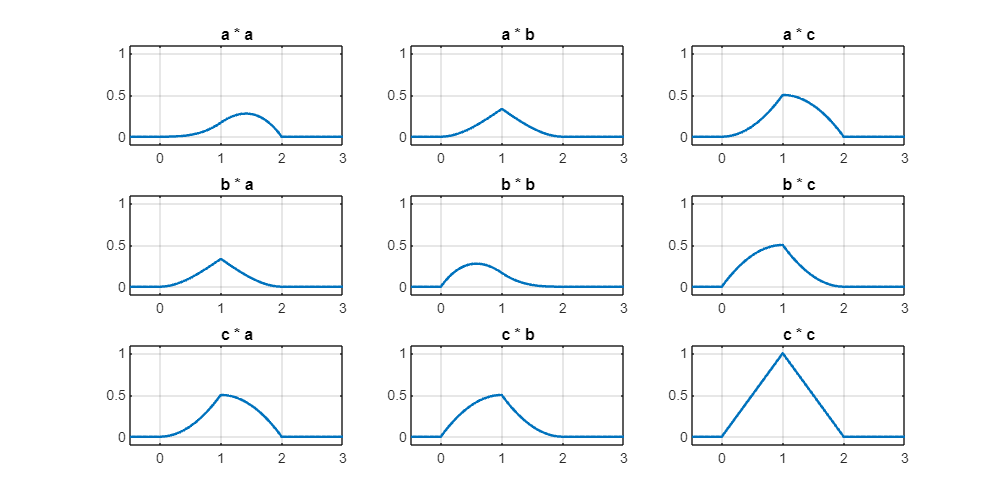
grid on;

idx = idx + 1;

end

end

کانولوشن‌های به دست‌ آمده:



با توجه به صورت پروژه و نتایج به دست آمده، کانولوشن‌های داخل صورت پروژه به شرح زیر است:

A graph of a function

AI-generated content may be incorrect.

