دانشکده مهندسی برق و کامپیوتر

تمرین دوم یادگیری ماشین

زهرا ریحانیان

شماره دانشجویی: ۸۱۰۱۰۱۱۷۷

۱- برای بدست آوردن مقادیر بهینه پارامتر های مدل که θ است، باید از $R(\theta)$ نسبت به θ مشتق بگیریم. به صورت زیر عمل می کنیم:

$$\frac{\partial}{\partial\theta}(R(\theta)) = \frac{\partial}{\partial\theta}[\frac{1}{2N}||y - X\theta||_2^2 + \theta^T H\theta + \theta^T\theta + a^T\theta]$$

در اینجا عبارت $||y - X\theta||_2^2$ را باز می کنیم و داریم:

$$\frac{\partial}{\partial\theta}\big(R(\theta)\big) = \frac{\partial}{\partial\theta}\Big[\frac{1}{2N}(y - X\theta)^T(y - X\theta) + \theta^T H\theta + \theta^T\theta + a^T\theta\Big]$$

در عبارت $(y - X\theta)^T$، T را با توجه به خواص ترانهاده به داخل پرانتز می بریم:

$$\frac{\partial}{\partial\theta}\big(R(\theta)\big) = \frac{\partial}{\partial\theta}\Big[\frac{1}{2N}(y^T - \theta^T X^T)(y - X\theta) + \theta^T H\theta + \theta^T\theta + a^T\theta\Big]$$

حال حاصل $(y^T - \theta^T X^T)(y - X\theta)$ را محاسبه میکنیم و داریم:

$$\frac{\partial}{\partial\theta}\big(R(\theta)\big) = \frac{\partial}{\partial\theta}\Big[\frac{1}{2N}(y^T y - y^T X\theta - \theta^T X^T y + \theta^T X^T X\theta) + \theta^T H\theta + \theta^T\theta + a^T\theta\Big]$$

مشتق گیری را انجام می دهیم:

$$\frac{\partial}{\partial\theta}\big(R(\theta)\big) = \frac{1}{2N}(-y^T X - X^T y + 2X^T X\theta) + 2H\theta + 2\theta + a$$

برای بدست آوردن مقادیر بهینه باید حاصل مشتق بالا را مساوی صفر قرار دهیم:

$$\frac{1}{2N}(-y^T X - X^T y + 2X^T X\theta) + 2H\theta + 2\theta + a = 0$$

میدانیم $y^T X = X^T y$. پس داریم:

$$\frac{1}{2N}(-2X^T y + 2X^T X\theta) + 2H\theta + 2\theta + a$$

$$= \frac{1}{N}(-X^T y + X^T X\theta) + 2H\theta + 2\theta + a = 0$$

بنابراین θ به صورت زیر خواهد بود:

$$\theta = \left(\frac{X^T X}{N} + 2H + 2I\right)^{-1}(\frac{X^T y}{N} - a)$$

۲- الف) Regularization تکنیکی است که عملکرد مدل های رگرسیون خطی معمولی را بهبود می بخشد و فرایندی است که اطلاعاتی را اضافه میکند و در واقع محدودیتی را قرار می دهد تا از overfitting جلوگیری کند. در ادامه متداول ترین تکنیک های regularization را معرفی میکنیم:

L1 regularization (Lasso regularization):

مخفف Least Absolute Shrinkage and Selection Operator است. پنالتی L1 را به تابع هزینه اضافه می کند. L1 مجموع قدرمطلق های ضرایب است. در واقع تابع هزینه را به صورت زیر تغییر می دهد:

$$J(m) = \sum_{i=0}^{n} (\hat{y} - y_i)^2 + \lambda\, |slope|$$

ترم اضافه ی λ|slope| باید حداقل شود که تابع هزینه مینیمم شود. به حداقل رساندن slope به این معنی است که خط را کمتر شیب دار می کند و این باعث می شود که خط از تمام نقاط داده های آموزش عبور نکند و از overfitting جلوگیری می کند.

L2 regularization (Ridge regularization):

پنالتی L2 را به تابع هزینه اضافه می کند. L2 برابر مجموع مجذور ضرایب است. تغییری که در تابع هزینه ایجاد می کند به صورت زیر می باشد:

$$J(m) = \sum_{i=0}^{n} (\hat{y} - y_i)^2 + \lambda\, (slope)^2$$

در اینجا ترم پنالتی می تواند به صفر نزدیک شود اما چون ضرایب را به توان ۲ می رساند صفر نخواهد شد.

تفاوت ها:

۱- ترم پنالتی در L1 regularization برابر ضرب lambda در مجموع قدرمطلق های ضرایب است اما در L2 regularization برابر ضرب lambda مجموع مجذور ضرایب است.
۲- ضرایب در L1 regularization ممکن است به صفر کاهش یابد اما در L2 regularization این گونه نیست.
۳- می توان از L1 regularization برای انتخاب ویژگی استفاده کرد چون بعضی از ضرایب را ممکن است صفر کند اما L2 regularization این گونه نیست.

۴- L1 regularization از نظر محاسباتی فشرده تر از L2 regularization است.

ب)

$$J = \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda||\beta||_2^2$$

باید از J نسبت به β مشتق گرفت:

$$\frac{\partial J}{\partial \beta} = \frac{\partial}{\partial \beta}(\sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda||\beta||_2^2)$$

$$\frac{\partial J}{\partial \beta} = \frac{\partial}{\partial \beta}(\sum_{i=1}^{n} ({Y_i}^2 - 2Y_i X_i\beta + {X_i}^2\beta^2) + \lambda||\beta||_2^2)$$

$$\frac{\partial J}{\partial \beta} = \sum_{i=1}^{n} \left(-2Y_i X_i + 2{X_i}^2\beta\right) + 2\lambda\beta = 0$$

$$\sum_{i=1}^{n} (-2 X_i Y_i) + \sum_{i=1}^{n} 2{X_i}^2\beta + 2\lambda\beta = -2A^T Y + 2\beta A^T A + 2\lambda\beta = 0$$

$$\beta = (A^T A + \lambda I)^{-1} A^T Y$$

۳- الف) می توان مقدار بایاس که همان w_{k0} را به داخل summation برده و یک ستون تماما ۱ به X اضافه کنیم که محاسبات ساده تر شود و به فرم بسته بتوانیم بنویسیم. پس داریم:

$$P(Y = y_k|X) \propto exp(\sum_{i=0}^{d} w_{ki}X_i) \; for \; k = 1, \dots, K-1$$

نسبت احتمال هر کلاس نسبت به احتمال کلاس K محاسبه می کنیم و ln میگیریم و داریم:

$$ln\frac{P(Y = y_1|X)}{P(Y = y_K|X)} = \beta_1 . X_i$$

$$ln\frac{P(Y = y_2|X)}{P(Y = y_K|X)} = \beta_2 . X_i$$

……..

$$ln\frac{P(Y\ =\ y_{K-1}|X)}{P(Y\ =\ y_K|X)} =\ \beta_{K-1}\ .X_i$$

برای هر معادله، طرفین را به توان e می رسانیم و داریم:

$$P(Y\ =\ y_1|X) =\ P(Y\ =\ y_K|X)e^{\beta_1\ .X_i}$$
$$P(Y\ =\ y_2|X) =\ P(Y\ =\ y_K|X)e^{\beta_2\ .X_i}$$
$$\ldots\ldots$$
$$P(Y\ =\ y_{K-1}|X) =\ P(Y\ =\ y_K|X)e^{\beta_{K-1}\ .X_i}$$

با توجه به این قضیه که مجموع احتمال ها برابر ۱ می باشد، داریم:

$$P(Y\ =\ y_K|X) =\ 1 - \sum_{k=1}^{K-1} P(\ Y = y_k|X) =\ 1 - \sum_{k=1}^{K-1} P(\ Y = y_K|X)e^{\beta_k\ .X_i}$$

$$P(Y\ =\ y_K|X) = \frac{1}{1 + \sum_{k=1}^{K-1} e^{\beta_k\ .X_i}}$$

می توانیم از این برای بدست آوردن سایر احتمال ها نیز استفاده نمود:

$$P(Y\ =\ y_1|X) = \frac{e^{\beta_1\ .X_i}}{1 + \sum_{k=1}^{K-1} e^{\beta_k\ .X_i}}$$

$$P(Y\ =\ y_2|X) = \frac{e^{\beta_2\ .X_i}}{1 + \sum_{k=1}^{K-1} e^{\beta_k\ .X_i}}$$

....

$$P(Y\ =\ y_{K-1}|X) = \frac{e^{\beta_{K-1}\ .X_i}}{1 + \sum_{k=1}^{K-1} e^{\beta_k\ .X_i}}$$

به صورت عمومی می توان به صورت زیر نوشت:

$$P(Y\ =\ y_k|X) = \frac{e^{\beta_k\ .X_i}}{1+\sum_{j=1}^{K-1} e^{\beta_j\ .X_i}}\ \text{for k = 1,…, K-1}$$

ب) قانون طبقه بندی به این صورت خواهد بود که اگر قرار باشد کلاسی را به داده ای نسبت دهیم باید شرایط زیر برقرار باشد:

$$P(Y\ =\ y_k|X) \geq\ P(Y\ =\ y_i|X)\ for\ k = 1, \ldots, K\ and\ i \neq k$$

۴- الف) معادله خط رگرسیون خطی را به این صورت در نظر می گیریم:

$$y =\ \beta_1 x +\ \beta_0$$

در رگرسیون خطی هدف آن است که کمترین خطا را داشته باشیم. یعنی عبارت زیر کمینه شود:

$$S = \sum_{i=0}^{n} (y_i - (\beta_1 x_i + \beta_0))^2$$

به همین علت برای بدست آوردن β_0 و β_1 باید از S نسبت به آنها مشتق گرفت. ابتدا نسبت به β_1 مشتق میگیریم:

$$\frac{\partial S}{\partial \beta_1} = \frac{\partial}{\partial \beta_1} \sum_{i=1}^{n} (y_i - \beta_1 x_i - \beta_0)^2$$

$$\frac{\partial S}{\partial \beta_1} = \sum_{i=1}^{n} [2(y_i - \beta_1 x_i - \beta_0)(-x_i)]$$

$$\sum_{i=1}^{n} [2(y_i - \beta_1 x_i - \beta_0)(-x_i)] = 0$$

$$\sum_{i=1}^{n} y_i x_i - \beta_1 \sum_{i=1}^{n} {x_i}^2 - \beta_0 \sum_{i=1}^{n} x_i = 0$$

حال نسبت به β_0 مشتق میگیریم:

$$\frac{\partial S}{\partial \beta_0} = \frac{\partial}{\partial \beta_0} \sum_{i=1}^{n} (y_i - \beta_1 x_i - \beta_0)^2$$

$$\frac{\partial S}{\partial \beta_0} = \sum_{i=1}^{n} [2(y_i - \beta_1 x_i - \beta_0)(-1)]$$

$$\sum_{i=1}^{n} [2(y_i - \beta_1 x_i - \beta_0)(-1)] = 0$$

$$\sum_{i=1}^{n} y_i - \beta_1 \sum_{i=1}^{n} x_i - \beta_0 \sum_{i=1}^{n} 1 = 0$$

$$\sum_{i=1}^{n} y_i - \beta_1 \sum_{i=1}^{n} x_i - \beta_0 n = 0$$

در نهایت به معادله های زیر میرسیم:

$$\beta_0 = \frac{\sum_{i=1}^{n} y_i \ - \beta_1 \sum_{i=1}^{n} x_i}{n}$$

$$\beta_1 = \frac{n \sum_{i=1}^{n} x_i y_i - \ \sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i}{n \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}$$

حال با جای گذاری نمونه ها در معادله های بدست آمده داریم:

$$\beta_0 = 21.7 \ , \ \beta_1 = 3.47$$

برای بدست آوردن σ^2 از فرمول زیر استفاده می کنیم:

$$\sigma^2 = \frac{S}{n-2}$$

$$S = \sum_{i=0}^{n} (y_i - \beta_1 x_i - \ \beta_0)^2 = \ \sum_{i=0}^{10} (y_i - 3.47 x_i - \ 21.7)^2 = \ 223.07$$

$$\sigma^2 = \frac{223.07}{10-2} = 27.88$$

ب)

$$Var(\widehat{\beta_1}) = \frac{\sigma^2}{S_{xx}} \quad , \quad S_{xx} = \ \sum_{i=1}^{n} (x_i - \ \bar{x})^2$$

$$Var(\widehat{\beta_0}) = \frac{\sigma^2 \ \sum_{i=1}^{n} x_i^2}{n \ S_{xx}}$$

با جای گذاری خواهیم داشت:

$$Var(\widehat{\beta_1}) = 0.07$$

$$Var(\widehat{\beta_0}) = \ 10.20$$

ج) برای بدست آوردن مقدار کوریلیشن ابتدا نیاز داریم که کواریانس این دو پارامتر را بدست آوریم:

$$Cov(\beta_0, \beta_1) = \frac{-\sigma^2 \ \sum_{i=1}^{n} x_i}{n \ S_{xx}}$$

با جای گذاری خواهیم داشت:

$$Cov(\beta_0, \beta_1) = \ -0.74$$

فرمول کوریلیشن به صورت زیر می باشد:

$$\text{corr} = \frac{\text{Cov}(\beta_0, \beta_1)}{\sigma_{\beta_0} * \sigma_{\beta_1}}$$

که σ_{β_0} و σ_{β_1} انحراف معیار به ترتیب β_0 و β_1 می باشد. با جای گذاری خواهیم داشت:

$$\text{corr} = \frac{-0.74}{\sqrt{0.07}\,\sqrt{10.20}} = -0.85$$

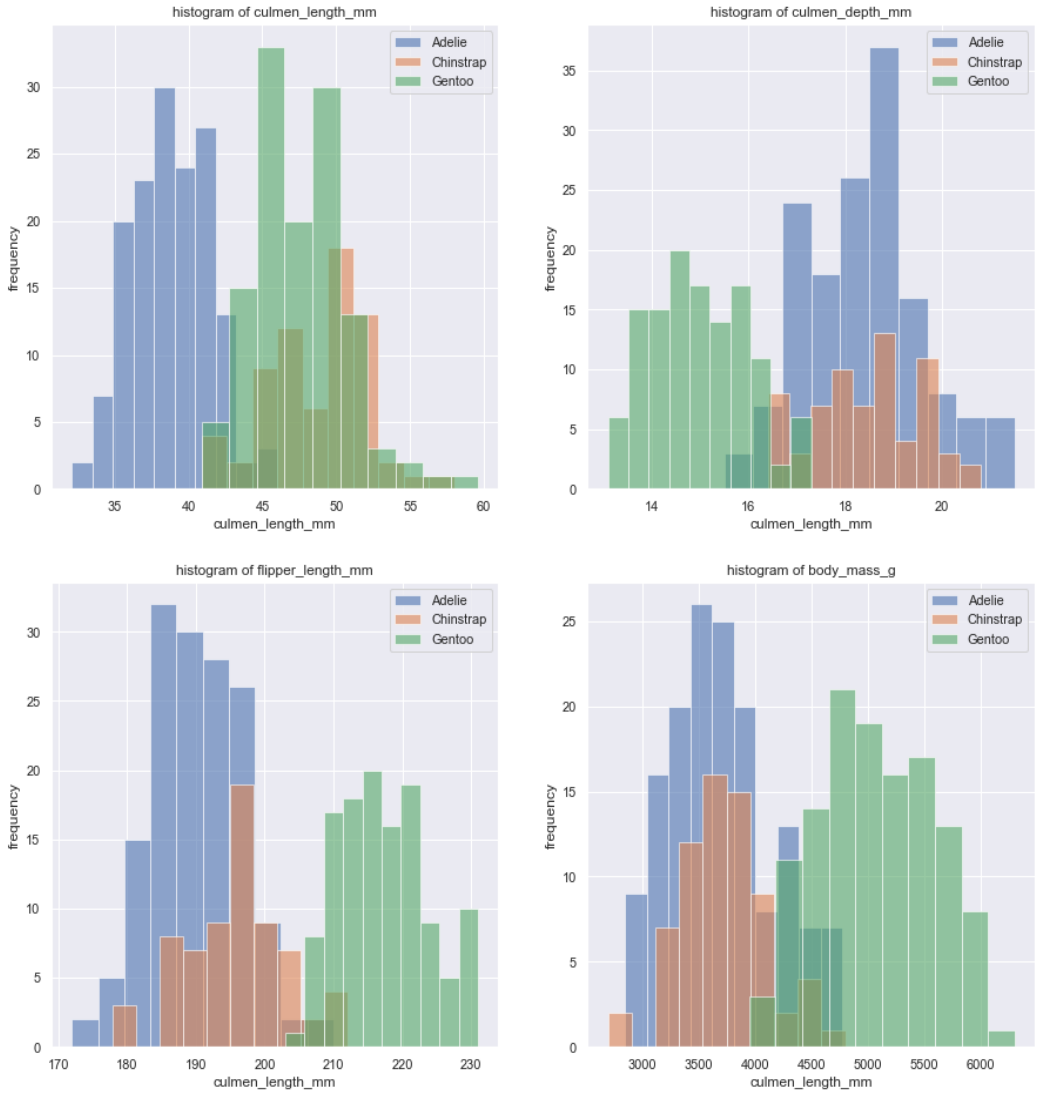
۵- جزئیات کدها به همراه کامنت ها در فایل جوپیتر نوت بوک موجود می باشد. در این جا توضیح مختصری در مورد کد ها و روند کار گفته خواهد شد.

الف) برای رسم نمودار نقاط، ویژگی ها[1]ی موجود که از ستون دوم به بعد هستند را در نظر گرفتم. در دو حلقه ی تو در تو، هر دو ویژگی را به صورت یکتا، یعنی دوتایی تکراری که فقط جایشان عوض شده باشد را در این پیمایش نداریم، بررسی کردم. یک حلقه هم به ازای هر کلاس قرار دادم که داده های مربوط به هر کلاس با رنگ متفاوتی از بقیه جدا شود. سپس لیبل های مربوط به هر محور و legend را قرار دادم و نمودار را نمایش دادم. به این ترتیب ۶ نمودار نقاط بدست آمد.

از بین این نمودار ها، نمودار زیر با دقت بیشتری داده ها را جدا می کند. دلیلش آن است که در این نمودار داده های مربوط به هر کلاس از کلاس های دیگر تقریبا جدا تر از بقیه نمودار ها و پراکندگی هر کلاس هم کمتر است و می توان گفت که داده ی مربوط به هر کلاس در همان ناحیه مربوط به خودش متمرکز تر است. بنابراین می توان با دقت بیشتری کلاس ها را جدا نمود.

[1] features

برای رسم هیستوگرام برای هر ویژگی، یک حلقه برای پیمایش ویژگی ها قرار دادم. سپس یک حلقه دیگر در درون آن برای پیمایش کلاس های مختلف قرار دادم که توزیع کلاس های مختلف در آن ویژگی را نمایش دهم. سپس عنوان نمودار و لیبل هر نمودار را ست کردم و legend قرار دادم و در انتها نمودار ها را نمایش دادم. هیستوگرام ها به صورت زیر بدست آمد:



شکل ۱ نمایش توزیع کلاس های مختلف به ازای هر ویژگی در هیستوگرام

ب) برای این قسمت، ابتدا لازم بود که الگوریتم طبقه بند logistic Regression را پیاده سازی کنم. در ابتدا یک تابع برای استاندارد سازی داده ها پیاده سازی کنم که داده های استاندارد را به این طبقه بند دهم. که روند استاندارد سازی این طور است که به ازای هر ویژگی، میانگین داده های آن ویژگی را از تک تک داده های مربوط به آن کم کرده و سپس حاصل را تقسیم بر انحراف معیار آن ویژگی می کنیم.

حال به شرح پیاده سازی الگوریتم Logistic Regression می پردازیم. این الگوریتم به این صورت است که داده های دو کلاسه را طبقه بندی می کند. برای این کار ابتدا برای هر ویژگی یک وزن اولیه در نظر می گیرد و یک ترکیب خطی را تشکیل میدهد. مثلا اگر ویژگی ها را X در نظر بگیریم و وزن آن ها در w باشد، به صورت $w^T X$ در می آید و یک مقدار bias هم در نظر می گیرد که با w_0 نشان می دهیم و به صورت زیر در می آید:

$$z = w^T X + w_0$$

در کد این الگوریتم هم ابتدا به وزن ها مقدار اولیه 0 می دهیم. همچنین برای اینکه بتوان به صورت فرم بسته و ساده تری محاسبات را انجام داد، یک ستون تمام ۱ به X اضافه می کنیم. بنابراین وزن ها هم یکی بیشتر در نظر می گیریم و چون این ستون تمام ۱ را به اول X اضافه کردیم، درایه ی اول weight همان bias خواهد بود. بنابراین z به صورت زیر خواهد شد:

$$z = w^T X$$

سپس در یک حلقه که تعداد تکرارش را خودمان تنظیم می کنیم به این صورت عمل می کند:
ابتدا با استفاده از تابع sigmoid احتمال کلاس ۱ بودن y را حساب میکند. فرمول تابع sigmoid که در کد هم موجود می باشد به صورت زیر می باشد:

$$g(z) = \frac{1}{1 + e^{-z}}$$

با جای گذاری z داریم:

$$h_w(x) = \frac{1}{1 + e^{-w^T X}}$$

سپس خطا که اختلاف بین حاصل مرحله قبل و y آموزش که به عنوان ورودی دریافت کرده است را حساب می کند. حاصل ضرب این مقدار خطا با X.T را بدست می آوریم و در aplpha که خودمان تنظیم می کنیم ضرب می کنیم و از وزن هایی که برای ویژگی ها داریم، کم می کنیم و وزن ها را به روز می کنیم. در نهایت با استفاده از تابع cost هزینه مدل را حساب می کنیم. این تابع بر پایه ی فرمول زیر است (m اندازه نمونه است):

$$J(w) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} log\ h_w(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_w(x^{(i)}))]$$

در واقع مراحلی که برای بروز رسانی w انجام دادیم، همان مشتق $J(w)$ بر حسب w است، برای اینکه هزینه را حداقل کند.

این از شرح حلقه که بیان شد و تعداد تکرار آن را خودمان تعیین می کنیم که من در اینجا مقدار پیش فرض را ۴۰۰ در نظر گرفتم.

در اینجا ابتدا به پیش پردازش های لازم برای داده های آموزش و تست می پردازم. ابتدا مقادیر *NaN* را از دیتاست حذف کردم و *Index* آن را بازنشانی کردم. سایز داده آموزش را ۸۰ درصد کل داده ها در نظر گرفتم و بقیه را برای تست کنار می گذارم. برای انتخاب داده های آموزش به صورت عمل کردم که به طور تصادفی و البته به صورت یکتا (یعنی عدد تکراری تولید نکند)، به اندازه ۸۰ درصد داده ها عدد که از ۰ تا سایز نمونه است، تولید کردم که این اعداد را همان *index* داده های آموزش در نظر می گیرم. پس بقیه داده ها را برای تست جدا می کنم.

چون طبقه بند *logistic regression* برای داده های دو کلاسه است، از تکنیک *one against all* استفاده می کنیم که بتوانیم از این طبقه بند استفاده کنیم. برای این کار من سه مدل جدا ایجاد کردم که در هر کدام یکی از کلاس ها برچسب ۱ دارد و بقیه برچسب ۰ دارند. مدل را روی هر کدام فیت می کنم. سپس تابع *probability* را روی هر کدام صدا می زنم. کار این تابع به این صورت است که احتمال ۱ بودن کلاس را با استفاده از تابع *sigmoid* و وزن هایی که در مرحله فیت بدست آورده، محاسبه می کند. بعد از این مرحله، برای هر عضو نمونه، آن کلاسی انتخاب می شود که احتمال بالاتری دارد که با استفاده از یک حلقه *for* و *argmax* این عمل انجام می شود.

در مرحله بعد ماتریس آشفتگی و معیار های کلاس بندی با توجه به فرمول هایی که در کد هم اشاره شده، محاسبه شد.

ماتریس آشفتگی ای که بدست آوردم به صورت زیر می باشد:

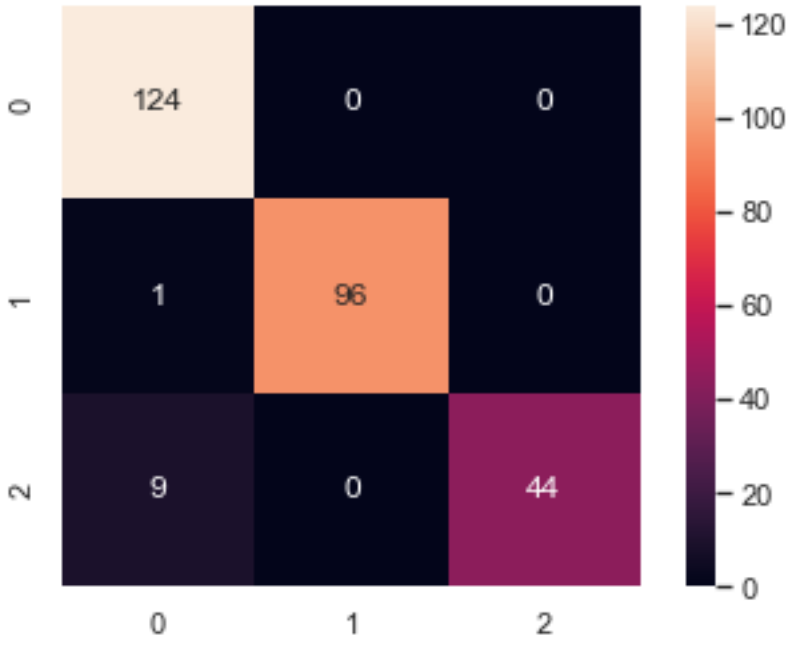
```
array([[34,  1,  1],
       [ 0, 22,  0],
       [ 1,  0, 10]])
```

معیار های دقت، *recall*، *precision*، *jaccard* و *f1 score* به شرح زیر می باشد:

```
accuracy = 0.9565217391304348
recall = [0.9444444444444444, 1.0, 0.9090909090909091]
precision = [0.9714285714285714, 0.9565217391304348, 0.9090909090909091]
jaccard = [0.918918918918919, 0.9565217391304348, 0.8333333333333334]
f1_score = [0.9577464788732395, 0.9777777777777777, 0.9090909090909091]
```

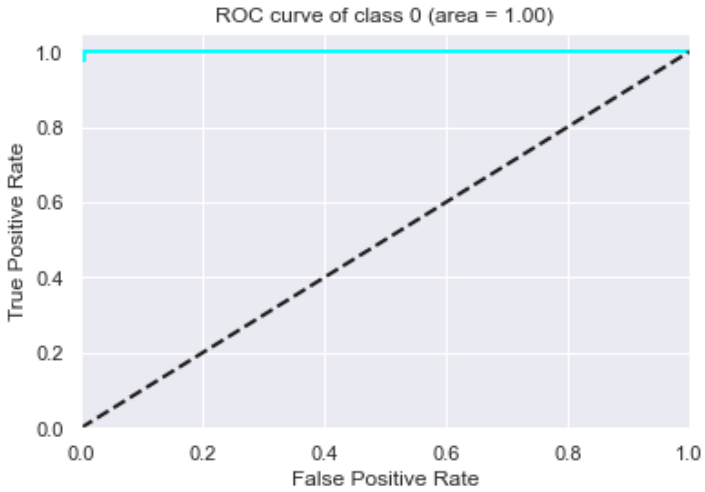
ج) در این قسمت با استفاده از کتابخانه *scikit-learn* داده ها جداسازی و سپس با استفاده از *logistic regression* با تکنیک *one vs rest* کلاس بندی شدند. سپس ماتریس آشفتگی و معیار های کلاس بندی با استفاده از توابع این کتابخانه محاسبه شدند که به شرح زیر می باشند:

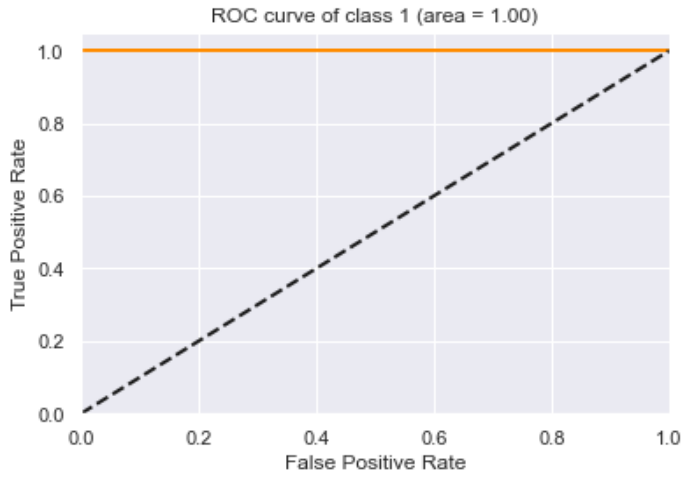
```
accuracy = 0.9635036496350365
recall = [1.         0.98969072 0.83018868]
precision = [0.92537313 1.         1.        ]
jaccard = [0.92537313 0.98969072 0.83018868]
f1_score = [0.96124031 0.99481865 0.90721649]
```

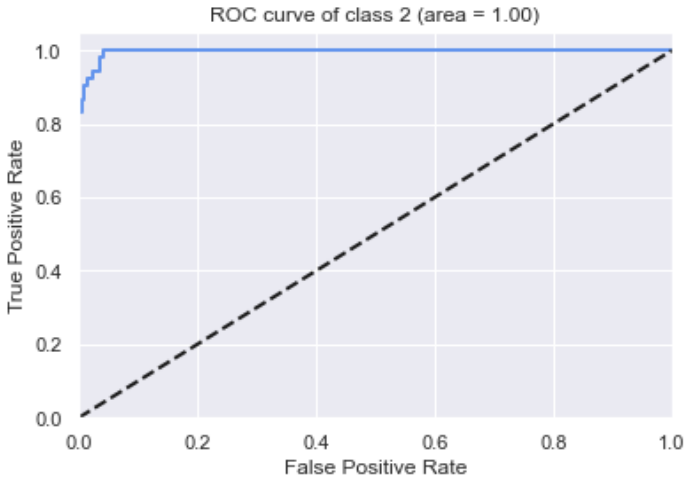


شکل ۲ ماتریس آشفتگی با استفاده از کتابخانه scikit-learn

در مرحله بعد با استفاده از همین کتابخانه برای هر کلاس نمودار ROC آن را رسم کردم که نتیجه آن به صورت زیر شد:







۶- در این سوال چون داده ها به صورت خطی جدا پذیر نیستند لازم است که آن ها به به فضای با مرتبه بالاتر ببریم. برای این کار از دو حلقه تو در تو استفاده کردم. به این ترتیب ستون هایی به دیتاست اضافه می شوند که هر کدام یکی از فرم های موجود در f(X) موجود در صورت سوال است. لیبل ستون ها هم به این صورت نام گذاری کردم که عدد اول توان عدد اول و سپس یک کاما و عدد بعدی توان عدد دوم باشد. الگوریتم logistic regression را به همان صورتی که در سوال ۵ توضیح دادم، پیاده سازی کردم. با این تفاوت که در اینجا از L2 Regularization استفاده کردم که این موجب اندکی تفاوت در محاسبه هزینه (cost) و مرحله به روز رسانی وزن ها می شود. در اینجا از تابع predict استفاده کردم که بعد از محاسبه ی احتمال کلاس ۱ بودن که با تابع sigmoid بدست می آورد، آن را با یک threshold که در اینجا ۰.۵ است مقایسه می کند. اگر بزرگتر مساوی ۰.۵ باشد کلاس ۱ وگرنه کلاس ۰ است.

مراحل جداسازی داده به آموزش و تست، مشابه سوال ۵ قسمت ب انجام شده است. بعد از فیت کردن داده های آموزش، داده های تست را با استفاده از تابع predict کلاس بندی می کنیم. سپس ماتریس آشفتگی و معیار های کلاس بند را بدست می آوریم که به صورت زیر بدست آمد:

ماتریس آشفتگی:

```
array([[10,  1],
       [ 2, 11]])
```

معیار های کلاسبند:

```
accuracy = 0.875
recall = 0.8461538461538461
precision = 0.9166666666666666
jaccard = 0.7857142857142857
f1_score = 0.8799999999999999
```

دقت این کلاس بند برابر 0.875 بدست آمد که این یعنی ۸۷.۵ درصد درست کلاس بندی را انجام داده و پیش بینی خوبی داشته است.

در مرحله بعد مرز تصمیم گیری را رسم کردم. برای این کار ابتدا ۵۰۰ داده برای x1 و x2 از 0.75- تا 1 تولید کردم و با استفاده از meshgrid آن ها را رند و در یک آرایه قرار دادم. سپس با استفاده از وزن هایی که از مرحله فیت بدست آوردم و توان هایی که روی هر ستون مشخص کردم، تابع F که بر حسب X1 و X2 است را بدست آوردم. در مرحله بعد ابتدا نمودار نقاط هر کلاس را رسم کردم و سپس مرز تصمیم گیری را با استفاده از تابع contour موجود در کتابخانه matplotlib رسم کردم که به صورت زیر نتیجه حاصل شد:

شکل ۳نمایش مرز تصمیم گیری