به نام خدا

کارگاه برنامه نویسی متلب

پروژه

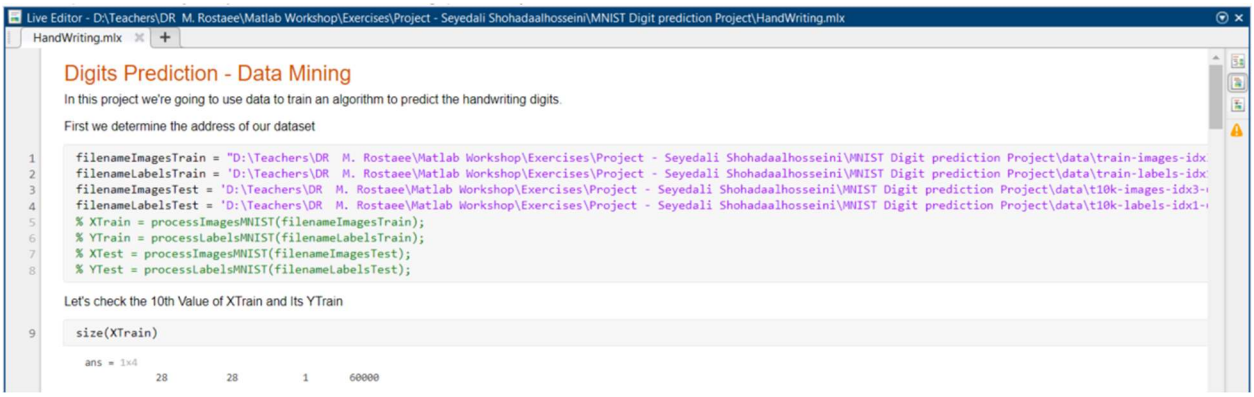
استاد: دکتر میثم روستایی

دانشجو: سید علی شهداالحسینی

بهار ۱۴۰۱

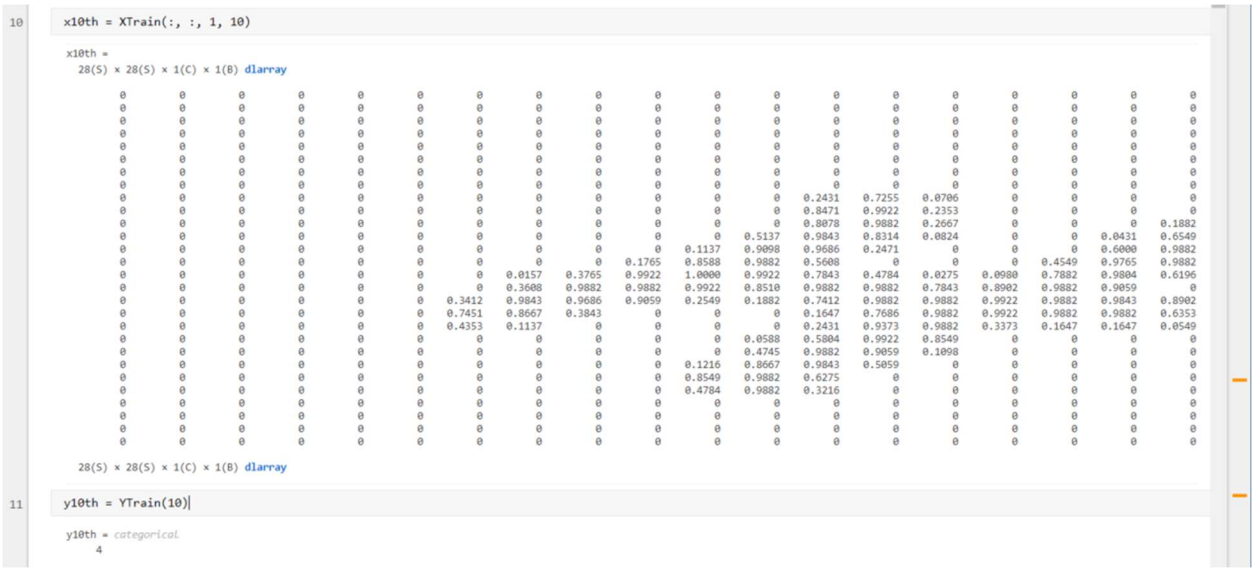
پروژه پیاده سازی مدل پیشبینی کننده اعداد به کمک دیتاست MNIST

در ابتدا dataset مورد نیاز این پروژه را دانلود از سایت http://yann.lecun.com/exdb/mnist دانلود کردیم و در دایرکتوری جاری خود قرار دادیم. قطعه کد نوشته شده برای پیاده سازی این پروژه را در ادامه مشاهده خواهید نمود.



در خطوط ۱ تا ۸ سعی کردیم تا دیتاست های دانلود شده خود را از دایرکتوری مورد نظر بخوانیم.

در خط ۹ به کمک متد size ابعاد دیتاست خود را بدست آوردیم. این دیتاست دارای ابعاد 28*28*1*60,000 میباید که به این معنا میباشد که در یک ردیف ۶۰هزار داده(تصویر) وجود دارد که ابعاد هر یک از این ها ۲۸*۲۸ میباشد.



در خط ۱۰ام سعی شد تا تمامی پیکسل های تصویر دهم را نمایش دهیم.(لازم به ذکر است که در تصویر تمامی ۲۸ در ۲۸ پیکسل نیفتاده است.) اگر توجه کنید مشاهده میکنید که این داده ها از نوع داده ای dlarray میباشند.

در خط ۱۱ام از روی داده های خروجی Train مقدار عددی(category) داده شماره ۱۰ام را بدست آوردیم. به عبارتی ماتریسی که خط ۱۰ام به ما نشان داده است برابر مقدار ۴ میباشد.

```
We convert these values to matrix to show
12  x10th = extractdata(x10th);
13
14  figure(1);
15  imshow(x10th);

16  y10th
    y10th = categorical
        4
```

حال برای اینکه بتوانیم این داده را نمایش دهیم و اطمینان حاصل نماییم، ابتدا باید نوع این داده را از dlarray به نوعی دیگر تبدیل کنیم که در اینجا ما به کمک متد extractdata آن را به نوع double تبدیل کردیم تا بتوانیم مقدار مورد نظر را نمایش دهیم.

برای نمایش این مقدار ابتدا در خط ۱۴ام یک پنجره ای برای نمایش این تصویر با مقدار عددی ۱ که به مانند ID این پنجره میباشد ایجاد کردیم و سپس در خط۱۵ام از دستور imshow استفاده کردیم تا داده مورد نظر خود را در پنجره شماره یکی که برای آن در نظر گرفتیم نمایش دهیم. که همانطور که مشاهده میشود داده مورد نظر ما مطابق چیزی که انتظار داشتیم برابر با مقدار ۴ بود.

```
Let's show a couple
16  figure(2);
17
18  subplot(1, 3, 1);
19  data_dlarray = XTrain(:, :, 1, 1);
20  extractedData = extractdata(data_dlarray);
21  imshow(extractedData);
22
23  subplot(1, 3, 2);
24  data_dlarray = XTrain(:, :, 1, 2);
25  extractedData = extractdata(data_dlarray);
26  imshow(extractedData);
27
28  subplot(1, 3, 3);
29  data_dlarray = XTrain(:, :, 1, 3);
30  extractedData = extractdata(data_dlarray);
31  imshow(extractedData);
```

همچنین در ادامه نیز سعی شد تا ۳ تصویر را همزمان بر روی یک window قرار دهیم که در ابتدا ما پنجره شماره ۲ خود را ایجاد کردیم و سپس هربار به کمک دستور subplot مشخص می‌کنیم که تصویر مورد نظر ما در کدام قسمت نمایش داده شود. برای مشخص کردن مکان تصویر در ورودی subplot در آرگومان اول ردیفی که میخواهیم تصویر در آن قرار بگیرد را مشخص کردیم، در آرگومان دوم تعداد ستون های آن ردیف را ذکر کردیم و در آرگومان سوم شماره ستونی که میخواستیم تصویر در آن نمایش داده شود را ذکر کردیم. این کار را برای هر سه تصویر انجام دادیم و همچنین در ادامه اش مطابق آنچه از قبل انجام دادیم، ابتدا داده ای که میخواستیم نمایش دهیم را انتخاب کردیم، سپس نوع آن داده را از dlarray به Double تغییر دادیم و نهایتا آن را imshow کردیم.

```
32  YTrain(1)

    ans = categorical
        5

33  YTrain(2)

    ans = categorical
        0

34  YTrain(3)

    ans = categorical
        4

35  XTrain_Matrix = extractdata(XTrain);
36  class(XTrain_Matrix)

    ans = 'double'
```

در اینجا نیز در خطوط 32 تا 34 این سه عددی که در قسمت قبل نمایش دادیم را از دیتاست YTrain که نشان دهنده دسته بندی داده های XTrain ما میباشد دریافت کردیم و نمایش دادیم و از صحت خروجی اطمینان حاصل کردیم.

همچنین در خطوط ۳۵ و ۳۶ تمامی داده های train خود را به نوع عددی double تبدیل کردیم تا بتوانیم آنها را نمایش دهیم.

در پردازش تصاویر پیکسل های یک تصویر درواقع feature ها ما هستند که از آنها استفاده میشود تا الگوریتم آموزش ببیند که مثلا با داشتن فلان پیکسل ها دسته بندی فلان را داریم. در اینجا از آنجا که تصویر ما ۲۸ در ۲۸ میباشد پس ما ۷۸۴ فیچر داریم پس در مجموع ما ۶۰هزار داده داریم که هر داده ۷۸۴ فیچر دارد که این ۷۸۴ فیچر به ما category مشخص شده اش را میدهد که این category در دیتاست YTrain مشخص شده است.

پس ما میتوانیم یک ستون دیگر را به عنوان ستونی category به انتها این ۷۸۴ ستون که فیچر های ما هستند اضافه کنیم تا بتوانیم در مراحل آموزش از این ستون استفاده کنیم. برای انجام این کار ها قطعه کد زیر نوشته شده است.

```
37  Reshaped_Matrix = reshape(XTrain_Matrix, [60000, 28*28*1])

    Reshaped_Matrix = 60000×784
         0    0    0    0    0    0    0    0    0    0    0    0    0.5176    0    0    0    0.7451 ...
         0    0    0    0    0    0    0    0    0    0    0    0         0    0    0    0    0.9961
         0    0    0    0    0    0    0    0    0    0    0    0         0    0    0    0         0
         0    0    0    0    0    0    0    0    0    0    0    0         0    0    0    0         0
         0    0    0         0    0    0    0    0.0510    0    0    0.2549    0         0    0    0    0         0
         0    0    0    0.7137    0    0    0    0.7608    0    0    0.7176    0         0    0    0    0         0
         0    0    0    0.9882    0    0    0    0.9882    0    0    0.9686    0    0.2392    0    0    0         0
         0    0    0    0.9882    0    0    0    0.9882    0    0    0.6078    0    0.7412    0    0    0         0
         0    0.3137    0         0    0    0    0    0.9922    0    0         0    0    0.7333    0    0         0
         0    0.9961    0         0    0    0    0    0.8902    0    0         0    0    0.0745    0    0.0392    0         0
```

در این خط سعی کردیم ابتدا داده های خود را مرتب کنیم. برای اینکار از متد reshape استفاده کردیم که در آرگومان اول ماتریسی که میخواهد مرتب کند را دریافت میکند و در آرگومان دوم سطر و ستون را دریافت میکند که ما مطابق دیتاست خود ۶۰هزار سطر(رکورد) را در نظر گرفتیم و تعداد ستون های آن را نیز برابر تعداد پیکسل ها قرار دادیم که نهایتا خروجی را به ما نمایش داد.

```
    Adding YTrain(Category) as a column to the end of the dataset.
38  Reshaped_Matrix(:, end+1) = YTrain

    Reshaped_Matrix = 60000×785
         0    0    0    0    0    0    0    0    0    0    0    0    0.5176    0    0    0    0.7451 ...
         0    0    0    0    0    0    0    0    0    0    0    0         0    0    0    0    0.9961
         0    0    0    0    0    0    0    0    0    0    0    0         0    0    0    0         0
         0    0    0    0    0    0    0    0    0    0    0    0         0    0    0    0         0
         0    0    0         0    0    0    0    0.0510    0    0    0.2549    0         0    0    0    0         0
         0    0    0    0.7137    0    0    0    0.7608    0    0    0.7176    0         0    0    0    0         0
         0    0    0    0.9882    0    0    0    0.9882    0    0    0.9686    0    0.2392    0    0    0         0
         0    0    0    0.9882    0    0    0    0.9882    0    0    0.6078    0    0.7412    0    0    0         0
         0    0.3137    0         0    0    0    0    0.9922    0    0         0    0    0.7333    0    0         0
         0    0.9961    0         0    0    0    0    0.8902    0    0         0    0    0.0745    0    0.0392    0         0
```

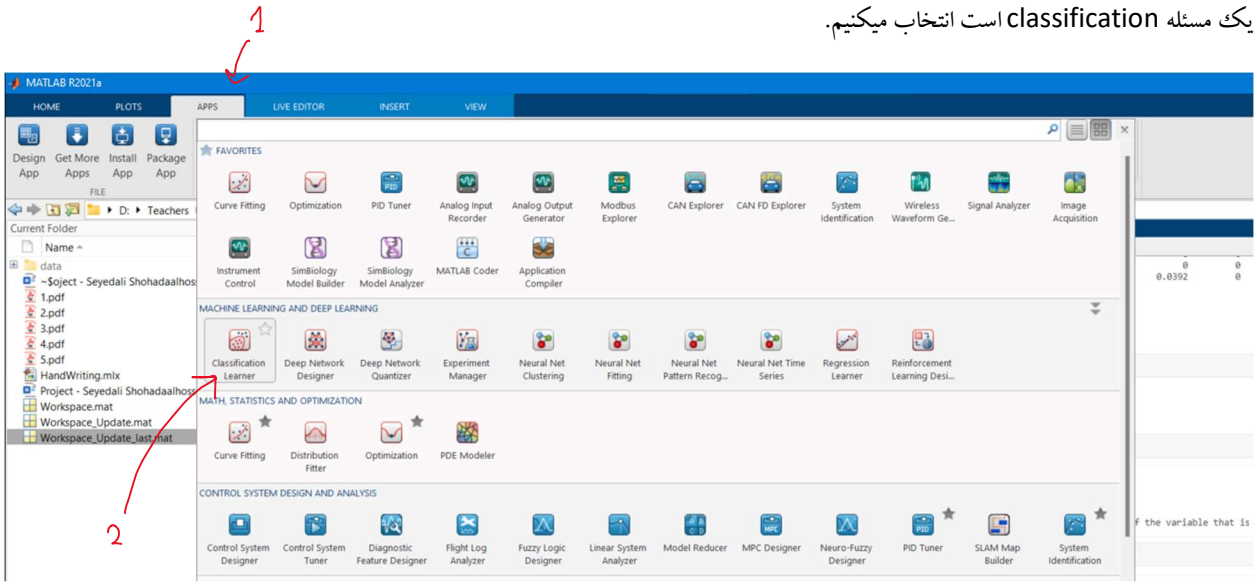
در این خط نیز آمدیم و در تمامی سطر های ستون آخر category خود را که در YTrain بود قرار دادیم. البته باید توجه شود که چون تعداد دو ماتریس با هم برابر بود، یعنی ۶۰هزار رکورد و ۶۰هزار دسته بندی آن مشخص شده بود، این assignment درست بود و خطا نداد.

```
39  subTrainingSet = Reshaped_Matrix(1:60000, :)

    subTrainingSet = 60000×785
         0     0     0     0     0     0     0     0     0     0     0     0    0.5176     0     0     0    0.7451 ...
         0     0     0     0     0     0     0     0     0     0     0     0         0     0     0     0    0.9961
         0     0     0     0     0     0     0     0     0     0     0     0         0     0     0     0         0
         0     0     0     0     0     0     0     0     0     0     0     0         0     0     0     0         0
         0     0     0     0     0     0     0     0    0.0510     0     0    0.2549     0     0     0     0     0
         0     0     0    0.7137     0     0     0    0.7608     0     0    0.7176     0     0     0     0     0
         0     0     0    0.9882     0     0     0    0.9882     0     0    0.9686     0    0.2392     0     0     0
         0     0     0    0.9882     0     0     0    0.9882     0     0    0.6078     0    0.7412     0     0     0
         0    0.3137     0     0     0     0     0    0.9922     0     0     0     0     0.7333     0     0     0
         0    0.9961     0     0     0     0     0    0.8902     0     0     0     0     0.0745     0    0.0392     0
         ⋮

    and their labels

40  subTrainingSet_Y = YTrain(1:60000);
```

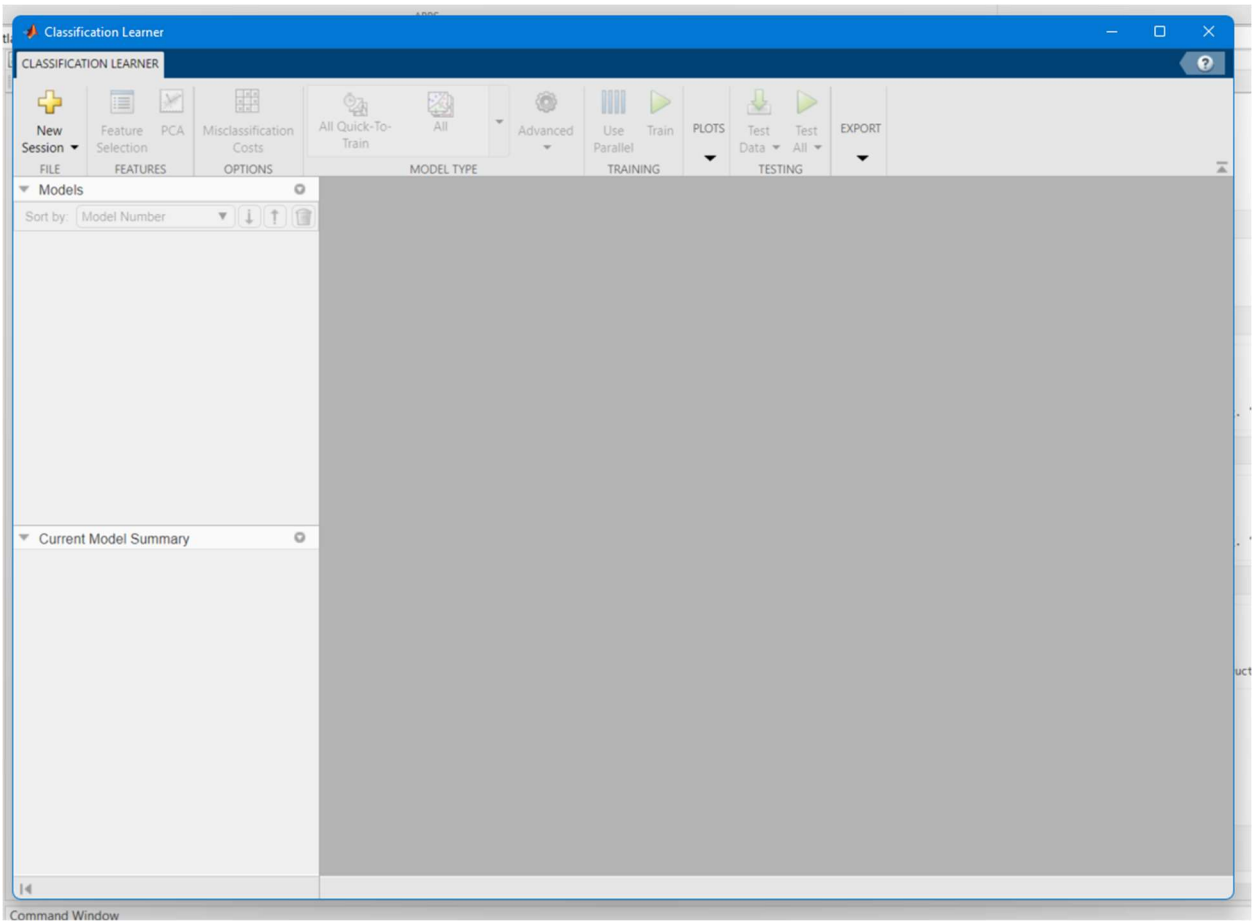
در خط ۳۹ و ۴۰ نیز سعی کردیم زیر مجموعه ای از داده های خود را دریافت کنیم برای آموزش دادن، اینکار را برای این انجام دادیم که فرآیند آموزش زمانبر نشود.

در مرحله بعدی ما میبایست train داده های خود را آغاز کنیم. در این قطعه کد ما ۳ train با مقدار داده های متفاوت و الگوریتم های متفاوت انجام دادیم، در ادامه یک مورد را به عنوان نمونه بررسی میکنیم.

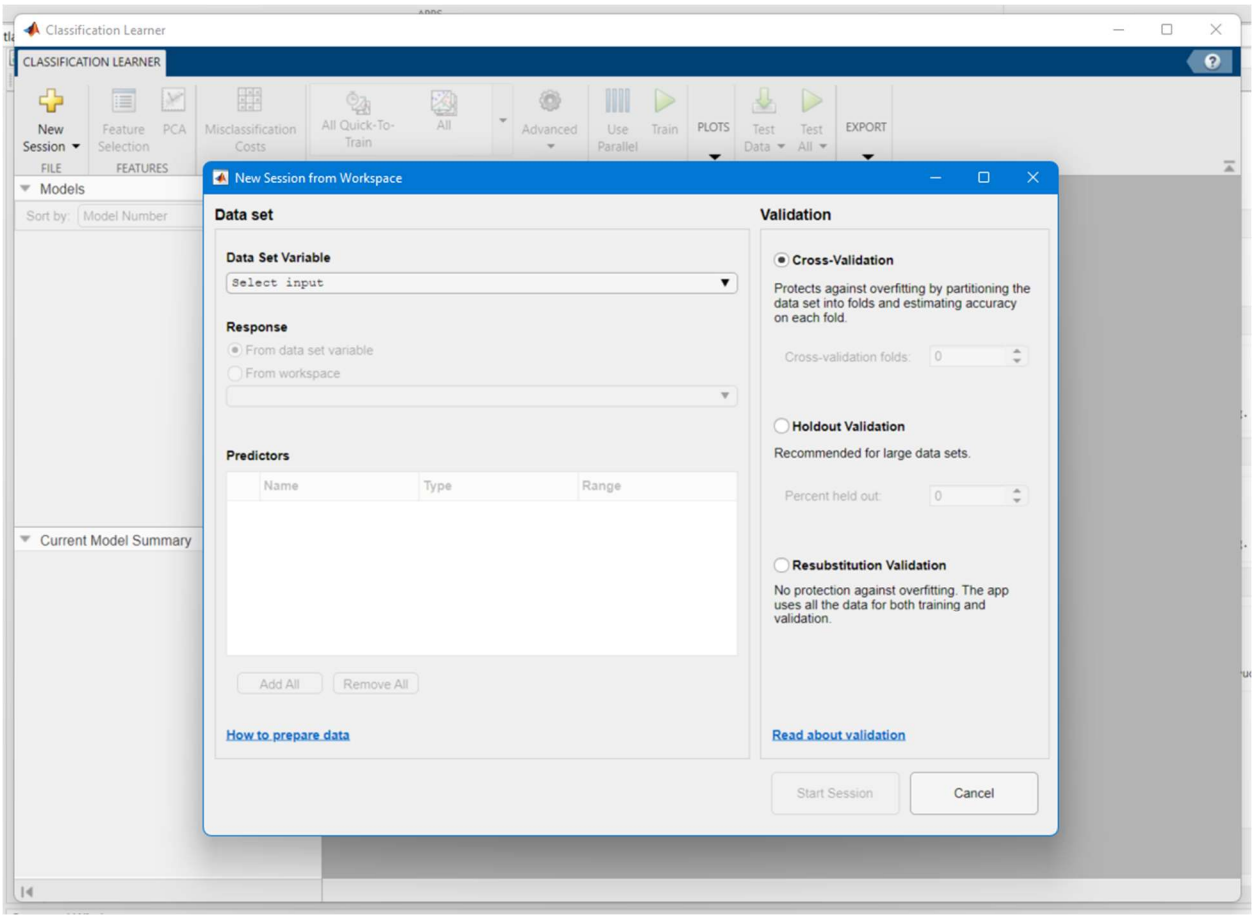
برای train کردن داده های انتخابی، ابتدا از منو بالا وارد بخش apps ها میشویم. و سپس متد classification را از آنجایی که این مسئله ما یک مسئله classification است انتخاب میکنیم.



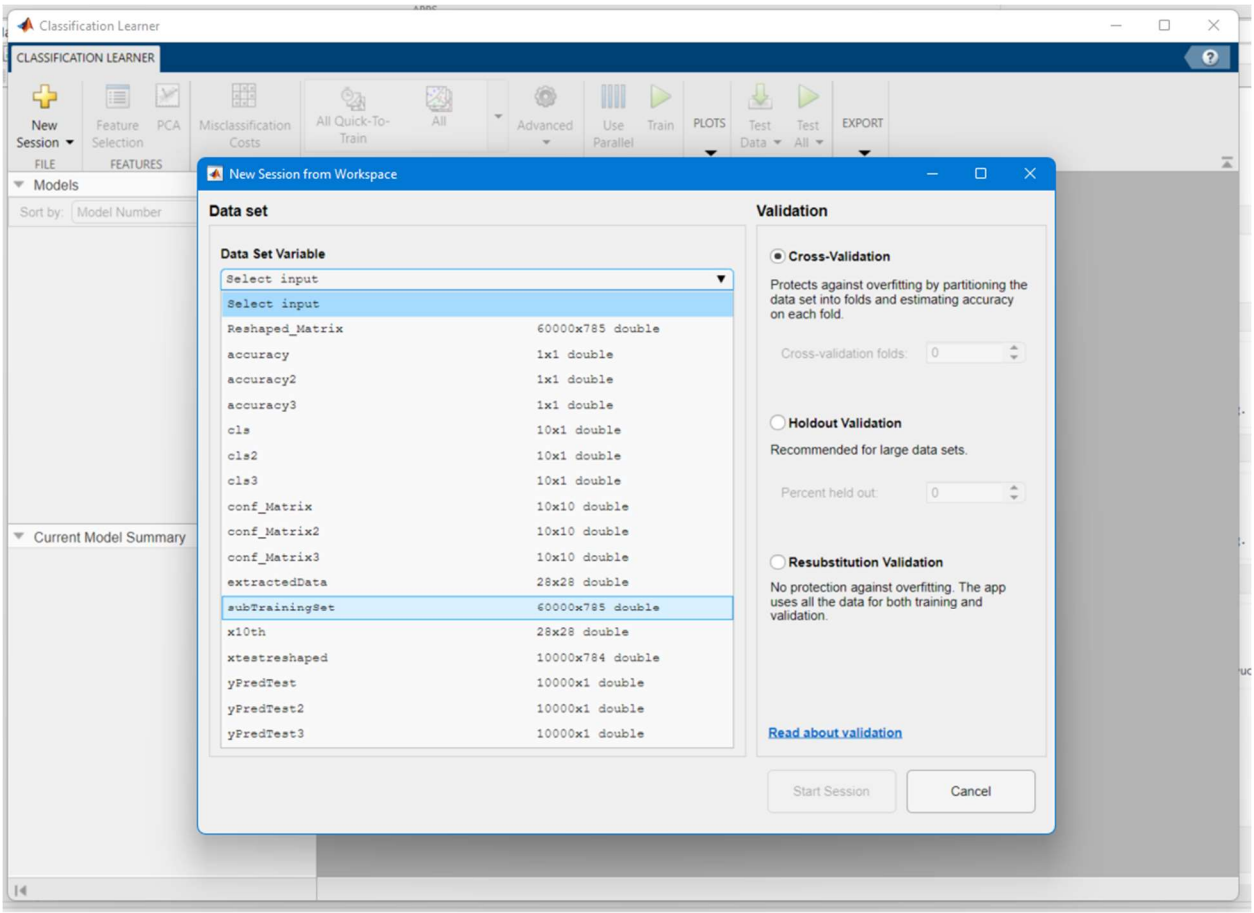
سپس منتظر میمانیم تا پنجره زیر load شود.



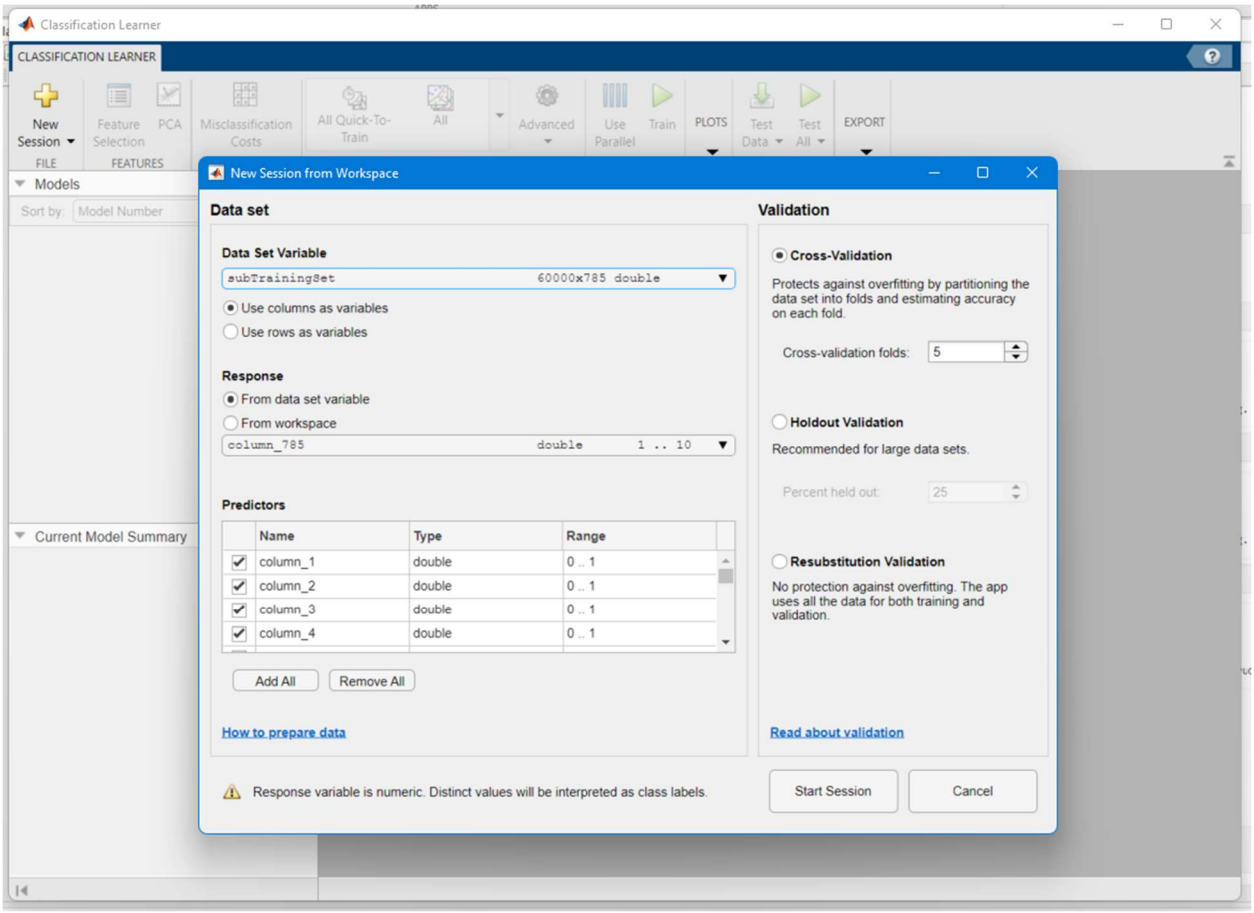
در این پنجره بر روی New Session کلیک میکنیم و سپس From Workspace را انتخاب میکنیم تا پنجره زیر باز شود.



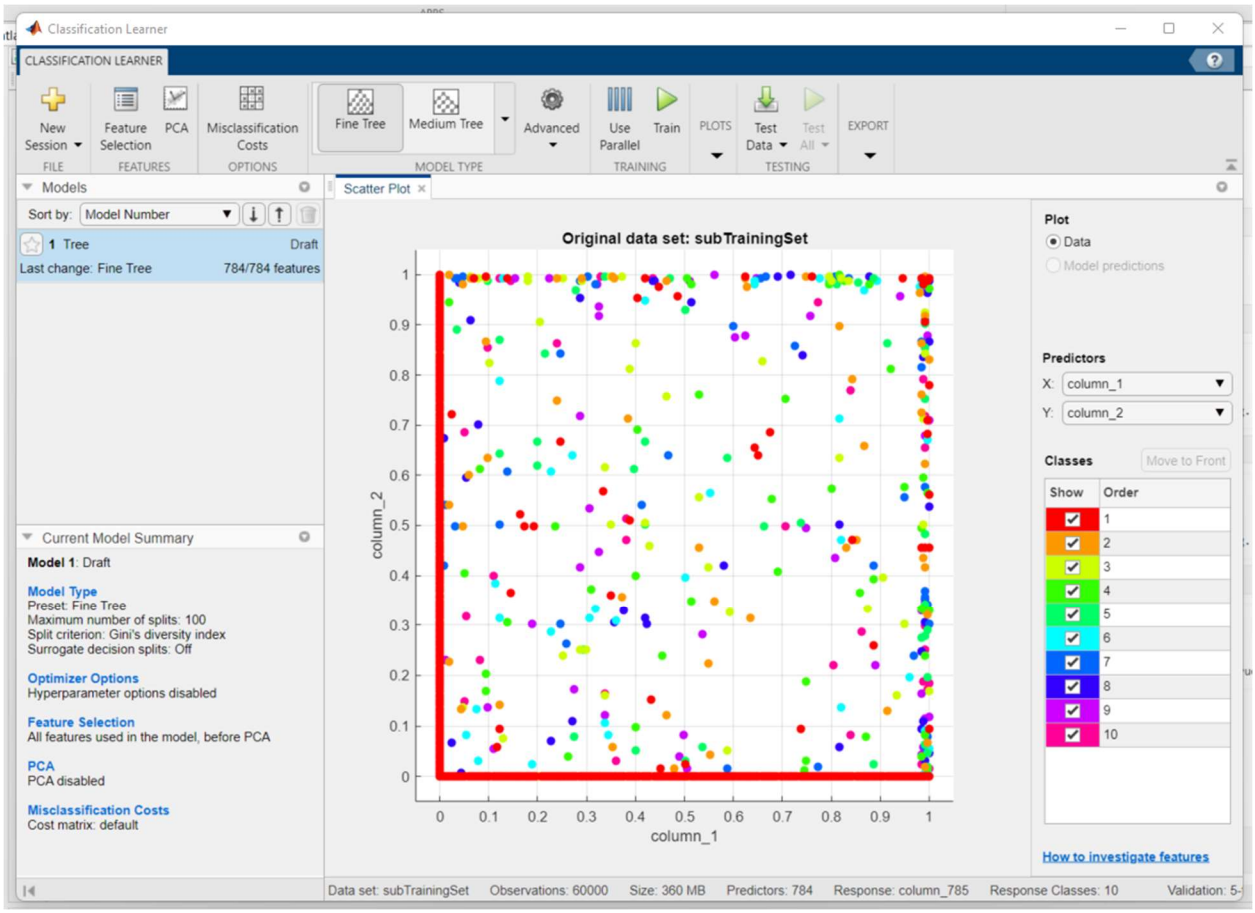
در این پنجره از بخش Dataset Variable دیتاست مورد نظر خود را انتخاب میکنید. این دیتا ست را ما پیشتر در روند برنامه ایجاد کردیم که در تصویر زیر مشاهده میکنید.



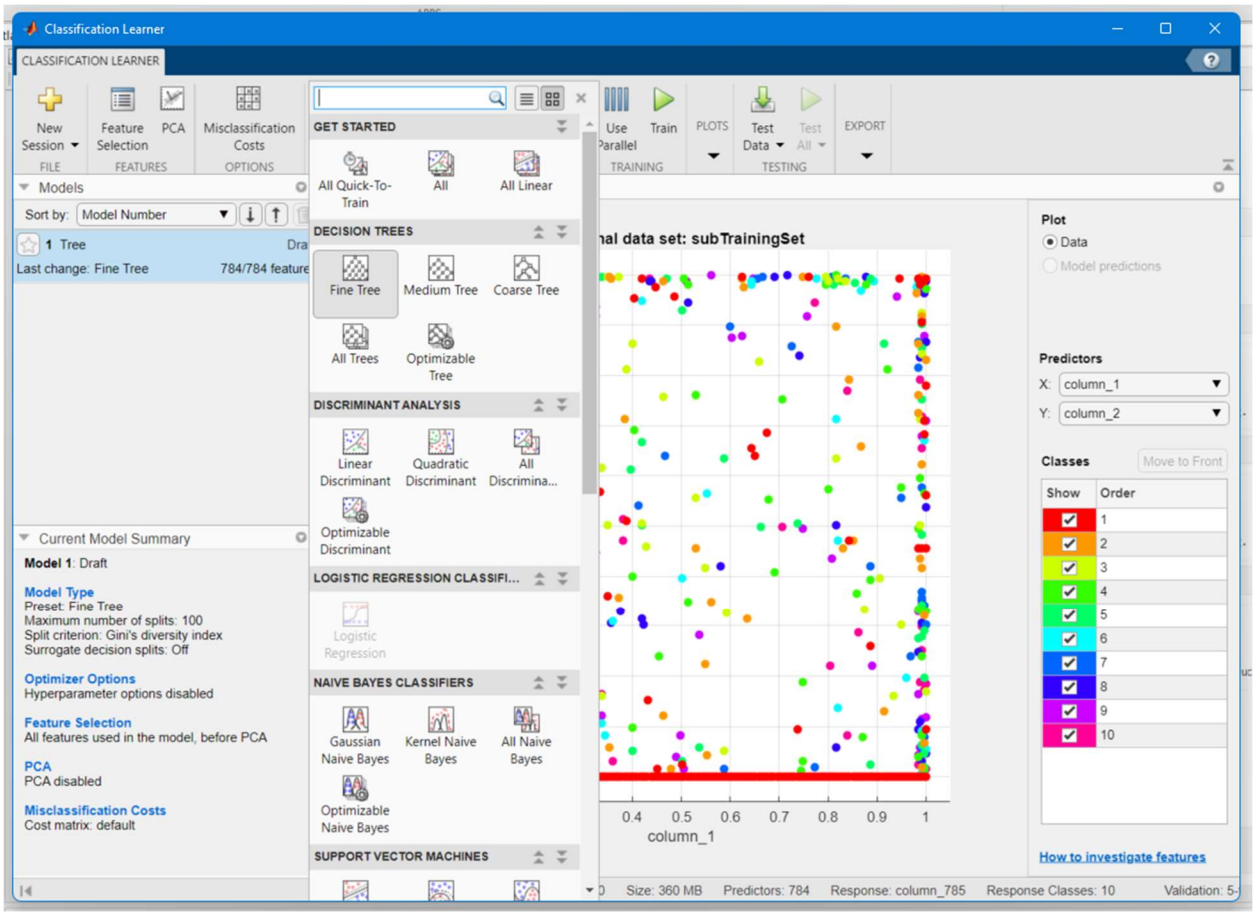
بعد از انتخاب دیتاست مورد نظر، در مرحله بعدی از بخش Response ستونی که جواب مقادیر این دیتاست هست را انتخاب میکنیم، یعنی ستون 785ام که خروجی داده های خود را در آن قرار دادیم.



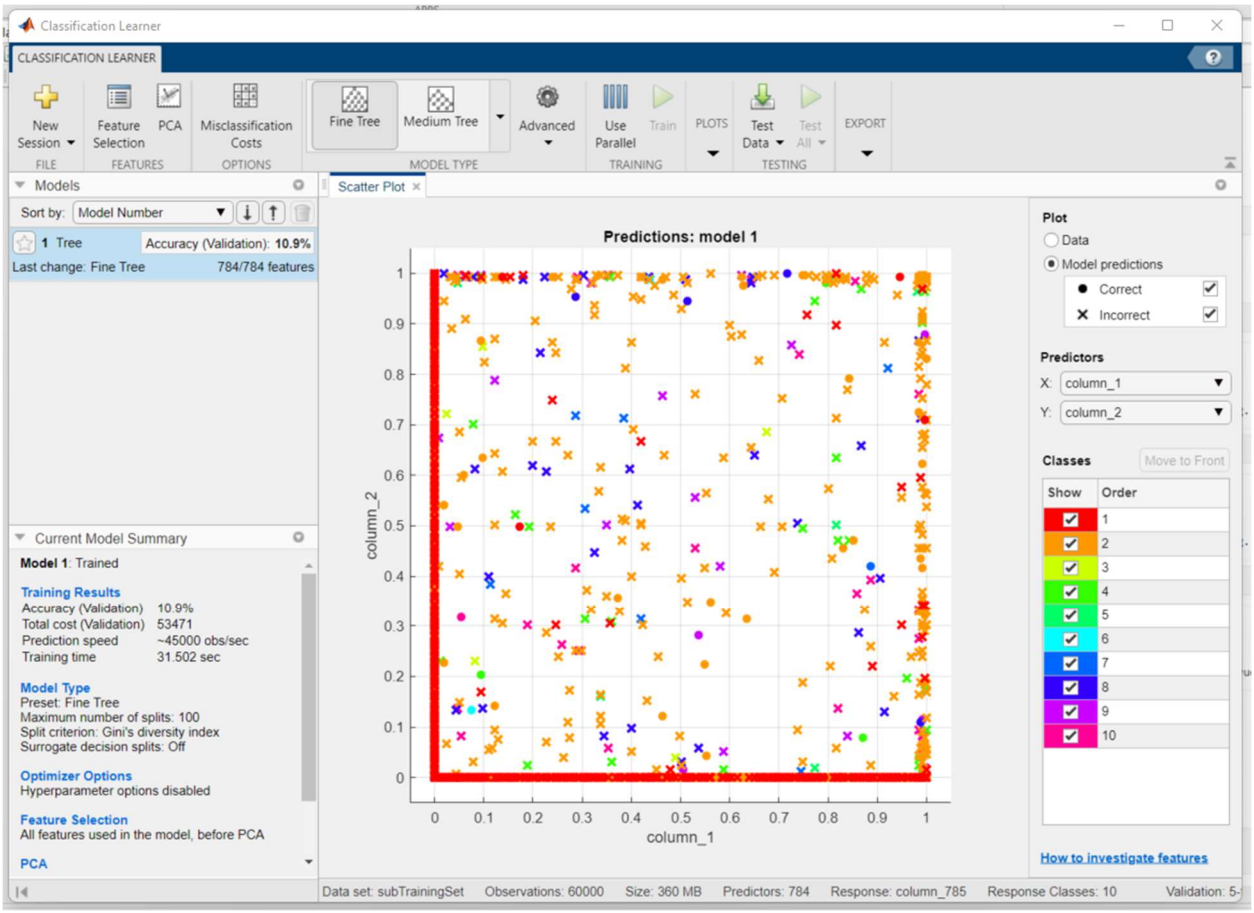
بعد از این بخش بر روی Start Session کلیک میکنیم و کمی بعد پنجره زیر نمایش داده میشود.



در این پنجره در ردیف پایین میتوانید اطلاعات دیتاست خود را مشاهده کنید. سپس در ادامه از منو بالا الگوریتم مورد نظر خود را برای train انتخاب میکنید.



که ما برای مثال Fine Tree را انتخاب کردیم. بعد از انتخاب الگوریتم مورد نظر بر روی Train کلیک میکنیم و منتظر میمانیم تا آموزش مدل ما به اتمام برسد. نهایتا بعد از اتمام تصویر زیر را به عنوان خروجی مشاهده می کنید.



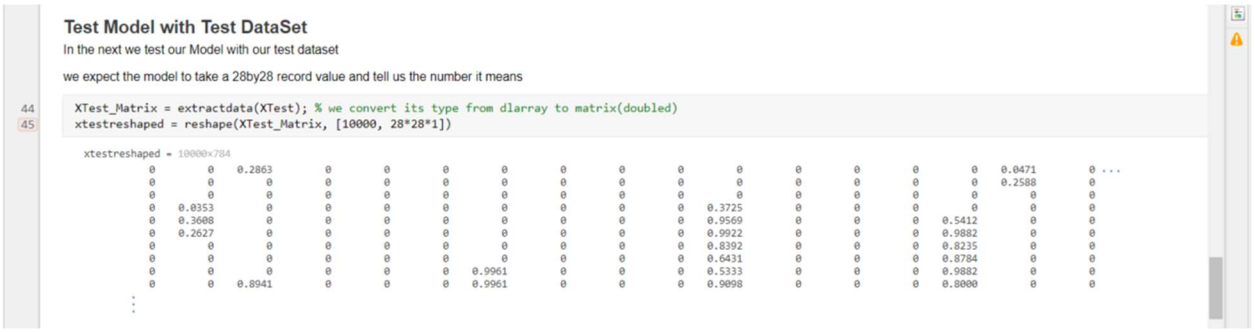
در این تصویر accuracy ۱۰درصد برای مدل ما ذکر شده است.

کار آموزش الگوریتم تا به اینجا به اتمام رسید و حالا میتوانیم این مدل آموزش دیده خود را ذخیره کنیم تا در برنامه از آن استفاده کنیم. برای اینکار نیز بر روی EXPORT کلیک میکنید و مدل خود را با نام مورد نظر خود ذخیره میکنید.

بعد از ذخیره این پنجره را ببندید. اگر به Workspace خود در نرم‌افزار متلب نگاه کنید، مدل شما در آن قابل رویت است. شما میتوانید در ادامه از این نام مدل استفاده کنید برای پیشبینی کردن داده های مورد نظرتان.

```
##########################################
Here we must train our model. Below is the trained model
41  trainedModel

trainedModel = struct with fields:
           predictFcn: @(x)exportableModel.predictFcn(predictorExtractionFcn(x))
    ClassificationTree: [1×1 ClassificationTree]
                 About: 'This struct is a trained model exported from Classification Learner R2021a.'
           HowToPredict: 'To make predictions on a new predictor column matrix, X, use: ↵  yfit = c.predictFcn(X) ↵replacing 'c' with the name of the variable that is this struct, e.g. 'train

42  trainedModel_30000

trainedModel_30000 = struct with fields:
           predictFcn: @(x)exportableModel.predictFcn(predictorExtractionFcn(x))
    ClassificationTree: [1×1 ClassificationTree]
                 About: 'This struct is a trained model exported from Classification Learner R2021a.'
           HowToPredict: 'To make predictions on a new predictor column matrix, X, use: ↵  yfit = c.predictFcn(X) ↵replacing 'c' with the name of the variable that is this struct, e.g. 'train

43  trainedModel_LiDis_30000

trainedModel_LiDis_30000 = struct with fields:
                predictFcn: @(x)exportableModel.predictFcn(predictorExtractionFcn(x))
    ClassificationDiscriminant: [1×1 ClassificationDiscriminant]
                 About: 'This struct is a trained model exported from Classification Learner R2021a.'
           HowToPredict: 'To make predictions on a new predictor column matrix, X, use: ↵  yfit = c.predictFcn(X) ↵replacing 'c' with the name of the variable that is this struct, e.g

##########################################
```

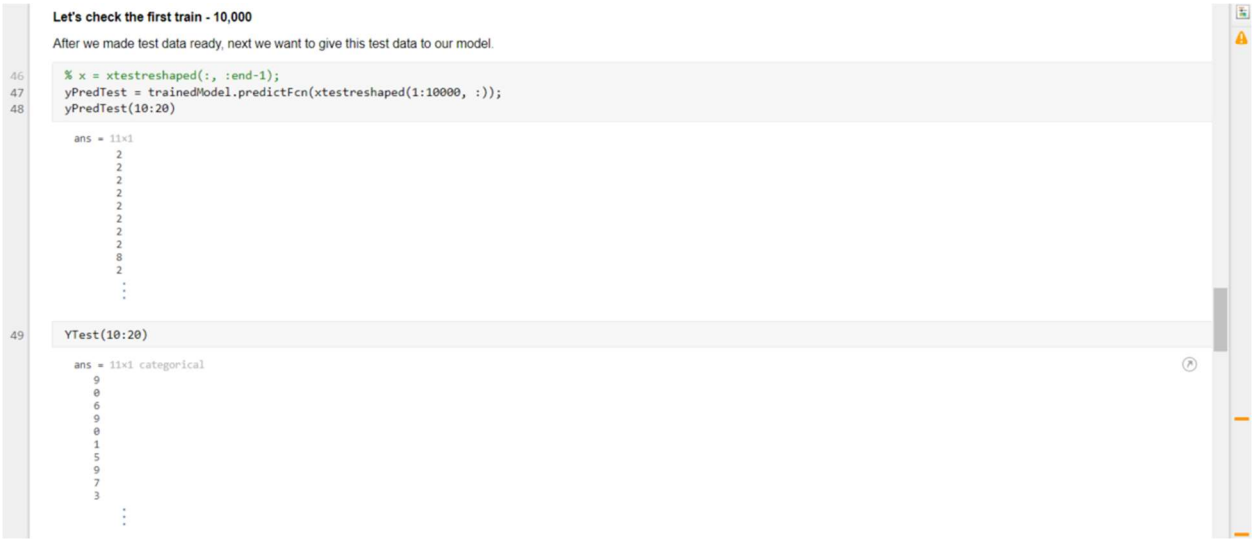
در این بخش سه مدل آموزش دیده شده توسط بنده را مشاهده میفرمایید.



در اینجا داده های تست مورد نظر را برای تست گرفتن مدل آموزش دیده، آماده کردیم.

خروجی هر سه مدل در این برنامه تست شد که یکی از اینها را در ادامه تشریح میکنم.

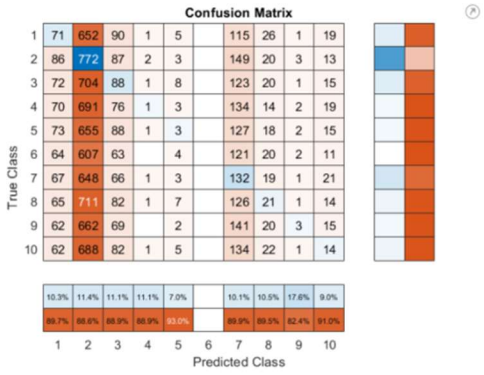
مدل اول



به کمک متد trainedModel.predictFcn داده تست مورد نظر خود را به مدل آموزش دیده خود با نام TrainedModel ارسال کردیم.
خروجی این متد پیشبینی هایی میباشد که این مدل بر روی داده های تست ما انجام داده است. که این پیشبینی های را در خط ۴۸ مشاهده میکنید و در خط ۴۹ دسته بندی اصلی این داده های تست را مشاهده میفرمایید.

در ادامه سعی کردیم به کمک تابع ConfusionMat ماتریس سردرگمی این مدل را بدست آوریم و آن را بررسی کنیم. خروجی این ماتریس کلاس(یعنی دسته بندی های ما) و خود ماتریس میباشد.

```
figure(3)
Confusion_Chart = confusionchart(conf_Matrix, cls, ...
    "Title", "Confusion Matrix", ...
    "ColumnSummary","column-normalized", ...
    "RowSummary","row-normalized");
```



Next we're calculating accuracy. It is equal to the division of the all successful predict number to the all predict number

```
accuracy = sum(double(YTest(:)) == yPredTest,'all')/numel(yPredTest) * 100
```

```
accuracy = 11.0500
```

سعی شد با این قطعه کد این خروجی را بهتر مورد بررسی قرار دهیم. همانطور که در تصویر مشاهده میفرمایید در ردیف ها ما کلاس های واقعی را داریم و در ستون ها مقدار پیشبینی شده آن را داریم. برای مثال برای مقدار کلاس ۱ ما ۷۱ پیشبینی درست داشتیم و ۶۵۲بار اشتباها مقدار ۲ پیشبینی شد و به همین ترتیب.

و نهایتا میزان accuracy این مدل خود را بدست آوردیم که برابر ۱۱ درصد بود.

در ادامه همین کار ها را با مدل های دیگری که آموزش دادیم انجام دادیم که از توضیحات آن عبور میکنیم. تصاویر ادامه قطعه کد به شرح زیر است.

Let's test the second train - 30,000

```
yPredTest2 = trainedModel_30000.predictFcn(xtestreshaped(1:10000, :));
yPredTest2(10:20)
```

```
ans = 11×1
     2
     2
     2
     2
     2
     2
     2
     2
     3
     3
     ⋮
```

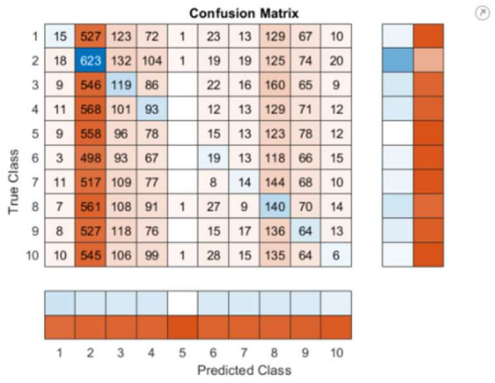
```
YTest(10:20)
```

```
ans = 11×1 categorical
     9
     0
     6
     9
     0
     1
     5
     9
     7
     3
     ⋮
```

```
[conf_Matrix2, cls2] = confusionmat(double(YTest(:)),yPredTest2(:))
```

```
conf_Matrix2 = 10×10
    15   527   123    72     1    23    13   129    67    10
    18   623   132   104     1    19    19   125    74    20
     9   546   119    86     0    22    16   160    65     9
    11   568   101    93     0    12    13   129    71    12
     9   558    96    78     0    15    13   123    78    12
     3   498    93    67     0    19    13   118    66    15
    11   517   109    77     0     8    14   144    68    10
     7   561   108    91     1    27     9   140    70    14
     8   527   118    76     0    15    17   136    64    13
    10   545   106    99     1    28    15   135    64     6

cls2 = 10×1
     1
     2
     3
     4
     5
     6
     7
     8
     9
    10
```

```
figure(4)
Confusion_Chart = confusionchart(conf_Matrix2, cls2, ...
    "Title", "Confusion Matrix", ...
    "ColumnSummary","column-normalized", ...
    "RowSummary","row-normalized");
```



Next we're calculating accuracy. It is equal to the division of the all successful predict number to the all predict number

```
accuracy2 = sum(double(YTest(:)) == yPredTest2,'all')/numel(yPredTest2)
```

```
accuracy2 = 0.1093
```

Let's test the second train - 30,000 - Linear Discriminant

```
yPredTest3 = trainedModel_LiDis_30000.predictFcn(xtestreshaped(1:10000, :));
yPredTest3(10:20)
```

```
ans = 11×1
     4
    10
    10
     3
     3
     3
     7
     6
     3
     3
     ⋮
```

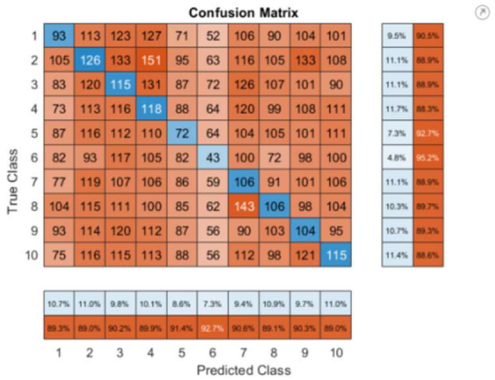
```
YTest(10:20)
```

```
ans = 11×1 categorical
     9
     0
     6
     9
     0
     1
     5
     9
     7
     3
     ⋮
```

```
70  [conf_Matrix3, cls3] = confusionmat(double(YTest(:)),yPredTest3(:))
```

```
conf_Matrix3 = 10×10
    93   113   123   127    71    52   106    90   104   101
   105   126   133   151    95    63   116   105   133   108
    83   120   115   131    87    72   126   107   101    90
    73   113   116   118    88    64   120    99   108   111
    87   116   112   110    72    64   104   105   101   111
    82    93   117   105    82    43   100    72    98   100
    77   119   107   106    86    59   106    91   101   106
   104   115   111   100    85    62   143   106    98   104
    93   114   120   112    87    56    90   103   104    95
    75   116   115   113    88    56   112    98   121   115

cls3 = 10×1
     1
     2
     3
     4
     5
     6
     7
     8
     9
    10
```

```
71  figure(5)
72  Confusion_Chart = confusionchart(conf_Matrix3, cls3, ...
73      "Title", "Confusion Matrix", ...
74      "ColumnSummary","column-normalized", ...
75      "RowSummary","row-normalized");
```



Next we're calculating accuracy. It is equal to the division of the all successful predict number to the all predict number

```
76  accuracy3 = sum(double(YTest(:)) == yPredTest3,'all')/numel(yPredTest3)
```

```
accuracy3 = 0.0998
```

با تشکر