به نام خدا

گزارش پروژه ۲

نگین سفاری ۸۱۰۱۹۷۵۲۵

مبانی مهندسی پزشکی

دکتر بدیعی

negin safari

بخش اول

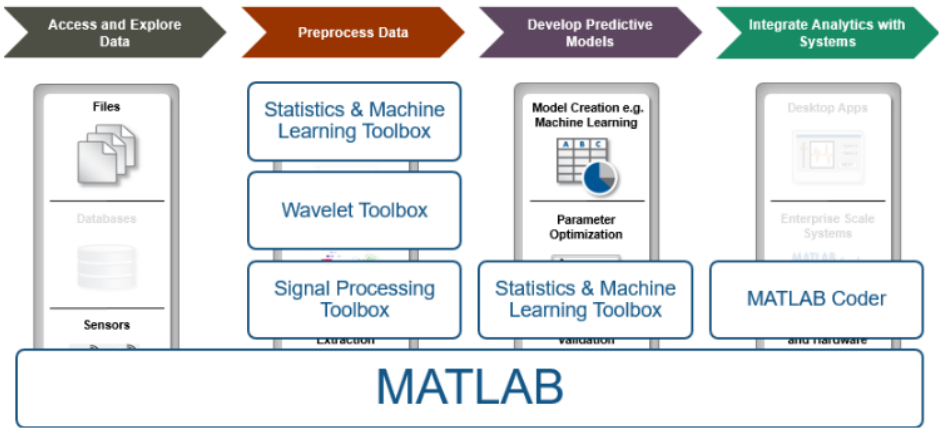
ویدیو دیده شد.

بخش دوم

در مثال داده شده به کمک ECG و صوت قلب بیماری قلبی تشخیص داده میشود. برای انجام این کار تعدادی داده برای آموزش و تعدادی داده برای تست به همراه لیبل های آن ها را داریم.

برای انجام این کار از یادگیری ماشین استفاده شده است. طبق ویدیو یادگیری ماشین ۴ مرحله دارد.

1. Access and explore data
2. Preprocess data
3. Develop predictive models
4. Integrate analytics with systems

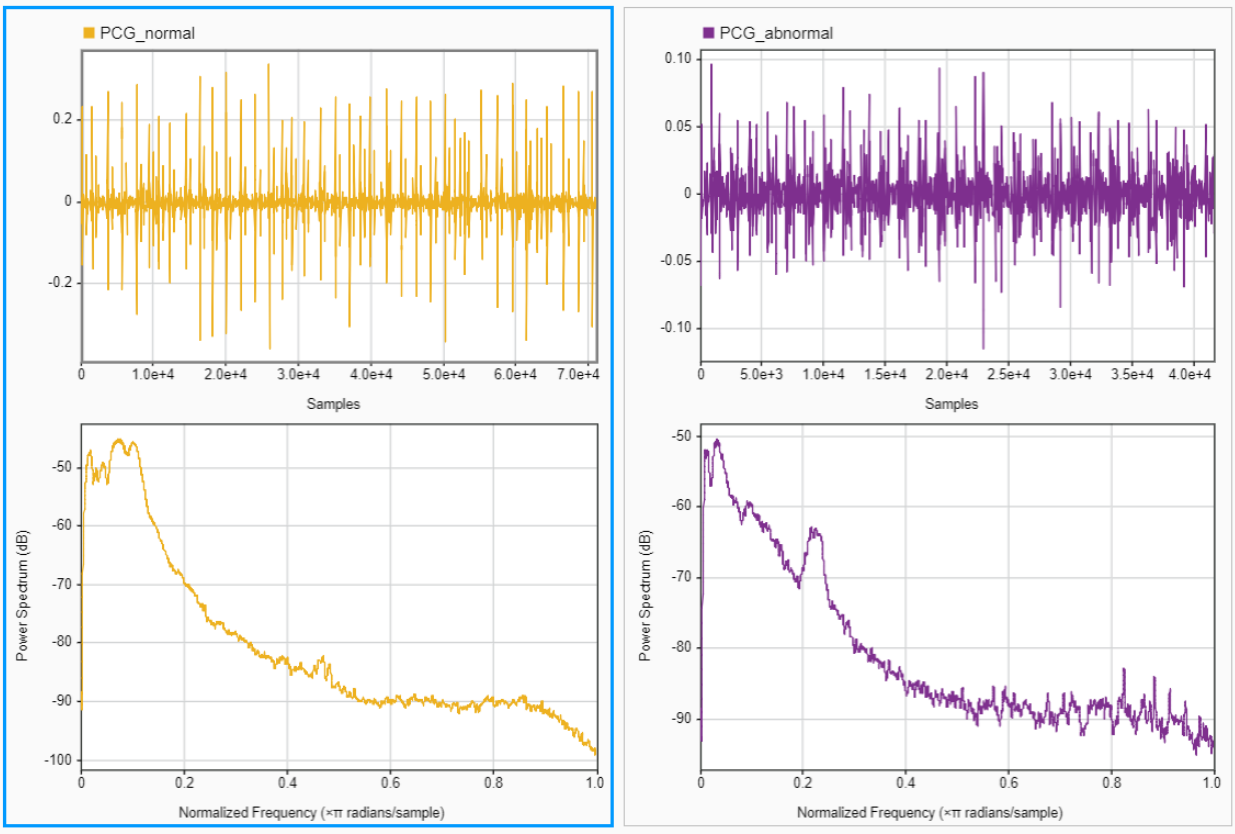


شکل ۱: مراحل آموزش و تست ماشین

در ادامه در این مثال این مراحل را طی میکنیم و به توضیح آن ها میپردازیم.

❖ مرحله ی Access and explore data

در این بخش به داده های ورودی دسترسی پیدا میکنیم و در کد یک صدای normal و یک صدای abnormal خوانده میشوند و پس از ران شدن کد صدای هر یک شندیده میشود. پس از آنکه اطلاعات صدا ها در ساختارهایی ذخیره شدند، نمودار های power spectrum بر حسب فرکانس و amplitude بر حسب زمان به دست می آیند. به کمک نمودار بر حسب فرکانس متوجه میشویم که سیگنال مربوط به صوت normal در فرکانس های پایین توان بیشتری دارد و با توجه به نمودار زمانی بین ضربان ها سکوت است و موج بدون نویز است. اما در سیگنال abnormal در نمودار زمانی میتوان دید که نویز بیشتری در طی سیگنال وجود دارد و در فرکانس های بالاتر مقدار دارد. در نمودار power spectrum در سیگنال abnormal یک جهش در اطراف فرکانس زاویه ای 0.2 radian موجود است که این در حالت normal دیده نمیشود.



شکل ۲ : نمودار های یک نمونه از صوت برای فرد نرمال و غیر نرمال

در این بخش نام تمامی صدا های موجود به همراه لیبل 1 و 1- که به ترتیب معنای normal و abnormal بودن صدا هست خوانده میشوند (موجود در فایل REFRENCE.csv) و در refrence_table ذخیره سازی میشوند. این داده ها داده های مربوط به training اند و به منظور یاددادن به ماشین ذخیره میشوند.

❖ مرحله Preprocess data

در این بخش قبل از آنکه وارد فرآیند یاددهی به ماشین شویم باید از داده های در دسترس تعدادی خاصیت را خارج کنیم. این خواص داده هایی مانند داده های آماری انحراف معیار، میانگین، مد و ... و یا ویژگی های فرکانسی مانند دامنه و amplitude در فرکانس های سیگنال ها هستند. تعداد کل این خواص ۲۷ عدد است که تمام آن ها وارد feature_table میشوند. بدین ترتیب در مراحل بعدی میتوانیم به راحتی به آنها دسترسی پیدا کنیم. تابع extractFeatures این داده ها را میگیرد و وارد feature_table میکند.

❖ مرحله Develop predictive models

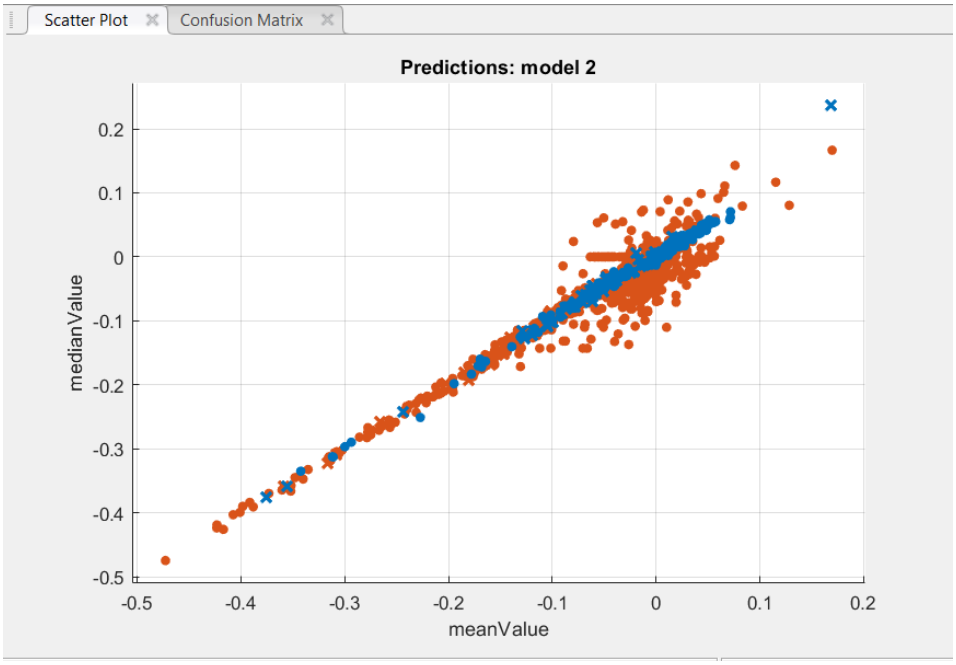
در این قسمت به کمک classification learner ۱۳۰۱۵ داده را تقسیم بندی میکنیم. از این داده ها مشخص میکنیم که 30% به منظور training استفاده شوند و باقی به منظور تست استفاده شوند. حال به کمک مدل های مختلف training را بر روی ماشین انجام میدهیم و پلات های Confusion Matix و accuracy را بررسی میکنیم.

در این قسمت به کمک ۶ روش Tree و Fine KNN و Linear SVM و Quadratic SVM و Logistic Regression و Ensemble عملیات train انجام شده است و نتایج به شرح زیر است.

1 ☆ Tree Last change: Disabled PCA	Accuracy: 89.9% 27/27 features
2 ☆ KNN Last change: Fine KNN	Accuracy: **94.5%** 27/27 features
3 ☆ SVM Last change: Linear SVM	Accuracy: 87.0% 27/27 features
4 ☆ SVM Last change: Quadratic SVM	Accuracy: 90.3% 27/27 features
5 ☆ Logistic Regression Last change: Logistic Regression	Accuracy: 86.1% 27/27 features
6 ☆ Ensemble Last change: Bagged Trees	Accuracy: 93.0% 27/27 features

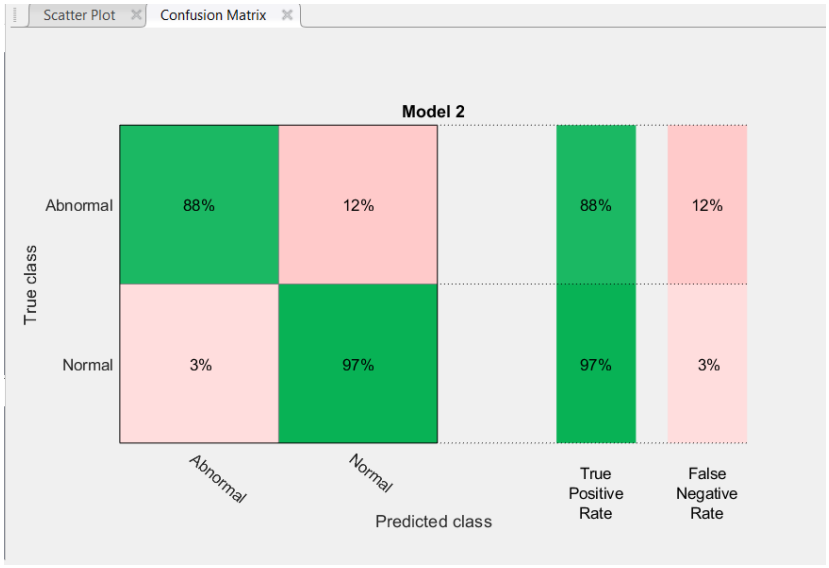
شکل ۳ : نتایج accuracy با روش های مختلف

طبق نتایج Fine KNN بیشترین میزان accuracy که برابر 94.5% میباشد، است.

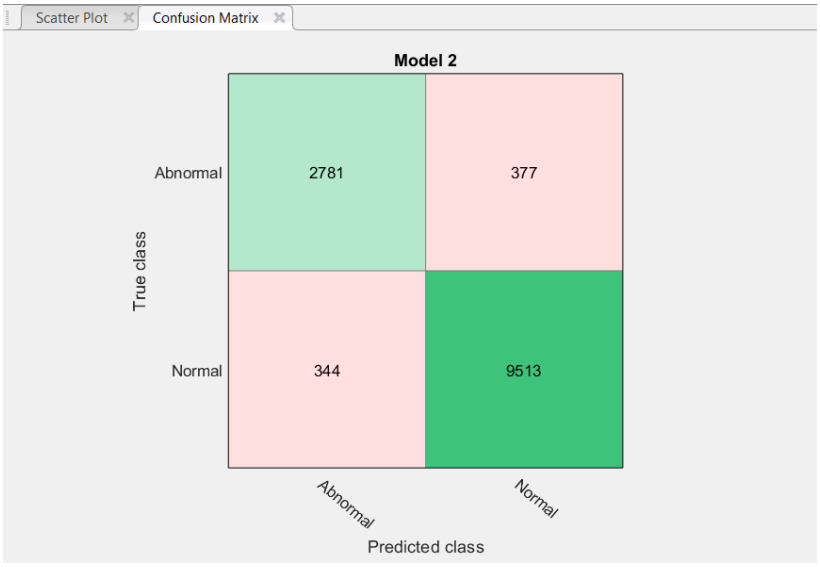


شکل ٤ : scatter plot برای روش Fine KNN

در scatter plot بخش هایی که به صورت نقطه مشخص شده اند به معنای درستی تشخیص ماشین در مرحله ی تست است و ضربدر ها به معنای تشخیص اشتباه ماشین اند.



شکل ۵: confusion matrix با نمایش درصدی



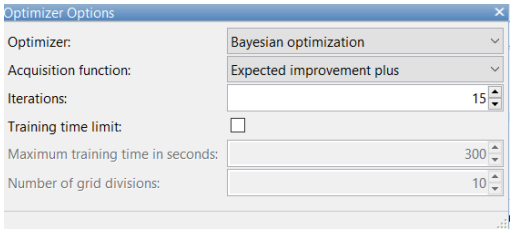
شکل ٦: confusion matrix با نمایش عددی

با توجه به confusion matrix میتوان فهمید که ماشین در تشخیص نتایج برای موج هایی که واقعا normal اند، عملکرد بهاری داشته است. اما در تشخیص موج های abnormal ضعیف تر عمل کرده است که این خوب نیست. زیرا در علم پزشکی تشخیص و اعلام آنکه یک فرد بیمار است، اهمیت بیشتری از آن دارد که فرد سالم باشد. اگر به فردی که واقعا بیمار است نتیجه غلط داده شود ممکن است جانش در خطر بیفتد. بنابراین باید درصد تشخیص سلامت افراد توسط ماشین برای افرادی کی واقعا بیمار اند را کمتر کنیم. علت این اتفاق آن است که داده های در اختیار ماشین که مربوط به موج های abnormal است، 3158 داده است ولی داده های مربوط به امواج normal برابر 9857 عدد است. بنابراین این ماشین به شکل بایاس شده train شده است و منطقی است که در تشخیص بیمار های واقعی ضعیف تر عمل کرده است.

بنابراین باید به بهبود این درصد بپردازیم. در کل این روند باید بدانیم که از هر جا که ما پیشرفتی داشته باشیم قطعا هزینه ای بابت آن پرداخت کرده ایم که این هزینه میتواند درصد درستی تشخیص افراد سالم باشد.

یکی از عواملی که با اصلاح آن به بهبود این مورد کمک میکند، کم کردن تعدادی از ویژگی های در دسترس ماشین به عنوان predictor است. دلیل آن است که گاها وقتی اطلاعات زیادی داریم برخی به اندازه باقی اطلاعات مفید نیستند و تنها باعث بایاس بیشتر میشوند که منجر به تشخیص اشتباه افراد واقعا بیمار میشود. بنابراین لازم است ویژگی هایی که مفید نیستند را حذف کنیم تا حجم محاسبات ماشین کاسته شود که به این کار Feature Selection می گویند. در مثال حل شده از Neighborhood Component Analysis بدین منظور استفاده شده تا ویژگی هایی که وزن و اهمیت بیشتری در طبقه بندی دارد را انتخاب کند. در این مثال با انجام این کار از بین 27 ویژگی تنها 15 ویژگی مفید انتخاب شده است و بدین ترتیب مساحت سخت افزار و منابع مورد نیاز ماشین کاسته میشود و به نسبت پیچیدگی محاسبات هم کم میشود. در حالاتی که تعداد داده های در دسترس مقدار زیادی اند(بالای 70) باعث رخ دادن over fitting میشود و این زیادی داده ها باعث تشخیص غلط ماشین میشود. در این مثال با این کار تنها مساحت سخت افزار و منابع مورد نیاز ماشین کاسته میشود و به نسبت پیچیدگی محاسبات هم کم میشود.

روش مفیدی که انجام میشود استفاده از hyperparameter optimization است. در این حالت به ویژگی ها وزنی داده میشود که با آن درصد اهمیت به هر پارامتر تغییر میکند. در این حالت ممکن است پارامترهایی در تعیین جواب نهایی موثر تر از باقی واقع شوند. در مثال انجام شده مدل KNN بیشترین accuracy را به ما داد. بنابراین نوع optimizable KNN را انتخاب میکنیم و تنظیمات optimizer را به صورت شکل زیر انجام میدهیم. میدانیم که 15 ایتریشن با توجه به دیتاست ما مقدار نسبتا زیادی است.



شکل ۷ : تنظیمات optimizer

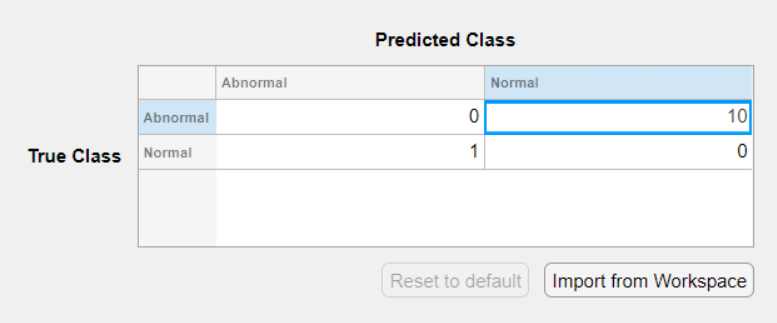
زمانی که به تنهایی این تنظیمات را بر روی optimizable KNN پیاده میکنیم. میزان accuracy مقداری افزایش میابد.

شکل ۸: میزان accuracy با روش optimizable KNN

همانطور که مشخص است accuracy از 94.5% به 95.1% افزایش یافته است.

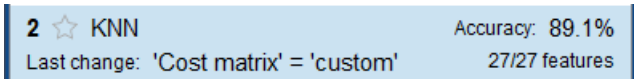
روش مفید دیگر تغییر دادن misclassification cost است. با تعیین مقادیر در این ماتریس به نوعی هزینه لازم برای پرداخت در ازای تشخیص اشتباه تعیین میشود. با توجه به تغییراتی که داده ایم و در شکل زیر مشخص است، ما cost برای تشخیص نادرست افراد واقعا abnormal را از 1 به 10 افزایش داده ایم. در این مثال چون داده های موجود برای افراد normal و abnormal در تعداد متفاوت و چشمگیری است، نیاز است که این هزینه ها تغییر کند. بدین ترتیب اگر فرد ناسالم به عنوان فرد سالم تشخیص داده شود هزینه بیشتری باید پرداخته شود.

در طی این تغییرات میزان accuracy کاسته میشود ولی در تشخیص افراد واقعا بیمار عملکرد بهتری دارد.



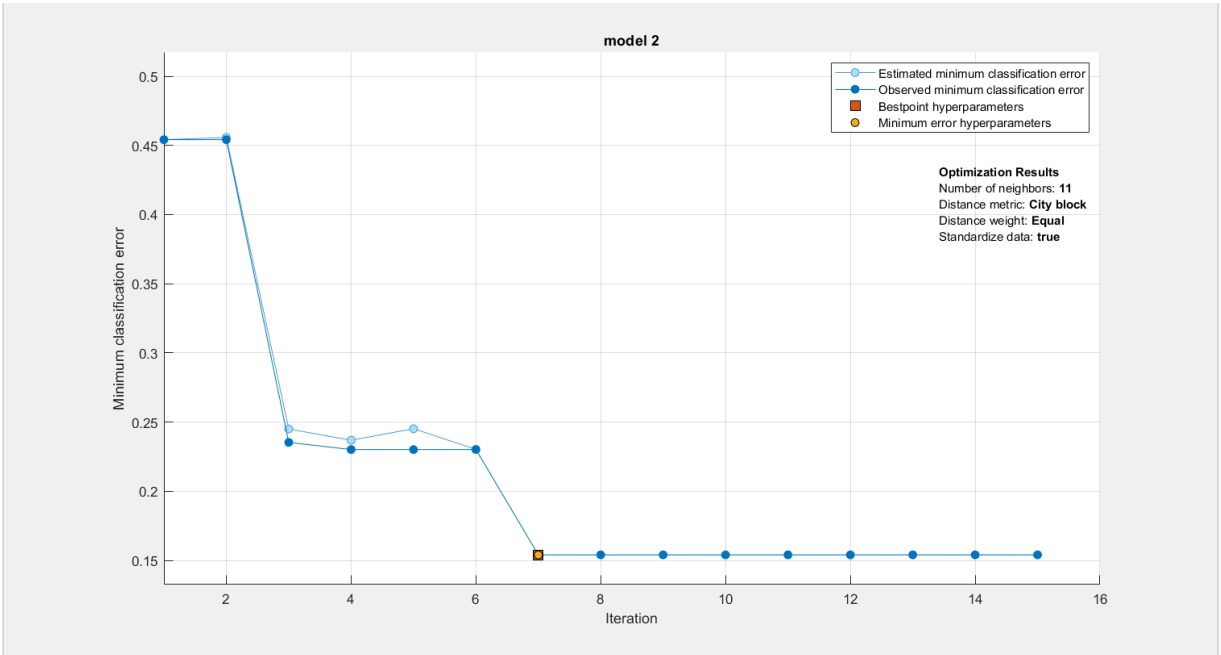
شکل ۹: تنظیمات ضرایب misclassification

در ادامه نتایج حاصل از انجام این تغییرات موجود است و به بررسی آن ها میپردازیم. لازم به یادآوری است که در این قسمت تعداد feature ها ثابت است و تنها misclassification cost تغییر یافته و از مدل optimizable استفاده شده است.



شکل ۱۰: میزان accuracy با اضافه کردن تغییرات در ضرایب misclassification

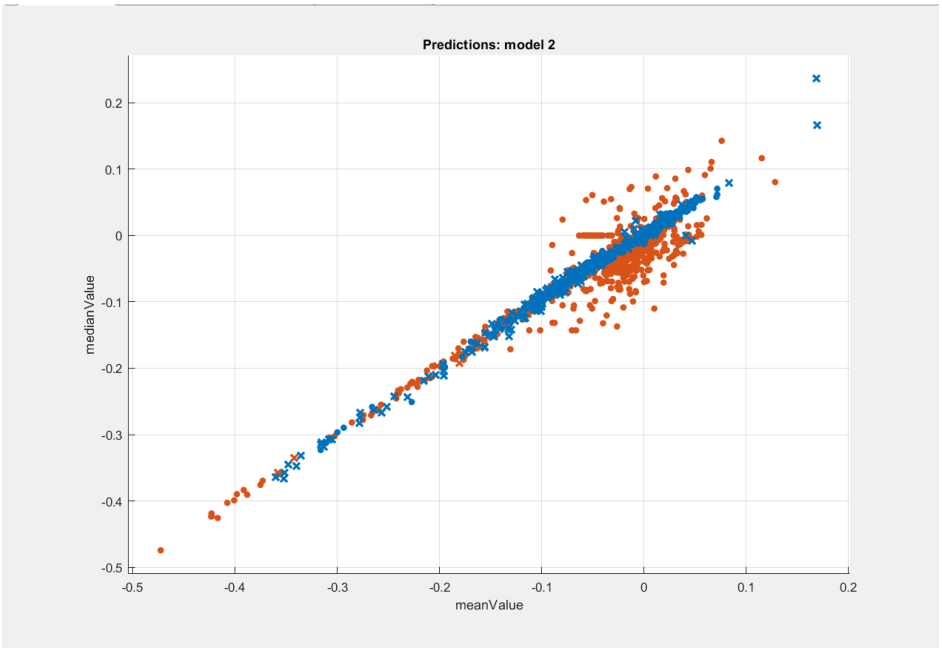
همانطور که دیده میشود accuracy به 89.1% رسیده و از آن مقداری کم شده است.



شکل ۱۱ : نمودار optimization result در ایتریشن ها

نمودار قبل نشان دهنده مینیم مقدار classification error در طول ایتریشن های محاسبات است. با افزایش ایتریشن ها این ارور کاسته شده است.

شکل ۱۲ : نمودار confusion matrix با درصد پس از بهبود متود



شکل ۱۳ : نمودار scatter plot پس از بهبود متود

در نمودار بالا نقطه ها نشاندهنده تشخیص درست ماشین و ضربدر ها نمایانگر تشخیص غلط ماشین اند.

همانطور که از confusion matrix مشخص است درصد تشخیص درست افراد واقعا abnormal به طرز چشمگیری زیاد شده است. اما از آنجا که قبلا اشاره شد(☺ there is no free lunch) درصد تشخیص صحیح افراد واقعا سالم اندکی کاسته شده است. در کل به مدل مطلوبمان رسیده ایم.

❖ مرحله Integrate analytics with systems

در این مرحله که میدانیم به مدل مطلوب مان رسیده ایم میتوانیم کد به زبان C تولید کنیم که سیگنال های رکورد شده را به همراه فرکانس نمونه برداری آن ها به عنوان ورودی بگیرد و کلاس بندی مورد نظرمان را بر روی این صدا ها انجام دهد و در خروجی تعیین کند کدام normal و کدام از نظر ماشین abnormal بوده اند. از این کد زبان C میتوان در سایر سیستم ها نیز بهره گرفت.

در این مرحله به کمک کد تولید شده میتوان عملیات validation را انجام داد. به این گونه که یک ست از داده های رندوم که جواب نهایی واقعی آن ها را میدانیم را به ماشین بدهیم و در خروجی آن تشخیص ماشین را برای هر داده ببینیم و میزان صحت و دقت کار ماشین را بدین طریق ببینیم.

بخش سوم

در این قسمت با کمک لینک زیر داده هایی مربوط به وضعیت دسته ای از افراد بدست آمده است.

Index of /ml/machine-learning-databases/adult (uci.edu)

در این داده ها برای هر فرد ۱۵ ویژگی ذکر شده است. این ویژگی ها مربوط به شغل، سن، محل زندگی، مدت زمان خانه بودن و ... در افراد است. خروجی مورد نظر وزن افراد است. در این جمع آوری داده ها هدف آن بوده که اعلام کنند افرادی که سبک و طریقه زندگی یکسانی دارند، رنج وزن نسبتا یکسانی دارند.

برای مرحله Training از 32561 داده موجود در فایل adult.scv استفاده شده است. خروجی مورد نظر یا به شکل >50K و یا به شکل <=50K است. در اصل قصد آن بوده که به ماشین یاد دهیم با داشتن اطلاعاتی از یک فرد تشخیص دهد وزن کمتر از 50 kg دارد و یا آنکه وزن فرد بیشتر و یا مساوی 50 kg است.

توضیحات دقیق تر انواع ویژگی افراد در فایل adult.txt موجود است.

Probability for the label '>50K' : 23.93% / 24.78% (without unknowns)

Probability for the label '<=50K' : 76.07% / 75.22% (without unknowns)

باید توجه داشته باشیم که در این ست از داده ها تعداد افراد با وزن بالای 50 kg به مراتب کمتر از افراد با وزن کمتر است. بنابراین انتظار میرود ماشین در تشخیص افراد با وزن بالا موفق تر باشد.

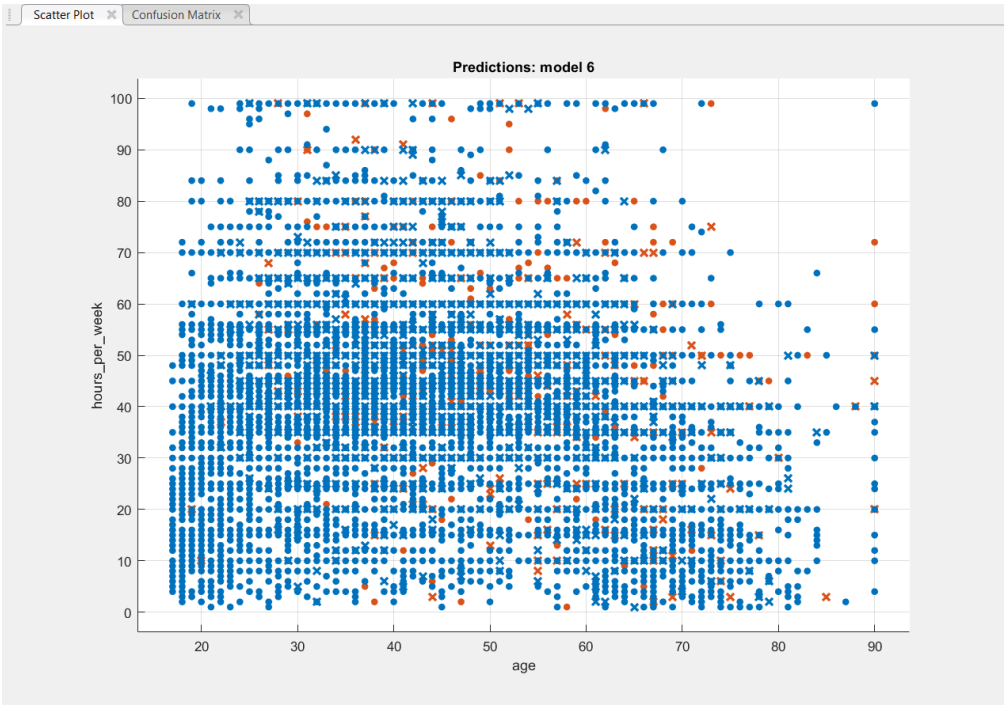
در تعیین مدل باید به این نکته توجه داشته باشیم که به دلیل کم بودن ویژگی ها برای تشخیص وزن (۱۴ عدد) در کل مساحت سخت افزاری و منابع مصرفی کم خواهد بود و نیازی به کاهش دادن آن نیست.

پس از آنکه داده ها را به بخش Classification Learner دادیم، با چندین روش مختلف ماشین را Train میکنیم. نتایج حاصل از accuracy برای روش های مختلف تست شده به شرح زیر است.

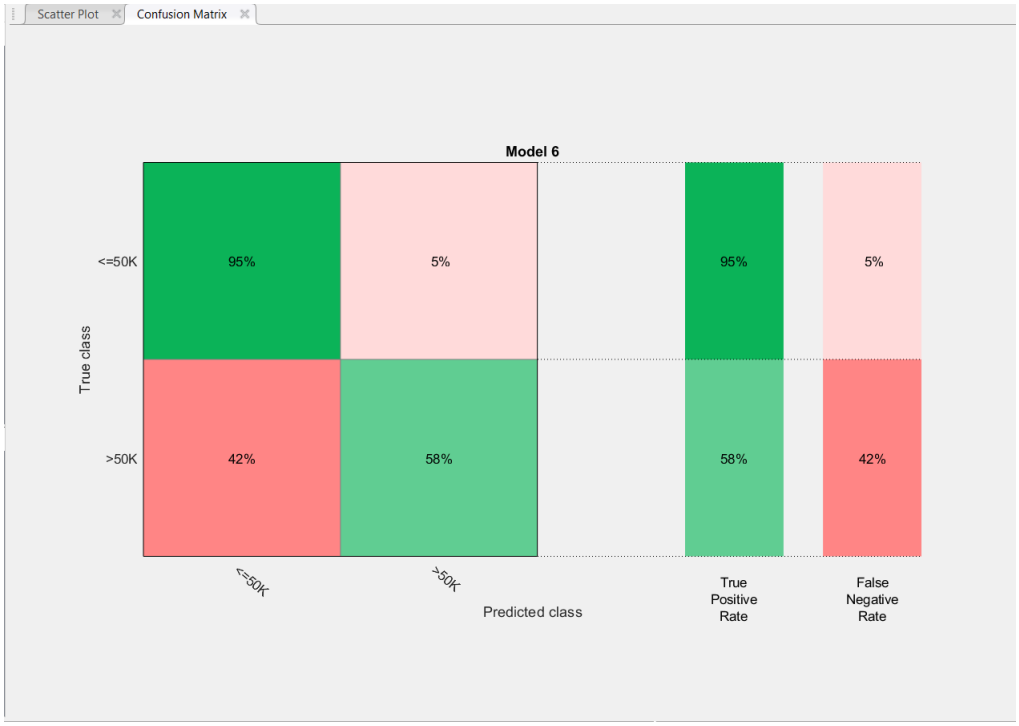
1 ☆ Tree Last change: Disabled PCA	Accuracy: 85.9% 14/14 features
2 ☆ Logistic Regression Last change: Logistic Regression	Accuracy: 85.2% 14/14 features
3 ☆ Naive Bayes Last change: Gaussian Naive Bayes	Accuracy: 83.3% 14/14 features
4 ☆ SVM Last change: Linear SVM	Accuracy: 85.1% 14/14 features
5 ☆ SVM Last change: Quadratic SVM	Accuracy: 85.4% 14/14 features
6 ☆ Ensemble Last change: Boosted Trees	Accuracy: **86.1%** 14/14 features

شکل ۱۴: نتایج accuracy پس از امتحان روش های مختلف train

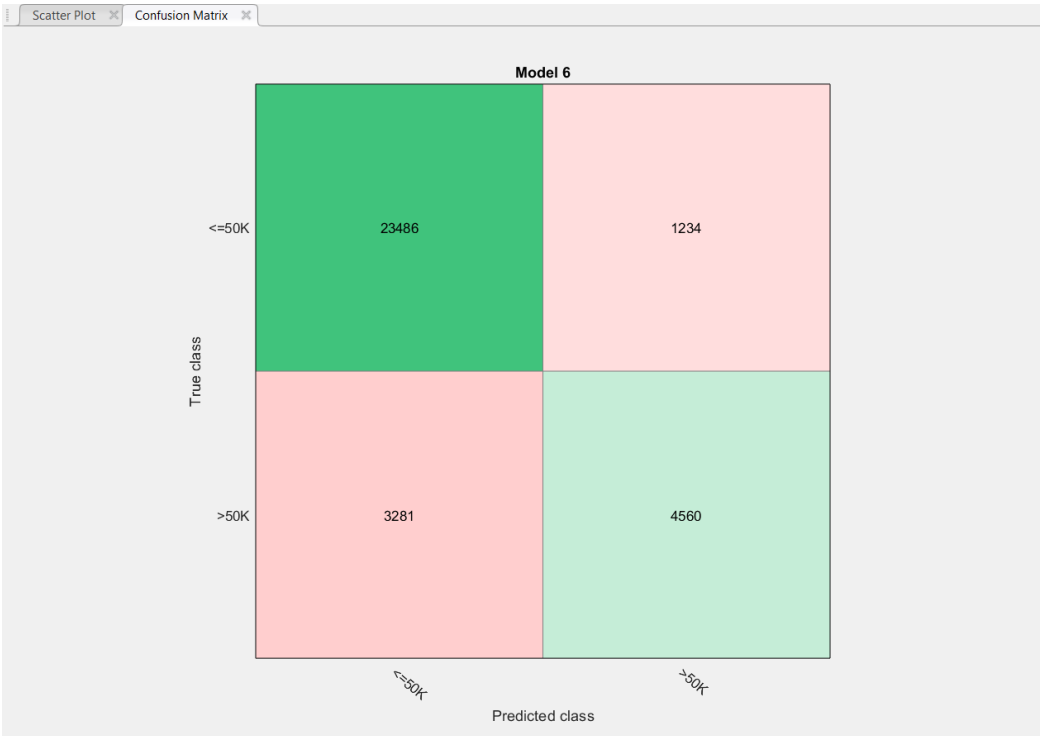
همانطور که مشخص است، بالاترین accuracy مربوط به روش Ensemble Boosted Trees است. حال confusion matrix و نمودار scatter plot را میبینیم.



شکل ۱۵: scatter plot مربوط به روش Ensemble



شکل ۱۶ : confusion matrix با درصد پیش از بهبود روش



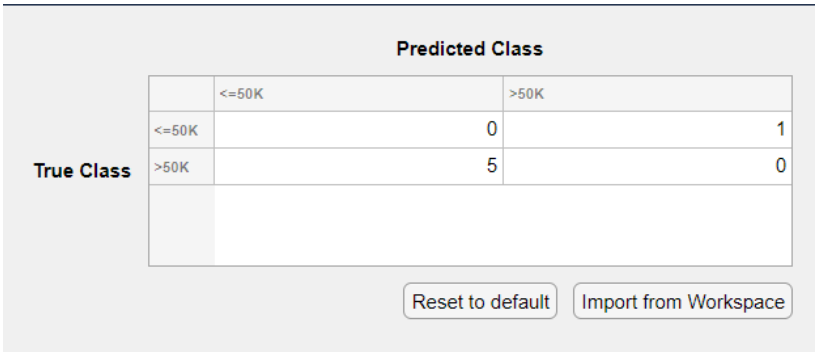
شکل ۱۷ : confusion matrix بامقدار پیش از بهبود روش

در نمودار scatter plot قسمت های مشخص شده با نقطه مربوط به افرادی است که تشخیص درست بر آنها توسط ماشین انجام شده و قست های ضربدر برای تشخیص های اشتباه ماشین است.

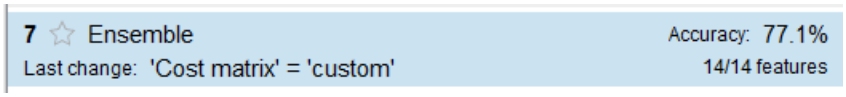
از نمودار confusion matrix میتوان متوجه شد که توانایی ماشین در تشخیص افراد با وزن کمتر یا مساوی 50 kg بسیار بیشتر از تشخیص افراد با وزن بالاتر است. علت آن کمتر بودن افراد موجود با وزن بیشتر از 50 kg نسبت به باقی است. حال بسته به آنکه زیاد بودن وزن مهم تر است و یا کمتر بودن آن میتوان تغییراتی در روش train ماشین انجام داد تا برای آن دسته با اهمیت بالاتر دقت بیشتری داشته باشد.

با توجه به نمودار confusion matrix برای 3281 نفر تشخیص اشتباه کم وزن بودن توسط ماشین را داشته ایم که نیاز است این مورد اصلاح شود.

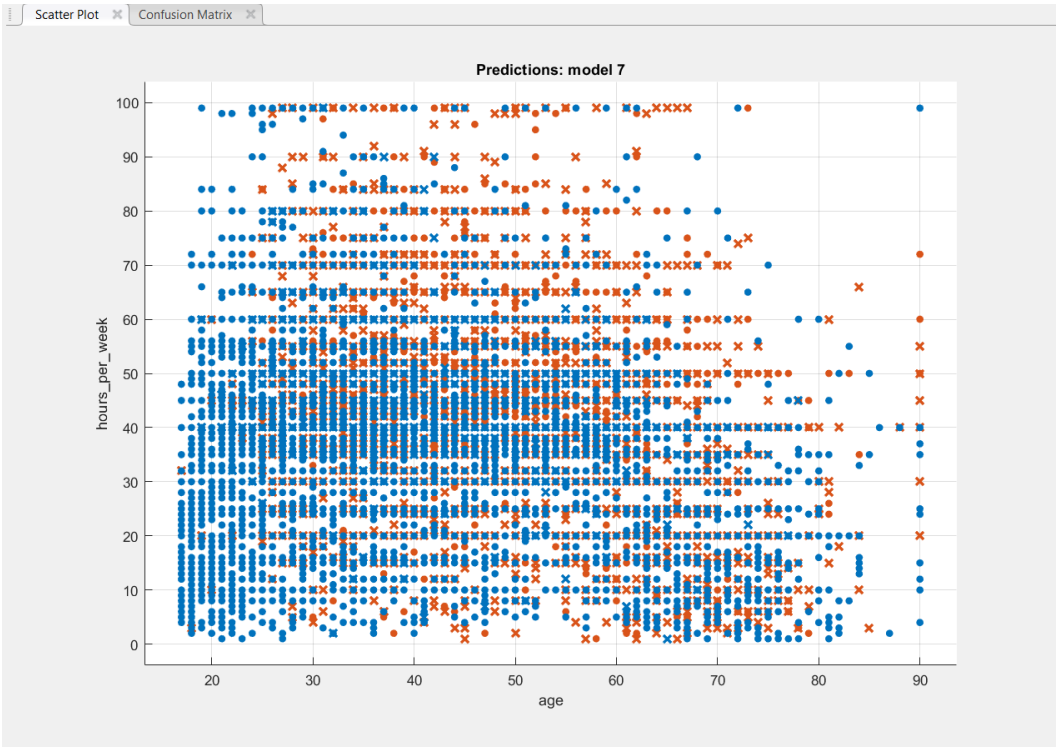
در ادامه برای بالاتر بردن دقت تشخیص افراد بالای 50 kg و رسیدن به یک تعادل نسبی در تشخیص درست، ماتریس Missclassification Costs را به شکل زیر در آوردیم. بدین ترتیب برای ماشین هزینه تشخیص غلط افراد با وزن بالا بیشتر از قبل میشود و در نهایت این بهبود نسبی مشهود است.



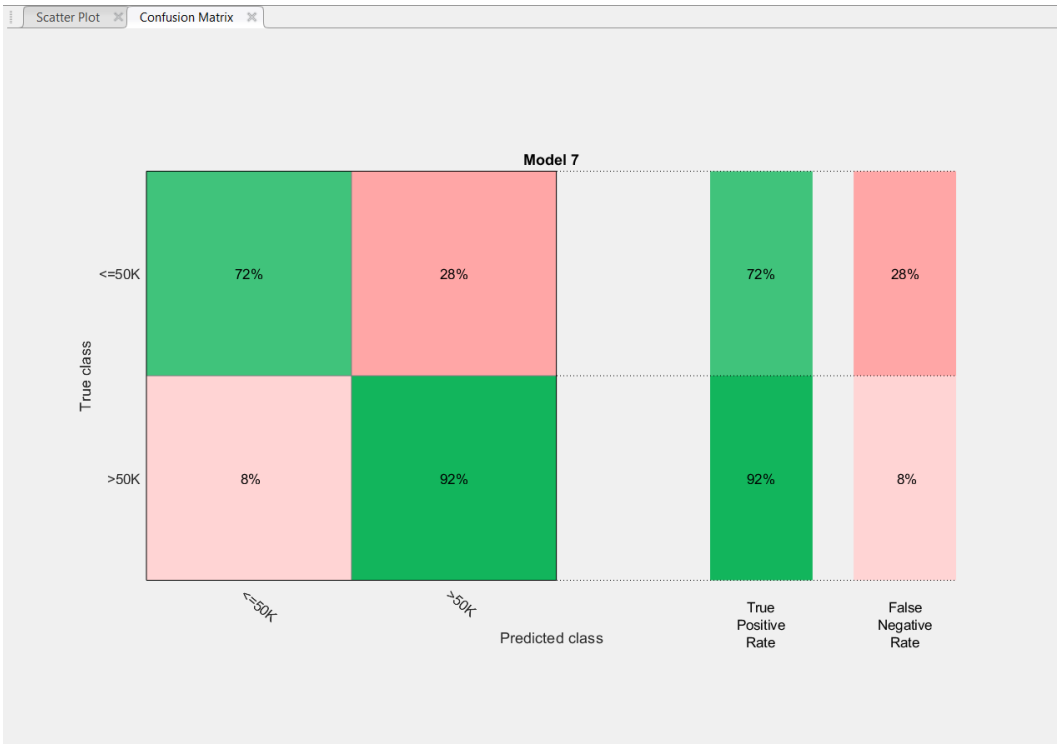
شکل ۱۸ : تنظیمات misclassification cost



شکل ۱۹ : میزان accuracy پس از بهبود متود



شکل ۲۰: scatter plot پس از بهبود روش



شکل ۲۱ : confusion matrix با درصد پس از بهبود روش

همانطور که انتظار میرفت با این تغییر درصد تشخیص درست افراد با وزن بیشتر بالا تر رفته و به اندازه مطلوبی برای هر دو تشخیص صحیح افراد کم وزن و افراد پر وزن رسیده ایم. از آنجا که باید به نحوی هزینه برای این افزایش درصد تشخیص درست افراد پر وزن داده باشیم، با توجه به میزان accuracy میتوان گفت accuracy کاسته شده است و به 77.1% رسیده است.

حال با کمک گزینه ی generate function از ماشین مد نظر و مطلوبمان تابع ای به دست می آوریم تا از آن برای انجام تست روی سری داده با نام adult_test.csv استفاده کنیم.در این مرحله عملیات validation را به نوعی بر روی ماشین پیاده میکنیم و چک میکنیم که درصد های TN و TPو FN و FP چه میزان با نتایج مربوط به train و test آن تشابه دارد. این دیتا ست دارای اطلاعات 16281 فرد است. برای انجام این کار نتایج داده های adult_test را در ماتریسی به نام actual_result ذخیره میکنیم و در table دیگری داده های بدون پاسخ را قرار میدهیم و آن table جدید را به ورودی تابع trainedClassifier.predictFcn() تشکیل شده میدهیم.

پس از آماده شدن نتایج، تعداد پاسخ های 50K< و 50K=> میشماریم و به ترتیب کد موجود و با کمک جواب های ذخیره شده در ماتریس actual_result درصد TN و TP و FN و FP را مشخص میکنیم. سپس به کمک درصد های به دست آمده sensivity و specificity را محاسبه و گزارش میکنیم.

در این محاسبات برای افراد با وزن تشخیص داده شده ی کمتر از 50Kg نتیجه ی positive به دست می آید و برای افراد با وزن بیشتر از 50Kg نتیجه ی negative در نظر گرفته میشود. برای مثال TP درصد افرادی را میدهد که واقعا کم وزن اند و کم وزن توسط ماشین تشخیص داده شده اند و TN درصد افرادی را میدهد که واقعا پر وزن اند و پر وزن توسط ماشین تشخیص داده شده اند.

به کمک تکه کد های زیر تعداد افراد واقعا با وزن کم و افراد واقعا با وزن بالا را به دست می آوریم.
و در ادامه با چک کردن نتایج محاسبات ماشین و شمارش آن ها، و چک کردن صحت آن ها خواسته های سوال ۴ را به دست می آوریم.

```
%% part 4 calculations:
actual_low_weight_num = 0;
actual_high_wight_num = 0;
for i = 1:(length(machine_prediction))
    if(isequal(actual_result{i,1},'<=50K.'))
        actual_low_weight_num = actual_low_weight_num + 1;
    end
    if(isequal(actual_result{i,1},'>50K.'))
        actual_high_wight_num = actual_high_wight_num + 1;
    end
end

TN = 0; TP = 0; FN = 0; FP = 0;
for i = 1:(length(machine_prediction))
    if(isequal(actual_result{i,1},'<=50K.') && isequal(machine_prediction{i,1},'<=50K'))
        TP = TP +1;
    end
    if(isequal(actual_result{i,1},'>50K.') && isequal(machine_prediction{i,1},'>50K'))
        TN = TN +1;
    end
    if(isequal(actual_result{i,1},'<=50K.') && isequal(machine_prediction{i,1},'>50K'))
        FN = FN +1;
    end
    if(isequal(actual_result{i,1},'>50K.') && isequal(machine_prediction{i,1},'<=50K'))
        FP = FP +1;
    end
end
```

شکل ۲۲ : کد متلب شمارش هر دسته از تشخیص ها و مقادیر درست

```
TNpercentage = (TN/actual_high_wight_num)*100;
TPpercentage = (TP/actual_low_weight_num)*100;
FPpercentage = (FP/actual_high_wight_num)*100;
FNpercentage = (FN/actual_low_weight_num)*100;

sensitivity = TP/(TP + FN);
specificity = TN/(TN + FP);
```

شکل ۲۳ : کد محاسبه درصد های خواسته شده

```
TN = 3527
TP = 9047
FN = 3388
FP = 319
TN percentage = 9.170567e+01
TP percentage = 7.275432e+01
FN percentage = 2.724568e+01
FP percentage = 8.294332e+00
sensitivity = 7.275432e-01
specificity = 9.170567e-01
```

شکل ۲۴ : نتایج خواسته شده

همانطور که مشخص است نتایج حاصل از تست کردن ماشین بر روی داده های جدید بسیار نزدیک به نتایج به دست آمده در مرحله ی train است. پس توانسته ایم ماشین ای را train کنیم که با حساسیت 0.727 و با اختصاصیت 0.917 جواب هایی در ازای ورودی اش تعیین میکند.

در قسمت train و test درصد TN برابر 92% بود و حال با داده های validation مقدار TN برابر 91.7% شده که بسیار نزدیک اند. در قسمت train و test درصد TP برابر 72% بود و حال با داده های validation مقدار TP برابر 72.7% شده که بسیار نزدیک اند. در قسمت train و test درصد FN برابر 28% بود و حال با داده های validation مقدار FN برابر 27.2% شده که بسیار نزدیک اند. در قسمت train و test درصد FP برابر 8% بود و حال با داده های validation مقدار FP برابر 8.2% شده که بسیار نزدیک اند.

در کل میتوان گفت یک ماشین را به طریقه صحیح ای با کمک متود Ensemble Boosted Trees ، train کرده ایم که با دقت مطلوبی افراد با وزن بالاتر از 50Kg و افراد با وزن کمتر از 50Kg را از هم متمایز کند.

راهنمای فایل های آپلود شده:

- در پوشه matlabCodes در فایل p3.m خواسته های مربوط به سوال ۳ و ۴ موجود است.
- در پوشه matlabCodes در فایل trainClassifier کد متلب تولید شده برای train ماشین موجود است.
- در پوشه matlabCodes در پوشه adult همان دیتاست انتخابی و دانلود شده به همراه توضیحاتش در adult.txt موجود است.