گزارش تمرین کامپیوتری 3

سیگنال ها و سیستم ها

دکتر اخوان

پریسا محمدی 810101509

سامان دوچی طوسی 810101420

سوال 1)

بخش 1

```
function charDataset = generateCharacterMapping()
    % Define a string with all the characters
    characterList = ['a':'z', ' ', '.', ',', '!', '"', ';'];

    % Initialize an empty cell array
    charDataset = cell(2, length(characterList));

    % Complete the first row with characters from the list
    for idx = 1:length(characterList)
        charDataset{1, idx} = characterList(idx);
    end

    % Complete the second row with 5-bit binary codes for each character
    for idx = 1:32
        if idx <= length(characterList)
            charDataset{2, idx} = dec2bin(idx-1, 5); % Convert decimal index to 5-bit binary string
        else
            break; % Exit loop if we exceed the character list length
        end
    end
end
```

تصویر 1 _ مپ کردن کاراکتر های مد نظر به 5 بیت باینری

تابع generateCharacterMapping هدف اش ساخت یک مجموعه داده (dataset) از حروف و کاراکترهای مشخصی است و به هر کاراکتر یک کد باینری ۵ بیتی اختصاص می‌دهد. این تابع به صورت زیر عمل می‌کند:

لیستی از کاراکترهای مورد نظر، شامل حروف کوچک الفبای انگلیسی، فضای خالی(space) ، نقطه، کاما، علامت تعجب، سمی کالن و دابل کوتیشن است را در متغیر characterList ذخیره می‌کند یک آرایه سلولی به نام charDataset با دو ردیف و به اندازه‌ی تعداد کاراکترهای موجود در characterList ایجاد

می‌کنیم. در حلقه‌ی اول، کاراکترها را در ردیف اول این آرایه سلولی ذخیره می‌کنیم .در حلقه‌ی دوم، برای هر کاراکتر یک کد باینری ۵ بیتی تولید کرده و آن را در ردیف دوم آرایه قرار می‌دهیم. این کار توسط تابع dec2bin انجام می دهیم که عدد ده‌دهی (index) را به باینری ۵ بیتی تبدیل می‌کند .در این تابع در نهایت مجموعه‌ای از کاراکترها و کدهای باینری مربوط به آن‌ها را به صورت یک آرایه سلولی خروجی می‌دهیم.

بخش 2

```
function textMsg = binaryToText(binarySeq, mapping)
    textMsg = '';
    % Convert binary sequence in chunks of 5 bits to characters
    for k = 1:5:length(binarySeq) - 4
        charBits = binarySeq(k:k+4);
        % Find character matching the 5-bit binary chunk
        for i = 1:length(mapping(2, :))
            if strcmp(mapping{2, i}, charBits)
                decodedChar = mapping{1, i};
                break;
            end
        end
        if exist('decodedChar', 'var') && decodedChar == ';'
            break;
        elseif exist('decodedChar', 'var')
            textMsg = [textMsg decodedChar];
        end
    end
end
```

تصویر 2_ پیدا کردن کاراکتر متناسب با 5 بیت باینری

تابع binaryToText هدف اش تبدیل دنباله‌ای باینری به متن قابل خواندن است. مراحل عملکرد این تابع به شرح زیر است:

تابع یک رشته خالی به نام textMsg برای ذخیره‌ی پیام نهایی ایجاد می‌کند. تابع دنباله باینری را در اسلایس های ۵ بیتی پردازش می‌کند. در هر تکرار، ۵ بیت از دنباله خوانده می‌شود و در متغیر charBits ذخیره می‌شود. با استفاده از حلقه‌ای دیگر، تابع به جستجو در نقشه mapping می‌پردازد تا کاراکتری را که با charBits تطابق دارد پیدا کند. اگر کاراکتر مطابق پیدا شود، به متغیر decodedChar اختصاص داده می‌شود. اگر کاراکتر پیدا شده برابر با `';` باشد، تابع به پردازش پایان می‌رسد. در غیر این صورت، کاراکتر به

textMsgافزوده می‌شود .این تابع در نهایت دنباله‌ای از کاراکترها را که معادل دنباله باینری ورودی است، به صورت یک متن باز می‌گرداند.منطق این تابع به گونه‌ای طراحی شده است که با ساختار دنباله باینری که در توابع قبلی تولید شده، همخوانی داشته باشد و پیام اصلی را به درستی بازیابی کند.

بخش 3

```
function encodedImage = encoder(mapping, grayImg, textMsg)

    windwDim = [5, 5];
    varThreshold = 70;

    % Convert the text message to a binary sequence using the mapping data
    binaryMessage = convertToBinary(textMsg, mapping);

    % Locate the semicolon index in the mapping data and append its binary representation to binaryMessage
    semicolonIdx = find([mapping{1, :}] == ';', 1);
    binaryMessage = [binaryMessage mapping{2, semicolonIdx}];

    % Duplicate the image to preserve original data for encoding
    encodedImage = grayImg;
    [imgRows, imgCols] = size(grayImg);
    bitIdx = 1;
```

```
    % Traverse the image in blocks defined by windwDim
    for m = 1:windwDim(1):imgRows-4
        for n = 1:windwDim(2):imgCols-4
            imgBlock = grayImg(m:m+4, n:n+4);
            % Check if block's variance surpasses the set threshold
            if var(double(imgBlock(:))) > varThreshold
                for row = m:m+4
                    for col = n:n+4
                        if bitIdx > length(binaryMessage), break; end
                        % Modify LSB of pixel to match current message bit
                        encodedImage(row, col) = bitset(encodedImage(row, col), 1, binaryMessage(bitIdx) - '0');
                        bitIdx = bitIdx + 1;
                    end
                end
            end
        end
    end

    if bitIdx <= length(binaryMessage)
        error('Message exceeds available space in the image.');
    end
end
```

```
function binStr = convertToBinary(text, mapping)
    binStr = '';
    for char = text
        idx = find([mapping{1, :}] == char, 1);
        if ~isempty(idx)
            % Append binary equivalent of character to binStr
            binStr = [binStr mapping{2, idx}];
        else
            error(['Character ', char, ' not present in mapping dataset.']);
        end
    end
end
```

تصویر 3_ پیاده سازی encoder

تابع encoder با استفاده از یک مپ کاراکتر به باینری و روش هایی معین، پیامی متنی را به تصویر خاکستری (encode)می‌کند. مراحل آن به شرح زیر است:

تابع کمکیconvertToBinary ، هر کاراکتر از پیام متنی را با استفاده از mapping به دنباله‌ای باینری تبدیل می‌کند .کد باینری کاراکتر `;` (به عنوان نشانه‌ی پایان پیام) به انتهای دنباله باینری پیام اضافه می‌شود .تابع encoder تصویر را به بلاک‌هایی با ابعاد 5 در 5 تقسیم می‌کند و هر بلاک را ارزیابی می‌کند تا ببیند آیا واریانس آن از آستانه مشخصی بیشتر است .در بلاک‌هایی که واریانس آن‌ها بیشتر از آستانه تعیین‌شده است، بیت‌های پیام در بیت کم‌اهمیت (LSB) هر پیکسل از بلاک ذخیره می‌شوند. این کار از طریق تاب bitset انجام می‌شود.

در نهایت، تابع چک می‌کند که آیا تمام بیت‌های پیام در تصویر رمزگذاری شده‌اند، در غیر این صورت، خطا می‌دهد که فضا کافی نیست.

بخش 4

```
function decodedMsg = decoder(encodedImg, mapping)
    % Set window dimensions and variance threshold within the function
    windwDim = [5, 5];
    varThreshold = 70;

    binMsg = '';
    [imgRows, imgCols] = size(encodedImg);

    % Loop over image blocks based on window dimensions
    for m = 1:windwDim(1):imgRows-4
        for n = 1:windwDim(2):imgCols-4
            imgBlock = encodedImg(m:m+4, n:n+4);
            if var(double(imgBlock(:))) > varThreshold
                for row = m:m+4
                    for col = n:n+4
                        binMsg = [binMsg, num2str(bitget(encodedImg(row, col), 1))];
                    end
                end
            end
        end
    end

    % Convert binary message back to text
    decodedMsg = binaryToText(binMsg, mapping);
end
```

تصویر 4_ پیاده سازی decoder

تابع decoder پیام رمزگذاری‌شده را از تصویر بازیابی می‌کند. این تابع منطق مشابهی با تابع encoderدارد تا دقیقا پیام باینری به صورت درست استخراج شود. مراحل اش به این صورت است:

ابعاد پنجره و آستانه واریانس به طور ثابت داخل تابع تعریف شده‌اند (همانند تابع encoder)

تابع decoder تصویر را تقسیم‌بندی کرده به بلوک های 5 در 5 . اگر واریانس یک بلوک از آستانه مشخص بیشتر باشد، بیت کم‌اهمیت (LSB) هر پیکسل در بلوک خوانده می‌شود و به رشته باینری پیام (binMsg) اضافه می‌شود. در پایان، رشته باینری با استفاده از تابع binaryToTextبه متن تبدیل می‌شود و پیام اصلی استخراج‌شده (decodedMsg) به دست می‌آید.

ساختار و منطق این decoder به‌گونه‌ای است که دقیقا با encoder همخوانی داشته باشد. به عبارت دیگر، همان بلاک‌های تصویر که با استفاده از واریانس بالا در encoder انتخاب شده‌اند، در اینجا نیز مورد بررسی قرار می‌گیرند تا بیت‌های پیام به درستی استخراج شوند.

بخش 5

```
clc;
clear;
close all;


% Select an image file using a file dialog
[file, path] = uigetfile({'*.jpg;*.bmp;*.png;*.tif'}, 'Choose an image');
if isequal(file, 0)
    disp('User selected Cancel');
    return;
end

% Read the selected image
imagePath = fullfile(path, file);
rgbImage = imread(imagePath);

% Convert to grayscale if the image is in RGB
if size(rgbImage, 3) == 3
    grayImage = rgb2gray(rgbImage);
else
    grayImage = rgbImage;
end

% Define character dataset and message
mapping = generateCharacterMapping();
textMessage = 'signal;';

% Encode and decode the message
encodedImg = encoder(mapping, grayImage, textMessage);
decodedText = decoder(encodedImg, mapping);


% Display the decoded message
disp('Decoded Message:');
disp(decodedText);

% Display the original grayscale and encoded images side by side
figure;
subplot(1,2,1);
if size(rgbImage, 3) == 3
    imshow(grayImage);
    title('Grayscale Image Before Encoding');
else
    imshow(rgbImage);
    title('Image Before Encoding');
end
axOriginal = gca;

subplot(1,2,2);
imshow(encodedImg);
title('Image After Encoding');
axEncoded = gca;


% Save the encoded image in the same directory as the input image
outputPath = fullfile(path, 'encoded_image.png');
imwrite(encodedImg, outputPath);
disp(['Encoded image saved as: ', outputPath]);
```

تصویر 5_ چک کردن مراحل encode و decode با یک عکس ورودی دلخواه

در این مرحله هدف، رمزگذاری یک پیام درون یک تصویر و سپس بازیابی آن پیام است. مراحل کار به این صورت است:

کاربر با استفاده از یک پرامپت برای دریافت فایل، یک تصویر را انتخاب می‌کند. اگر کاربر دکمه Cancelرا بزند، برنامه متوقف می‌شود. اگر تصویر رنگی RGBباشد، به تصویر خاکستری تبدیل می‌شود تا برای رمزگذاری آماده شود.

تابع generateCharacterMapping برای تولید مجموعه‌ای از کاراکترها و معادل باینری آن‌ها فراخوانی می‌شود. سپس پیام متناظر با کاراکترها (در اینجا signal;) تعریف می‌شود.

با استفاده از تابع encoder، پیام در تصویر خاکستری رمزگذاری می‌شود و سپس با استفاده از تابع decoder، پیام بازیابی می‌شود. پیام بازیابی شده درcommand windowنمایش داده می‌شود.

تصویر خاکستری اصلی (یا تصویر رنگی اصلی در صورت عدم تبدیل به خاکستری) و تصویر پس از رمزگذاری در یک پنجره نمایش داده می‌شوند.

در نهایت، تصویر رمزگذاری‌شده در همان دایرکتوری که تصویر اصلی در آن قرار دارد، با نام encoded_image.pngذخیره می‌شود.

تصویر5.1 – عکس خروجی حاصل از اجرای کد

```
Command Window
  Decoded Message:
  signal
  Encoded image saved as: C:\Users\EMTOO\Desktop\UT_HW\Signals_&_Systems\CA\3\CA2\encoded_image.png
fx >>
```

تصویر 5.2 – متن دیکود شده داخل تصویر

سوال 2)

```
close all;
clc;clear;
% Reading Images
PCB=imread('PCB.jpg');
IC=imread('IC.png');
PCB=rgb2gray(PCB);
IC=rgb2gray(IC);

figure
imshow(PCB)
hold on


function findMatches(pcb, ic, edgeColor)
    [rowpcb, colpcb] = size(pcb);
    [rowic, colic] = size(ic);
    for i = 1:(rowpcb - rowic)
        for j = 1:(colpcb - colic)
            a = double(ic);
            b = double(pcb(i:(i + rowic - 1), j:(j + colic - 1)));
            makhraj = sqrt(sum(sum(a .* a)) * sum(sum(b .* b)));
            soorat = sum(sum(a .* b));
            coff = soorat / makhraj;
            if coff >= 0.9
                rectangle('Position', [j, i, colic, rowic], 'EdgeColor', edgeColor, 'LineWidth', 3);
            end
        end
    end
end

% Calling function
findMatches(PCB, IC, 'r');
IC=imrotate(IC,180);
findMatches(PCB,IC, 'g');
```

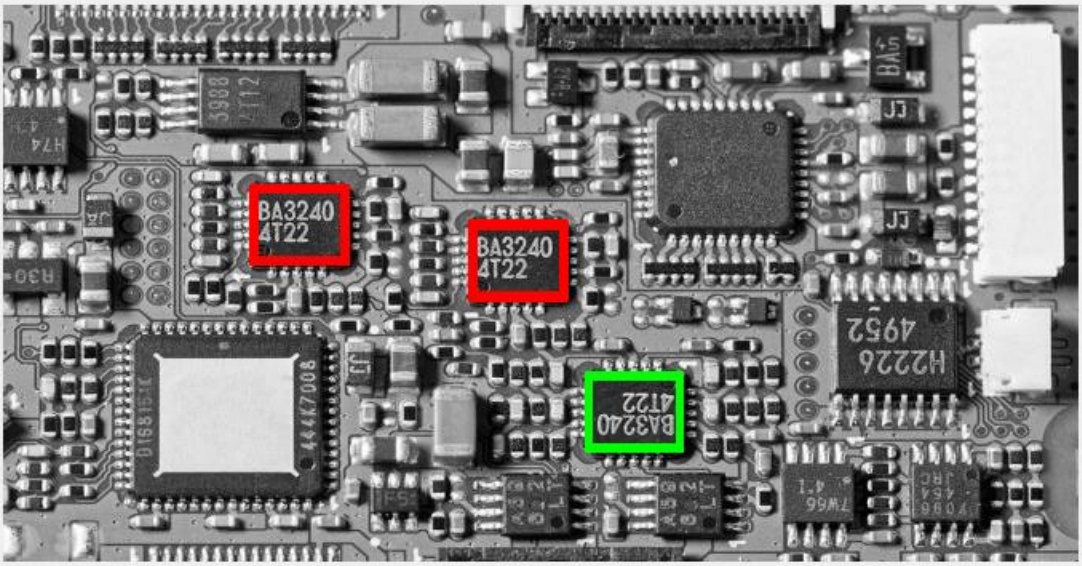
تصویر6 _کد نوشته شده برای پیدا کردن IC

استراتژی حل مسئله: در ابتدا دو تصویر را میخوانیم و آن ها را خاکستری میکنیم. سپس از تابعی که توضیحات نوشته شده است IC آن در قسمت زیر و بار دیگر تصویر برعکس شده آن را به عنوان ورودی میدهیم. استفاده میکنیم و یکبار تصویر

توضیحات تابع:

تصاویر ای سی ها به عنوان ورودی به تابع داده شده است. و تعداد ردیف ها و ستون ها را داخل یک آرایه ذخیره میکنیم.

سپس با استفاده از دو حلقه تصویر ای سی را در طول هر سطر و ستون حرکت میدهیم. سپس کورلیشن را با توجه به فرمول گفته شده در صورت سوال پیاده سازی میکنیم. و صورت و مخرج را جداگانه محاسبه میکنیم و اگر مقدار حاصل از کسر بیشتر از 0.9 را تشخیص میدهیم و دور آن داخل تصویر بورد مستطیل با توجه به بود ICرنگ گرفته شده به عنوان ورودی میکشیم.

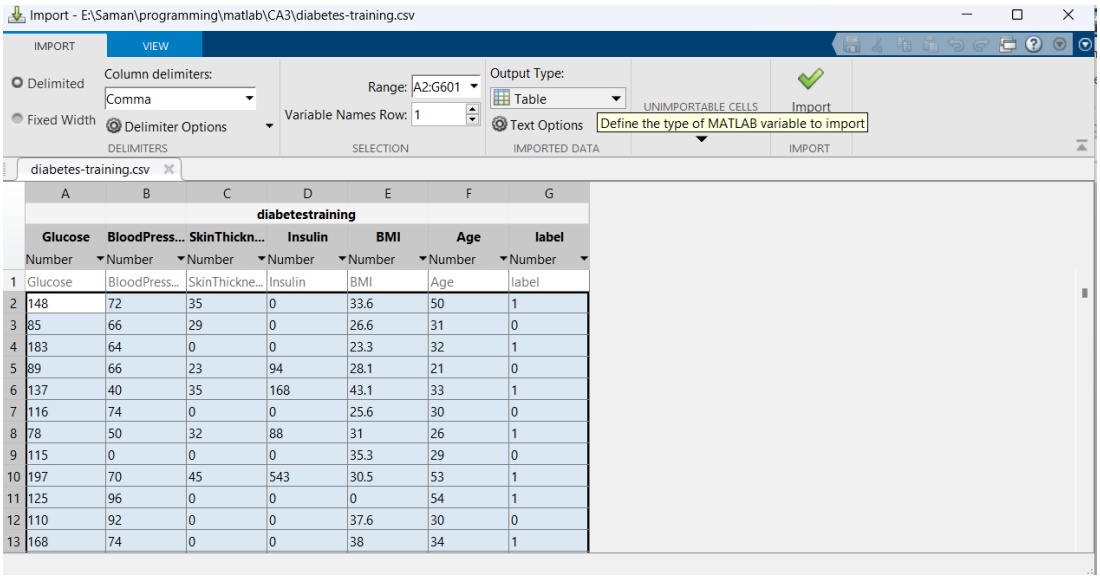


تصویر 7_ خروجی حاصل از اجرای کد بالا

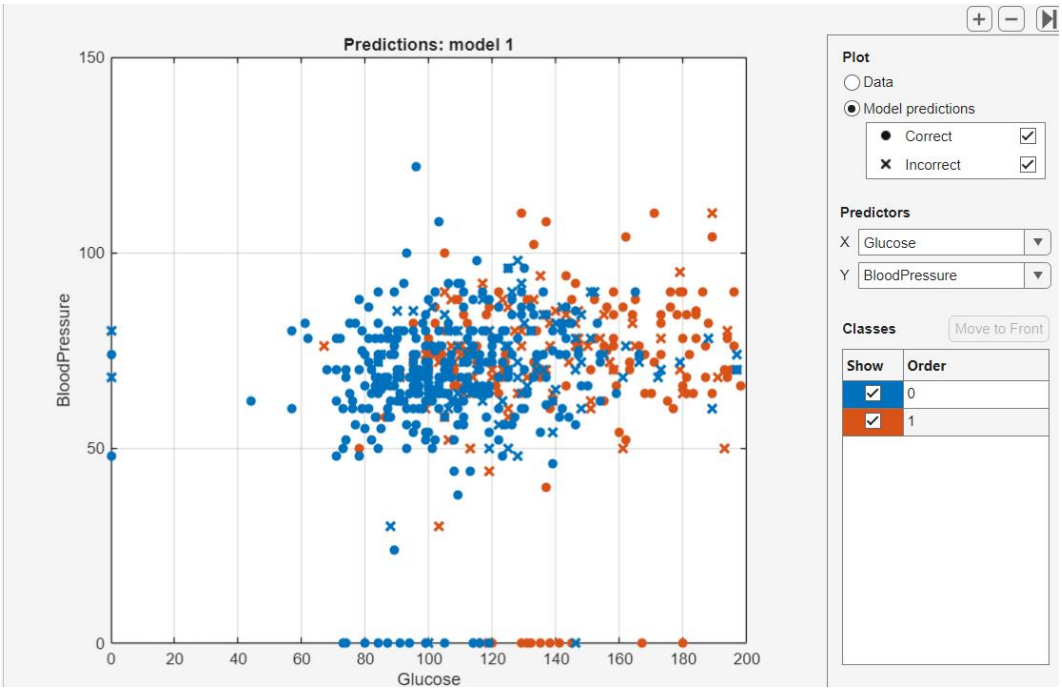
ای سی های معکوس شده را با رنگ سبز و ای سی های غیر معکوس را با رنگ قرمز مشخص کردیم.

سوال 3)

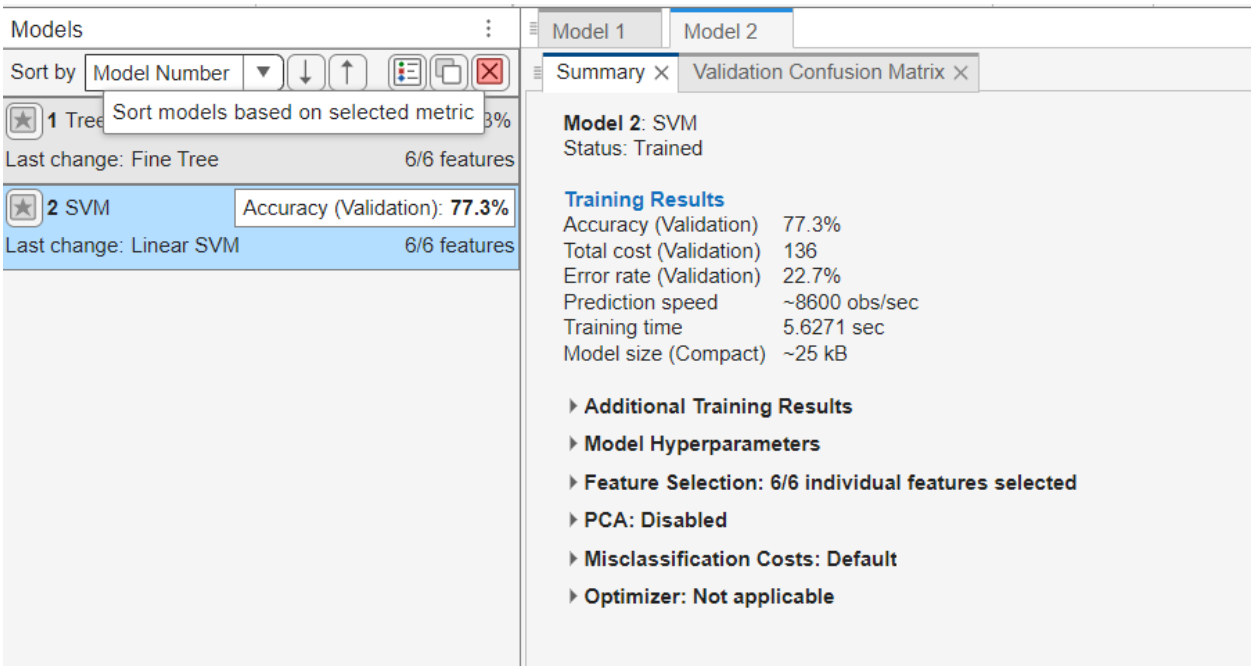
بخش1



تصویر 8 _ ایمپورت کردن اطلاعات داخل فایل ***csv***



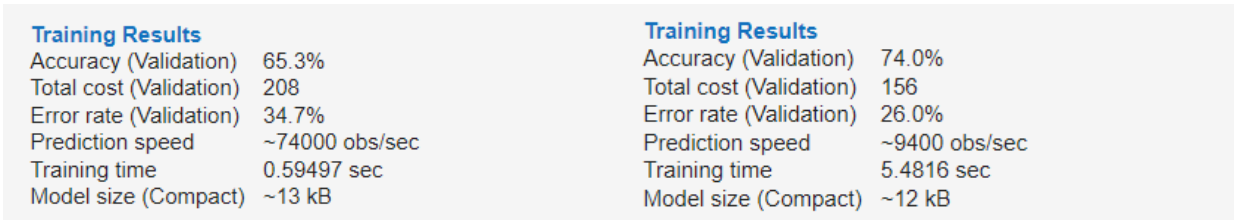
تصویر 9_نمودار اسکتر برای دو متغیر دلخواه



تصویر10_ترین کردن مدل Svm

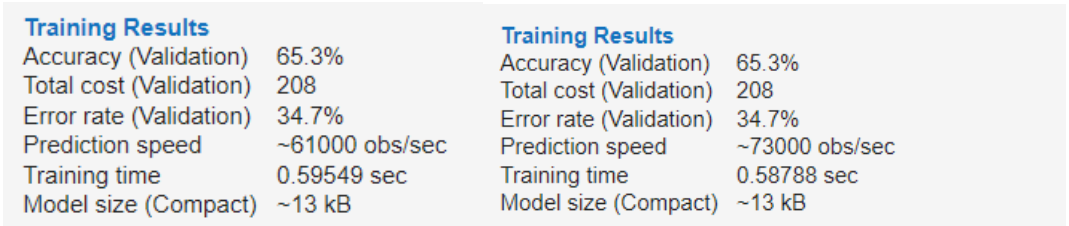
در تصویر بالا کار ها را به ترتیب گزارش انجام میدهیم و مدلمان را ترین میکنیم، همان طور که در تصویر مشخص است، مدلمان به دقت 77.3 درصد رسیده است.

بخش 2



تصویر 11_ترین با گلوکز

تصویر 12_ترین با فشار خون



تصویر 13 _ترین با ضخامت پوست تصویر 14 _ترین با انسولین

Training Results	
Accuracy (Validation)	65.3%
Total cost (Validation)	208
Error rate (Validation)	34.7%
Prediction speed	~72000 obs/sec
Training time	0.61743 sec
Model size (Compact)	~13 kB

Training Results	
Accuracy (Validation)	65.0%
Total cost (Validation)	210
Error rate (Validation)	35.0%
Prediction speed	~81000 obs/sec
Training time	2.8028 sec
Model size (Compact)	~13 kB

تصویر 15 _ترین با BMI تصویر 16 _ترین با سن

با مقایسه اعداد موجود در 6 تصویر قبل پی می بریم که دیتای گلوکز دارای بالا ترین ارتباط است.

بخش 3

```
clear;
% Loading Model
load('trainedModel.mat')

% Testing model
predictions = trainedModel.predictFcn(diabetestraining);
true_labels=diabetestraining(:,7);
array=table2array(true_labels);
counter=0;
for i=1:600
    if predictions(i)==array(i)
        counter=counter+1;
    end
end
accu_train=(counter/600)*100;
disp(accu_train);
```

تصویر 17 _بررسی عملکرد مدل

توضیحات کد: ابتدا مدل را لود میکنیم.

سپس برای همه داده مقدار حقیقیشان را با مقدار پیش بینی شده مقایسه میکنیم. و بررسی میکنیم چه تعداد به درستی پیشبینی شده اند.

و بعد درصد دیتا درست نمایش داده شده را نشان میدهیم که با عدد به دست امده در قسمت 3-1 یکسان هست.

```
>> p3
   77.5000
```

تصویر 18_دقت حاصل از مدل

```
% Testing model with diabetes-validation
predictions = trainedModel.predictFcn(diabetesvalidation);
true_labels=diabetesvalidation(:,7);
array=table2array(true_labels);
counter=0;
for i=1:100
    if predictions(i)==array(i)
        counter=counter+1;
    end
end
accu_train=(counter/100)*100;

disp(accu_train);
```

تصویر 20_کد بررسی کارکرد داده ولیدیشن

```
Command Window
>> p3
   77.5000

    78
```

تصویر 21_دقت حاصل از اجرای کد تصویر پیشین

دقت برای داده ها اصلی که با آن مدلمان ترین شد، برابر با 77.5 و برای داده های ولیدیشن برابر با 78 می باشد.