به نام خدا

دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

یادگیری ماشین

تکلیف چهارم

استاد درس: دکتر ناظرفرد

امیرحسین کاشانی

۴۰۰۱۳۱۰۷۱

amkkashani@gmail.com

نیم سال اول ۱۴۰۱-۱۴۰۲

فهرست

Q1

a)

روش‌های ensemble روش‌هایی هستند که با ترکیب estimatorهای متخلف و رای گیری بین آن‌ها سعی در رسیدن به generalizability / robustness در مدل خود دارند. و به دو دسته boosting , averaging تقسیم می‌شوند.

در روش‌های Averaging که Bagging نیز عضوی از آن‌ها است مدل‌های متفاوت مستقلا ساخته می‌شوند و در نهایت نظر آن‌ها باهم میانگین گرفته می‌شود. مدل نهایی عملکرد بهتری نسبت به هر یک مدل‌های ورودی خواهد داشت.(Averaging امکان موازی سازی را نیز دارد)

در روش‌های Boosting مدل به صورت sequential ایجاد می‌شود و امکان موازی سازی وجود ندارد و در هرگام سعی در بهبود نقاط یا فضایی از داده‌ها هستیم که در آن دقت یا کارآیی مدل ضعیف بوده است. این روش عموما از تجمیع مدل‌های ساده بدست می‌آید.

**استفاده از Bagging در مواجه با overfit کار آیی بیشتری نسبت به Boosting دارد و همچنین Bagging زمانی مناسب تر است که هر یک از base_estimatorها پیچیده به حساب بیایند ولی در Boosting عموما base_estimatorها ساده تر و ضعیف تر هستند.

b)

در گام اول می‌توان گفت که این کار در زمانی که داده‌های کم باشند ایجاد مشکل می‌کند و با حذف داده‌ها در هر انتخاب داده‌های کمتری برای ایجاد مدل‌ها خواهیم داشت و عدم وجود اشتراک بین داده‌های مدل‌های مختلف باعث می‌شود که خرد جمعی رخ ندهد بلکه هریک از مدل‌ها می‌تواند بر روی یک قسمت نظر دهد.

در گام دوم که جواب اصلی به سوال می‌باشد این است که در صورتی که انتخاب بدون جایگذاری باشد احتمال ورود داده‌ها به مدل‌های مختلف دیگر توزیع uniform (یکنواخت) نخواهد داشت و شانس حضور در مدل‌ها متفاوت خواهد بود و توزیع داده‌ها در مدل ما به ترتیب ایجاد مدل‌ها وابسته خواهد شد.

c)

این کار چندان روش مناسبی نیست.

با استفاده از مدل‌های پیچیده تر در روش‌های Boosting در اصل داریم از ماهیت Ensemble دور می‌شویم زیرا در Boosting با تمرکز بر اشتباهات مدل سعی در کاهش bias داریم و استفاده از مدل پیچیده از ابتدا bias پایینی دارد و اشتباهاتی کمی از این رو یا تعداد iteration کمی را باید اتخاذ کنیم یا اینکه مدل به سرعت overfit خواهد کرد. از این رو در Boosting بهتر است که مدل‌ها ساده باشند.

d)

برای حل این مساله به شکل دقیق نیازمند این هستیم که توزیع احتمالاتی بر روی نظرات اشتباه را داشته باشیم از این رو فرض می‌کنیم که حالات خطا به شکل یکنواخت در estimatorهای ما پخش شده است یعنی احتمال اینکه لیبل ۱ ، به اشتباه ۲ یا ۳ یا... تشخیص داده شود یکی است.

حال با این فرض اگر مدل ما صرفا رندوم تصمیم گیری می‌کرد صحت آن برابر ۱۰۰ تقسیم بر ۷ می‌شد که یعنی ۱۴.۲ ، پس مدل‌های ما از رندوم نیز ضعیف تر هستند.

در این شرایط اگر با اغماض ۱۴.۲ را برابر ۱۴ فرض کنیم در بهترین حالت مدل‌های ما رندوم هستند و رای گیری بین یک سری مدل رندوم یک مدل رندوم دیگر ایجاد می‌کند که در بهترین حالت همان ۱۴ درصد صحت را بر می‌گرداند ولی اگر بخواهیم دقیق تر بررسی کنیم مدل‌های ما در جهت منفی دارند رای گیری می‌کنند یعنی همگرایی و رای گیری آن‌ها بدلیل اینکه احتمال پاسخ درست کمتر از میانگین احتمالات اختصاص داده است، یک پاسخ غلط را به عنوان پاسخ نهایی پر رنگ می‌کند از این رو در این مورد به هیچ وجه نباید از رای گیری استفاده نمود.

e)

با توجه به اینکه Boosting بصورت sequential در حال اجرا است و در هر iteration نیازمند آن است که بر روی تمامی داده‌ها حرکت کند پس گزینه مناسبی برای دیتاست‌های بزرگ نیست اما از طرف دیگر Bagging می‌تواند بصورت موازی آموزش یابد و در نتیجه از لحاظ زمانی بهتر عمل می‌کند.

برای ابعاد زیاد نیز می‌توان گفت که مدل‌های ساده در نظر گرفته شده در Boosting شاید به نظر بیاید که ضعیف عمل کنند اما با توجه به اینکه می‌توان تعداد iterationها را بیشتر کرد بر این مساله نیز غلبه می‌کنند. در نتیجه خواهیم داشت

دیتاست بزرگ و ابعاد کم => Bagging

دیتاس و کوچک و ابعاد بزرگ => Boosting

f)

نظری ندارم

g)

g)

برای حل این مسأله باید دو معادله‌ی زیر برقرار باشند.

$$\begin{cases} \nabla f(x,y) = \lambda \nabla g(x,y) \\ g(x,y) = c \end{cases}$$

g-1)

$$\left.\begin{aligned} -2x &= \lambda \times 1 \\ -4y &= \lambda \times 1 \end{aligned}\right\} \rightarrow \left.\begin{aligned} x &= -\lambda/2 \\ y &= -\lambda/4 \end{aligned}\right\} \text{(I)} \qquad \begin{aligned} x &= 2/3 \\ y &= 1/3 \end{aligned}$$

$$x + y = 1 \xrightarrow{\text{(I)}} -3/4\,\lambda = 1 \rightarrow \lambda = \frac{-4}{3} \longrightarrow \text{(I)}$$

$$f(2/3, 1/3) = 2 - 4/9 - 2/9 = 12/9$$

g-2)

$$\left.\begin{aligned} 2x &= \lambda \times 1 \\ 2y &= \lambda \times 1 \end{aligned}\right\} \rightarrow x = y \qquad f(-1,-1) = 2$$

$$x = -1$$

$$y = -1$$

g-۳)

$$\left.\begin{array}{l} 3x^2 = 2x\lambda \\ 3y^2 = 2y\lambda \end{array}\right\} \longrightarrow \begin{array}{l} x=0 \\ y=0 \end{array} \text{ و یا } \quad x = y \rightarrow 3x = 2\lambda \rightarrow x = \frac{2\lambda}{3}$$

$$y = \frac{2\lambda}{3}$$

$$x^2 - 1 \geq 0 \rightarrow \frac{4}{9}\lambda^2 \geq 1 \rightarrow \begin{cases} \lambda > 3/2 \\ \lambda < 3/2 \end{cases}$$

x و y می‌توانند $-\infty$ و $-\infty$ باشند و معادلات پاسخ خاصی به

$f(-\infty, -\infty) = -\infty$ نمایش دهند.

h)

h)

با توجه به جایگیری نقاط

معادله‌ی خط بدست آمده از SVM برابر زیر می‌باشد

$x_2 = -x_1 + 3$ زیرا باید بر $x_2 - x_1$ عمود بوده و از نقاط آبی عبور کند.

i)

i)

kernel matrix → $K = K^T$ متقارن ، $\alpha^T K \alpha \geq 0$ Positive-semi-definite

i-1) $K(x,z) = k_1(x,z) + k_2(x,z)$

جمع دو ماتریس متقارن، متقارن خواهد شد و شرط positive semi-definite بودن را اثبات میکنیم.

$z^T K z = z^T (k_1 + k_2) z = z^T k_1 z + z^T k_2 z$ (I)

$z^T k_1 z \geq 0$ ، $z^T k_2 z \geq 0$ (I) → $z^T K z \geq 0$ ✓ positive semi-definite

بنابراین K معتبر است

i_۲) $k(x,z) = k_1(x,z) - k_2(x,z)$

تقارن مشابه مثال قبل برقرار است، positive semi-definite را نخواهد بود

مثال نقض ← k کرنل نیست

k_1 $\quad$ k_2 $\quad$ k

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

i_۳) $k(x,z) = \alpha k_1(x,z)$

شرط تقارن با توجه به اینکه α اسکالری باشد حفظ می‌شود

$$z^T k z = z^T(\alpha k_1) z = \alpha z^T k_1 z \Big\} \rightarrow \alpha z^T k_1 z \geq 0$$

$$z^T k_1 z \geq 0$$

به صورتی که $\alpha \geq 0$ کرنل معتبر است

PAPCO

مثال نقض برای حالتی که $\alpha < 0$ باشد

$\alpha = -1$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \alpha = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

k_1 $\quad$ k

i_۴) $k(x,z) = k_1(x,z)\, k_2(x,z)$

راه‌حل زیر ترجمه‌ی راه‌حل یافت شده در اینترنت می‌باشد

با توجه به فرض
$$\begin{cases} k_1(x,z) = (\phi^{(1)}(x))^T (\phi^{(1)}(z)) \\ k_2(x,z) = (\phi^{(2)}(x))^T (\phi^{(2)}(z)) \end{cases}$$

$$k(x,z) = k_1(x,z)\, k_2(x,z)$$

$$= \sum_i \phi_i^{(1)}(x)\, \phi_i^{(1)}(z) \sum_j \phi_j^{(2)}(x)\, \phi_j^{(2)}(z)$$

$$= \sum_i \sum_j \phi_i^{(1)}(x)\, \phi_i^{(1)}(z)\, \phi_j^{(2)}(x)\, \phi_j^{(2)}(z)$$

$$= \sum_i \sum_j \left[\phi_i^{(1)}(x)\, \phi_j^{(2)}(x)\right] \left[\phi_i^{(1)}(z)\, \phi_j^{(2)}(z)\right] \quad \text{def } \psi_{(i,j)} = \phi_i^{(1)} \phi_j^{(2)}$$

$$= \sum_{(i,j)} \psi_{(i,j)}(x)\, \psi_{(i,j)}(z) \rightarrow k(x,z) = \psi(x)^T \psi(z)$$

که این k کرنل است

PAPCO

j)

در حالت عادی یک داده زمانی منجر به محاسبه مجدد مرزها می‌شود که بین داده‌های مرزی بیفتد به بیان دیگر در جاده ی بین داده‌های دو کلاس قرار گیرد. برای این حالت می‌توانیم یک شرط کنترل قرار دهیم که در صورتی که میزان خطای سیستم از یک threshold کمتر بود محاسبه مجدد صورت نگیرد و در مواردی که اشتباه کم است اغماض کند .

K)

این شروط به ما اجازه می‌دهد تا در حل مسائل برنامه نویسی غیر خطی بتوانیم مانند روش لاگرانژ از شروطی به جز شرط تساوی نیز استفاده کنیم. هنگامی‌که مسئله اولیه محدب باشند شرایط KKT برای نقاط بهینه مسئله اولیه و مسئله دوگان صادق هستند، یا به عبارت دیگر فاصله دوگانی صفر می‌باشد.

تعریف مساله

$$\begin{aligned} \textbf{minimize} \quad & f(x) \\ \text{mtfc } \textbf{subject to} \quad & g_i(x) \leq 0; \quad i \in \{1, \ldots, m\} \\ & h_j(x) = 0; \quad j \in \{1, \ldots, \ell\} \end{aligned}$$

شروط KKT

۱. مساله دارای جواب باشد

۲. مساله دارای دوگان باشد

۳. شرط Complementary slackness

$$\mu_i g_i(x^*) = 0, \text{ for } i = 1, \ldots, m.$$

۴. شرط ایستا برقرار باشد

$$-\nabla f(x^*) = \sum_{i=1}^{m} \mu_i \nabla g_i(x^*) + \sum_{j=1}^{\ell} \lambda_j \nabla h_j(x^*),$$

با در نظر گرفتن شروط فوق و حل معادلات بدست آمده می‌توانیم مینمم تابع مورد نظر را بدست آوریم.

Q2

نتایج بدست آمده نشان می‌دهد که همه کرنل‌های ساخته شده در مجموعه داده آیریس عملکرد مشابهی داشته اند البته این تشابه را می‌توان به وجود کم داده‌های تست نیز مرتبط دانست. اما کرنل sigmoid در حالتی که به تنهایی گرفته شود عملکرد به شدت ضعیفی از خود نشان داده است.

```
kernel 1

accuracy : 0.9666666666666667
confusion matrix
 [[10  0  0]
 [ 0 13  0]
 [ 0  1  6]]
```

```
kernel 2

accuracy : 0.9666666666666667
confusion matrix
 [[10  0  0]
 [ 0 13  0]
 [ 0  1  6]]
```

```
kernel 3

accuracy : 0.9666666666666667
confusion matrix
 [[10  0  0]
 [ 0 13  0]
 [ 0  1  6]]
```

```
kernel 4

accuracy : 0.9666666666666667
confusion matrix
 [[ 9  0  0]
 [ 0  9  0]
 [ 0  1 11]]
```

```
RBF only

accuracy : 0.9666666666666667
confusion matrix
 [[10  0  0]
 [ 0 13  0]
 [ 0  1  6]]
```

```
sigmoid only

accuracy : 0.23333333333333334
confusion matrix
 [[ 0  0 10]
 [ 0  0 13]
 [ 0  0  7]]
```

Q3

a)

در این بخش random_state در بخش split و make_classification برابر ۴۲ قرار داده شده است.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((80000, 2), (20000, 2), (80000,), (20000,))
```

b)

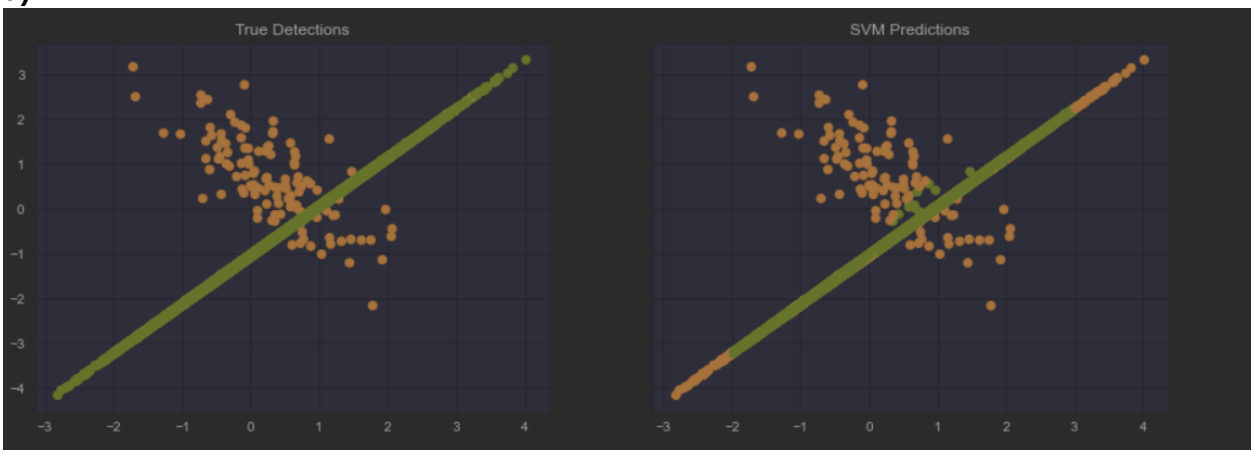
One_class_svm یک روش Unsupervised مبتنی بر SVM است که با به کارگیری SVM به دنبال ایجاد یک ابر صفحه برای تشخیص داده‌های پرت می‌باشد ملاک این دسته بندی تراکم داده‌ها و یادگیری توزیع آماری داده‌ها می‌باشد و بر اساس آن به فضاوت در رابطه با اینکه آیا این داده یک داده ی پرت است یا خیر می‌پردازد.

c)

با توجه به اینکه تعداد داده‌های پرت ما دسته به شدت کوچکی از داده‌های ما است پس هدف ما شناسایی هر چه بیشتر آن‌ها می‌باشد از این رو معیاری مانند Recall می‌تواند گزینه ای مناسب برای ارزیابی الگوریتم ما باشد.

	precision	recall	f1-score	support
0	0.99	0.98	0.99	1976
1	0.06	0.08	0.07	24
accuracy			0.97	2000
macro avg	0.52	0.53	0.53	2000
weighted avg	0.98	0.97	0.98	2000

d)



Q4

a)

فرض بر این است که سمت مثبت، سالم بودن نرم افزار در نظر گرفته شده است.

از دید امنیتی قطعا باید **False positive rate** کاهش یابد(مهم ترین معیار است) زیرا این که بتوانیم همه بدافزارها را شناسایی کنیم و به اشتباه تعدادی بد افزار را به عنوان نرم افزار سالم پیش بینی نکنیم اولویت اول است هر چند که این مهم در صورتی که بدست بیاید که تعدادی نرم افزار سالم را نیز با سخت گیری بیش از حد مخرب پیش بینی کنیم که False Negative را افزایش می‌دهد.

اما در سمت user باید هر دو مولفه در کنار یک دیگر وجود مهم هستند اما False Negative Rate اهمیت بیشتری در تجربه کاربری دارد زیرا در صورتی که False Negative افزایش یابد سیستم ما به اشتباه تعدادی از برنامه‌هایی که کاربر به آن‌ها نیاز داشته است را مخرب شناسایی می‌کند مثل windows defender که همیشه فقط فایل‌های کرک را به عنوان مخرب شناسایی می‌کند از این رو **False NegativeRate** در صورتی که از حد استاندارد بیشتر باشد برای تجربه کاربری لطمه بزرگ تری به حساب می‌آید.

b)

stratified sampling

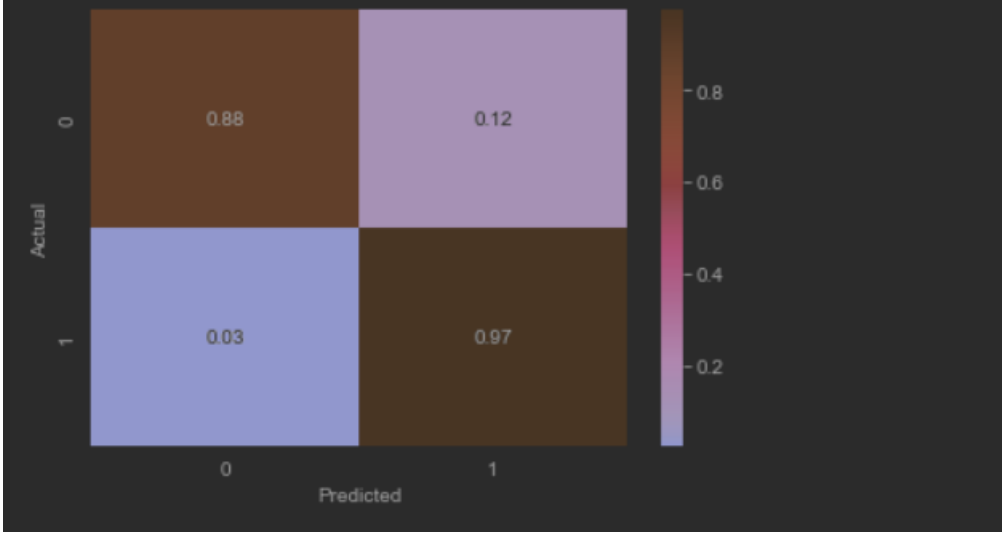
روشی است که در آن به جای نمونه برداری رندوم از داده‌ها به شکلی نمونه برداری می‌کنیم که نسبت توزیع داده‌ها در هر یک از دسته‌ها ثابت بماند به این معنی که اگر ۱ درصد از داده‌ها در داده‌های موجود مربوط به کلاس c هستند بعد از جداکردن داده test ,train در هر کدام یک درصد از داده‌ها مربوط به گروه c می‌باشد. کاربرد این روش بیشتر در داده‌هایی است که تعداد داده‌های یک دسته یا چند دسته به کم باشد یا به طور کلی دیتاست کوچکی داشته باشیم و در نتیجه در صورت عدم استفاده از این روش ممکن است جمعیت برخی از دسته‌ها به صفر میل کنند.

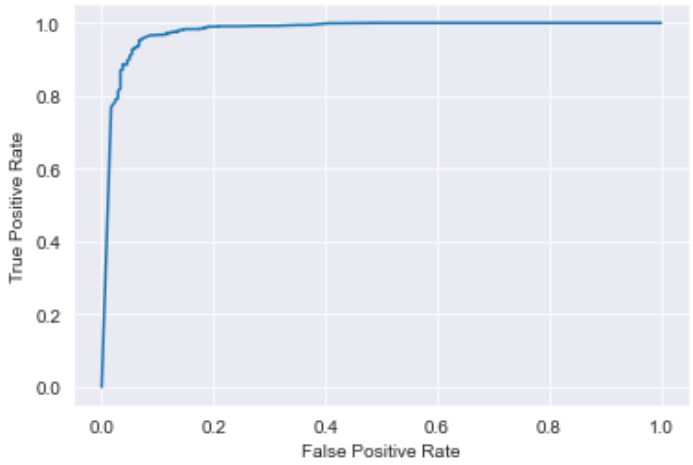
برای جداکردن داده‌ها به داده‌های آموزش و تست فیلد stratified در تابع train_test_split استفاده شده است. در ادامه در بخش e از تابع StratifiedShuffleSplit برای این کار استفاده گردیده است.

c)
bagging with KNN

	precision	recall	f1-score	support
0	0.90	0.88	0.89	240
1	0.97	0.97	0.97	886
accuracy			0.95	1126
macro avg	0.93	0.92	0.93	1126
weighted avg	0.95	0.95	0.95	1126

******
confusion_matrix



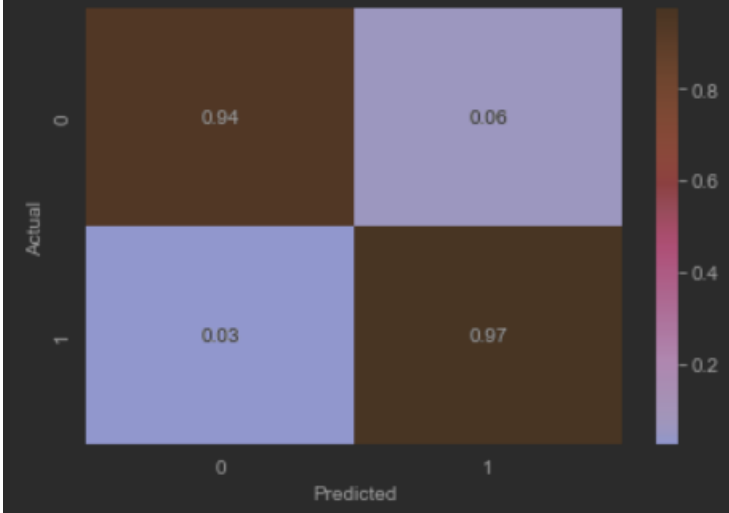


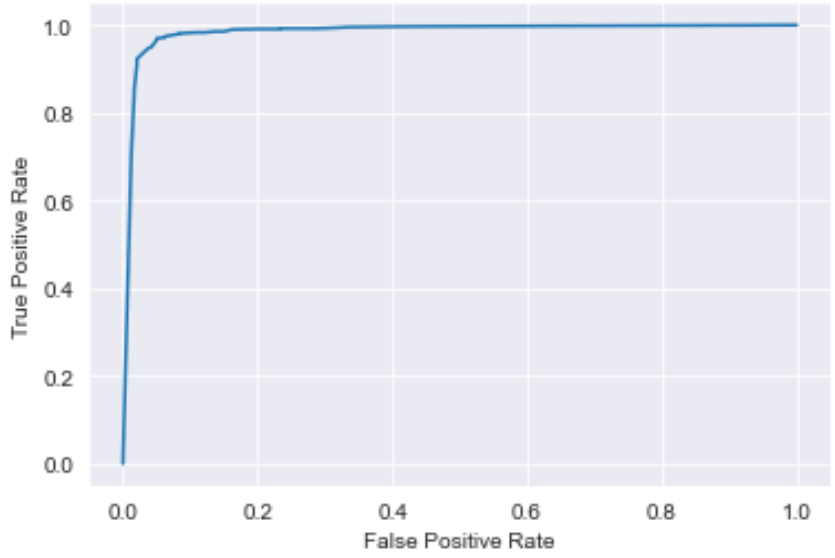
Bagging with tree

	precision	recall	f1-score	support
0	0.91	0.94	0.92	240
1	0.98	0.97	0.98	886
accuracy			0.97	1126
macro avg	0.95	0.96	0.95	1126
weighted avg	0.97	0.97	0.97	1126

******

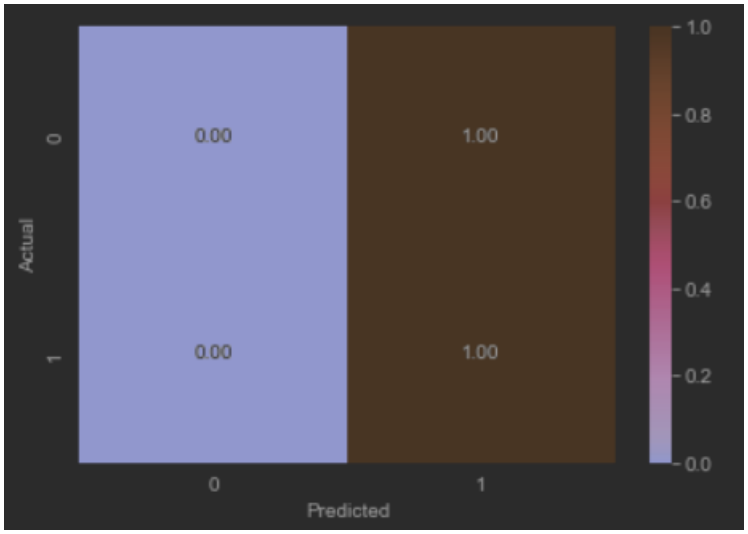
confusion_matrix

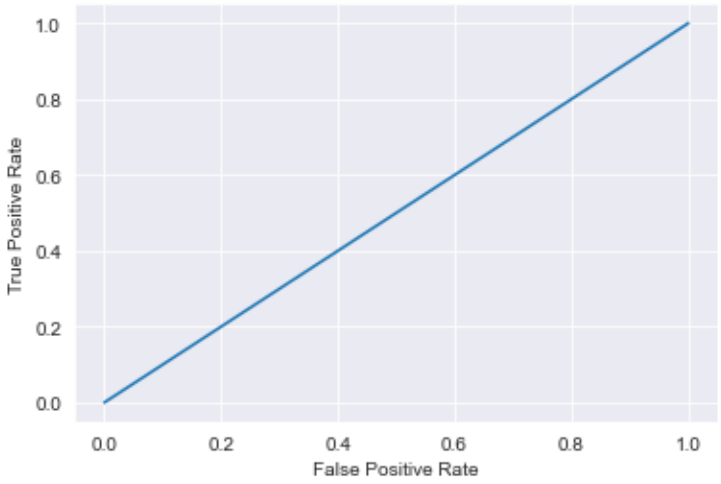




d)
Adaboost SVC

	precision	recall	f1-score	support
0	0.00	0.00	0.00	240
1	0.79	1.00	0.88	886
accuracy			0.79	1126
macro avg	0.39	0.50	0.44	1126
weighted avg	0.62	0.79	0.69	1126

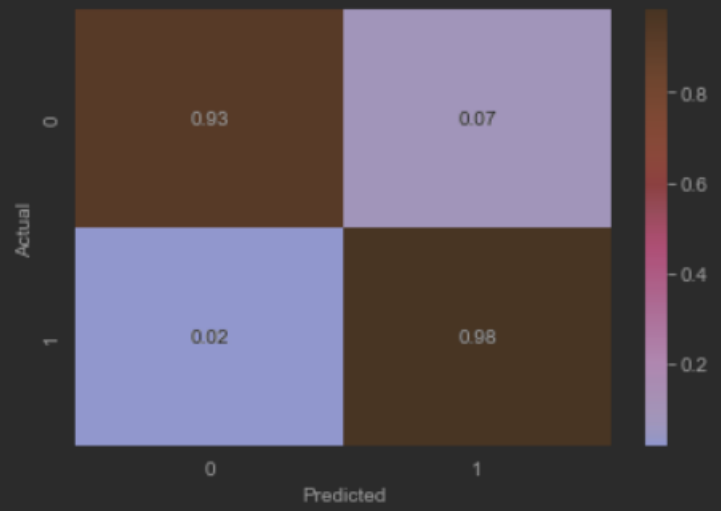


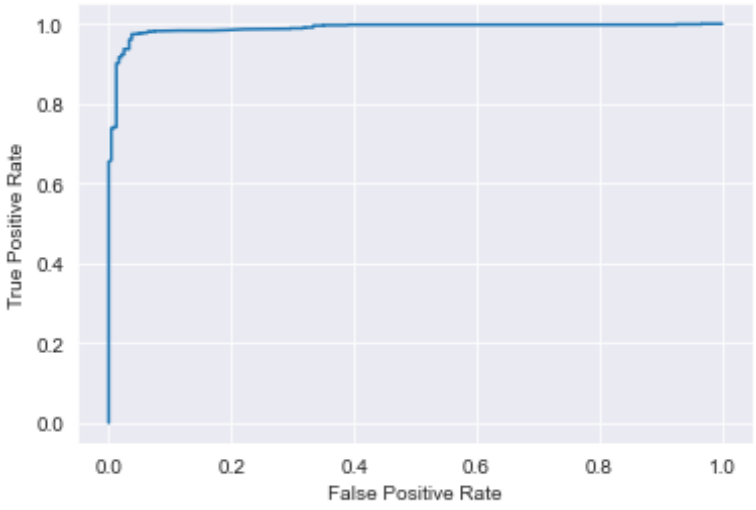


Ada Boost tree

	precision	recall	f1-score	support
0	0.93	0.93	0.93	240
1	0.98	0.98	0.98	886
accuracy			0.97	1126
macro avg	0.96	0.95	0.95	1126
weighted avg	0.97	0.97	0.97	1126

******
confusion_matrix





e)

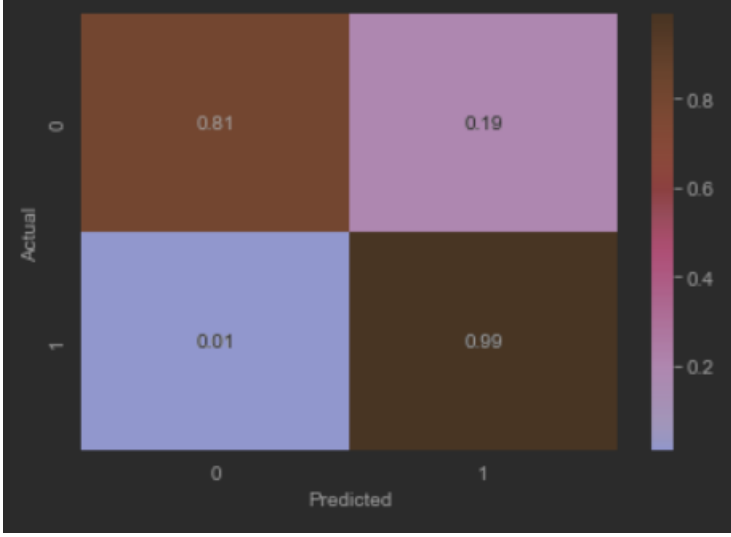
در این بخش از تابع StratifiedShuffleSplit برای ایجاد نمونه‌ها استفاده کرده ایم و ۱۰۰ بار این تابع را فراخوانی کرده و در هربار ۵ درصد داده را برداشته ایم.

f)

	precision	recall	f1-score	support
0	0.95	0.81	0.87	240
1	0.95	0.99	0.97	886
accuracy			0.95	1126
macro avg	0.95	0.90	0.92	1126
weighted avg	0.95	0.95	0.95	1126

******

confusion_matrix



g)

یک نمونه از ماتریس ۱۰۱ بعدی ایجاد شده

```
array([[1, 1, 1, ..., 1, 1, 1],
       [1, 1, 1, ..., 1, 1, 1],
       [1, 1, 1, ..., 1, 1, 1],
       ...,
       [0, 1, 0, ..., 1, 0, 0],
       [1, 1, 1, ..., 1, 1, 1],
       [1, 1, 1, ..., 1, 1, 1]], dtype=int64)
```

h)

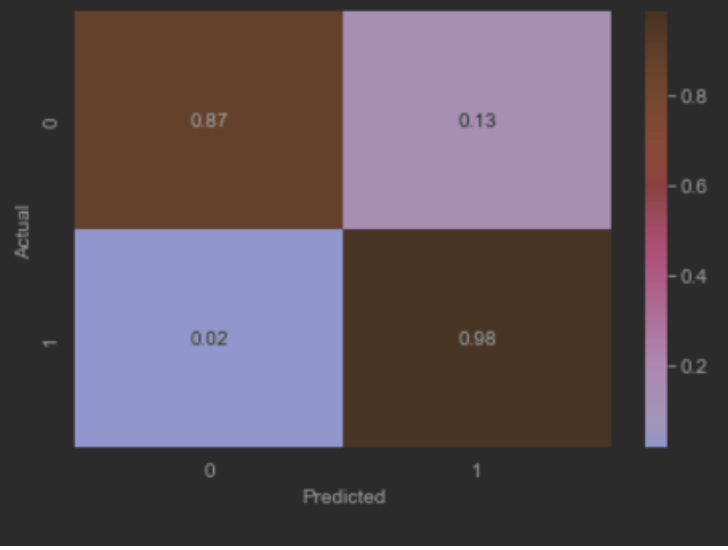
نتیجه svc

```
              precision    recall  f1-score   support

           0       0.93      0.87      0.90       240
           1       0.97      0.98      0.97       886

    accuracy                           0.96      1126
   macro avg       0.95      0.93      0.94      1126
weighted avg       0.96      0.96      0.96      1126

******
confusion_matrix
```

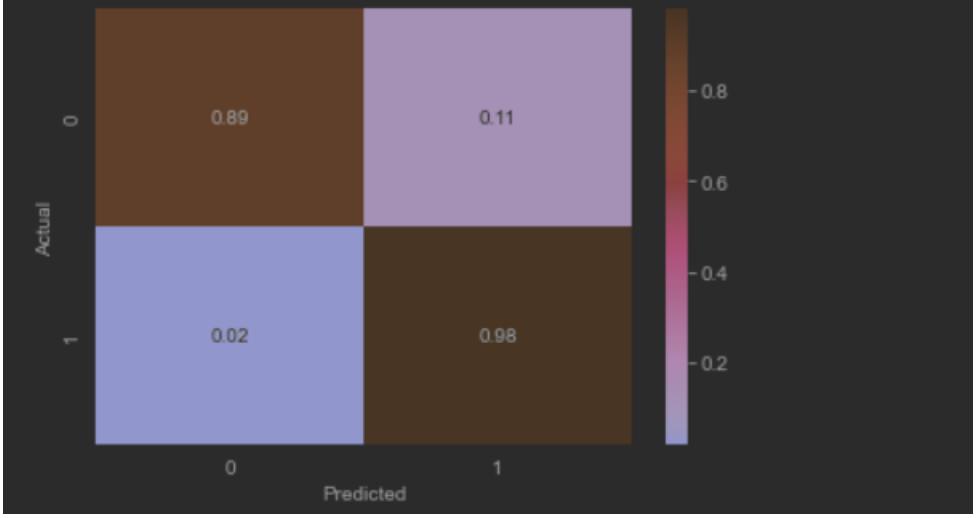


نتیجه decision tree

	precision	recall	f1-score	support
0	0.92	0.89	0.90	240
1	0.97	0.98	0.97	886
accuracy			0.96	1126
macro avg	0.95	0.93	0.94	1126
weighted avg	0.96	0.96	0.96	1126

******

confusion_matrix



به طور کلی می‌توان گفت که نتیجه بدست آمده در همه مدل‌ها قابل قبول است ومقایسه روش‌ها زمانی که همگی در بازه ۹۵ به بالا هستند زیاد عادلانه نیست و دلیل برتری آن‌ها می‌تواند صرفا به خاص بودن اون دیتاسیت مربوط باشد. اما به طور کلی مدل‌هایی که در بخش آخر تولید شده بودند توزیع اعداد در confusion ماتریس در مجموع بهتر بود به این معنی که FN یا FP وجود ندارد که خیلی متفاوت باشد. در بخش آخر نیز عملکرد درخت تصمیم بهتر از SVC بود.