حل تمرین پایان ترم

زهرا بهروزه

سوال اول

هنگامی ما از کرنل ها در SVM بهره می بریم که داده ها جدایی پذیر خطی نیستند و با این تکنیک ما داده ها را در فضای با بعد بیشتر منتقل میکنیم به شکلی که در آن فضا جدایی پذیر خطی باشند. کرنل های مختلف با توجه به ساختار های متفاوتی که دارند در مسائل مختلف متناظر با ساختارشان کاربرد بهتری خواهند داشت. مثال هایی برای کرنل های پر کاربرد کرنل های چبیشف، لژاندر و کرنل توابع شعاعی (Radial Basis Functions) هستند. کرنل RBF به دلیل شباهتش به الگوریتم K-Nearest شناخته شده است. کرنل چبیشف هم به دلیل پایه های متعامدی که دارد.

سوال دوم

داده های خود را به این شکل تولید می کنیم که ابتدا از مجموعه داده train (آموزش) دو ستون three_g و clock_speed را به عنوان داده های محور X و ستون four_g را به عنوان داده ی محور y در نظر می گیریم، سپس یک مدل SVM بر روی این داده ها اعمال می کنیم. همچنین داده های test (آزمایش) را هم به همان صورت از مجموعه داده آزمایش در فایل موبایل انتخاب می نماییم. مدل کلاس SVM خود را بر روی داده های آموزش خود فیت می کنیم و با استفاده از تابع predict و داده ی X_test_data به عنوان ورودی این تابع، از مدل می خواهیم که برآورد خود را از داده ی X_test_data تولید کند و مجموعه ای به شکل y_test_data تولید کند.

سوال سوم

شش حالت مختلف مدل SVM متشکل از چهار کرنل مختلف خطی، سیگموید، چندجمله ای و شعاعی با پارامترهای مختلف را انجام دادیم و مقدار خطای هر حالت با استفاده از تابع محاسبه خطا Calculate_error بدست آوردیم. تابع محاسبه خطا مقادیر داده ی واقعی که همان y_test_data است را با مقدار بدست آمده از مدل SVM، یک به یک مقایسه می کند و اگر یکسان نبودند به متغیر error value که داخل خود تابع با مقدار صفر تعریف شده است، یک واحد اضافه میکند. سپس مقدار این متغیر تقسیم بر طول آرایه ی داده ی واقعی می شود تا مقدار خطا را در بازه ی بین صفر و یک بصورت نرمالایز شده داشته باشیم.

سوال چهارم

در مدل های 2 تا 6، از روش حاشیه نرم یا Soft Margin استفاده شده است، چون کرنل های غیرخطی برای این مدل ها به کار گرفته ایم. اما در مدل 7 از کرنل خطی و روش حاشیه سخت یا Hard Margin استفاده کرده ایم.

سوال هفتم

درخت تصمیم درختی است که از ریشه به سمت پایین (برگ) رشد کرده است. در **الگوریتم درخت تصمیم** نمونه‌ها را دسته‌بندی می‌کنیم که در واقع دسته‌ها در انتهای گره‌های برگ قرار دارد. درخت تصمیم در مسائلی کاربرد دارد که بتوان آنها را به صورتی مطرح نمود که پاسخ واحدی به صورت نام یک دسته یا کلاس ارائه دهند. روش های ساخت درخت تصمیم معمولا به صورت بالا به پایین عمل می کنند به این معنی که ابتدا فضای ورودی به فضاهای کوچکتر تقسیم می‌شود، سپس فرآیند تقسیم بندی برای هر یک از این قسمت‌ها تکرار می شود. به عبارت دیگر در هنگام ساخت درخت، ابتدا ریشه ساخته می-شود، سپس هر یک از زیر شاخه ها به شاخه‌های دیگری تقسیم می‌شود و این فرآیند تکرار می شود.

الگوریتم ID3 یکی از ساده ترین الگور یتم های درخت تصمیم است. در این الگوریتم درخت تصمیم از بالا به پایین ساخته میشود. این الگوریتم با این سوال شروع میشود: کدام ویژگی باید در ریشه درخت مورد آزمایش، قرار بگیرد؟ برای یافتن جواب از معیار بهره اطلاعات استفاده میشود .با انتخاب این ویژگی، برای هر یک از مقادیر ممکن آن یک شاخه ایجاد شده و نمونه های آموزشی بر اساس ویژگی هر شاخه مرتب میشوند. سپس عملیات فوق برای نمونه های قرار گرفته در هر شاخه تکرار می شوند تا بهترین ویژگی برای گره بعدی انتخاب شود.

الگوریتم : CART این الگوریتم متغیرهای ورودی را برای یافتن بهترین تجزیه می آزماید تا شاخص ناخالصی حاصل از تجزیه کمترین مقدار باشد.در تجزیه دو ز یر گروه ایجاد میشوند و هر کدام در مرحله بعد به دو زیر گروه دیگر تقسیم میشوند و این روند ادامه دارد تا یکی از معیارهای توقف ارضا شود.درخت cart بازگشتی دودویی است.

تفاوت انواع درخت تصمیم: الگوریتم درخت تصمیم به گونه ای عمل میکند که سعی دارد گوناگونی و یا تنوع را در گره ها به حداقل ممکن برساند.این عدم یکنواختی در گره ها با استفاده از معیار های عدم خلوص قابل اندازه گیری است که مهم ترین و پرکاربردترین آن شاخص جینی میباشد.

اغلب تفاوت انواع درخت های تصمیم در همین معیار اندازه گیری عدم خلوص، شیوه شاخه بندی و هرس کردن گره های درخت میباشد.

سوال هشتم

از کلاس Decision Tree Classifier بهره می گیریم وبه دو روش زیر درخت تصمیم را می سازیم.

1. با استفاده از شاخص gini

2. با استفاده از ساختار entropy

ساختار کلی ساخت درخت دارای چهار تابع است. تابع اول تابع آموزش بر اساس شاخص جینی است. این تابع داده های مورد نیاز برای ساخت درخت تصمیم را می گیرد و درخت تصمیم را بر اساس شاخص gini می سازد. سپس داده های آموزش X و Y را به مدل برازش می کند و در نهایت درخت را به عنوان خروجی خواهیم داشت. تابع دوم ساختاری کاملا مشابه با تابع اول دارد و فقط به جای شاخص gini، از شاخص entropy برای ساخت مدل درخت تصمیم استفاده می شود. تابع سوم نیز برای محاسبه prediction تعریف شده است. این تابع با استفاده از تابع predict مقادیر y_test_data را با مدل ساخته شده تخمین می زند و در نتیجه به عنوان خروجی مقدار برآورد شده را باز می گرداند. تابع چهارم برای محاسبه دقت تخمین های بدست آمده تعریف می شود. در نهایت خروجی های این تابع که عبارتند از: امتیاز دقت تخمین، confusion matrix و گزارش کلاس بندی را، خواهیم داشت.

سوال نهم

تعداد سطوح و تعداد برگ های درخت تصمیم توسط کاربر تنظیم می شوند. در سوال قبل درخت تصمیمی با سه سطح و پنج برگ ساختیم. در این سوال درخت تصمیم هایی به ترتیب با 4 سطح و 4 برگ و 4 سطح و 5 برگ می سازیم. با مشاهده ای که انجام شد؛ نتیجه گرفتیم که خروجی های بدست آمده در این سوال با خروجی حالت قبل برابر هستند. اما این نتیجه به این معنی نیست که تغییر در تعداد برگ و سطح، تاثیری در نتیجه ندارد بلکه تغییرات در پارامتر های این درخت، باعث تغییر نتیجه خواهد شد.

سوال دهم

هرس یک روش فشرده سازی داده در الگوریتم های یادگیری ماشین که با از بین بردن بخشهایی از درخت که حیاتی نیستند و تکراری هستند، اندازه درختان تصمیم را کاهش می دهد. هرس پیچیدگی کلاس بند نهایی را کاهش می دهد و از این رو دقت پیش بینی را بهبود می بخشد. درختی که بیش از حد بزرگ باشد، بیش از حد متناسب با داده های آموزش است و نسبت به نمونه های جدید ضعیف است. یک درخت کوچک ممکن است اطلاعات مهم ساختاری مربوط به فضای نمونه را نداشته باشد. یک استراتژی معمول این است که درخت رشد کند تا زمانی که هر گره شامل تعداد کمی از نمونه ها باشد سپس از هرس برای حذف گره هایی استفاده می شود که اطلاعات بیشتری را ارائه نمی کنند.

حال زمانی نیاز به هرس کردن درخت هست که تعداد کلاس ها یا در کل دسته بندی های بدست آمده توسط درخت زیادتر از حد معمول باشد. در واقع به این دلیل زیاد تر از حد معمول است که در کلاس های بدست آمده داده های کمی وجود دارد و برخی از این داده ها در کلاس های مختلف در حدی به هم شبیه هستند که بتوان آن ها را در یک کلاس قرار داد. در این حالت در قسمت برگ یا سطح های بالاتر (معمولا تا دو سطح بالاتر از برگ) بعضی از نود ها حذف می شوند و داده هایی که از آن مسیر به کلاس حذف شده رفته بودند به کلاسی که داده های مشابه داده های کلاس حذف شده دارد، انتقال می یابند.

سوال سیزدهم

مهمترین مزایای درخت تصمیم:

مهم ترین فواید استفاده از این الگوریتم عبارتند از:

1- فهم ساده: هر کسی با اندکی مطالعه و آموزش می تواند، طریقه کار با درخت تصمیم را بیاموزد

2- کار کردن با داده های بزرگ و پیچیده: درخت تصمیم در عین سادگ ی می تواند با داده های پیچیده به راحتی کار کند و از روی آنها تصمیم بسازد .

3-استفاده مجدد آسان: در صورتی که درخت تصمیم برای یک مسئله ساخته شد، نمونه های مختلف از آن مسئله را می توان با آن درخت تصمیم محاسبه کرد.

4- قابلیت ترکیب با روش های دیگر: نت یجه درخت تصمیم را می توان با تکنیک های تصمیم سازی دیگر ترکیب کرده و نتایج بهتری بدست آورد.

سوال چهاردهم

C4.5با استفاده از پسوند الگوریتم ID3 کوینالن ، درخت تصمیم را از یک مجموعه تنظیم می کند. درخت تصمیم شامل شاخه هایی است که نتایج آزمایشات انجام شده بر روی ویژگی های انتخاب شده و گره ها ی برگ را نشان می دهد ، که به برچسب های کالس اختصاص داده شده است. 5 C4.پس از سا ختن درخت تصمیم ، سعی می کند با استفاده از یک مرحله هرس ، پیچیدگی آن را کاهش دهد و هدف آن حذف شاخه ها با حداقل کمک به دقت کلی است. هنگامی که درخت هرس شد ، می توان دانش را استخراج و به صورت قوانین در صورت وجود ارائه داد.

Grow & prune algorithms

Ripper

هرس افزایشی مکرر برای تولید خطا

یکی از کارآمدتر ین و مورد استفاده ترین الگوریتم های یادگیری قاعده است. این یک استراتژی تقسیم و تسخیر را برای حاکمیت استقرا اجرا می کند Ripper .برای تدوین یک مجموعه اولیه از قوانین برای هر کالس ، اصطلاحاً به آن افزوده می شود IREP .سپس ، مرحله بهینه سازی اضافی هر قانون را در مجموعه فعلی به نوبه خود در نظر می گیرد و دو قانون جایگزین از آنها ایجاد می کند: یک قانون جایگزینی و یک قانون تجدید نظر. پس از آن ، در مورد اینکه آیا مدل باید قاعده اصلی ، جایگزینی یا قانون تجدید نظر را بر اساس معیار حداقل طول توصیف حفظ کند ، تصمیم گیری می شود.

PART

نظریه تشدید تطبیقی تصوری

یک الگور یتم درخت تصمیم جزئی است. به طور خاص ، PARTبا توجه به استراتژی تقسیم و تسخیر ، مجموعه ای از قوانین را ایجاد می کند ، همه موارد را از مجموعه آموزش که تحت این قانون هستند حذف می کند و به صورت بازگشتی ادامه می دهد تا زمانی که هیچ موردی باقی بماند. برای ایجاد یک قانون واحد ، PARTیک درخت تصمیم C4 5.جزئی برای مجموعه موارد فعلی ایجاد می کند و برگ با بیشترین پوشش را به عنوان قانون جدید انتخاب می کند. پس از آن ، درخت تصمیم گیری جزئی به همراه موارد تحت پوشش قانون جدید از داده ها ی آموزش حذف می شود تا از تعمیم اولیه جلوگیری شود. این روند تکرار می شود تا زمانی که همه موارد با قوانین استخراج شده پوشش داده شوند.

CAMUR

طبقه بندی با مدل های مبتنی بر قانون جایگزین Multiple

بر اساس الگوریتم RIPPER ساخته شده است. مبانی متعدد و معادل قاعده را از طریق محاسبه تکراری مدل طبقه بندی مبتنی بر قانون استخراج می کند CAMUR .شامل یک مخزن دانش موردی پایگاه داده و یک ابزار پرس و جو است.

BIGBIOCL

نسخه بهبود یافته CAMUR است که برای کنترل صدها هزار ویژگی طراحی شده است. طبق استراتژی CAMUR، این برنامه برای یادگیری مدل های طبقه بندی چند گزینه ای و معادل از طریق حذف تکراری ویژگی های انتخاب شده طراحی شده است

Based on fuzzification FURIA

نسخه بهبود یافته الگوریتم RIPPER است. FURIA از الگوریتم اصلاح شده RIPPER به عنوان مبنا استفاده می کند و قوانین فازی و مجموعه قوانین نامرتب را یاد می گیرد. نقطه قوت اصلی این الگوریتم روش کشش قانون است ، که با حل فشار مشکل رکوردهای جدیدی که طبقه بندی می شود می تواند خارج از فضای تحت پوشش قوانین قبلی باشد. نمایش قوانین فازی نیز پیشرفته است ، اساساً ، یک قانون فازی از طریق جایگزینی فواصل با فواصل فازی ، یعنی مجموعه های فازی با عملکرد عضویت ذوزنقه ای ، بدست می آید.

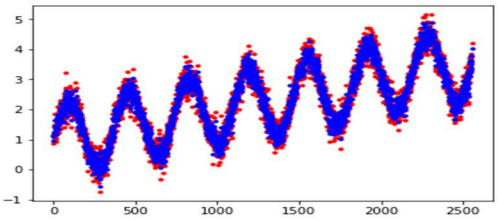
Based on probability estimation

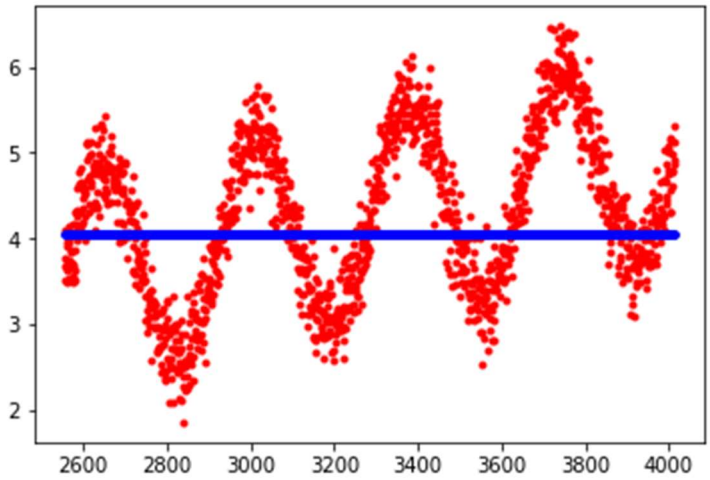
MLRules

یک الگوریتم القایی برای حل مسائل طبقه بندی از طریق برآورد احتمال است. ایده اصلی این است که از قوانین واحد به عنوان طبقه بندی کننده منفرد استفاده کرده و سپس سیستم طبقه بندی گروه را بر روی آنها پیاده سازی کنید. متفاوت از رویه های کلاسیک پوشاندن متوالی که به آن روشهای تقسیم و تسخیر نیز گفته می شود، قوانین جدید بدون تعدیل قوانینی که قبلا اضافه شده اند، اضافه میشوند. مزیت اصلی الگوریتم MLRules این واقعیت است که از یک روش آماری ساده و قدرتمند برای القای قوانین استفاده می شود ، که به نوبه خود منجر به گروه های نهایی با دقت پیش بینی بسیار بالا می شود.

سوال پانزدهم:

درختان تصمیم در مسائل رگرسیون برای داده های آموزش در مدل داده های سری زمانی مناسب است ؛ اما هر چه قدر که در آموزش مناسب است در مراحل ارزیابی (validation) و همینطور در ادامه در مرحله تست بسیار ضعیف است. مثلا در نمونه ای برای داده های غیر خطی از نوع متناوب با نوسان زیاد، یک خط روی این مدل به دست آورده است که در رگرسیون ، این نتیجه بسیار بد است :





عکس اول مربوط به آموزش و عکس دوم مربوط به مرحله دوم آموزش و قبل از تست ، یعنی مرحله ارزیابی است.

سوال شانزدهم

ستون Date را به شکلی که درخواست شده بود تغییر دادیم و در کد موجود است. ستون های مجموعه داده از نوع رشته ای بودند بنابراین ابتدا آن ها را به عدد یعنی نوع float تبدیل کردیم و سپس مدل را بر روی آن فیت کردیم. مراحل پاکسازی و تبدیل مختصرا بصورت ذیل است:

1. ابتدا از مقادیر ستون ها علامت کاما را حذف کردیم.
2. سپس از مقادیر ستون .Vol، علامت K که به معنای هزار است حذف کردیم.
3. در نهایت تبدیل مقادیر رشته ای به عددی انجام شد.
4. ممکن است برخی از مقادیری که رشته ای بودند مقدار nan بگیرند، که با کد زیر مشکل مرتفع شد

```
data = data.replace(np.nan, 0)
```

درنهایت داده های آموزش و آزمایش را جدا سازی کرده که در کد موجود می باشد و مدل یادگیری مان را خواهیم داشت.

سوال هفدهم

از شش مدل ذیل استفاده نمودیم در سوال که بدین شرح میباشند:

1. رگرسیون خطی (Linear Regression)
2. رگرسیون لاسو(Lasso Regression)
3. رگرسیون ریج (Ridge Regression)
4. روش SVR با کرنل شعاعی
5. روش SVR با کرنل چند جمله ای
6. شبکه عصبی LSTM با لایه ورودی یک نورون، لایه ی مخفی با چهار نورون و لایه ی خروجی با یک نورون

پنج مدل اول به یک شکل طراحی شده اند. ابتدا یک شی از کلاس تعریف شده برای روش موردنظر در پایتون می سازیم و سپس مدل خود را به داده فیت می کنیم. سپس مقادیر تخمینی برای مقدار y_test را با استفاده از تابع predict، می یابیم. خطای مدل را هم با استفاده از روش RMSE گزارش داده ایم.

در مدل LSTM، قبل فیت کردن مدل بر روی داده، ابتدا تمام مقادیر مجموعه داده را در آرایه ذخیره می کنیم و ستون های Date و Change% را از این آرایه حذف می کنیم و نوع آرایه را به float تغییر می دهیم. آرایه بدست آمده را با همان مقادیر گفته شده در سوال 16 برای داده های آموزش و آزمایش،

جداسازی می کنیم. آرایه ی داده آموزش دارای 3822 سطر و یک ستون و آرایه ی داده آزمایش دارای 119 سطر و یک ستون است. سپس با استفاده از تابع create dataset این آرایه ها را به ماتریس تبدیل می کنیم و به ماتریس های X train و X test حاصل دو ستون دیگر اضافه می کنیم که سه بعدی شود تا بتوانیم به عنوان تنسور از آنها استفاده کنیم زیرا ورودی اول تابع fit برای این نوع شبکه باید تنسور باشد.

سپس مدل sequential شبکه عصبی LSTM را پیاده سازی کردیم. تابع خطا و بهینه ساز تابع خطا به ترتیب عبارتند از:

Mean Square Error (MSE)
Adam Optimizer

مدل خود را بر روی داده ها برازش میدهیم. در اینجا از کل داده های آموزش در 100 مرحله استفاده می کنیم. در نهایت مقدار خطای RMSE را نمایش میدهیم.

سوال هجدهم

روش های ensemble learning روش های ترکیبی یادگیری هستند که یک یادگیرنده ی ترکیبی در این حالت، به روش های مختلف از مخلوط یا ترکیب نتیجه ی چند یادگیرنده ی ضعیف (روش های تکین یادگیری ماشین)، یک نتیجه با واریانس و بایاس کم بدست می آورد. از جمله روش های به نام این نوع یادگیری، روش های Bagging، Boosting، Voting و stacking است که به ترتیب سه مورد اول را مختصرا شرح میدهیم. مدل های تکین ما در اینجا رگرسیون خطی، ریج و لاسو هستند که از ترکیب کردن نتیجه های این سه روش، نتیجه های نهایی را خواهیم داشت.

Bagging

این روش بخش هایی از مجموعه داده را با طول های مشخص و مساوی انتخاب می کند و یک مدل یادگیری ماشین را روی این بخش های بدست آمده پیاده می کند. در نهایت خروجی هر درایه از آرایه ی مدل یادگیری ترکیبی میانگینی از درایه های نظیر در نتایج بدست آمده از مدل اجرا شده روی بخش های مجموعه داده خواهد بود. همچنین معمول ترین روش برای ترکیب نتیجه، روش میانگین است.

Voting

این روش بر اساس رای هایی که از مدل های تکین می گیرد خروجی نهایی را می سازد. برای پیاده سازی این روش از کتابخانه Voting Regressor استفاده می کنیم. به دلیل مشکلی در لیبل ها و شمارنده ی آرایه ی y_train، از تابع label encoder بهره بردیم تا لیبل های این آرایه را مناسب با تابع fit ، اصلاح کند.

Boosting

در این حالت به جای استفاده از یک مدل با ورودی های مختلف، از یک مدل چندین و چند بار به صورت سری مورد استفاده می شود به طوری که خروجی یک مرحله به عنوان ورودی مرحله بعدی به همان مدل داده می شود. در این سوال ما از روش AdaBoost بهره بردیم.

سوال نوزدهم

حالت اول: مدل **SVR** با کرنل شعاعی

test score: 48611.42 RMSE

حالت دوم : مدل کرنل چند جمله ای درجه دوم

test score: 39445.92 RMSE

حالت سوم : مشابه حالت اول اما با دنباله ای به طول 20 یادگیرنده ضعیف

test score: 48611.42 RMSE

حالت چهارم : مشابه حالت دوم اما با دنباله ای به طول 20 یادگیرنده ضعیف

test score: 39445.92 RMSE

سوال بیستم

الگوریتم Random Forest به عنوان یکی دیگر از روش های یادگیری ترکیبی برای دسته بندی و رگرسیون می‌باشد. جنگل‌های تصادفی برای درختان تصمیم که در مجموعهٔ آموزشی دچار بیش برازش می‌شوند، مناسب هستند. عملکرد جنگل تصادفی معمولا بهتر از درخت تصمیم است. این روش کاملا شبیه روش Bagging است، تنها با این تفاوت که در این روش از درخت تصمیم به عنوان مدل یادگیرنده ضعیف استفاده می شود.

در این سوال دو حالت از این الگوریتم را بررسی کردیم با درختانی از عمق به ترتیب 2 و 4 سطح.

سوال بیست و یکم

در سری سوالات 17 تا 20 خروجی ها همه به شکل RMSE محاسبه شده است که در کد ها مشخص میباشد. برای رسیدن به این خروجی هم از تابع آماده پایتون به اسم(mean square error (MSE استفاده شد و در نهایت با استفاده از تابع math.sqrt ، خروجی RMSE تولید شد.

برای تابع سوال 21 نیز به این شکل عمل کردیم که ابتدا فرمول خطای درصدی RMSE را با استفاده از توابع کتابخانه numpy حساب می کنیم. سپس شرط گفته شده را بررسی می کنیم که اگر این مقدار حساب شده که در واقع امتیاز (score) مقدار تخمینی است، بیش از 0.95 بود، یعنی خطای ما کمتر از 0.05 است و خروجی درست میگیریم. به عنوان مثال هم خروجی رندوم فارست چهارسطحی هم به این تابع داده شد و به دلیل اینکه این خروجی از 0.95 کمتر بود خروجی نادرست گرفتیم.

سوال بیست و دوم

به دیتاست یک ستون اضافه می کنیم به طوری که اگر مقدار قیمت open از مقدار قبلی آن بیشتر بود مقدار ستون جدید برابر با 1 و اگر کمتر باشد مقدار 1- می گیرد. در صورت مساوی بودن مقدار فعلی با مقدار بالایی ستون open هم مقدار 0 به ستون جدید تعلق می گیرد.

سپس شبکه عصبی LSTM را به همان شیوه ی قبل فقط این بار با مقادیر آزمایشی و آموزشی جدید که به ترتیب ستون جدید ساخته شده و ستون اپن است، پیاده سازی می کنیم.