بسمه تعالی

**دانشگاه صنعتی شریف**

**دانشکده ........**

**گزارش تمرین سری دوم**

**درس مبانی داده کاوی و کاربرد های آن**

**استاد : دکتر ......**

**گردآورنده :**

**نام و نام خانوادگی شماره دانشجویی**

خرداد 99

فهرست مطالب

[سوالات تحقیقی-تشریحی 2](#_Toc42176546)

[بخش اول: 2](#_Toc42176547)

[بخش دوم: 3](#_Toc42176548)

[گزارش کدنویسی با پایتون 6](#_Toc42176549)

[گام اول: 6](#_Toc42176550)

[گام دوم: 8](#_Toc42176551)

[گام سوم: 10](#_Toc42176552)

[گام چهارم: 13](#_Toc42176553)

# سوالات تحقیقی-تشریحی

## بخش اول:

الگوریتم های داده کاوی را می توان بر اساس نوع داده های ورودی (برچسب دار و یا بدون برچسب بودن) و همچنین ماهیت خروجی مورد انتظار دسته بندی کرد. به طور کلی هدف ما از داده کاوی انجام اعمالی نظیر پیش بینی[[1]](#footnote-1)، دسته بندی[[2]](#footnote-2) و یا رگرسیون[[3]](#footnote-3) است. بر این اساس، در صورتی که داده های ورودی ما بدون برچسب[[4]](#footnote-4) باشند، نمی توان از الگوریتم های دسته بندی استفاده کرد. یا در صورتی که متغیر مورد نظر ما، متغیری عددی و پیوسته (نظیر دمای هوا و یا قیمت) باشد، از الگوریتم های رگرسیون استفاده می کنیم. به طور کلی الگوریتم های داده کاوی برای هشت عملکرد زیر استفاده می شوند:

1. دسته بندی
2. قواعد انجمنی
3. تشخیص داده پرت و خارج از قاعده[[5]](#footnote-5)
4. خوشه بندی[[6]](#footnote-6)
5. رگرسیون
6. پیش بینی
7. الگو های متوالی[[7]](#footnote-7)
8. درخت های تصمیم[[8]](#footnote-8)

به طور کلی، بر اساس اینکه داده های ورودی دارای برچسب هدف هستند و یا خیر، مدل ها را به دو نوع نظارت شده[[9]](#footnote-9) و غیر نظارتی[[10]](#footnote-10) تقسیم می کنیم. در الگوریتم های نظارت شده می توان به الگوریتم های دسته بندی، رگرسیون و درخت های تصمیم اشاره کرد. برای مثال در طبقه بندی، نیاز داریم که طبقه مربوط به هر کدام از داده های آموزشی را بدانیم. در الگوریتم های غیر نظارتی نیز می توان به خوشه بندی و یا تشخیص داده پرت و خارج از قاعده اشاره کرد. در این دسته از الگوریتم ها، نیازی به برچسب و تابع هدف نمی باشد و اساس انجام عملیات بر روی داده ها شباهت آن ها با یکدیگر است.

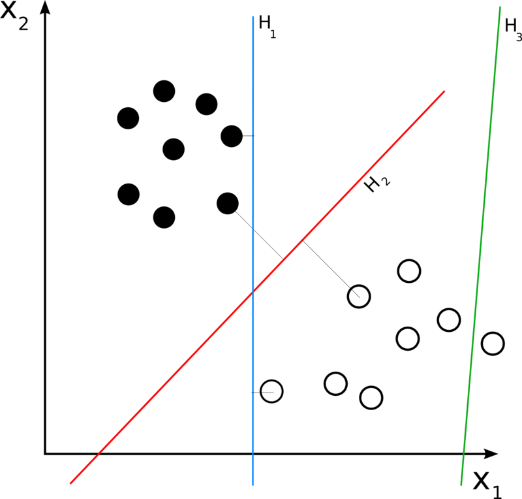
در دسته بندی دیگر نیز می توانیم الگوریتم ها را بر اساس نوع خروجی مورد انتظار دسته بندی کنیم. برای مثال در صورتی که انتظار داریم خروجی مورد نظر دسته[[11]](#footnote-11) مربوط به داده ها باشد، از الگوریتم های دسته بندی و یا درخت تصمیم استفاده می کنیم. یا در صورتی که انتظار داریم خروجی مورد انتظار متغیر عددی پیوسته ای باشد، از رگرسیون استفاده می کنیم. به طور کلی این الگوریتم ها را می توان بر اساس نوع خروجی به دسته های مختلفی تقسیم کرد که برخی از آنها عبارتند از:

1. خروجی برچسب
2. خروجی متغیر عددی پیوسته
3. خروجی خوشه ها

## بخش دوم:

1. ماشین بردار پشتیبان[[12]](#footnote-12) (به اختصار SVM) الگوریتم نظارت شده ای است که برای دو عمل دسته بندی و رگرسیون استفاده می شود. مبنای کاری این الگوریتم دسته بندی خطی داده ها است و عملکرد آن به صورتی است که خطی انتخاب شود که بیشترین حاشیه اطمینان را دارد. از آنجایی که این الگوریتم نظارت شده است، نیاز به داده های با برچسب دارد و بنابراین برای دیتاست هایی استفاده می شود که دارای برچسب متغیر هدف می باشند. ماشین بردار پشتیبان عملکرد نزدیکی به شبکه های عصبی[[13]](#footnote-13) دارد، به نحوی که یک SVM بدون هسته[[14]](#footnote-14) یک نرون شبکه عصبی با تابع هزینه متفاوت است.
2. بردار های پشتیبان[[15]](#footnote-15) مجموعه ای از نقاط در فضای چند بعدی داده ها است که مرز بین دسته ها را مشخص می کند. ابعاد این بردار همواره یک واحد کمتر از ابعاد داده ها می باشد. برای مثال در فضای دو بعدی بردار پشتیبان یک خط و در فضای سه بعدی بردار پشتیبان یک صفحه می باشد. هدف اصلی این الگوریتم یافتن بهترین مرزبندی میان داده های ورودی است، به گونه ای که بیشترین حاشیه[[16]](#footnote-16) (فاصله) ممکن را از تمام دسته ها داشته باشد.

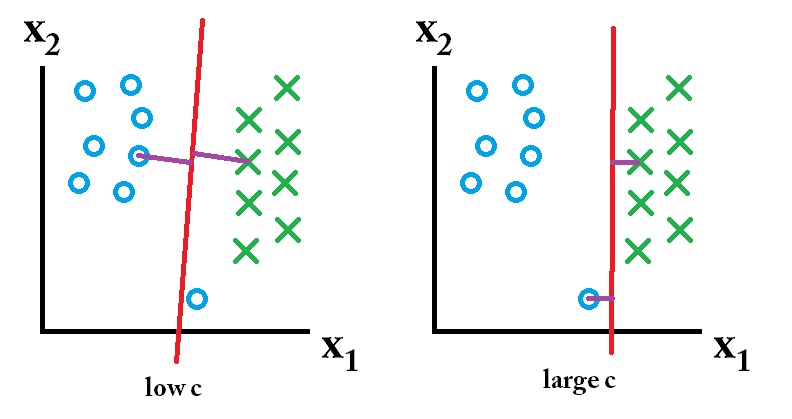
به عنوان مثال در شکل فوق، در صورتی که دو دسته داده های سفید و سیاه را داشته باشیم، خط H3 داده ها را به خوبی تقسیم نمی کند و بنابراین به عنوان یک بردار پشتیبان مورد قبول نیست. دو بردار H2 و H1 بردار های پشتیبان هستند که از بین آنها، H2 عملکرد بهتری دارد چرا که دو دسته را با بیشترین حاشیه ممکن جدا می کند. بردار H1 نیز این دو دسته را با حاشیه بسیار کمی جدا می کند که مطلوب نیست. به طور کلی، از آنجایی که داده های ورودی ممکن است دارای نویز و یا اعوجاج باشند، این الگوریتم تلاش می کند تا بردار پشتیبانی را انتخاب کند که بیشترین حاشیه ممکن را (که به آن حاشیه اطمینان نیز گفته می شود) داشته باشد.



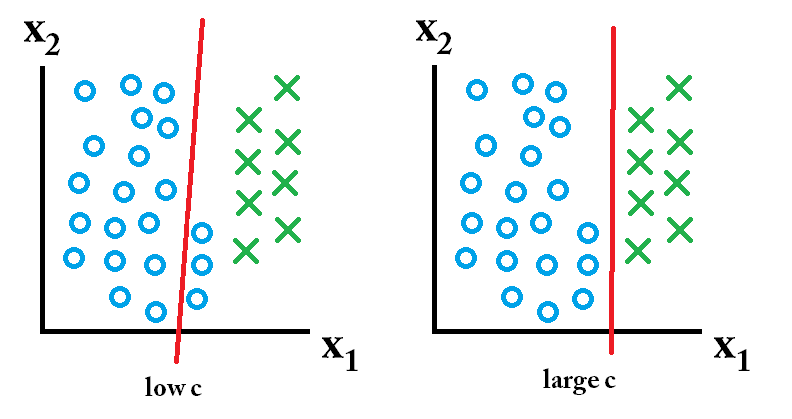
در صورتی که داده های بردار های آموزشی باشند که در دو دسته قرار دارند، هدف SVM پیدا کردن و است، به نحوی که تابع تصمیم گیری زیر برای همه (و یا اکثر) نمونه ها پیش بینی درستی انجام دهد:

1. پارامتر C که به آن پارامتر رگولاریزاسیون نیز می گویند، ضریبی است که هزینه مربوط به دسته بندی اشتباه داده ها را در فرایند بهینه سازی بیان می کند. در صورتی که مقدار آن افزایش یابد، داده هایی که با خطا دسته بندی شده اند تاثیر بیشتری در تابع خطا[[17]](#footnote-17) خواهد داشت و در صورتی که به بی نهایت میل کند، جواب نهایی به اصطلاح Hard Margin می شود. در صورتی که این پارامتر به صفر میل کند، تعداد دسته بندی های نادرست افزایش می یابد. بنابراین، مقادیر بزرگ C باعث می شود که الگوریتم به دنبال انتخاب بردار ها با حاشیه کمتری باشد، مادامی که تمام داده های ورودی به درستی دسته بندی شوند. مقادیر خیلی کوچک C نیز باعث می شود تا الگوریتم به دنبال انتخاب بردار ها با حاشیه بزرگتری باشد، حتی اگر در این حالت نقاطی با خطا دسته بندی شوند.

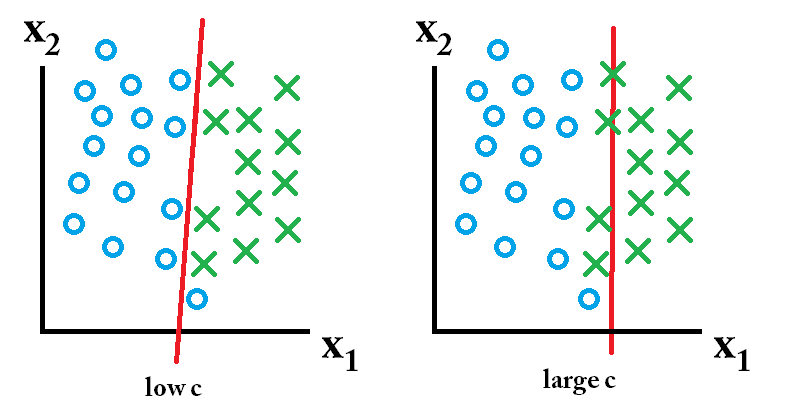
انتخاب این پارامتر به داده های ورودی بستگی دارد. برای مثال در شکل زیر عملکرد یک ماشین بردار پشتیبان در دو حالت با پارامتر C کم و زیاد را مشاهده می کنیم. در صورتی که مقدار پارامتر C کم باشد (شکل چپ)، حاشیه بین داده ها و خط زیاد شده که مطلوب ماست، اما یکی از داده های دسته آبی رنگ که داده پرت محسوب می شود، به اشتباه طبقه بندی می شود. در صورتی که مقدار این پارامتر زیاد باشد، تمامی داده ها به درستی دسته بندی می شوند اما حاشیه بسیار کم می شود.



حال کدام یک از این دو مدل بهتر است؟ این موضوع به داده هایی که قصد پیش بینی و دسته بندی آنها را داریم بستگی دارد. برای مثال در صورتی که داده ها به صورت زیر باشند، مدلی که پارامتر C آن بزرگتر است بهتر عمل می کند.



و در صورتی که داده ها به صورت زیر باشند، مدلی که پارامتر C آن کوچکتر است، عملکرد بهتری دارد.

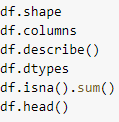


بنابراین مقدار به خصوص و مشخصی برای این پارامتر، که عملکرد مدل را بهبود دهد وجود نداشته و تاثیر آن بر روی مدل به داده های ورودی و همچنین داده های تست ما بستگی دارد.

# گزارش کدنویسی با پایتون

## گام اول:

در ابتدا به بررسی مشخصات و ویژگی های دیتاست موجود می پردازیم. برای خواندن این دیتاست در محیط پایتون[[18]](#footnote-18) از کتابخانه[[19]](#footnote-19) pandas استفاده شده است. اولین مرحله بررسی و تحلیل دیتاست است که با دستورات زیر صورت می گیرد:

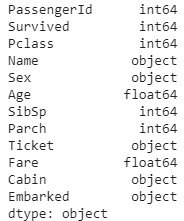
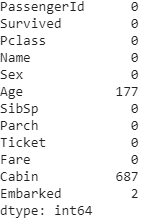


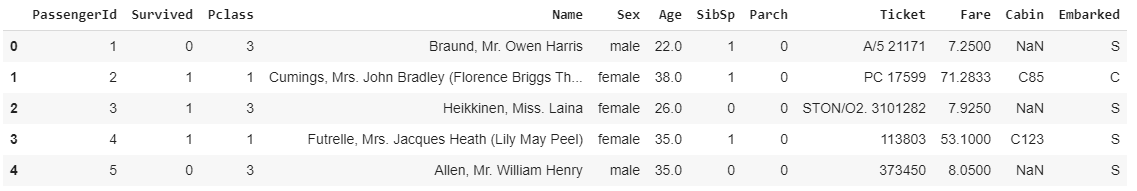
با فراخوانی تابع shape() متوجه می شویم که این دیتاست دارای 12 ستون (ویژگی) و 891 سطر (نمونه) است. با تابع columns لیست 12 ویژگی که در دستور کار نیز آمده است، به نمایش در می آید. در ادامه با استفاده از تابع describe()، مشخصات آماری ویژگی های عددی این دیتاست را به دست می آوریم. از آنجایی که این دیتاست دارای 7 ویژگی عددی و 5 ویژگی غیر عددی می باشد، دستور describe فقط خصوصیات آماری ویژگی های عددی را نمایش می دهد.



اطلاعات آماری که این دستور نمایش می دهد عبارتند از تعداد، میانه، انحراف از معیار، مینیمم و ماکزیمم و چارک ها. برای مثال مشاهده می شود که میانگین سن مسافران کشتی تایتانیک 29 سال بود یا میانگین کرایه پرداختی آن ها 32 دلار است.

همانطور که اشاره شد، این دیتاست 7 ویژگی عددی و 4 ویژگی غیر عددی دارد که با استفاده از تابع dtypes می توانیم آن ها را تشخیص دهیم. برای بررسی مقادیر از دست رفته[[20]](#footnote-20) در دیتاست نیز از دستور isna().sum() استفاده می کنیم که مشاهده می شود این دیتاست در سه ستون سن، شماره کابین و محل سوار شدن به کشتی دارای داده های از دست رفته می باشد که در ادامه به آنها میپردازیم. در نهایت، با فراخوانی توابع head() می توانیم چند سطر اول این دیتاست را مشاهده کنیم.

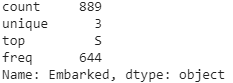




## گام دوم:

با بررسی دقیق تر دیتاست، متوجه می شویم که متغیر هدف یا همان دسته، متغیر زنده ماندن است که متغیر باینری بوده و دو مقدار صفر و یک را میپذیرد. بنابراین در این مسئله با یک متغیر و دو دسته سر کار داریم. بدیهی است که نام و نام خانوادگی مسافران، شماره بلیط و آی دی آنها تاثیری در زنده ماندن آنها ندارد، بنابراین با اطمینان می توانیم این سه ویژگی را از مجموعه داده خود حذف کنیم. در ادامه به برطرف کردن داده های از دست رفته می پردازیم. چرا که الگوریتم های دسته بندی با وجود داده های از دست رفته، عملکرد مناسبی ندارند.

سه متغیر کابین، سن و محل سوار شدن به کشتی[[21]](#footnote-21) دارای مقادیر از دست رفته هستند. ویژگی کابین که بیانگر شماره کابین مسافران است دارای 687 مقدار از دست رفته (از 891 نمونه) است. به عبارت دیگر، 77% از مقادیر این ویژگی از دست رفته اند و تخمین و یا بازیابی آنها غیر ممکن است. بنابراین این ویژگی را نیز به کل از مجموعه داده های خود حذف می کنیم. ویژگی بعدی، محل سوار شدن به کشتی است که فقط دارای 2 مقدار از دست رفته است. از آنجایی که این ویژگی به صورت رسته ای[[22]](#footnote-22) است، این 2 مقدار از دست رفته را با مقداری که بیشترین فراوانی را در دیتاست دارد جایگزین می کنیم. با استفاده از دستور describe پی می بریم که فراوان ترین مقدار 'S' است و بنابراین این دو مقدار از دست رفته را با آن جایگزین می کنیم.



ویژگی آخر سن است که دارای 177 مقدار از دست رفته می باشد. در اینجا از روش میانگین و انحراف از معیار استفاده می کنیم، بدین صورت که به تعداد مقادیر از دست رفته متغیر، در بازه [Mean-STD, Mean+STD] اعداد تصادفی با توزیع یکنواخت تولید می کنیم. سپس مقادیر از دست رفته را با این اعداد تصادفی جایگزین می کنیم. بدین صورت، مشخصات آماری دیتاست تغییری نمی کند و فقط داده های از دست رفته جایگزین می شوند.

از سوی دیگر، باید ویژگی های رسته ای را به ویژگی های عددی تبدیل کنیم، چرا که در اکثر مواقع در فرایند دسته بندی استفاده از این ویژگی ها به صورت مستقیم امکان پذیر نمی باشد. بنابراین با استفاده از تابع map، ویژگی های جنسیت و محل سوار شدن به کشتی را به ویژگی های عددی تبدیل می کنیم. بدین صورت که مسافران مرد، دارای ویژگی جنسیت صفر و مسافران زن دارای ویژگی جنسیت یک می باشند. همچنین مقادیر 'S'، 'C' و 'Q' نیز به ترتیب به 0، 1 و 2 نظیر می شوند.

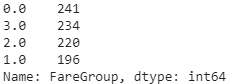
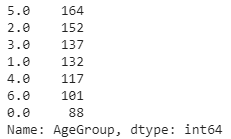
در نهایت دو ویژگی سن و کرایه بلیط دارای مقادیر منحصر به فرد بسیاری هستند که می تواند در عملکرد مدل های دسته بندی نظیر درخت تصمیم گیری، تاثیر منفی بگذارد. بنابراین آن ها را به گروه های متفاوتی تقسیم می کنیم. ویژگی سن را به 7 گروه زیر تقسیم می کنیم:

1. کوچکتر از 15 سال
2. بین 15 و 20 سال
3. بین 20 و 25 سال
4. بین 25 و 30 سال
5. بین 30 و 35 سال
6. بین 35 و 45 سال
7. بزرگتر از 45 سال

و کرایه بلیط را نیز به 4 دسته زیر تقسیم می کنیم:

1. کمتر از 8 دلار
2. بین 8 و 14 دلار
3. بین 14 و 30 دلار
4. بیشتر از 30 دلار

این بازه ها به گونه ای انتخاب شده اند که تعداد اعضای گروه ها تقریبا در یک محدوده باشند (برای مثال یک گروه دارای 5 عضو و گروه دیگر دارای 200 عضو نباشد). این موضوع را می توانیم با اجرای دستور value\_counts بررسی کنیم:

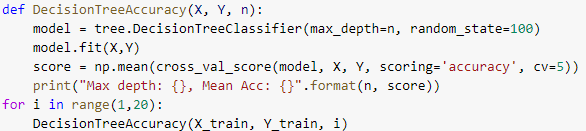
 

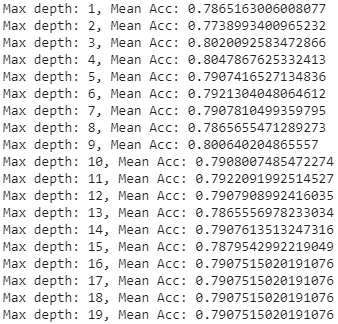
در ادامه، ویژگی زنده ماندن را در یک متغیر دیگر که نام آن Y است و بیانگر برچسب می باشد ریخته و با استفاده از تابع train\_test\_split که در کتابخانه sklearn موجود است، داده های فوق را به داده های آموزشی[[23]](#footnote-23) و تست[[24]](#footnote-24) تقسیم می کنیم. 20% داده ها را به تست و 80% آن ها به داده‌های آموزشی اختصاص می دهیم. این تقسیم به صورت تصادفی انجام می شود و بنابراین در هر بار اجرای برنامه، داده های آموزشی و تست با یکدیگر متفاوت شده و در نتیجه ممکن است نتایج مختلفی بگیریم. برای اینکه در هر بار اجرای برنامه به نتایج ثابتی برسیم، پارامتر random\_state این برنامه را برابر مقدار ثابتی (مثلا 1000) قرار می‌دهیم.



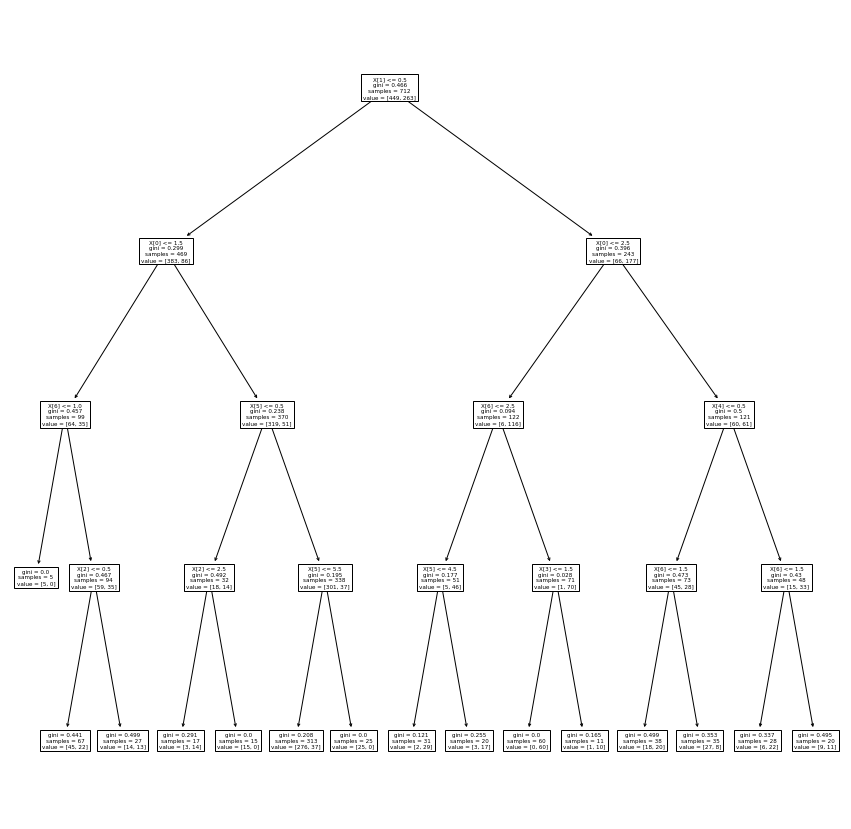
## گام سوم:

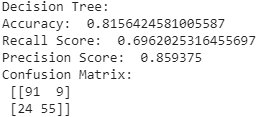
در ابتدا به مدل درخت تصمیم می پردازیم. این مدل در کتابخانه sklearn موجود است و سه پارامتر مهم آن max\_depth، min\_samples\_split و min\_samples\_leaf می باشد که از این سه تاثیر گذار ترین پارامتر max\_depth است که عمق درخت را بیان می کند. بدین منظور تابعی مینویسیم که به ازای مقادیر مختلف این پارامتر، دقت تست مدل را گزارش کند. سپس این تابع را به ازای max\_depth 1 تا 20 اجرا کرده و بهترین مقدار این پارامتر را می یابیم. در اینجا پارامتر random\_stateاین تابع را برابر 100 قرار می دهیم تا در هر بار اجرای تابع، اثر تصادفی نتایج از بین برود. در اینجا برای دقت بیشتر، از فرایند Cross Validation استفاده شده است که در آن داده های آموزشی را به 5 قسمت تقسیم می کنیم و در هر بار یکی از این قسمت ها را به عنوان تست و بقیه را به عنوان داده آموزشی به مدل می دهیم. در نهایت میانگین دقت این 5 بار را به عنوان دقت نهایی مدل بر می گردانیم.





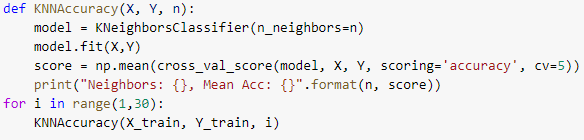
مشاهده می کنیم که به ازای حداکثر عمق 4، مدل بهترین دقت تست را به ما می دهد. پس به ازای عمق 4 یک درخت تصمیم ایجاد کرده و متریک های آن را به عنوان بهترین مدل گزارش می کنیم. این درخت تصمیم به شکل زیر است:

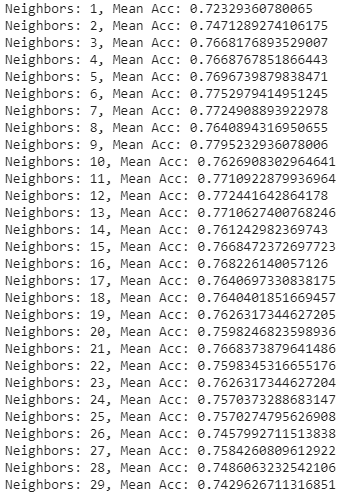




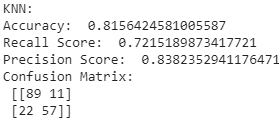
مقادیر TP, TN, FP, FN آن به ترتیب برابر 24، 9، 91، 55 است و امتیاز Recall آن 0.696 و امتیاز Precision آن 0.859 و امتیاز Accuracy آن 0.815 است. این مقادیر از طریق فرمول های زیر بدست می آیند:

در ادامه به مدل کی-نزدیکترین همسایه[[25]](#footnote-25) (یا به اختصار KNN) می پردازیم. این مدل دارای یک پارامتر کلیدی به نام n\_neighbors است که همان مقدار K می باشد. همانند قسمت قبل، با استفاده از یک تابع و حلقه مقدار بهینه K را تعیین می کنیم.



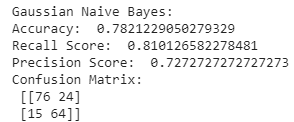


مشاهده می کنیم که به ازای K=9 به بیشترین دقت می رسیم بنابراین مدل را به ازای این مقدار اجرا کرده و پارامتر های خروجی آن را گزارش می کنیم.



مقادیر TP, TN, FP, FN آن به ترتیب برابر 22، 11، 89، 57 است و امتیاز Recall آن 0.721، امتیاز Precision آن 0.838 و امتیاز Accuracy آن 0.815 است. دقت شود Accuracy که در اینجا گزارش می شود مربوط به دقت تست است و از مقدار دقت حاصل ازCross Validation ، به علت تعداد بیشتر نمونه ها، کمی بیشتر است.

در نهایت به مدل Gaussian Naïve Bayes می پردازیم. این مدل پارامتر قابل تنظیم[[26]](#footnote-26) کردنی ندارد و بنابراین آن را به صورت ساده اجرا کرده و دقت آنرا گزارش می کنیم.

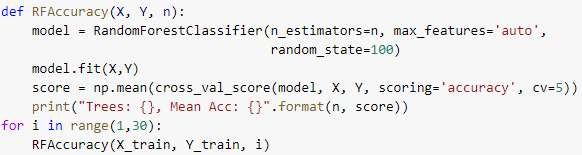


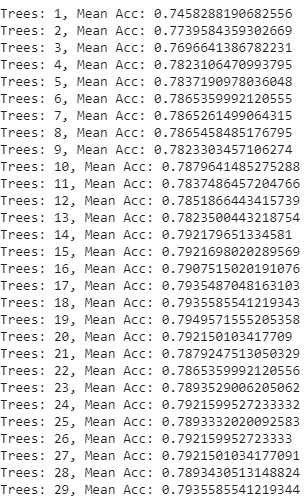
مقادیر TP, TN, FP, FN آن به ترتیب برابر 15، 24، 76، 64 است و امتیاز Recall آن 0.810، امتیاز Precision آن 0.727 و امتیاز Accuracy آن 0.782 است.

به نظر می رسد که از بین مدل های فوق، هر دو مدل KNN و درخت تصمیم عملکرد بهتری را نسبت به مدل Naïve Bayes دارند، با این تفاوت که مدل درخت تصمیم امتیاز Precision بیشتری نسبت به مدل KNN دارد، اما امتیاز Recall آن کمتر است. همچنین در هر دو حالت تعداد نمونه هایی که به اشتباه دسته بندی شده اند برابر 33 می باشد. در این مسئله، دقت Accuracy مدل اهمیت بیشتری نسبت به دو دقت دیگر دارد که برای هر دو مدل KNN و درخت تصمیم برابر بدست آمد.

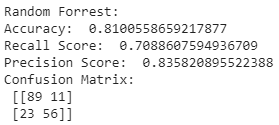
## گام چهارم:

یکی دیگر از مدل هایی که عملکرد خوبی در دسته بندی دارد، مدل جنگل تصادفی[[27]](#footnote-27) است که در واقع مجموعه ای از درخت های تصمیم می باشد. این مدل یک پارامتر تعیین کننده n\_estimators دارد که برابر تعداد درخت ها در هر جنگل می باشد. همانند قسمت های قبل مقدار بهینه این پارامتر را محاسبه می کنیم:



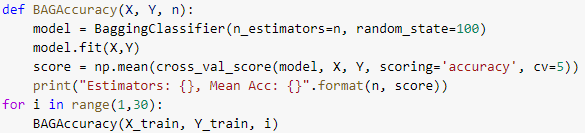


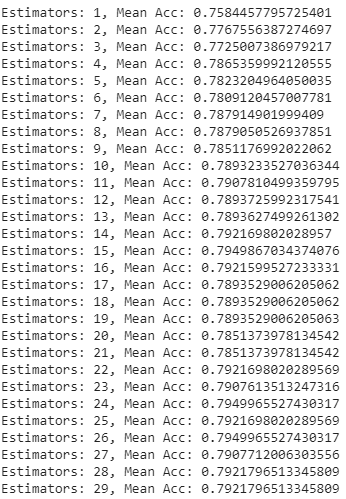
مشاهده می کنیم که به ازای n=18 (18 درخت در یک جنگل)، بیشترین مقدار دقت حاصل از Cross Validation به دست می آید. پس مدل ار به ازای این پارامتر بار دیگر اجرا می کنیم و دقت تست آن را محاسبه می کنیم:



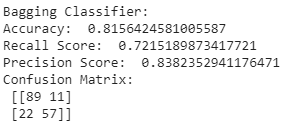
مقادیر TP, TN, FP, FN آن به ترتیب برابر 23، 11، 89، 56 است و امتیاز Recall آن 0.708، امتیاز Precision آن 0.835 و امتیاز Accuracy آن 0.810 است.

مدل دیگر Bagging Classifier است که همانند جنگل تصادفی، یک پارامتر تعیین کننده دارد که تعداد تخمین دهنده های مدل است. همانند قسمت قبل عمل می کنیم:



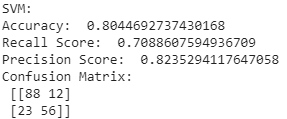


مشاهده می کنیم که به ازای n=24 (24 تخمین زننده)، بیشترین مقدار دقت حاصل از Cross Validation به دست می آید. پس مدل ار به ازای این پارامتر بار دیگر اجرا می کنیم و دقت تست آن را محاسبه می کنیم:



مقادیر TP, TN, FP, FN آن به ترتیب برابر 22، 11، 89، 57 است و امتیاز Recall آن 0.721، امتیاز Precision آن 0.838 و امتیاز Accuracy آن 0.815 است.

در نهایت، در صورت استفاده از مدل ماشین بردار پشتیبان خطی، مقادیر زیر به دست می آید:



از بین مدل های فوق، سه مدل Bagging Classifier، KNN و درخت تصمیم نتایج بسیار مشابهی را ارائه دادند که به نظر این سه مدل مدل های بهینه برای دسته بندی این دیتاست هستند.

1. Prediction [↑](#footnote-ref-1)
2. Classification [↑](#footnote-ref-2)
3. Regression [↑](#footnote-ref-3)
4. Label [↑](#footnote-ref-4)
5. Anomaly [↑](#footnote-ref-5)
6. Clustering [↑](#footnote-ref-6)
7. Sequential Patterns [↑](#footnote-ref-7)
8. Decision Trees [↑](#footnote-ref-8)
9. Supervised [↑](#footnote-ref-9)
10. Unsupervised [↑](#footnote-ref-10)
11. Class [↑](#footnote-ref-11)
12. Support Vector Machine [↑](#footnote-ref-12)
13. Neural Networks [↑](#footnote-ref-13)
14. Kernel [↑](#footnote-ref-14)
15. Support Vectors [↑](#footnote-ref-15)
16. Margin [↑](#footnote-ref-16)
17. Loss Function [↑](#footnote-ref-17)
18. Python [↑](#footnote-ref-18)
19. Library [↑](#footnote-ref-19)
20. Missing Values [↑](#footnote-ref-20)
21. Embarked [↑](#footnote-ref-21)
22. Categorical [↑](#footnote-ref-22)
23. Train [↑](#footnote-ref-23)
24. Test [↑](#footnote-ref-24)
25. K-Nearest Neighbor [↑](#footnote-ref-25)
26. Tune [↑](#footnote-ref-26)
27. Random Forrest [↑](#footnote-ref-27)