پروژه درس داده کاوی

مریم واقعی[[1]](#footnote-1)

مژگان پیوندیان شعرباف

|  |  |  |
| --- | --- | --- |
| اطلاعات گزارش |  | چکیده |
| **تاریخ:**  **15 بهمن 1400** |  | در اين گزارش‏‌ قرار است پروژه ی خود در درس داده کاوی را با تکیه بر دانش آموخته شده در کلاس و ابزار‌ها و کتابخانه های زبان برنامه نویسی پایتون، تشریح کنیم. این پروژه به دو بخش فاز اول که شامل شناخت داده و پیش پردازش روی داده ها و فاز دوم شامل ساخت مدل ها و ارزیابی آنها و همچنین پیدا کردن الگوهای پرتکرار و قوانین انجمنی تقسیم می‌شود.  ما در این پروژه برای ارزیابی مدل های بدون نظارت از روش بصری سازی خوشه ها و برای مدل های با نظارت از دو روش ارزیابی accuracy score و k-fold استفاده کردیم. همچنین در مدل های با نظارت، برای درک بهتر میزان درستی و نادرستی پیش بینی کلاس ها ماتریس confusion برای داده های آموزش و داده های تست را رسم کرده ایم. |
| **واژگان كليدي:**  داده کاوی  زبان برنامه نویسی پایتون  پیش پردازش داده ها  الگوریتم های بی نظارت  الگوریتم های با نظارت  الگوریتم Kmeans  الگوریتم DBScan  شبکه عصبی  الگوریتم SVM  الگوریتم KNN  درخت تصمیم  الگوریتم bayes  الگوهای پرتکرار  قوانین انجمنی |  |

فهرست مطالب

[مقدمه 3](#_Toc94800569)

[بخش اول: آشنایی با مجموعه داده 4](#_Toc94800570)

[بخش دوم: تحلیل مجموعه داده 6](#_Toc94800571)

[2.1- خصوصیات هر ویژگی 7](#_Toc94800572)

[2.2- کشیدگی داده های هر ستون به کمک نمودار هیستوگرام 7](#_Toc94800573)

[2.3- صحت، معتبر بودن و کامل بودن داده ها 9](#_Toc94800574)

[2.4- نمودار هیستوگرام برای تمام ویژگی ها 10](#_Toc94800575)

[2.5- شناسایی داده های پرت در هر ویژگی 10](#_Toc94800576)

[2.6- ماتریس عدم شباهت 16](#_Toc94800577)

[2.7-بررسی همبستگی ویژگی ها 17](#_Toc94800578)

[2.8-نمایش نمودار scatter برای ستون های همبسته 18](#_Toc94800579)

[بخش سوم: پیش پردازش داده ها 20](#_Toc94800580)

[3.1-پاکسازی داده ها 20](#_Toc94800581)

[3.2- افزونگی در سطح ویژگی و رکورد 20](#_Toc94800582)

[3.3- کاهش در سطح رکورد 20](#_Toc94800583)

[3.4- نرمال سازی داده ها 21](#_Toc94800584)

[3.5- کاهش ابعاد داده ها 21](#_Toc94800585)

[بخش چهارم: مدل سازی و داده کاوی 21](#_Toc94800586)

[4.1- پیاده سازی الگوریتم های بی نظارت 21](#_Toc94800587)

[4.1.1- الگوریتم kmeans 21](#_Toc94800588)

[4.1.2- الگوریتم DBScan 22](#_Toc94800589)

[4.1.3- مقایسه دو الگوریتم Kmeans و DBScan 23](#_Toc94800590)

[4.2- پیاده سازی الگوریتم های با نظارت 24](#_Toc94800591)

[4.2.1- مدل سازی شبکه عصبی 24](#_Toc94800592)

[4.2.2- الگوریتم طبقه بندی SVM 26](#_Toc94800593)

[4.2.3- الگوریتم طبقه بندی KNN 28](#_Toc94800594)

[4.2.4- الگوریتم طبقه بندی bayes 32](#_Toc94800595)

[4.2.5- الگوریتم طبقه بندی decision tree 33](#_Toc94800596)

[4.2.6- روش ensemble 36](#_Toc94800597)

[بخش پنجم: الگوهای پرتکرار و قوانین انجمنی 38](#_Toc94800598)

[5.1- الگوهای پرتکرار 38](#_Toc94800599)

[5.2- قوانین انجمنی 42](#_Toc94800600)

[نتیجه گیری 45](#_Toc94800601)

[منابع و مراجع 46](#_Toc94800602)

# مقدمه

در گذشته و در عصری که اینترنت و تکنولوژی برای بشریت معنا نداشت داده ها و دانش و اطلاعات صرفا در ذهن انسان ها، روی دیوار غارها، برگ درختان، پوست حیوانات و بعدها روی کاغذ ذخیره می‌شد، اما امروزه با پیشرفت تکنولوژی ما در دنیایی از داده ها غرق شده ایم و داده ها به جای هک شدن بر روی دیوار غارها و برگ درختان امروزه در گوشی های همراه، کامپیوتر های شخصی، لپ تاپ ها و سرور ها ذخیره شده اند و زندگی ما انسان ها آنچنان به این داده ها وابسته شده که گویی اگر روزی تمام این داده ها و اطلاعات از بین بروند عملا ما انسان ها به عصر انسان های اولیه بازگشته ایم و زنذگی ما به اندازه همان زمان سخت و طلاقت فرسا خواهد شد.

تمام این ها نشان می‌دهد که چه میزان داده ها در زندگی انسان ها نقش حیاتی و مهمی دارند و بنابراین باید این داده ها را حفظ کرد و علاوه بر حفظ آنها، از این داده ها دانش جدید بدست آورد و در اینجاست که داده کاوی و استخراج اطلاعات اهمیت و معنا پیدا می‌کند.

داده کاوی این قابلیت را ارائه می دهد که داده ها را با دیدی جدید مشاهده کنیم، و ارتباط ها و الگوهایی را که قبلاً مورد توجه نبوده اند، کشف کنیم.

ما در این پروژه سعی کردیم تا با تکیه بر 4 مرحله اصلی داده کاوی یعنی جمع آوری داده، پیش پردازش داده، داده کاوی و درنهایت ارزیابی الگو ها بتوانیم در نهایت به مدل هایی کارا برای مجموعه داده های خود برسیم.

در فاز جمع آوری داده ما از مجموعه داده ی المپیک توکیو 2021 استفاده کردیم[[2]](#footnote-2). سپس در فاز پیش پردازش عملیات پاکسازی داده ها شامل حذف افزونگی ها، پر کردن داده های از دست رفته، کاهش ابعاد در سطح ویژگی و داده را انجام دادیم تا داده ها برای مرحله بعد یعنی فاز داده کاوی آماده شوند. ما در فاز داده کاوی با اعمال الگوریتم های مختلف داده کاوی از جمله الگوریتم های kmeans و dbscan برای داده های بدون برچسب و الگوریتم های mlp و svm و knn و bayes و decision tree و ensemble برای داده های دارای برچسب سعی کردیم تا مدل هایی را متناسب با داده های خود ایجاد کنیم و درنهایت نیز در فاز ارزیابی ما با روش های ارزیابی با محاسبه ی دقت با متریک های accuracy و k-fold، دقت مدل ها را مورد بررسی و ارزیابی قرار دادیم.

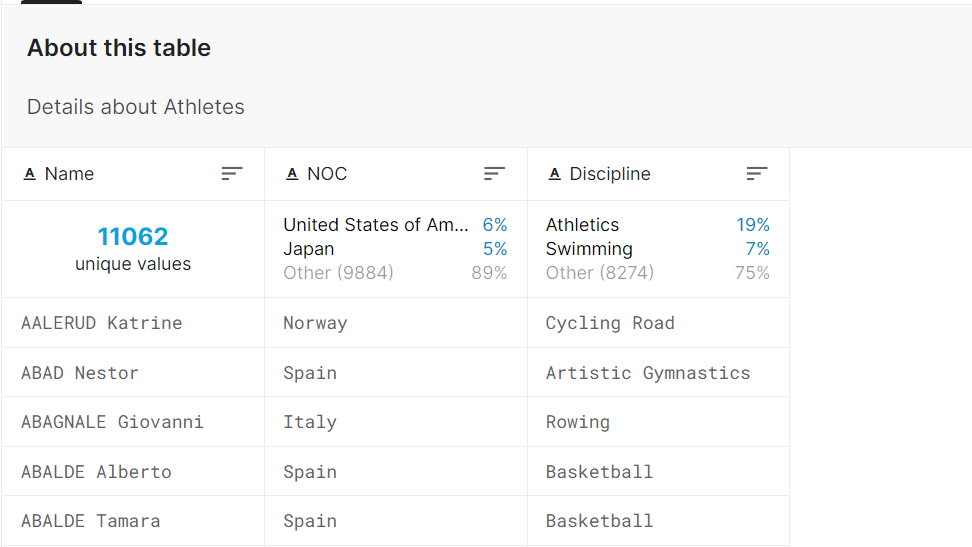
# بخش اول: آشنایی با مجموعه داده

ما در این پروژه از مجموعه داده 2021\_olympics استفاده کرده ایم که منبع اصلی این دیتاست سایت kaggle [[3]](#footnote-3) می باشد.

این دیتاست شامل 5 فایل با نام های زیر است:

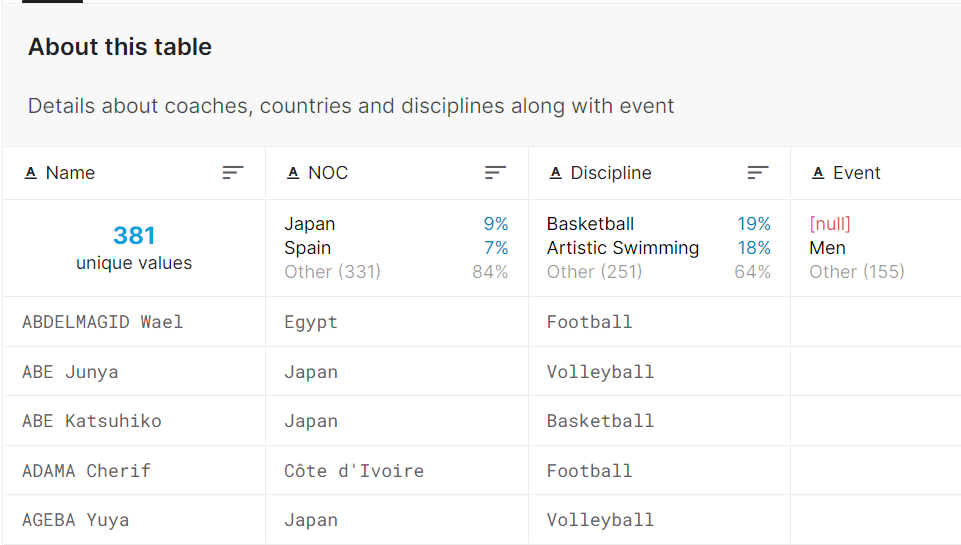
1. Athletes.xlsx
2. Coaches.xlsx
3. EntriesGender.xlsx
4. Medals.xlsx
5. Teams.xlsx

در فایل Athletes.xlsx اطلاعات ورزشکاران گزارش شده که این شامل سه ویژگی نام ورزشکاران، نام کشورشان و رشته ورزشی آنها می باشد. این فایل 11062 ردیف منحصر به فرد دارد.



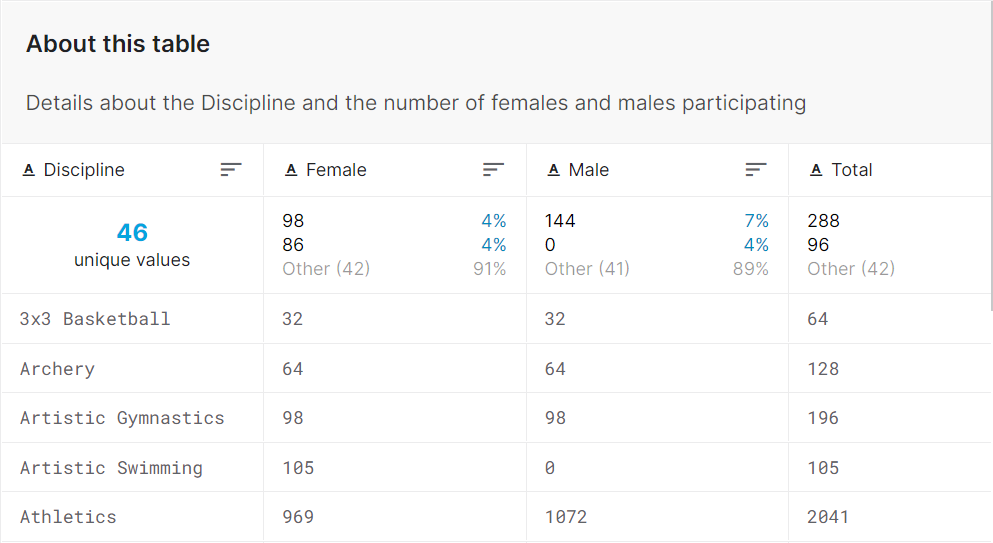
تصویر 1- جزئیات اطلاعات ستون ها در فایل Athletes.xlsx

در فایل Coaches.xlsx اطلاعات مربیان تیم ها از جمله نام آنها، نام کشور و نام رشته ورزشی و همچنین رویدادی که در آن حضور دارند را نشان میدهد. در این فایل یکسری مقادیر null در ستون events وجود دارد که در فاز اول مقادیر آن را پر خواهیم کرد. این فایل همچنین شامل 381 ردیف منحصر به فرد می‌باشد.



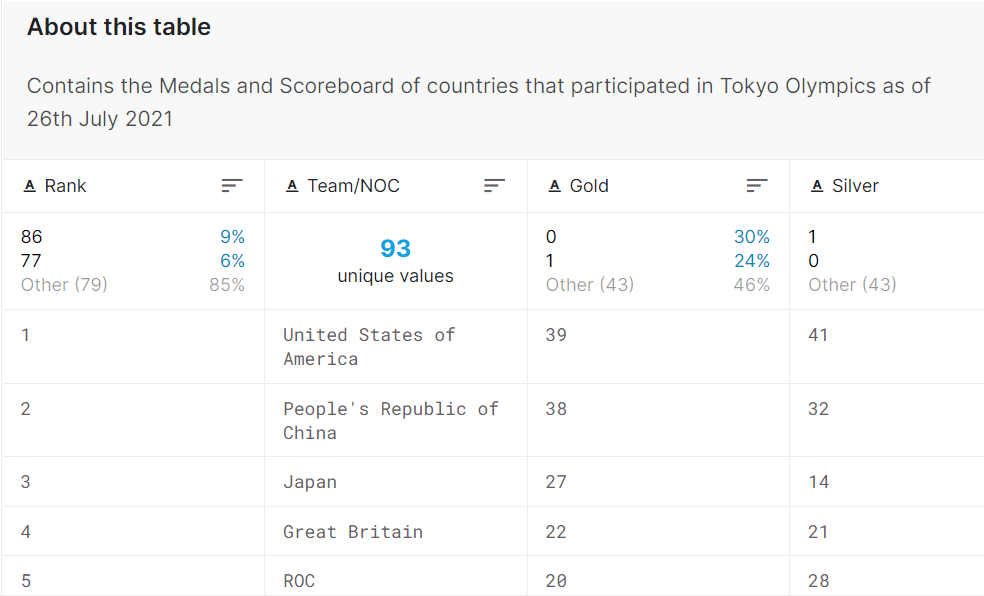
تصویر 2 - جزئیات اطلاعات ستون ها در فایل Coaches.xlsx

فایل EntriesGender.xlsx اطلاعاتی در مورد تعداد خانم ها و آقایان و تعداد کل ورزشکاران در هر رشته ورزشی را نشان می دهد که این فایل شامل 42 ردیف منحصر به فرد میباشد.



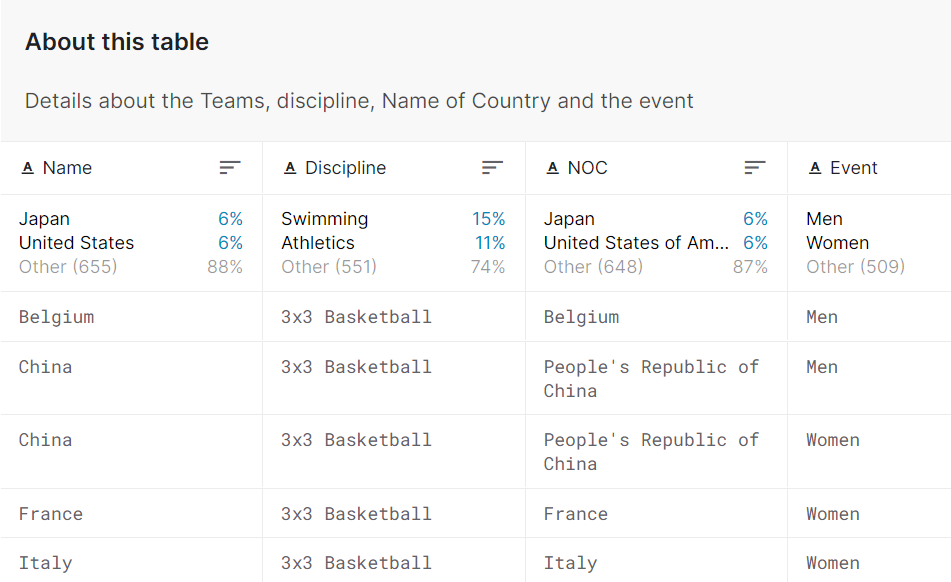
تصویر 3 - اطلاعات ستون ها در فایل EntriesGender.xlsx

فایل Medals.xlsx اطلاعاتی در مورد نام کشور و تعداد مدال های طلا، نقره و برنز و همچنین رتبه هر کشور به ما میدهد. این فایل شامل 93 ردیف منحصر به فرد می‌باشد.



تصویر 4 - اطلاعات ستون ها در فایل Medals.xlsx

در نهایت در فایل Teams.xlsx ما اطلاعاتی شامل نام تیم، رشته ورزشی، نام کشور و رویدادی که آن تیم در آن شرکت میکند را مشاهده میکنیم. این فایل نیز شامل 743 ردیف می باشد.



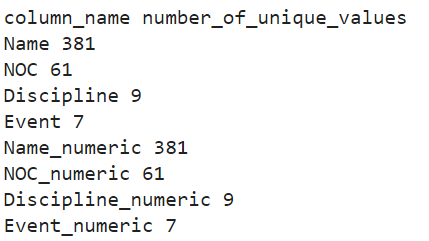
تصویر 5 - اطلاعات ستون ها در فایل Teams.xlsx

# بخش دوم: تحلیل مجموعه داده

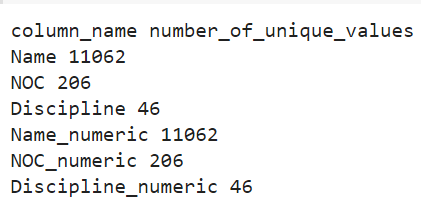
در ابتدا و قبل از شروع پیش پردازش چون داده های ما بیشتر از جنس رشته هستند، فلذا ما برای این ستون ها، یک ستون جدید به هر دیتافریم مربوط به هر فایل اضافه کردیم که در آن مقادیر رشته ای هر ستون را به مقادیر عددی نگاشت کردیم.

این کار به این دلیل است ما که در ادامه و در برخی مدل ها و الگوریتم ها نمیتوان از مقادیر رشته ای استفاده کرد. در نتیجه لازم است تا نگاشت مقادیر رشته ای به اعداد را داشته باشیم.

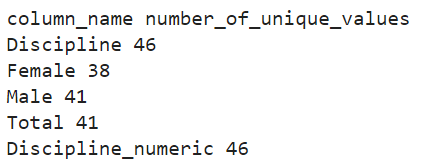
در ابتدا ما به بررسی مجموعه داده پرداختیم. ابتدا تعداد مقادیر منحصر به فرد در هر ستون را بدست آوردیم که به شرح زیر میباشد:



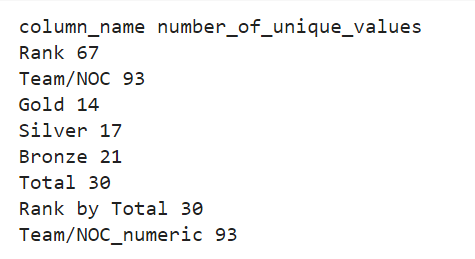
تصویر 6- تعداد مقادیر منحصر به فرد در ستون های فایل Coaches.xlsx



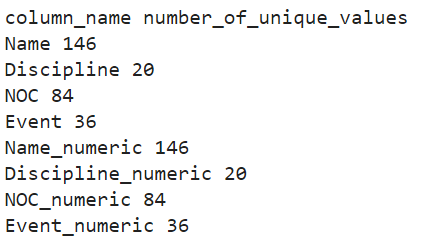
تصویر 7 - تعداد مقادیر منحصر به فرد در ستون های فایل Athletes.xlsx



تصویر 8 - تعداد مقادیر نحصر به فرد در ستون های فایل EntriesGender.xlsx



تصویر 9 - تعداد مقادیر منحصر به فرد در ستون های فایل Medals.xlsx



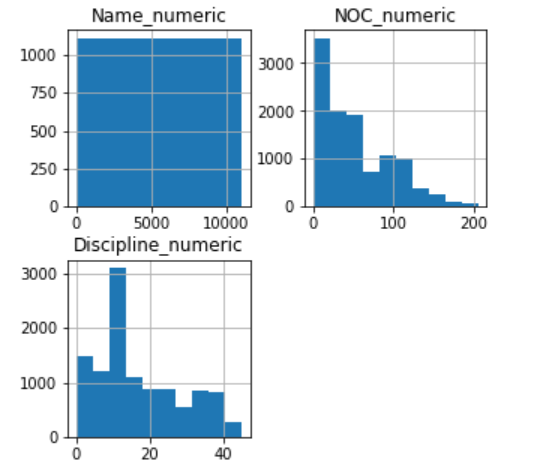
تصویر 10 - تعداد مقادیر منحصر به فرد در ستون های فایل Teams.xlsx

## 2.1- خصوصیات هر ویژگی

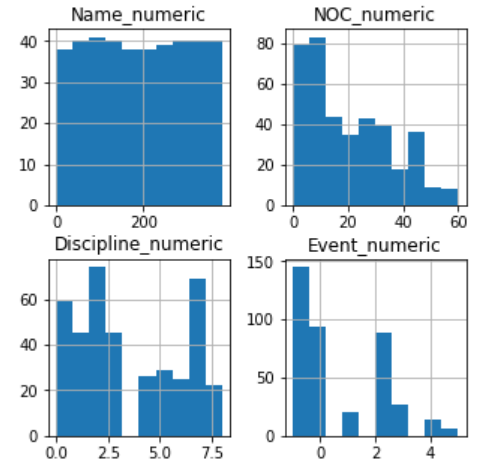
ما اطلاعات ستون ها در تمام دیتافریم ها شامل نام، نوع داده، بازه مقادیر، میانگین، میانه، مد و کمترین و بیشترین مقدار را بدست آوردیم و در متغیر دیکشنری تمام این اطلاعات را ذخیره کردیم.

## 2.2- کشیدگی داده های هر ستون به کمک نمودار هیستوگرام

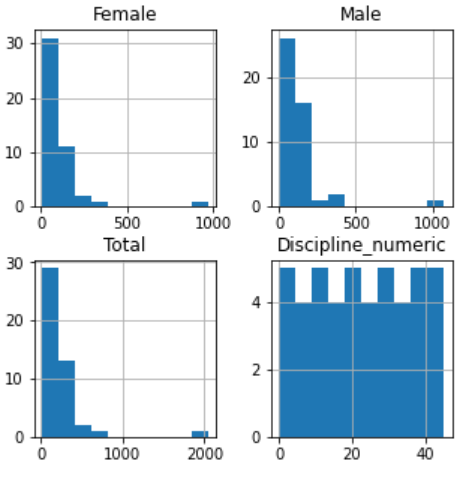
پس از آن ما نحوه توزیع داده ها و چولگی داده ها را به کمک نمودار هیستوگرام نمایش دادیم. که به صورت زیر میباشد:



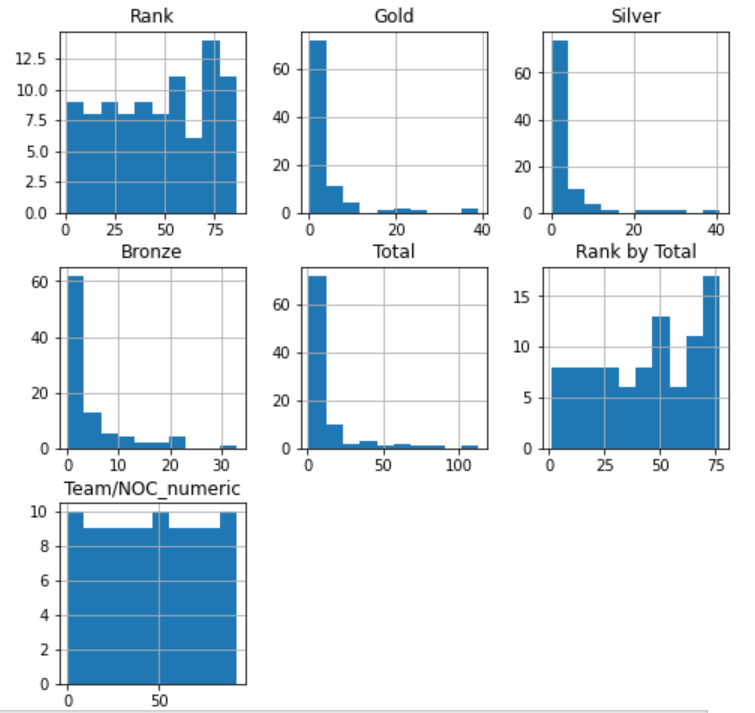
تصویر 11 - نمودار هیستوگرام برای ستون های عددی دیتافریم athletes\_df



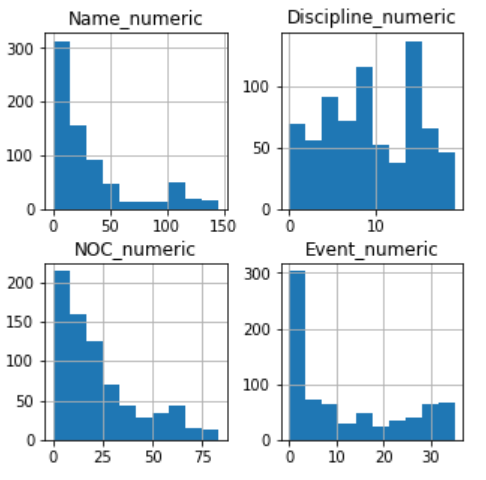
تصویر 12 - نمودار هیستوگرام ستون های عددی در دیتافریم coaches\_df



تصویر 13 - نمودار هیستوگرام مربوط به ستون های عددی دیتافریم entries\_gender\_df



تصویر 14 - نمودار هیستوگرام ستون های عددی در دیتافریم medals\_df



تصویر 15 - نمودار هیستوگرام ستون های عددی در دیتافریم teams\_df

## 2.3- صحت، معتبر بودن و کامل بودن داده ها

برای چک کردن کامل بودن داده ها ما به کمک متد info در کتابخانه pandas برای هر دیتافریم بررسی کردیم که هر ستون چه تعداد داده از دست رفته داریم. که خب با بررسی که انجام دادیم متوجه شدیم تنها ستونی که مقدار از دست رفته دارد ستون Events از دیتافریم coaches\_df می باشد. که مقادیر از دست رفته را در بخش های بعدی با مقادیر مناسب پر میکنیم.

برای چک کردن صحت داده ها از آنجایی که تمام ستون های عددی ما تعداد را نشان میدهند مثل تعداد خانم ها و آقایان در رشته های ورزشی یا تعداد مدال ها و... نبابراین چون نشان دهنده تعداد است و تعداد نمیتواند عدد منفی باشد پس یکی از چک هایی که روی مقادیر عددی انجام دادیم این است که این مقادیر آیا نامنفی هستند یا خیر.

مورد بعد که برای صحت مقادیر چک کردیم این است که در دیتافریم entries\_gender ما تعداد خانم ها و آقایان و تعداد کل آنها را در هر رشته ورزشی داریم. بنابراین تعداد کل ورزشکاران در یک رشته ورزشی باید برابر با جمع تعداد خانم ها و آقایان در آن رشته ورزشی باشد پس این مورد را هم چک کردیم که آیا تعداد کل برابر با مجموع تعداد خانم ها و آقایان است یا خیر.

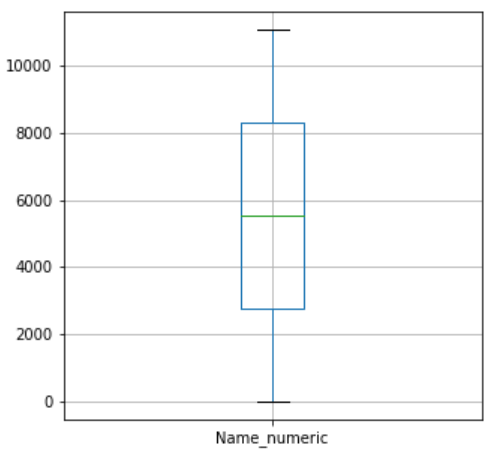
بعد از بررسی هایی که انجام دادیم برای صحت و معتبر بودن داده ها خوشبختانه ما داده نامعتبر و نادرست در دیتافریم ها نداشتیم.

## 2.4- نمودار هیستوگرام برای تمام ویژگی ها

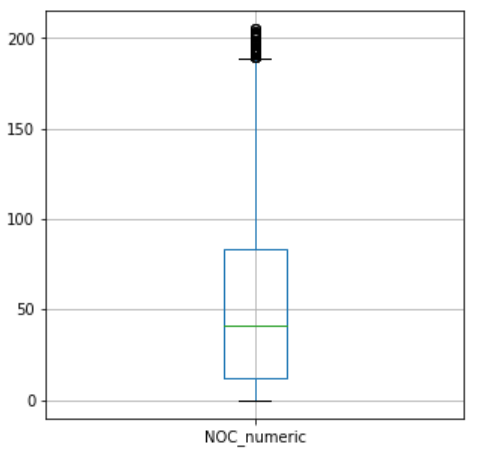
ما در بخش کشیدگی داده های هر ستون از نمودار هیستوگرام استفاده کردیم، فلذا برای مشاهده نمودار هیستوگرام هر ویژگی میتوانید به بخش 1.2 مراجعه کنید.

## 2.5- شناسایی داده های پرت در هر ویژگی

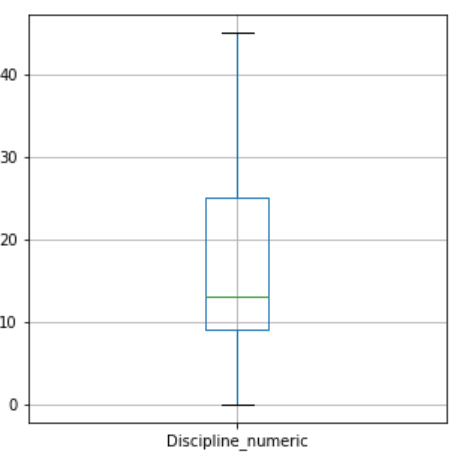
برای شناسایی داده های پرت در هر ویژگی از نمودار boxplot استفاده کردیم.



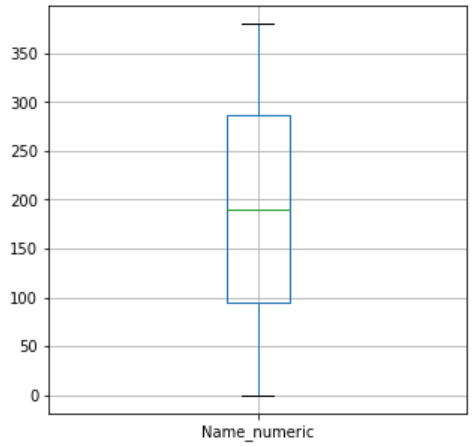
نمودار 1- نمودارجعبه ای ستون Name\_numeric از دیتافریم athletes\_df



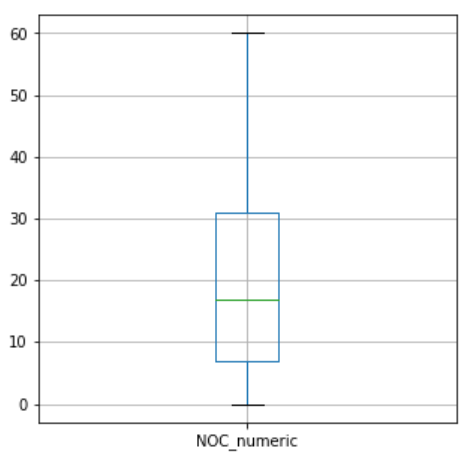
نمودار 2- نمودار جعبه ای ستون NOC\_numeric از دیتافریم athletes\_df



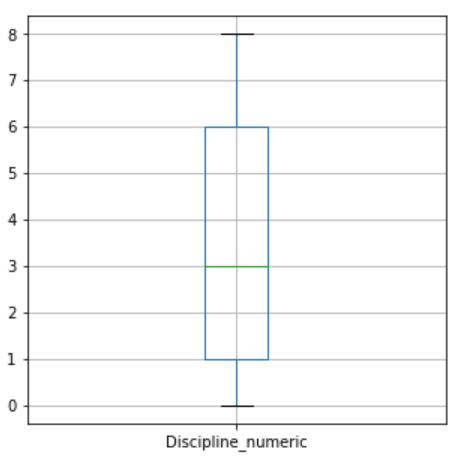
نمودار 3 - نمودار جعبه ای ستون Discipline\_numeric از دیتافریم athletes\_df



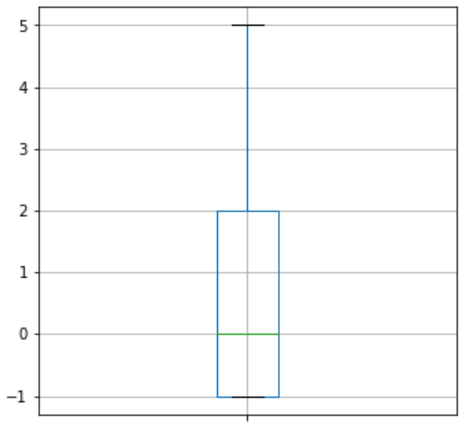
نمودار 4 - نمودار جعبه ای ستون Name\_numeric از دیتافریم coaches\_df



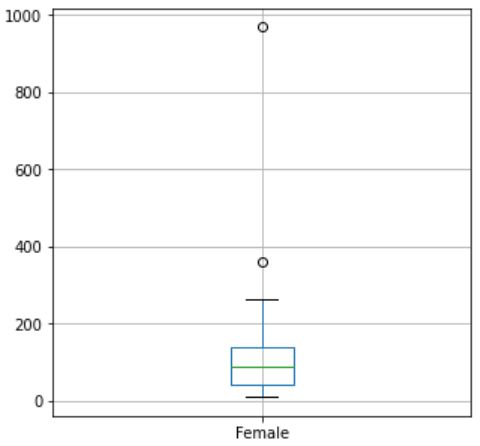
نمودار 5 - نمودار جعبه ای ستون NOC\_numeric از دیتافریم coaches\_df



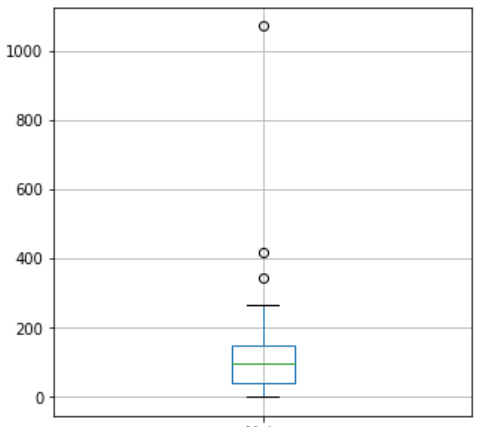
نمودار 6 - نمودار جعبه ای ستون Discipline\_numeric



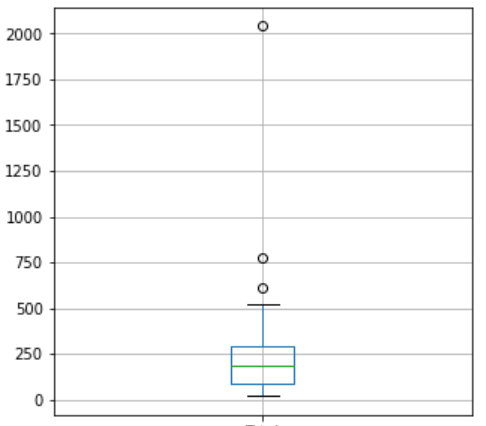
نمودار 7 - نمودار جعبه ای ستون Event\_numeric از دیتافریم coaches\_df



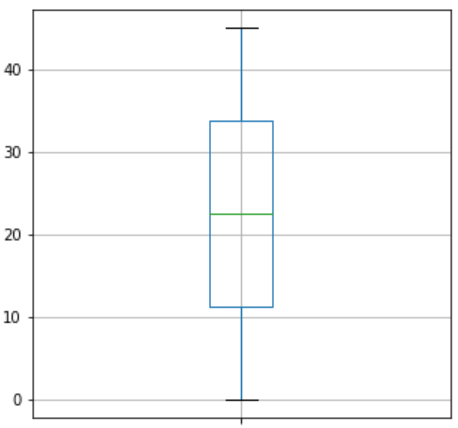
نمودار 8 - نمودار جعبه ای ستون Female از دیتافریم entries\_gender\_df



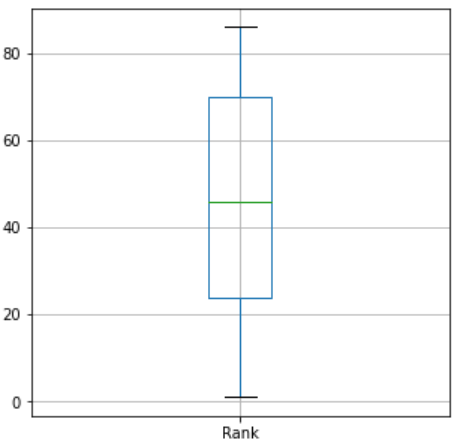
نمودار 9 - نمودار جعبه ای ستون Male از دیتافریم entries\_gender\_df



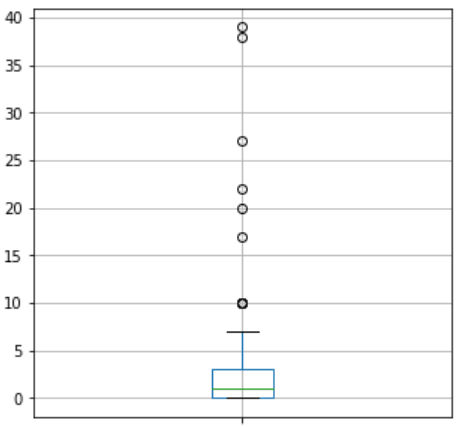
نمودار 10 - نمودار جعبه ای ستون Total از دیتافریم entries\_gender\_df



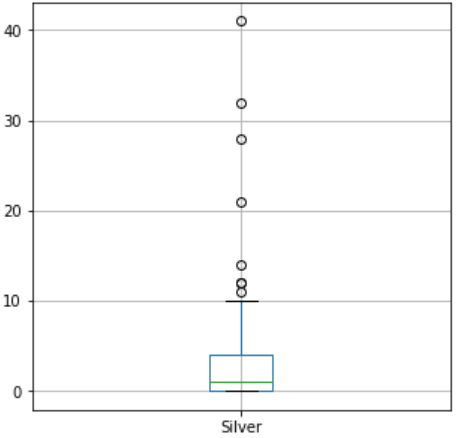
نمودار 11 - نمودار جعبه ای ستون Discipline\_numeric از دیتافریم entries\_gender\_df



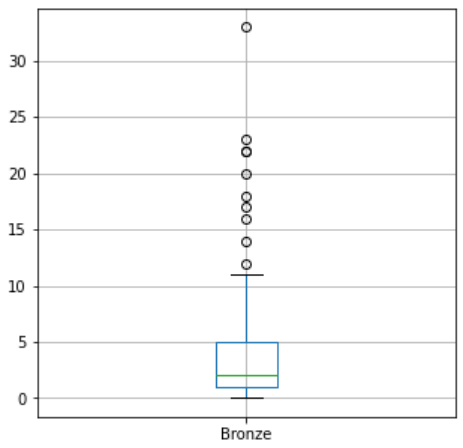
نمودار 12 - نمودار جعبه ای ستون Rank از دیتافریم medals\_df



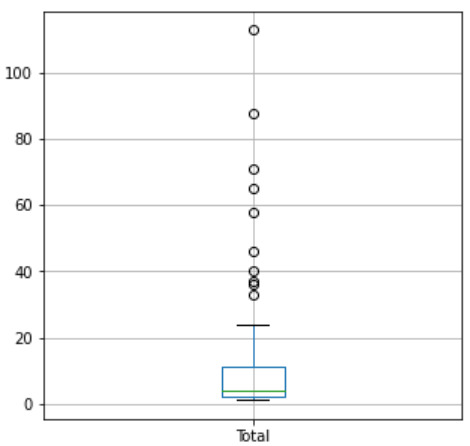
نمودار 13 - نمودار جعبه ای ستون Gold از دیتافریم medals\_df



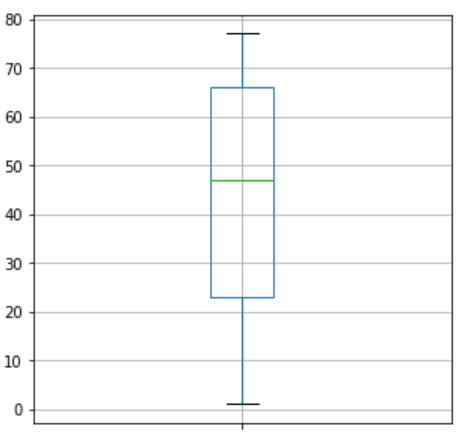
نمودار 14 - نمودار جعبه ای ستون Silver از دیتافریم medals\_df



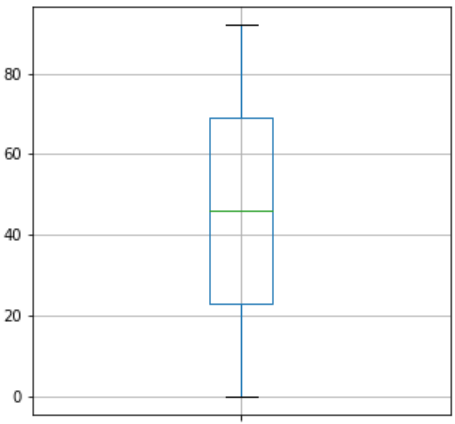
نمودار 15 - نمودار جعبه ای ستون Bronze از دیتافریم medals\_df



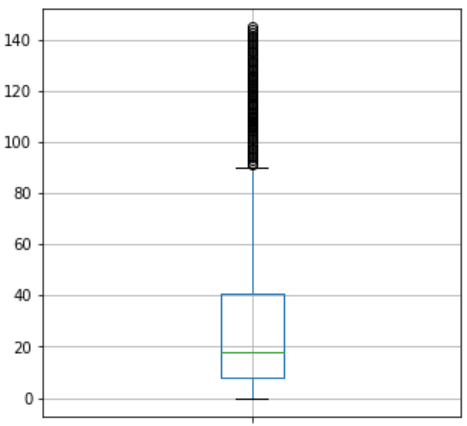
نمودار 16 - نمودار جعبه ای ستون Total از دیتافریم medals\_df



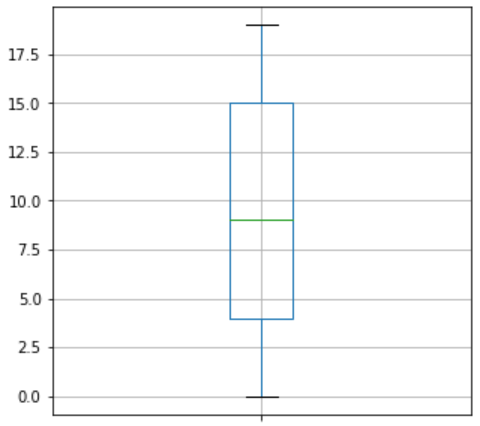
نمودار 17 - نمودار جعبه ای ستون Rank by Total از دیتافریم medals\_df



نمودار 18 - نمودار جعبه ای ستون Team/NOC\_numeric از دیتافریم medals\_df



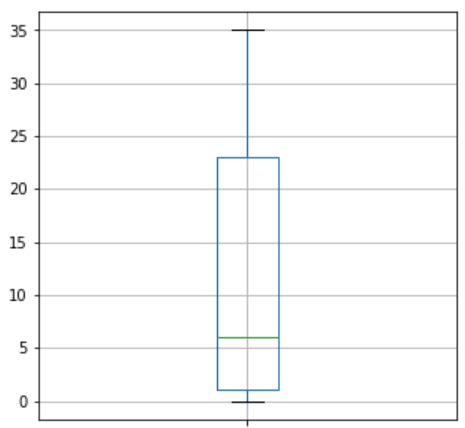
نمودار 19 - نمودار جعبه ای ستون Name\_numeric از دیتافریم teams\_df



نمودار 20 - نمودار جعبه ای ستون Discipline\_numeric از دیتافریم teams\_df



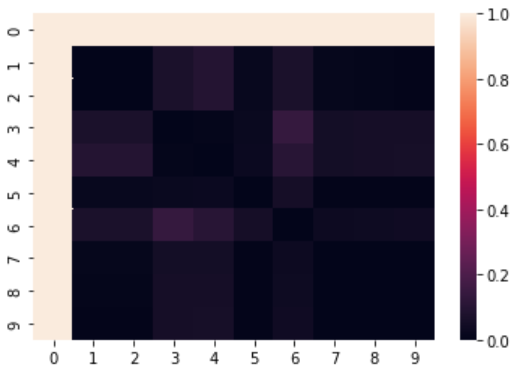
نمودار 21 - نمودار جعبه ای ستون NOC\_numeric از دیتافریم teams\_df



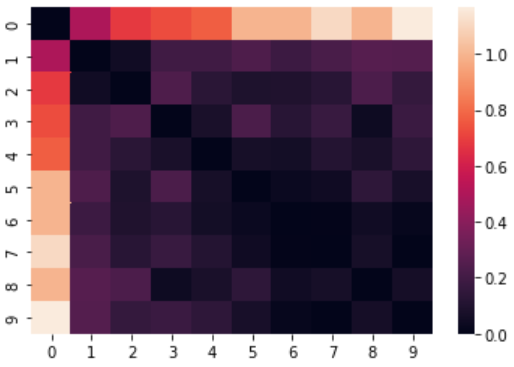
نمودار 22 - نمودار جعبه ای ستون Event\_numeric از دیتافریم teams\_df

## 2.6- ماتریس عدم شباهت

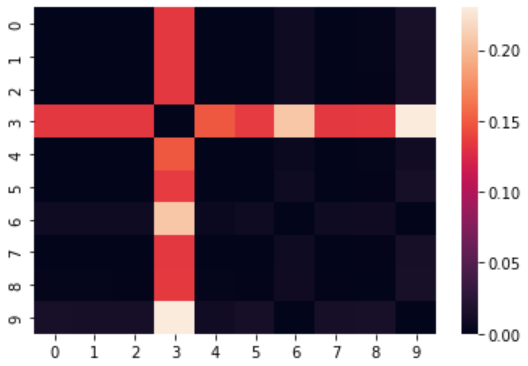
ما ماتریس عدم شباهت را با استفاده از شباهت کسینوسی برای حداکثر 10 ردیف از هر دیتافریم محاسبه کردیم و مقادیر ماتریس عدم شباهت را با نمودار heatmap نمایش داده ایم که به شرح زیر است:



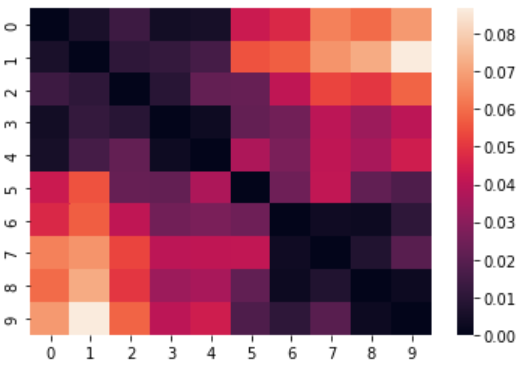
نمودار 23 - ماتریس عدم شباهت 10 ردیف از دیتافریم athletes\_df



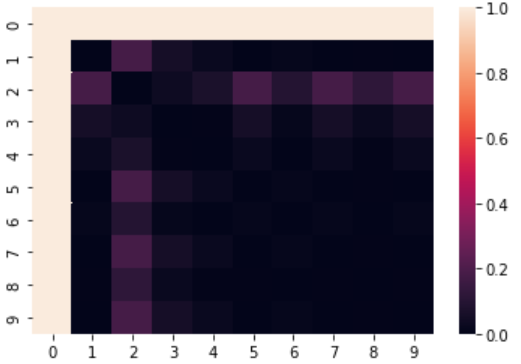
نمودار 24 - ماتریس عدم شباهت 10 ردیف از دیتافریم coaches\_df



نمودار 25 - ماتریس عدم شباهت 10 ردیف از دیتافریم entries\_gender\_df



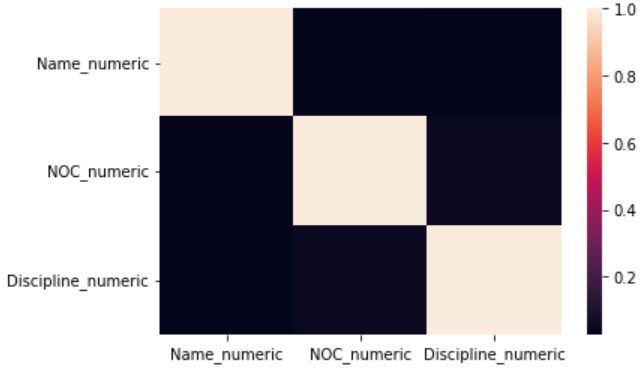
نمودار 26 - ماتریس عدم شباهت 10 ردیف از دیتافریم medals\_df



نمودار 27- ماتریس عدم شباهت 10 ردیف از دیتافریم teams\_df

## 2.7-بررسی همبستگی ویژگی ها

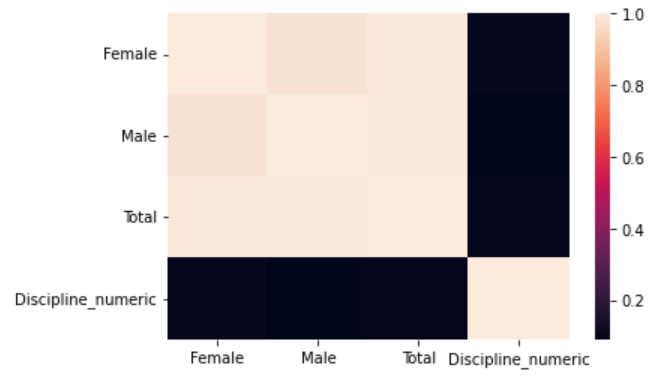
ما همبستگی ویژگی ها را به کمک معیار correlation و یا استفاده از متد corr در کتابخانه pandas بدست آوردیم و سپس با کمک نمودار heatmap همبستگی آنها را نشان داده ایم:



نمودار 28 - نمودار heatmap همبستگی ویژگی ها در دیتافریم athletes\_df



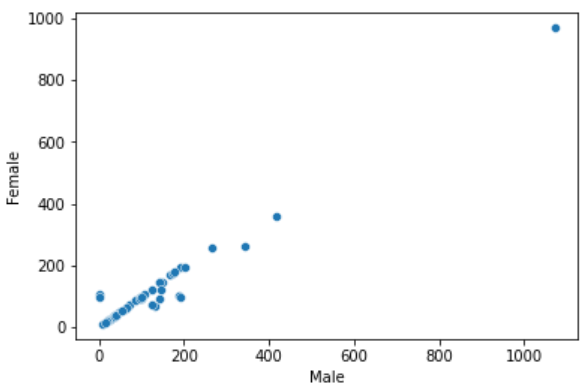
نمودار 29 - نمودار heatmap همبستگی ویژگی ها در دیتافریم coaches \_df



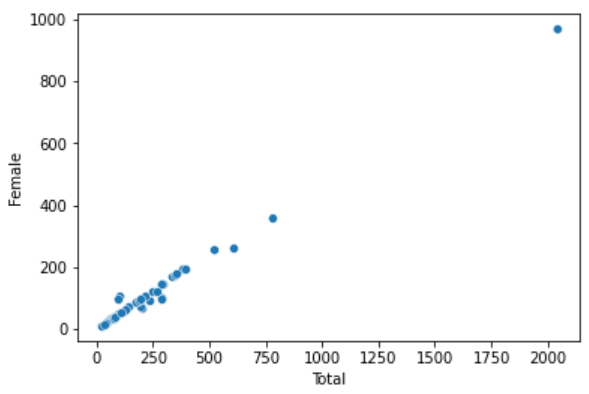
نمودار 30 - نمودار heatmap همبستگی ویژگی ها در دیتافریم entries\_gender\_df

## 2.8-نمایش نمودار scatter برای ستون های همبسته

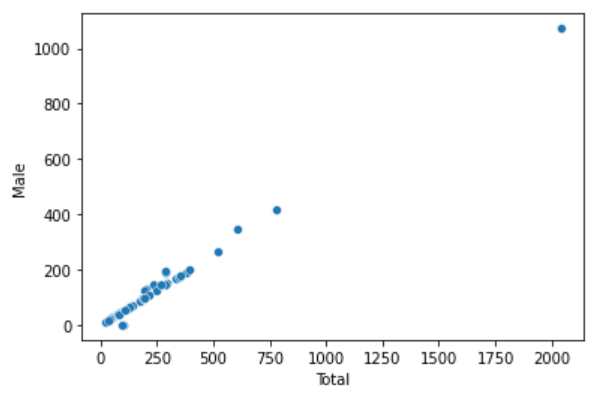
ما برای ستون هایی که مقدار همبستگی آنها بیشتر از 0.95 بوده است نمودار scatter را رسم کردیم که به صورت زیر است:



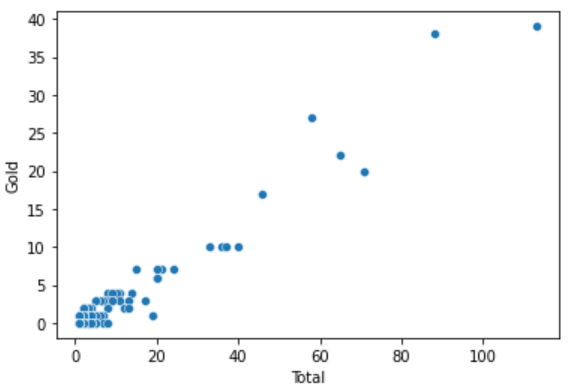
نمودار 31 - نمودار scatter دو ستون Male و Female از دیتافریم entries\_gender\_df



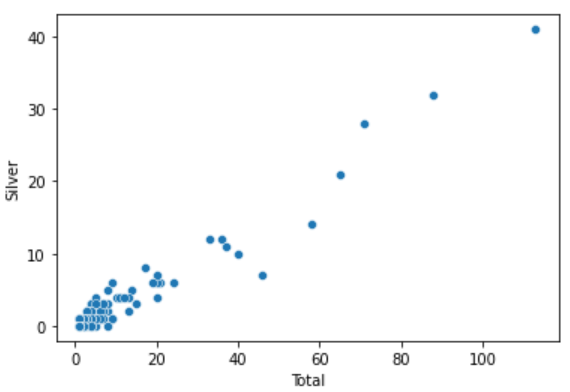
نمودار 32 - نمودار scatter برای دو ستون Total و Female از دیتافریم entries\_gender\_df



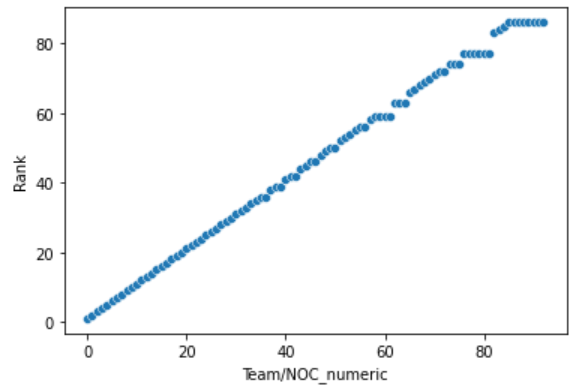
نمودار 33 - نمودار scatter برای دو ستون Total و Male از دیتافریم entries\_gender\_df



نمودار 34 - نمودار scatter برای دو ستون Total و Gold از دیتافریم medals\_df



نمودار 35 - نمودارscatter برای دو ستون Total و Silver از دیتافریم medals\_df



نمودار 36 - نمودار scatter برای دو ستون Team/NOC\_numeric و Rank از دیتافریم medals\_df

# بخش سوم: پیش پردازش داده ها

## 3.1-پاکسازی داده ها

در بخش قبل دیدیم که تنها ستونی که داده از دست رفته دارد، ستون Event از دیتافریم coaches\_df می باشد. حال ما برای اینکه مقادیر از دست رفته را پر کنیم از الگوریتم knn استفاده کردیم و داده ها را براساس ستون Events به دو دسته train و test تقسیم کردیم که در دسته train داده هایی هستند که ستون Event آن ها مقدار دارد و در دسته test داده هایی هستند که ستون Event آنها مقدار ندارد و میخواهیم مقدار آن را پیش بینی کنیم. ما نزدیکترین داده به داده ی مدنظر را در نظر گرفتیم و مقدار داده ی از دست رفته را برابر با مقدار آن ستون در داده ی نزدیک به آن قرار دادیم.

در بخش بعد برای حذف داده های پرت از دیتافریم ها، داده هایی که از q1 - 1.5\* IQR کمتر و از q3 + 1.5 \* IQR بیشتر بودند را به عنوان داده پرت در نظر گرفتیم و آن ها را حذف کردیم.

## 3.2- افزونگی در سطح ویژگی و رکورد

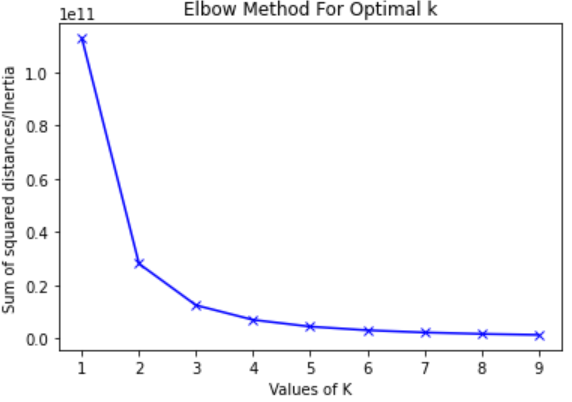
ما برای حذف ستون هایی که افزونگی دارند دو چیز را در نظر گرفتیم:

1. برای ستون های رشته ای ما در ابتدای کار ستون های عددی را ایجاد کردیم که نگاشتی از ستون های رشته ای به اعداد هستند، پس ما اکنون نیازی به ستون های رشته ای نداریم و نگاشت عددی آنها را داریم فلذا تمام ستون های رشته ای را حذف کردیم.
2. از بین دو ستون همبسته که مقدار همبستگی بین آنها بیشتر از 0.95 بود، نیز یک ستون را حذف و دیگری را حفظ کردیم.

برای حذف افزونگی در سطح رکورد نیز ابتدا بررسی کردیم کدام دیتافریم ها داده های تکراری دارند، سپس با استفاده از متد drop\_duplicates از کتابخانه pandas، از بین ردیف های تکراری، اولین ردیف تکراری را حفظ کردیم و باقی ردیف های تکراری را حذف کردیم.

## 3.3- کاهش در سطح رکورد

برای کاهش در سطح رکورد با بررسی دیتافریم ها متوجه شدیم که همه دیتافریم ها به جز دیتافریم athletes\_df تعداد داده خیلی کمی دارند. در دیتافریم athletes\_df تعداد داده 11029 تا می باشد که برای کاهش تعداد از روش نمونه برداری از خوشه ها استفاده کردیم. به این ترتیب که ابتدا بهترین تعداد خوشه ها برای خوشه بندی داده های athletes\_df به کمک روش kmeans را پیدا کردیم(که تعداد کلاستر مناسب آن با توجه به نمودار زیر 3 می باشد).



نمودار 37- نمودار روش Elbow که در آن مجموع مربعات فاصله ها با توجه به تعداد خوشه ها نشان داده شده.

پس از مشخص شدن تعداد مناسب خوشه ها، با روش kmeans داده های athletes\_df را خوشه بندی کردیم و پس از آن به تعداد 2000 نمونه از هر خوشه به صورت رندوم برداشتیم.

## 3.4- نرمال سازی داده ها

برای بخش نرمال سازی داده ها ما سه روش MinMaxScaler و Normalizer از کتابخانه sklearn و zscore از کتابخانه stats استفاده کردیم. اما به این دلیل که zscore نحوه توزیع شدگی داده ها را تغییر نمیدهد و چولگی داده ها را حفظ میکند و فقط بازه ی داده ها را استاندارد سازی میکند، بنابراین تصمیم گرفتیم که از این نرمال سازی استفاده کنیم.

## 3**.5- کاهش ابعاد داده ها**

ما برای کاهش ابعاد داده ها از آنجایی که دیتافریم ها 3 یا 4 یا 6 تا بعد داشتند لذا از روش PCA استفاده کردیم و داده ها را به فضایی بردیم که همه دیتافریم ها 2 بعد دارند و درنتیجه ابعاد داده ها را کاهش دادیم.

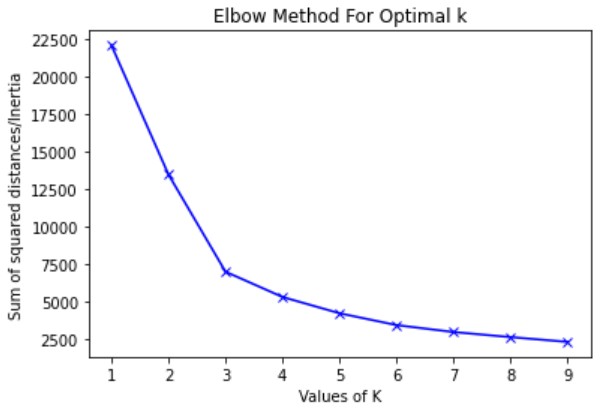
# بخش چهارم: مدل سازی و داده کاوی

## 4.1- پیاده سازی الگوریتم های بی نظارت

برای این بخش ما از داده های دیتافریم athletes\_df استفاده کرده ایم. به این صورت که NOC\_numeric (نگاشت عددی ستون NOC) و Discipline\_numeric (نگاشت عددی ستون Discipline) را برای هر ورزشکار داریم و میخواهیم بر اساس مقدار این دو ویژگی برای ورزشکاران آنها را خوشه بندی کنیم.

### 4.1.1- الگوریتم kmeans

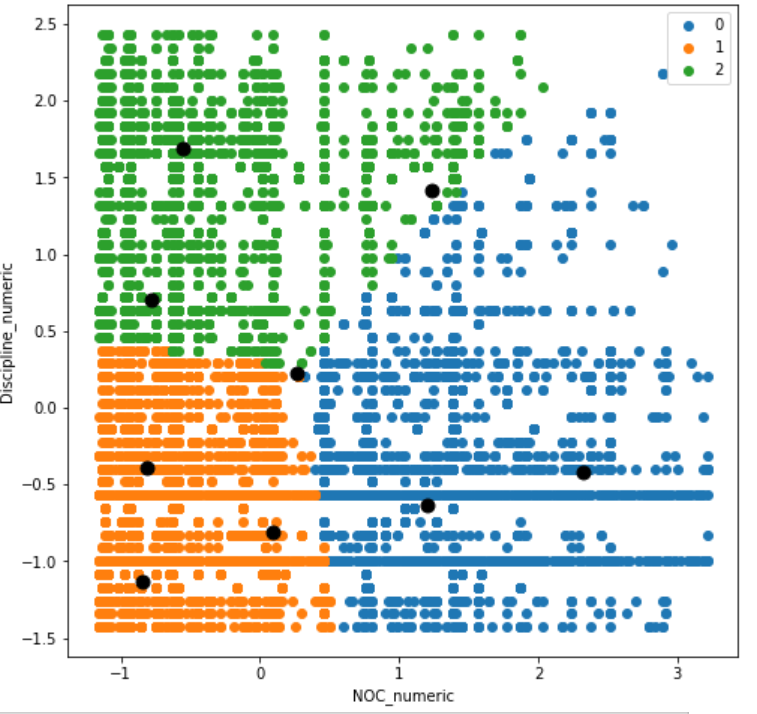
برای kmeans ابتدا باید تعداد بهینه خوشه ها را بدست آوریم که برای اینکار از روش Elbow و خطای SSE استفاده میکنیم. نمودار خطاها برای تعداد مختلف خوشه های kmeans به صورت زیر می باشد:



نمودار 38 - نمودار خطای مدل kmeans در تعداد مختلف خوشه ها

همانطور که میبینیم مقدار بهینه k در تعداد خوشه 3 رخ داده است. بنابراین تعداد خوشه را برابر با این مقدار میگذاریم و داده های نرمال شده را با آن خوشه بندی میکنیم.

بعد از خوشه بندی داده ها، نمودار خوشه های داده ها به صورت زیر می باشد:

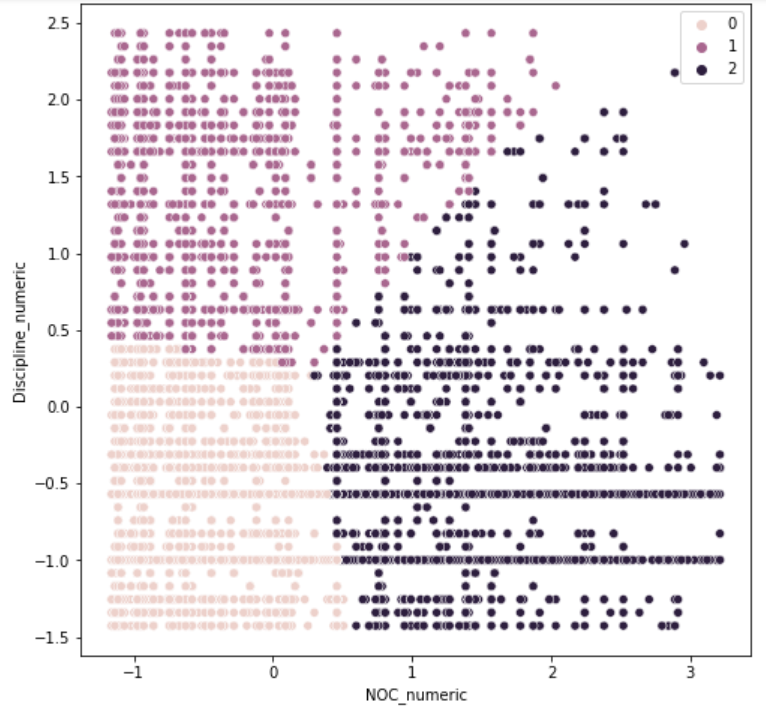


نمودار 39 - نمودار داده های خوشه بندی شده با مدل kmeans با k = 3

### 4.1.2- الگوریتم DBScan

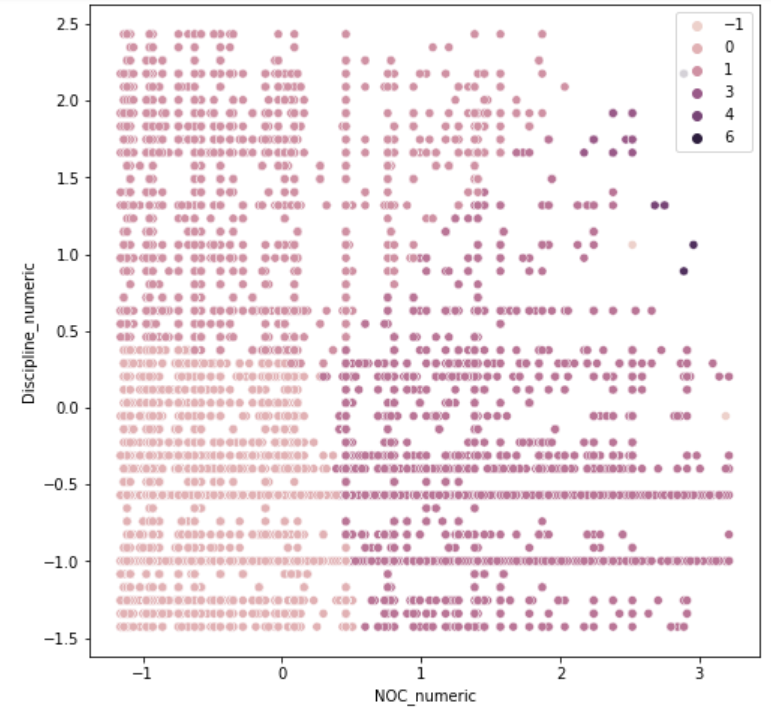
برای الگوریتم DBScan ابتدا با min\_samples=2 و eps=2 و eps=3 امتحان کردیم که خب تمام داده ها در یک خوشه قرار گرفتند. البته این دور از انتظار نبود چون داده ها نرمال هستند فلذا range مقادیر محدود است.

سپس با min\_samples=2 و eps=1 و eps=0.5 امتحان کردیم که برای هر دو خوشه ها به صورت زیر ایجاد شد:



نمودار 40 - نمودار خوشه بندی الگوریتم DBScan با min\_samples=2 و eps=1 و eps=0.5

برای eps=0.25 نیز تعداد خوشه ها برابر 6 تا شد و به صورت زیر است که همانطور که میبینیم با این مقدار الگوریتم کارا نیست:



نمودار 41 - نمودار خوشه بندی الگوریتم DBScan با min\_samples=2 و eps=0.25

پس با مقادیر مختلفی که امتحان کردیم متوجه شدیم که با eps=1 و eps=0.5 الگوریتم کارا ترین خوشه بندی را دارد.

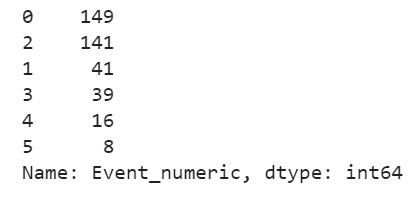
### 4.1.3- مقایسه دو الگوریتم Kmeans و DBScan

با مشاهده نمودار 39 و نمودار 40 متوجه میشویم که خوشه بندی kmeans و dbscan برای این داده ها مشابه هم است و در نتیجه از لحاظ میزان کارایی هر دو مثل هم هستند.

## 4.2- پیاده سازی الگوریتم های با نظارت

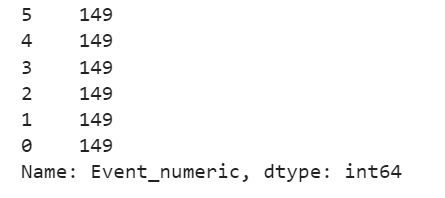
نکته: در الگوریتم های با نظارت ما از دیتافریم coaches\_df استفاده میکنیم. به این صورت که ستون های NOC\_numeric (نگاشت شده نام کشور به عدد) و Discipline\_numeric (نگاشت شده رشته ورزشی به عدد) داده ها را تشکیل دهند و Event\_numeric (نگاشت نوع رویداد ورزشی به عدد) برچسب داده ها می باشد.

چالش: ابتدا برای نامتوازن بودن داده های کلاس ها چالش داشتیم که مقدار کلاس ها به صورت زیر بود:



تصویر 16- تعداد داده ها در هر کلاس قبل از متوازن سازی کلاس ها

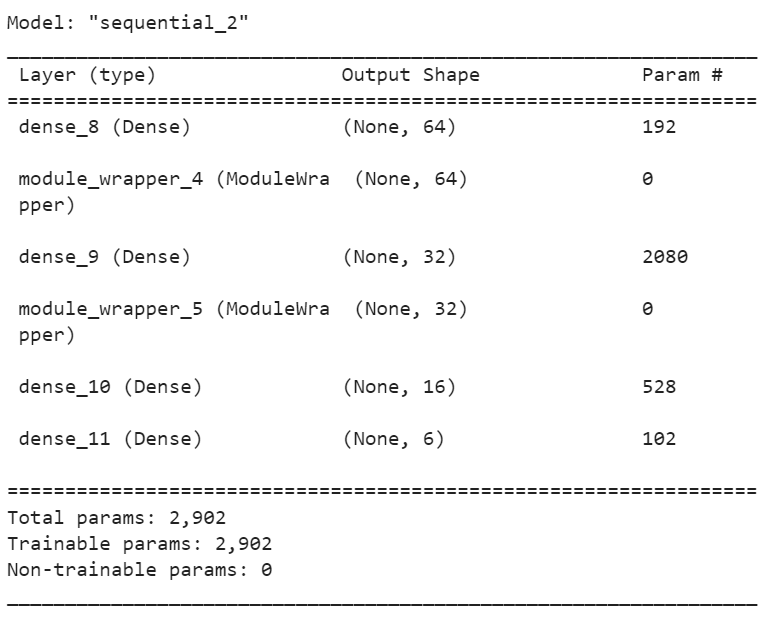
پس ما با استفاده از RandomOverSampler در کتابخانه imblearn داده های کلاس های کمتر را over sample کردیم و در نهایت پس از متوازن سازی تعداد داده ها به صورت زیر شد:



تصویر 17 - تعداد داده ها در هر کلاس بعد از متوازن سازی کلاس ها

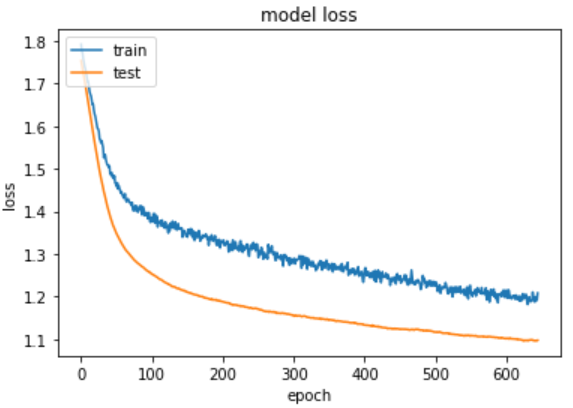
### 4.2.1- مدل سازی شبکه عصبی

معماری شبکه ای که در ابتدا در نظر گرفتم به صورت زیر می باشد:



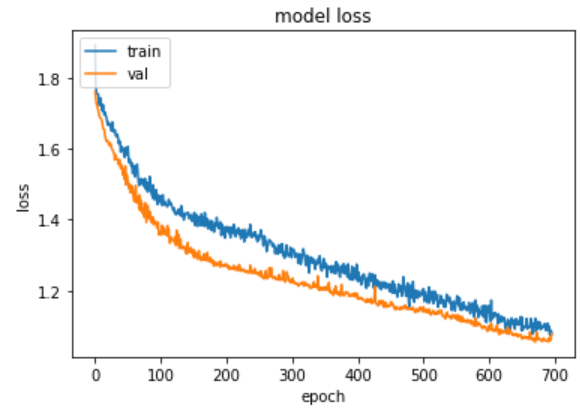
تصویر 18 - معماری ابتدایی شبکه عصبی

با در نظر گرفتن optimizer = Adam و learning\_rate = 0.0001 و loss=’categorical\_crossentropy’ و در نظر گرفتن early\_stopping با paitent = 20 نمودار loss مربوط به داده های train و validation به صورت زیر می باشد:



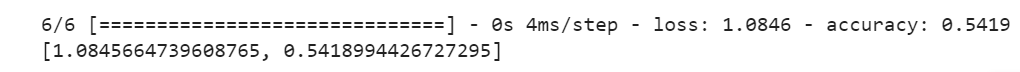
نمودار 42- نمودار loss شبکه عصبی ابتدایی

بعد از تغییرات زیاد در تعداد لایه های شبکه، تعداد نورون ها در لایه های Dense و حذف و اضافه کردن لایه های Dropout مختلف و تغییر optimizer و سایز batch و ... در نهایت به نمودار Loss زیر رسیدم:

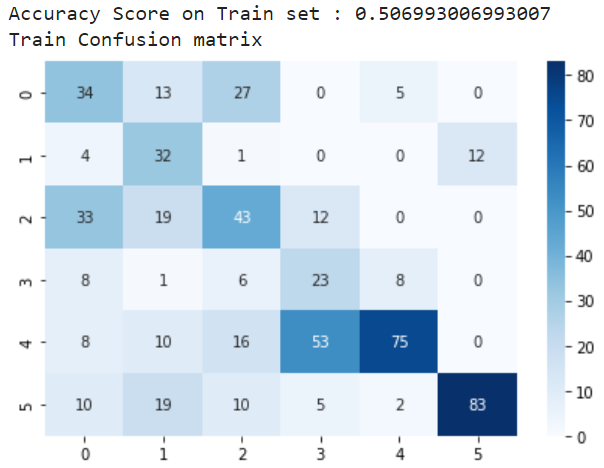


نمودار 43 - نمودار loss شبکه عصبی نهایی

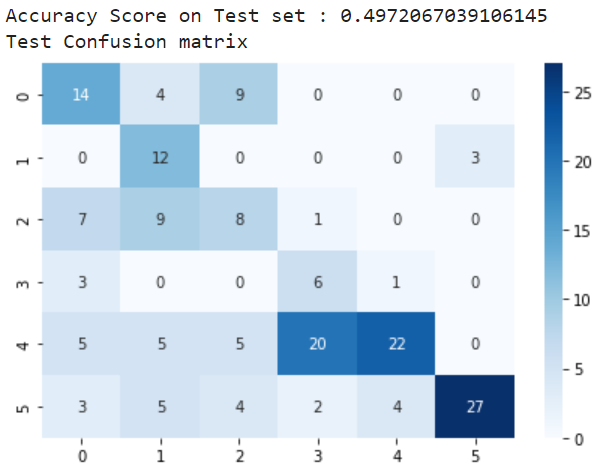
و در نهایت روی داده های test به دقت زیر رسیدم:



همچنین ماتریس confusion برای داده های تست و داده های آموزش به صورت زیر می باشد:



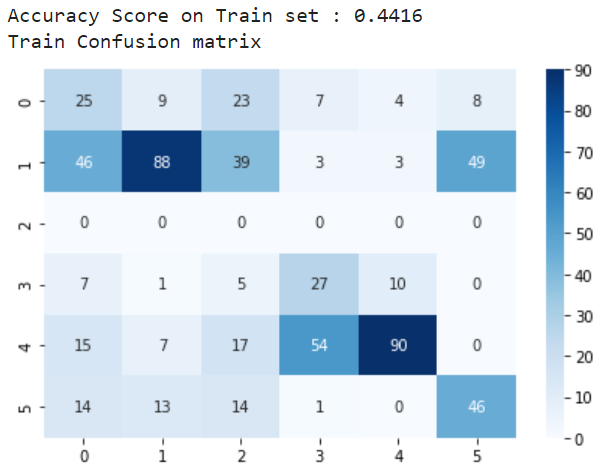
تصویر 19 - ماتریس confusion برای داده های آموزش



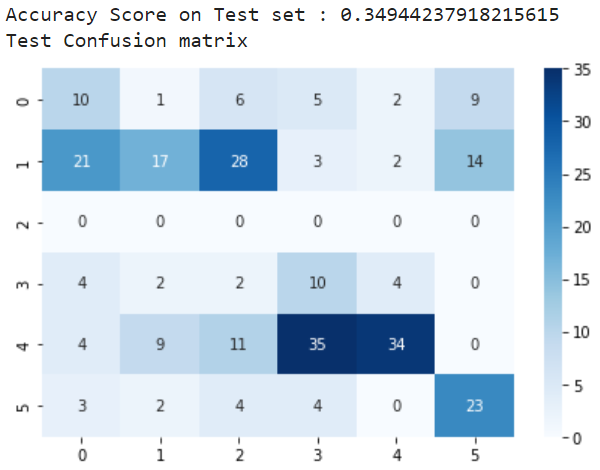
تصویر 20 - ماتریس confusion برای داده های تست

### 4.2.2**- الگوریتم طبقه بندی SVM**

تلاش 1: از مدل svm با کرنل linear استفاده کردم و دقت را اندازه گرفتم:

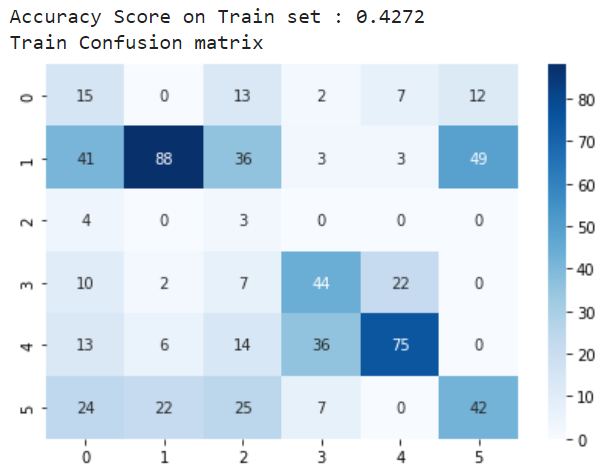


تصویر 21 - ماتریس confusion برای داده های آموزش برای مدل svm با کرنل خطی

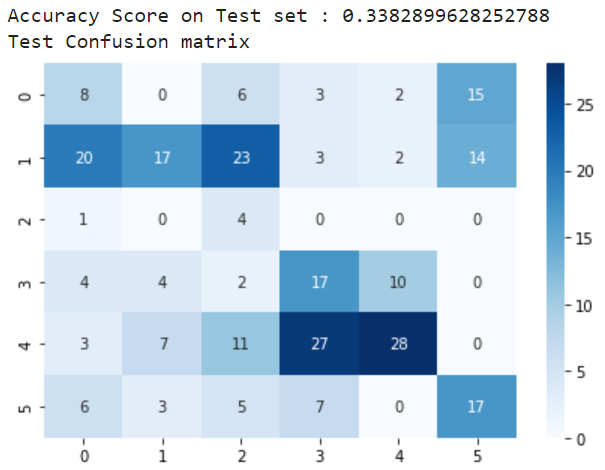


تصویر 22 - ماتریس confusion برای داده های تست برای مدل svm با کرنل خطی

تلاش 2: از مدل svm با کرنل rbf استفاده کردم و دقت را اندازه گرفتم:

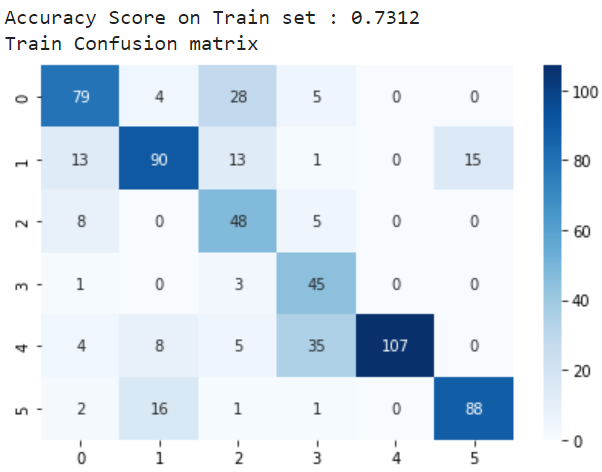


تصویر 23 – ماتریس confusion برای داده های آموزش برای مدل svm با کرنل rbf

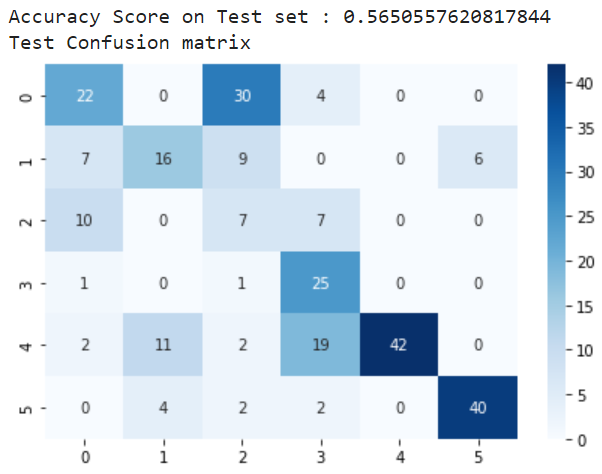


تصویر 24 - ماتریس confusion برای داده های تست برای مدل svm با کرنل rbf

تلاش3: در 2 حالت قبل مقدار پارامتر gamma در حالت پیش فرض و با مقدار scale بود. با توجه به اینکه دقت داده های آموزش در حالتی که با کرنل rbf بود خیلی بیشتر از حالت خطی بود، پس به نظرم رسید که باید مقدار gamma را امتحان کنم تا ببینم با تغییرات gamma چه خروجی خواهم داشت. پس مقدار gamma را از حالت scale به حالت auto تغییر دادم و به دقت زیر رسیدم:



تصویر 25 – ماتریس confusion برای داده های آموزش برای مدل svm با کرنل rbf و مقدار gamma = auto



تصویر 26 - ماتریس confusion برای داده های تست برای مدل svm با کرنل rbf و مقدار gamma = auto

با این تغییر به نظر می آید که دقت مدل به دقت بالایی رسیده. همچنین با روش ارزیابی k-fold به دقت زیر رسیدم:

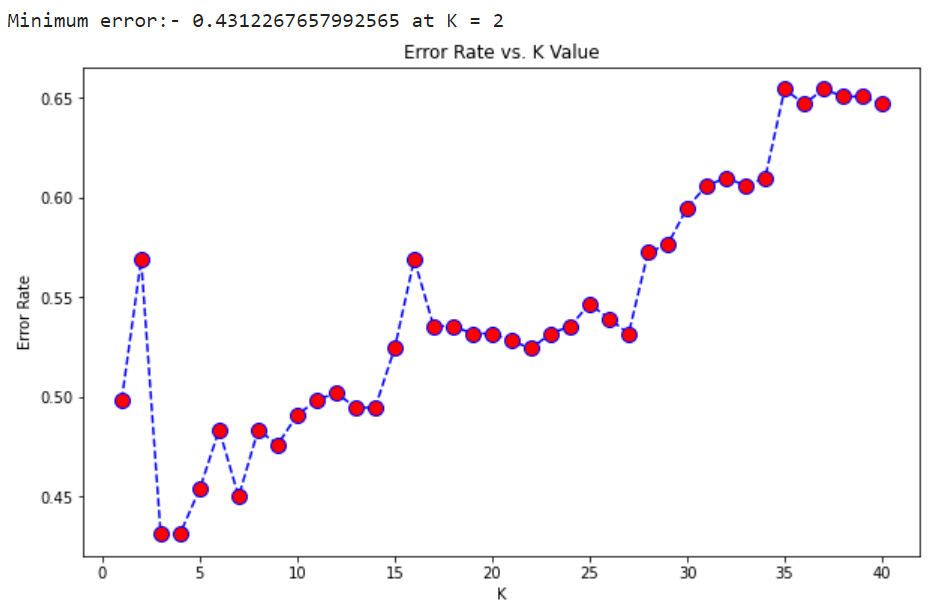


تلاش 4: مقدار gamma را اعداد 3، 10 ، 100، 200، 300، 500 و 1000 قرار دادم اما همان دقت های بالا با چند صدم تفاوت بود.

### 4.2.3- الگوریتم طبقه بندی KNN

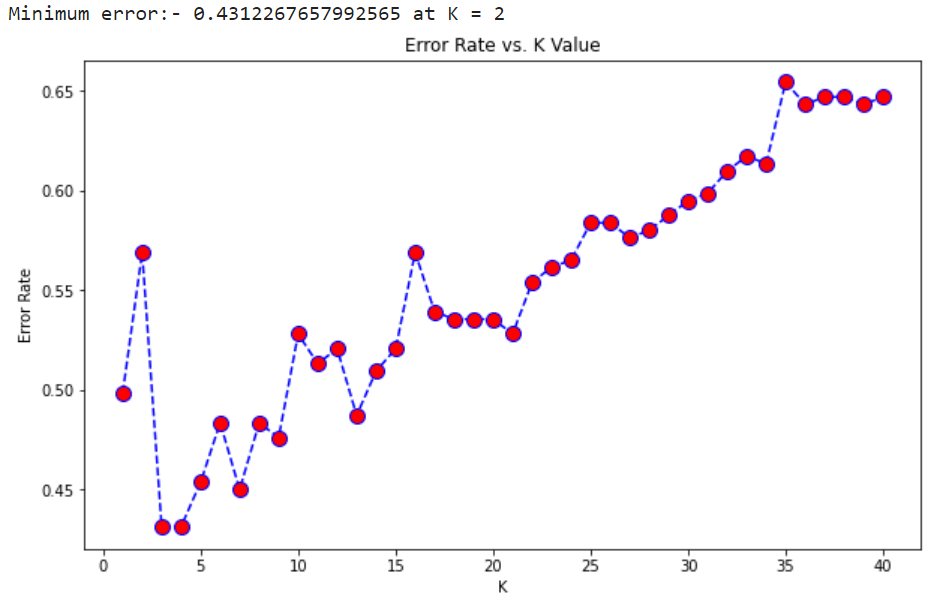
برای پیدا کردن مقدار بهینه تعداد همسایه ها، برای تعداد همسایه از 1 تا 40 مقدار خطا را محاسبه کردیم و نمودار خطا ها را رسم کردیم که به صورت زیر می باشد:

برای metric = ‘minkowski’ و p=1 (یعنی فاصله منهتن):



نمودار 44 - نمودار خطای مدل knn با متریک فاصله منهتن

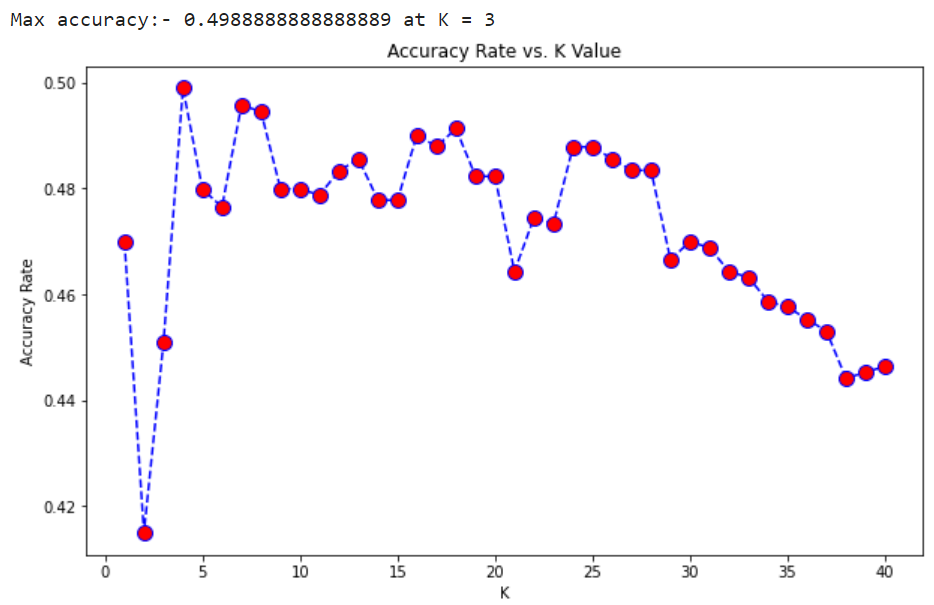
برای metric = ‘minkowski’ و p=2 (یعنی فاصله اقلیدسی):



نمودار 45 - نمودار خطای مدل knn با متریک فاصله اقلیدسی

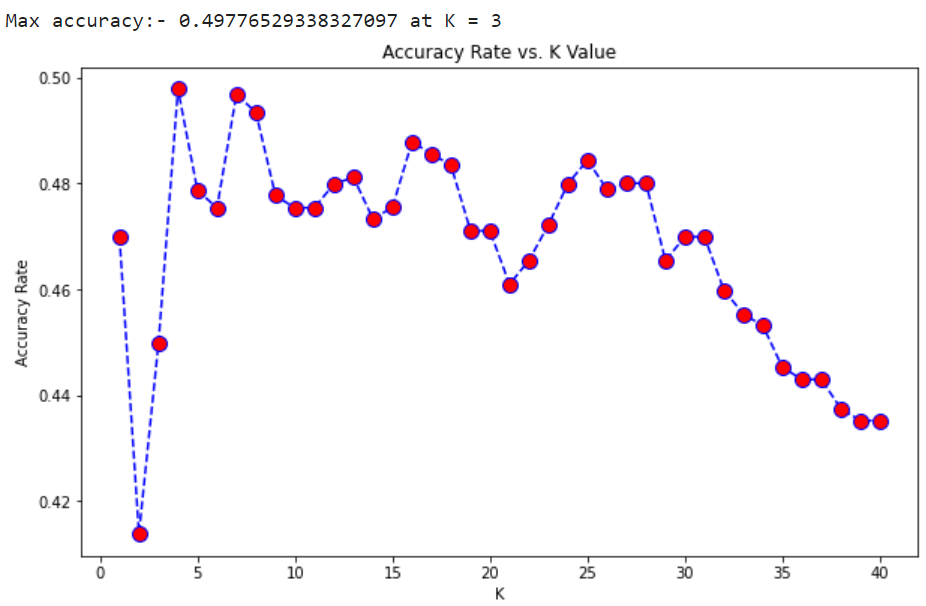
پس از این ما مقدار دقت را براساس روش k-fold برای تعداد همسایه 1 تا 40 محاسبه کردیم که نمودار آن به صورت زیر است:

برای metric = ‘minkowski’ و p=1 (یعنی فاصله منهتن):



نمودار 46 - نمودار دقت مدل knn با متریک فاصله منهتن

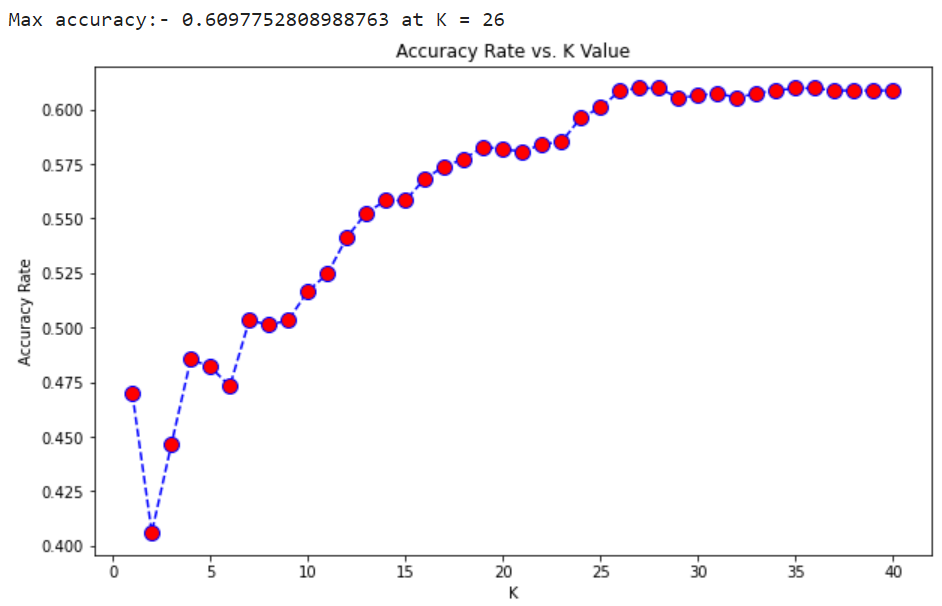
برای metric = ‘minkowski’ و p=2 (یعنی فاصله اقلیدسی):



نمودار 47 - نمودار دقت مدل knn با متریک فاصله اقلیدسی

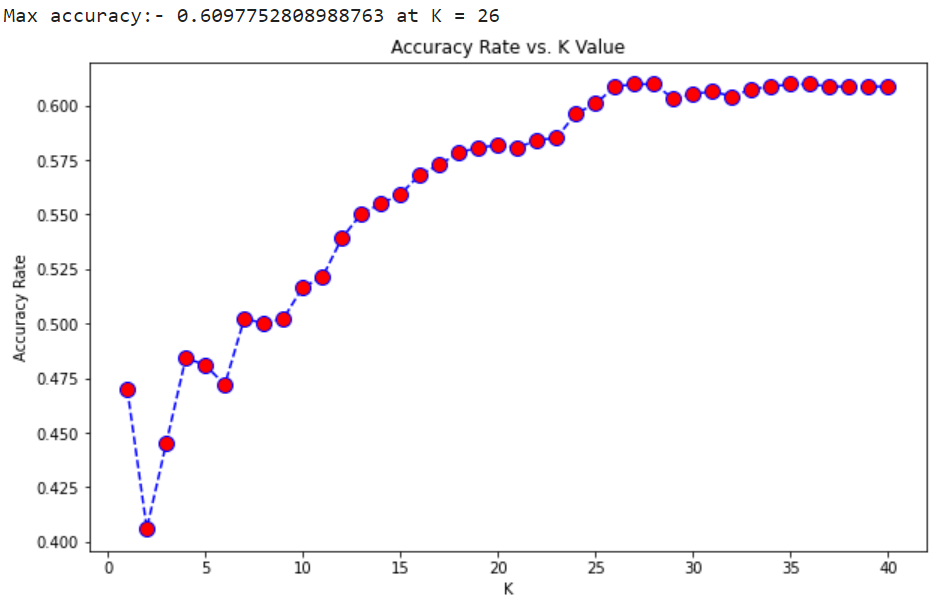
تا اینجا که بررسی کردیم وزن همسایه ها یکسان بود. حال وزن همسایه ها براساس فاصله تغییر میکند که با این تغییر ما دوباره دقت را با k-fold بدست آوردیم:

برای metric = ‘minkowski’ و p=1 (یعنی فاصله منهتن):



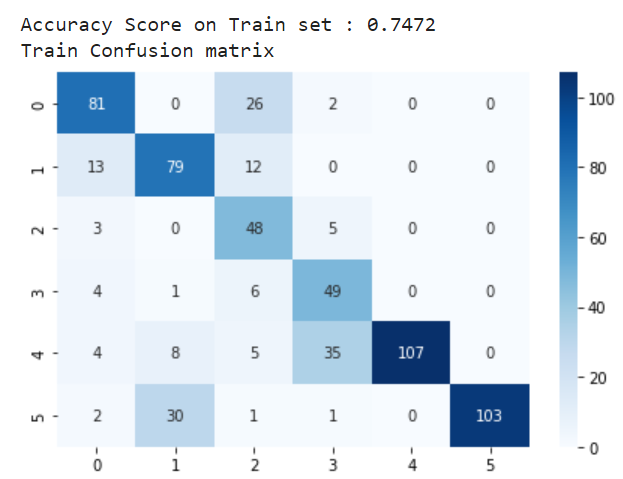
نمودار 48 - نمودار دقت مدل knn با متریک فاصله منهتن و با وزن دهی به همسایه ها

برای metric = ‘minkowski’ و p=2 (یعنی فاصله اقلیدسی):

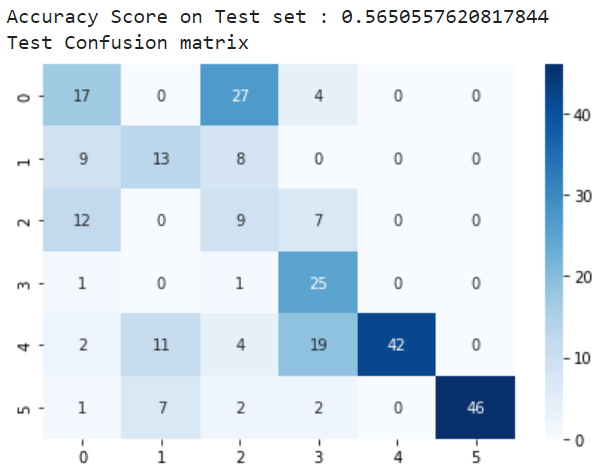


نمودار 49 - نمودار دقت مدل knn با متریک فاصله اقلیدسی و وزن دهی به همسایه ها

در نهایت با بررسی ها تصمیم گرفتیم که با تعداد همسایه 15 و متریک فاصله اقلیدسی مدل knn را آموزش دهیم که در نهایت این مدل روی داده های آموزش و تست دقت های زیر را دارد:



تصویر 27- ماتریس confusion برای داده های آموزش برای مدل نهایی knn



تصویر 28 - ماتریس confusion برای داده های تست برای مدل نهایی knn

و با روش ارزیابی k-fold دقت به صورت زیر می باشد:

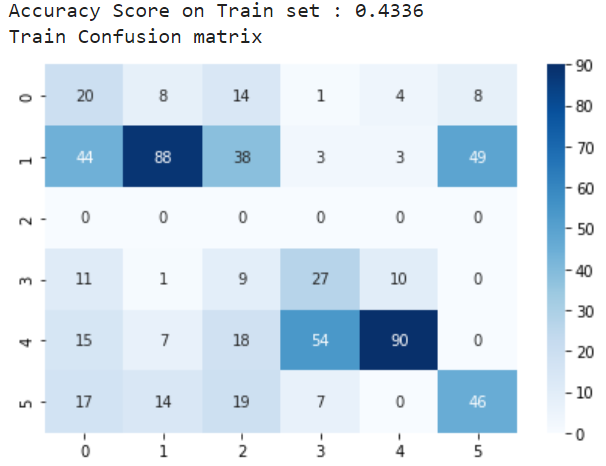


### 4.2.4- الگوریتم طبقه بندی bayes

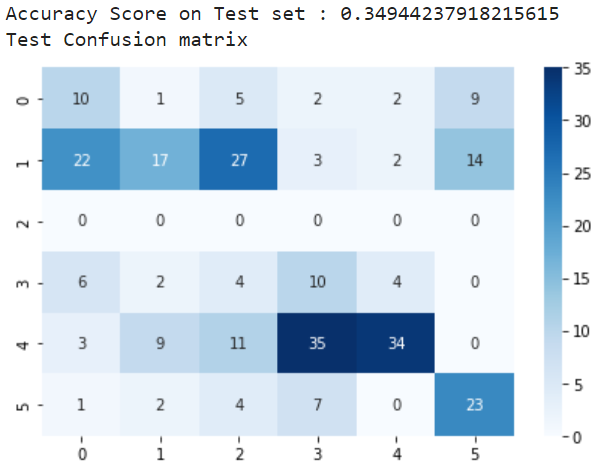
در این بخش ابتدا با استفاده از متد correlated\_attrs که در بخش های قبل پیاده شده چک کردیم که آیا در دیتافریم coaches\_for\_model (همان دیتافریم مورد استفاده در بخش الگوریتم های با نظارت) ویژگی های همبسته وجود دارد یا نه و از آنجایی که ویژگی همبسته ای وجود نداشت در نتیجه میتوانیم از naive Bayesian classifier استفاده کنیم.

بنابراین ما این مدل را ایجاد کردیم و روی داده ها آن را train کردیم.

ماتریس confusion برای مدل bayes برای داده های تست و آموزش به صورت زیر می باشد:



تصویر 29 - ماتریس confusion برای داده های آموزش برای مدل bayes



تصویر 30 - ماتریس confusion برای داده های تست برای مدل bayes

پس از آن با روش ارزیابی k-fold دقت آن را محاسبه کردیم که نتیجه به صورت زیر شد:



### 4.2.5- الگوریتم طبقه بندی decision tree

تغییرات پارامتر ها:

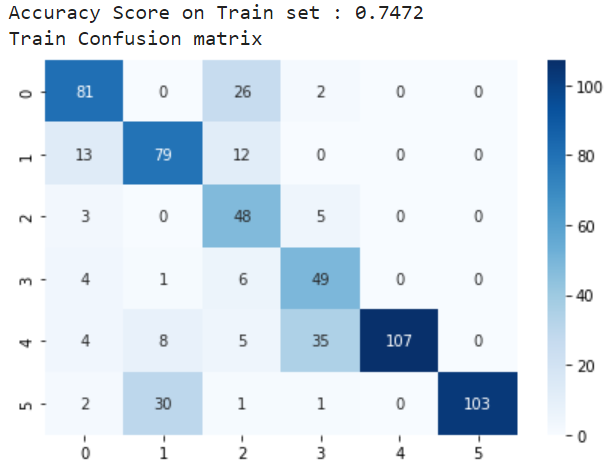
1. پارامتر criterion: این پارامتر متد انتخاب بهترین attribute برای جدا کردن داده ها را نشان میدهد که دو مقدار gini و entropy را دارد که gini برای روش gini impurity و entropy برای روش information gain می باشد. با مقدار دادن هر دو این روش ها درصد دقت مدل با روش k-fold یکسان و برابر با مقدار زیر شد:



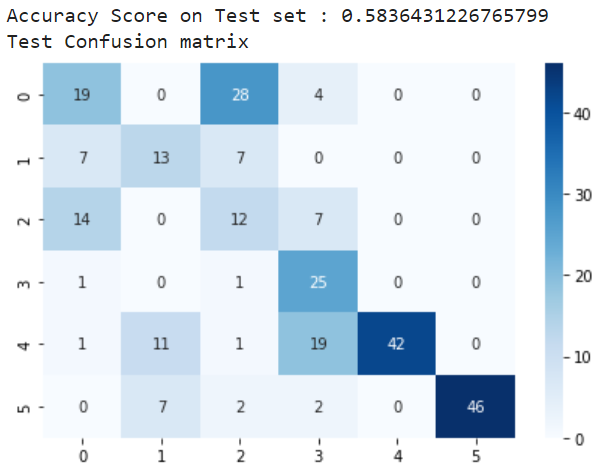
1. پارامتر splitter: این پارامتر با دو مقدار best و random استراتژی مورد استفاده برای انتخاب تقسیم در هر گره را مشخص میکند. که مقدار best بهترین تقسیم رو برای هر گره انتخاب میکند و random بهترین انتخاب تصادفی را برای تقسیم در هر گره در نظر میگیرد. با مقدار best درصد دقت با k-fold همچنان 0.62 می باشد، اما با random مقدار دقت یک صدم (یعنی 1 درصد) افزایش و به مقدار 0.63 میرسد.
2. Max\_depth: بیشترین عمق مجاز برای درخت را تعیین میکند که مقدار پیش فرض ندارد و در نتیجه عمق درخت هر مقدار میتواند باشد. من مقدار عمق را از 5 تا 10 تغییر دادم و هرچه عمق را بیشتر میکردم دقت افزایش پیدا میکرد.
3. ccp\_alpha: هرس پیچیدگی حداقل هزینه، به صورت بازگشتی گره با ضعیفترین پیوند را انتخاب و هرس میکند. ضعیفترین پیوند با آلفای موثر مشخص میشود، جایی که گره های با آلفای کمتر از ccp\_alpha هرس میشوند. مقدار پیش فرض این متغیر صفر است به این معنی که به صورت پیش فرض درخت هرس نمیشود.

#### 4.2.5.1- درخت تصمیم بدون هرس

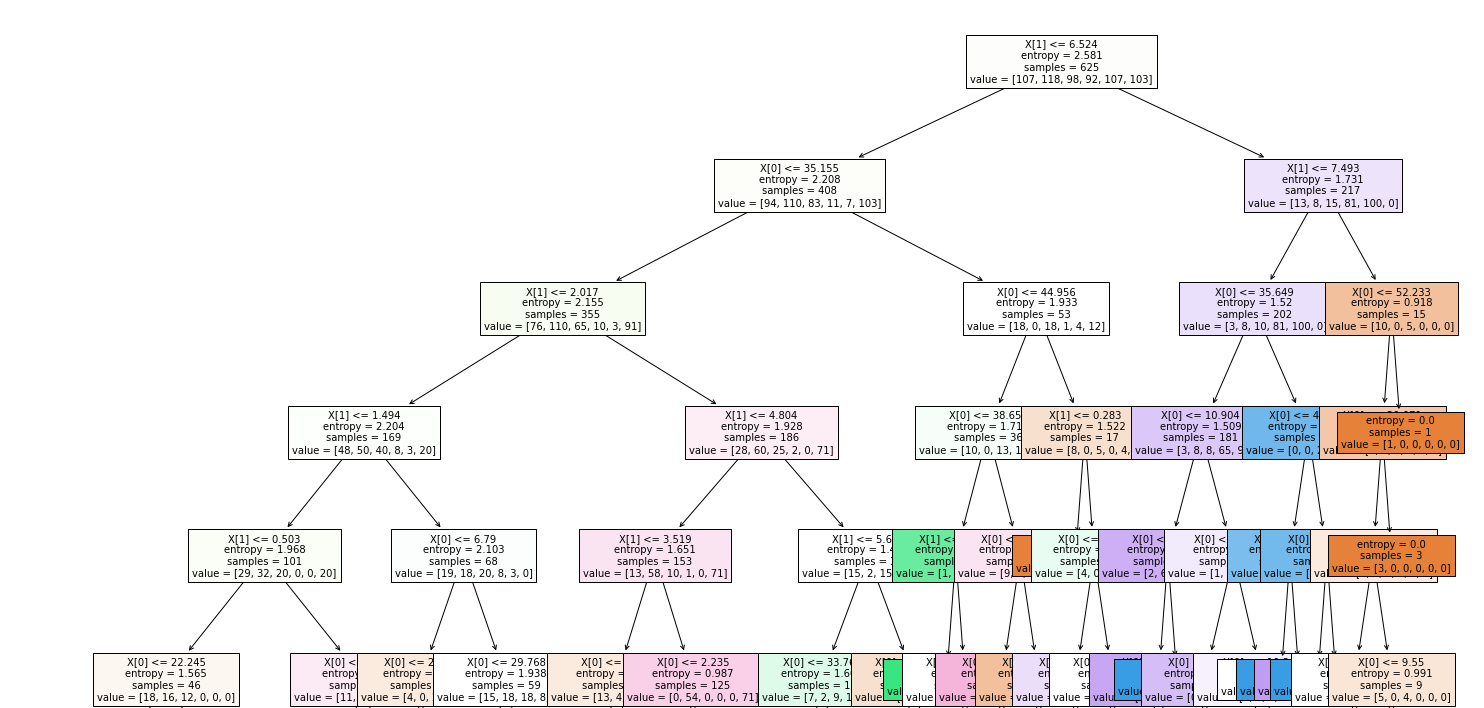
ماتریس confusion برای داده های آموزش و تست برای مدل درخت تصمیم به صورت زیر می باشد:



تصویر 31 - ماتریس confusion برای داده های آموزش برای مدل درخت تصمیم با criterion = "entropy" و splitter = "random"



تصویر 32 - ماتریس confusion برای داده های آموزش برای مدل درخت تصمیم با criterion = "entropy" و splitter = "random"



تصویر 33 - شمایل درخت تصمیم با criterion = "entropy" و splitter = "random"

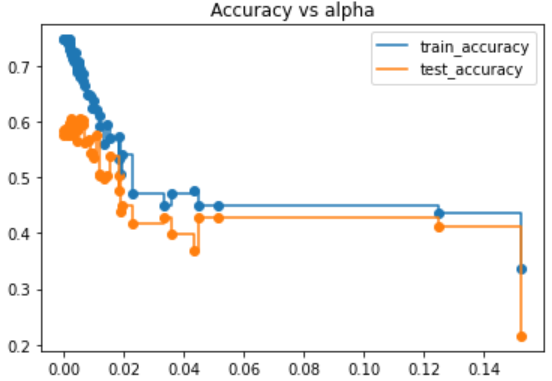
همچنین دقت مدل با روش k-fold به صورت زیر می باشد: 

#### 4.2.5.2- درخت تصمیم هرس شده

در بین پارامتر های معرفی شده برای درخت تصمیم پارامتری به نام ccp\_alpha معرفی کردیم که برای هرس درخت کاربرد دارد و مورد استفاده قرار میگیرد.

برای هرس درخت تصمیم ابتدا باید مقدار بهینه برای ccp\_alpha را پیدا کنیم. ابتدا برای ccp\_alpha های مختلف مدل را train می کنیم و تمام مدل ها را در آرایه قرار میدهیم. سپس برای همه مدل ها داده های آموزش و تست را پیش بینی میکنیم و سپس با استفاده از برچسب واقعی و برچسب پیش بینی شده دقت را برای هر مدل محاسبه میکنیم.

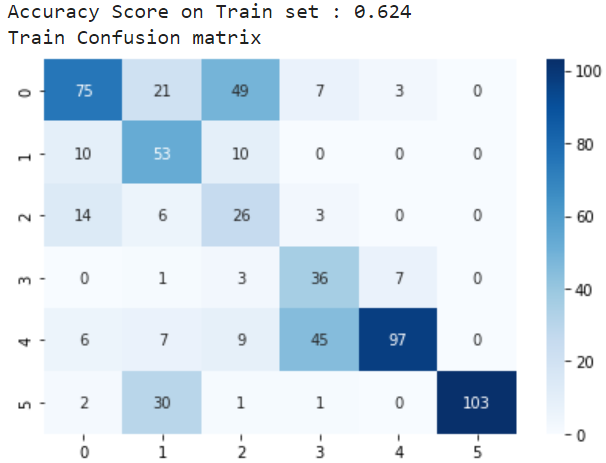
جدول زیر دقت داده های آموزش و تست براساس مقدار alpha می باشد:



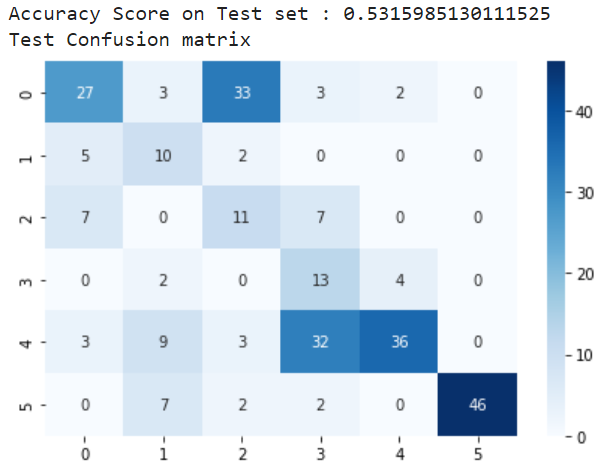
تصویر 34 - دقت داده های آموزش و تست برای مدل ها براساس مقدار alpha

بر اساس این جدول تصمیم گرفتیم تا مقدار ccp\_alpha را برابر با 0.01 قرار دهیم.

در نتیجه ماتریس confusion مدل هرس شده با criterion = “entropy” و splitter = “random” و ccp\_alpha = 0.01 به صورت زیر می باشد:



تصویر 35 - ماتریس confusion برای داده های آموزش برای مدل درخت تصمیم با ccp\_alpha = 0.01



تصویر 36 - ماتریس confusion برای داده های آموزش برای مدل درخت تصمیم با ccp\_alpha = 0.01

همچنین دقت مدل هرس شده با روش k-fold به صورت زیر می باشد:

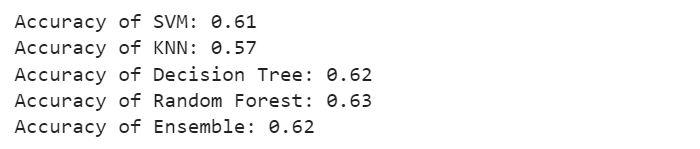


### 4.2.6- روش ensemble

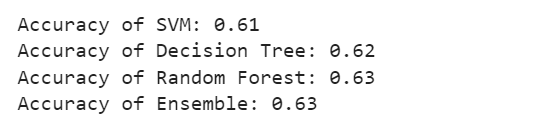
#### 4.2.6.1 – استفاده از VotingClassifier

در اینجا ما سعی کردیم بهترین الگوریتم های طبقه بندی که در اینجا دقت بالاتری داشتند را استفاده کنیم. بنابراین در این مرحله ما از مدل svm با دقت 61%، knn با دقت 57%، decision tree با دقت 62% استفاده کردیم. همچنین از دسته بند RandomForestClassifier نیز استفاده کردیم که دقت آن 63% می باشد.

تلاش1: وقتی از این 4 الگوریتم در ensemble استفاده کردیم، دقت هر یک به صورت زیر شد:



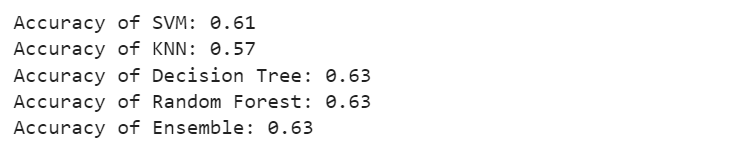
تلاش2: ما احتمال دادیم به خاطر اینکه دقت knn از همه پایین تر است در نتیجه دقت مدل ensemble را نیز کاهش دهد بنابراین در گام بعدی knn را از بین دسته بند ها حذف کردیم و دوباره دقت را محاسبه کردیم که به صورت زیر شد:



همانطور که میبینیم به دلیل نبودن knn در مدل جدید، دقت 1 درصد افزایش پیدا کرد.

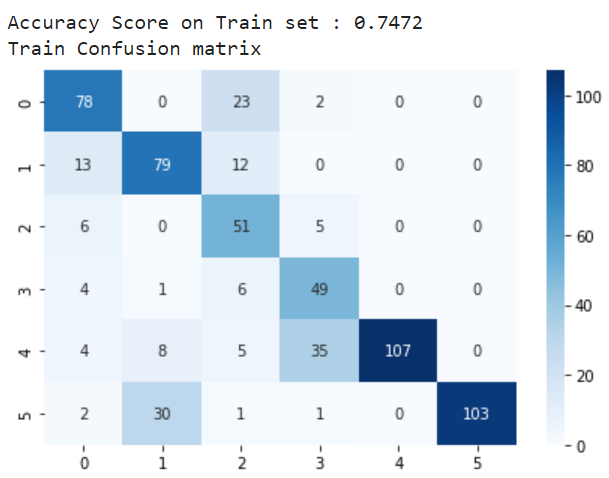
تلاش3: همچنین مقدار پارامتر voting را از hard (که از برچسب های پیش بینی شده برای رای گیری استفاده میکند) به soft (که برچسب کلاس را بر اساس ماکزیمم مجموع احتمالات پیش بینی شده، تعیین میکند) تغییر دادم، اما دقت همچنان 63% بود.

تلاش4: با استفاده از پارامتر weights به مدل ها وزن دادم. به این صورت که به svm وزن 1، به knn وزن 0.5، به درخت تصمیم وزن و random forest وزن 3 دادم، که این باعث شد دقت مدل ensemble، 63% شود. به عبارتی با وزن دهی به تمام مدل های در نظر گرفته شده از جمله knn توانستیم دقت را از حالت اول یک درصد افزایش دهیم:

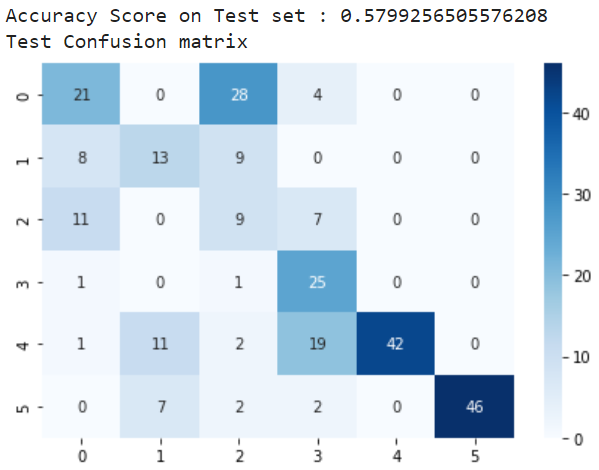


همچنین در VotingClassifier یک پارامتر n\_jobs می باشد که تعداد job هایی که به صورت موازی برای fit کردن داده ها روی مدل ها اجرا میشوند را تعیین می کند که به صورت پیش فرض مقدار آن None می باشد که به این معناست که یک job اجرا میشود و در حالتی که مقدار را -1 بگذاریم به این معناست که از تمام processor ها استفاده کند.

در نهایت ماتریس confusion برای داده های آموزش و آزمون برای مدل نهایی ensemble به صورت زیر می باشد:



تصویر 37- ماتریس confusion برای داده های آموزش برای مدل ensemble



تصویر 38 - ماتریس confusion برای داده های آزمون برای مدل ensemble

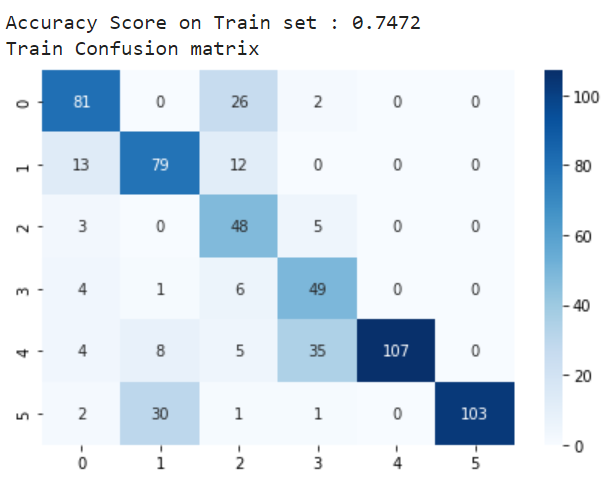
#### 4.2.6.2- استفاده از ExtraTreesClassifier

این کلاس تعدادی درخت تصادفی موسوم به درخت های اضافی را روی مجموعه داده آموزش fit میکند و از روش میانگین گیری برای بهبود دقت و کنترل overfit استفاده میکند.

برای این حالت من تعداد estimator ها را ( پارامتر n\_stimators) را 10 قرار دادم. همچنین پارامتر criterion که روش پیدا کردن بهترین ویژگی برای split را نشان میدهد برابر با entropy قرار دادم، که در نهایت با تعیین این پارامتر ها دقت مدل 63% شد.



ماتریس confusion برای داده های آموزش و آزمون برای مدل ensemble از نوع ExtraTrees به صورت زیر می باشد:



تصویر 39- ماتریس confusion برای داده های آموزش برای مدل ExtraTreesClassifier



تصویر 40- ماتریس confusion برای داده های آزمون برای مدل ExtraTreesClassifier

# بخش پنجم: الگوهای پرتکرار و قوانین انجمنی

## 5.1- الگوهای پرتکرار

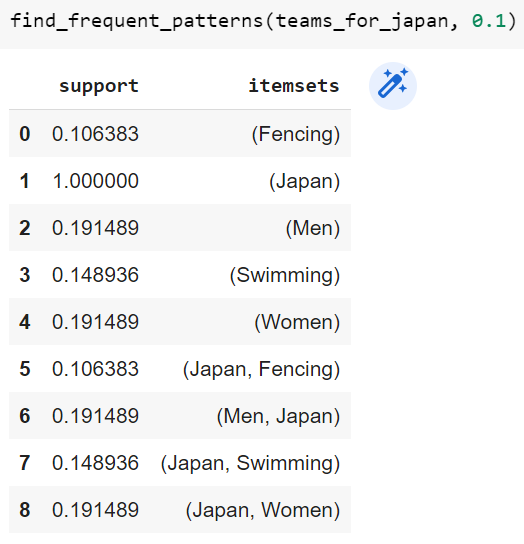
ما برای پیدا کردن الگوهای پرتکرار ابتدا ستون های عددی که نگاشتی از ستون های رشته ای هستند را حذف کرده ایم چون ستون های رشته ای متناظر با آنها را در دیتافریم ها داریم.

پس از آن با استفاده از TransactionEncoder داده های داخل دیتافریم را به فرم encode شده درآوردیم و پس از آن الگوریتم apriori را روی دیتافریم مربوطه اعمال کردیم.

نکته- با توجه به اینکه در دیتاست ما داده های ما حالت گزارش دارند و معمولا اکثر سطر ها unique هستند، بنابراین به طور کلی الگوی پرتکرار با support بالا در آن وجود ندارد، با این حال یکسری قوانین را بدست آوردیم که در ادامه آنها را مشاهده میکنیم.

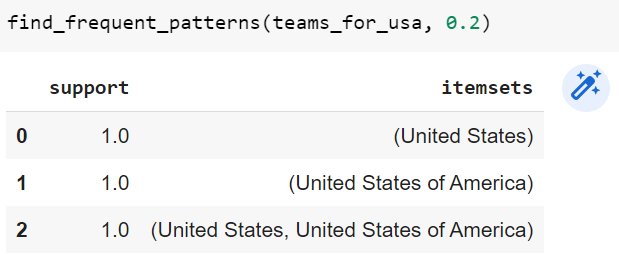
دیتافریم تیم ها که شامل نام تیم، رشته ورزشی، نام کشور و نوع رویداد است:

در این دیتافریم سطر هایی که مربوط به کشور ژاپن و آمریکا هستند از همه بیشترند. پس به عنوان مثال تصمیم گرفتیم برای تیم های کشور ژاپن و آمریکا الگوی پرتکرار پیدا کنیم:



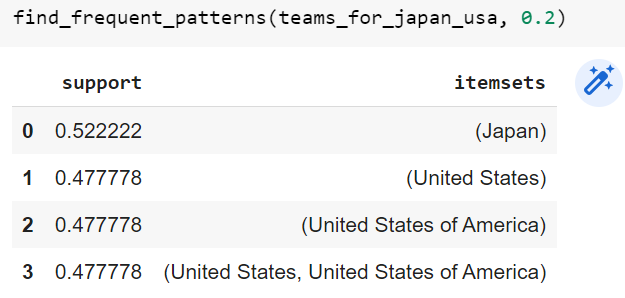
تصویر 41- الگوهای پرتکرار مربوط به تیم های کشور ژاپن با minsup = 0.1

همانطور که میبینیم تمام itemset ها به غیر از (Japan) مقدار support کمتر از 0.2 دارند که بسیار پایین است.



تصویر 42 - الگوهای پرتکرار مربوط به تیم های کشور آمریکا با minsup = 0.2

حال الگوی پرتکرار برای تیم های ژاپن و آمریکا (با هم) میخواهیم بدست آوریم:

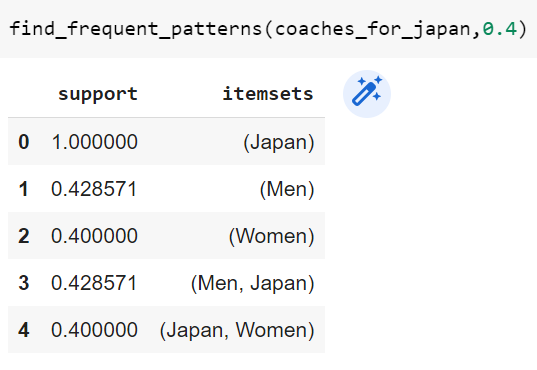


تصویر 43 - الگوی پرتکرار مربوط به تیم های کشورهای آمریکا و ژاپن با minsup = 0.2

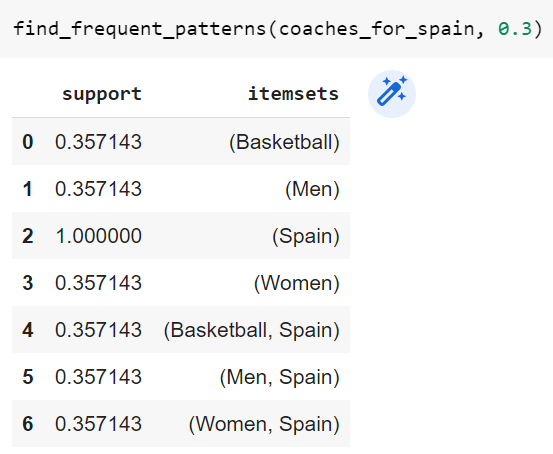
همانطور که میبینیم از ترکیب تیم های کشور های ژاپن و آمریکا الگوی پرتکرار جدیدی مشاهده نشد.

دیتافریم مربیان شامل نام مربی، نام کشور، رشته ورزشی و نوع رویداد می باشد:

ابتدا نام مربی را از دیتافریم حذف میکنیم چون مقدار آن منحصر به فرد است و کمکی در پیدا کردن الگوی پرتکرار به ما نمیکند. کشور های ژاپن و اسپانیا بیشترین تعداد مربی را دارند در نتیجه تصمیم گرفتیم تا برای این دو کشور الگوهای پرتکرار را پیدا کنیم:

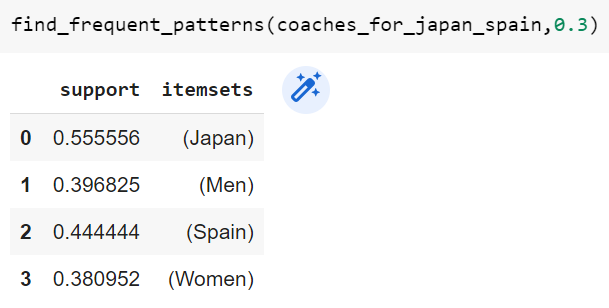


تصویر 44 - الگوهای پرتکرار مربیان کشور ژاپن با min\_sup = 0.4



تصویر 45 - الگوهای پرتکرار مربیان کشور اسپانیا با min\_sup = 0.4

حال میخواهیم از الحاق داده های مربیان ژاپن و اسپانیا الگوهای پرتکرار استخراج کنیم:



تصویر 46 - الگوهای پرتکرار مربیان کشور های ژاپن و اسپانیا با minsup = 0.3

همچنین برای یکی از کشور هایی که تعداد مربیان آن کم است نیز میخواهیم الگوهای پرتکرار را پیدا کنیم، در اینجا کشور فرانسه به تعداد 10 مربی دارد که میخواهیم برای آن الگوهای پرتکرار را پیدا کنیم:



تصویر 47- الگوهای پرتکرار مربیان کشور فرانسه با minsup = 0.2

فایل ورزشکاران شامل نام ورزشکار، نام کشور، رشته ورزشی می باشد:

در ابتدای کار نام ورزشکاران را از دیتافریم حذف میکنیم چون مقادیر آن منحصر به فرد است و در پیدا کردن الگوهای پرتکرار کمک نمیکند.

در این فایل کشور های آمریکا، ژاپن و استرالیا بیشترین تعداد ورزشکار را دارند پس برای این ورزشکاران این کشور ها الگوهای پرتکرار را پیدا میکنیم:



تصویر 48 - الگوهای پرتکرار ورزشکاران کشور آمریکا با minsup = 0.2



تصویر 49 - الگوهای پرتکرار ورزشکاران کشورژاپن با minsup = 0.1

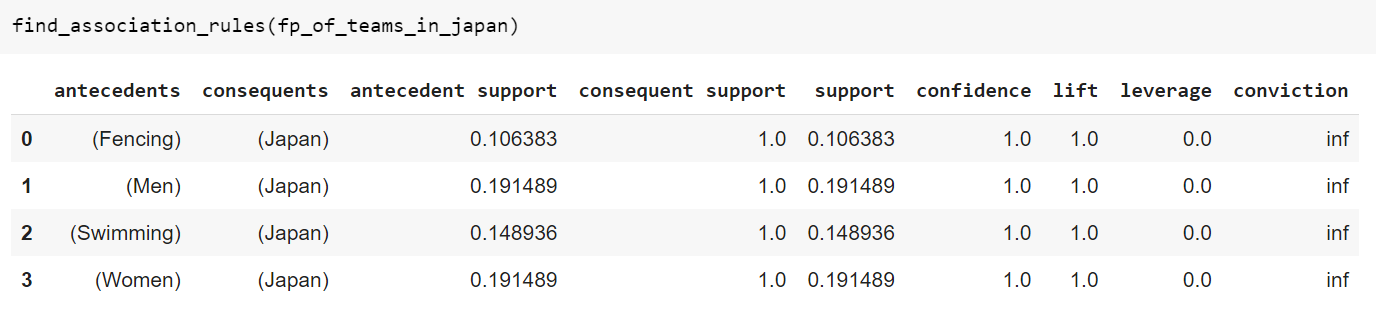


تصویر 50 - الگوهای پرتکرار ورزشکاران کشور استرالیا با minsup=0.1

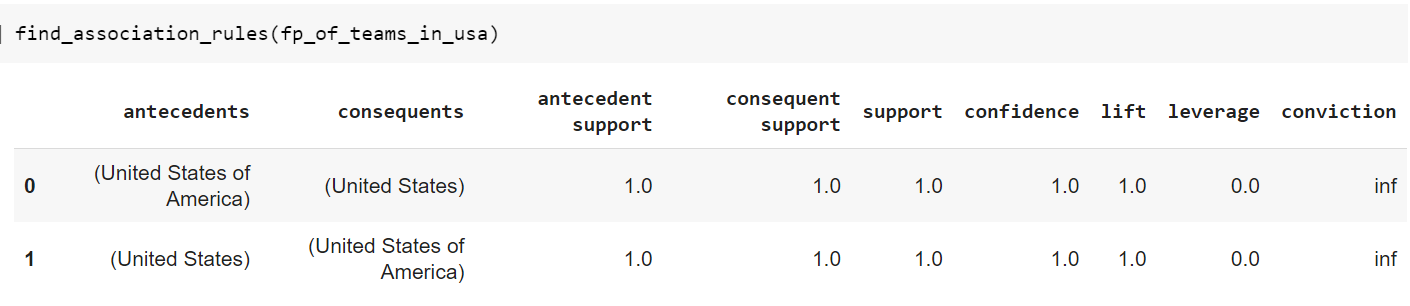
همانطور که میبینیم برای این ها نیز الگوهای پرتکرار با support بالا پیدا نشده است.

## 5.2- قوانین انجمنی

حال برای الگوهای پرتکراری که در بخش قبل پیدا کردیم، قوانین انجمنی را استخراج میکنیم. در این بخش از معیار همبستگی lift برای ارزیابی قوانین استفاده کردیم. لازم به ذکراست که در اینجا کمترین مقدار اطمینان 0.6 در نظر گرفته شده است که با توجه به این قوانین انجمنی به صورت زیر می باشد:

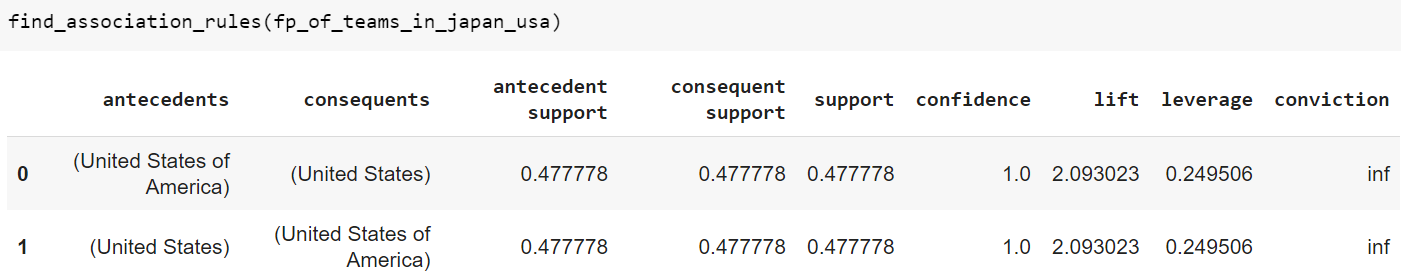


تصویر 51- قوانین انجمنی برای تیم های کشور ژاپن با minsup = 0.1 و minconf = 0.6



تصویر 52 - قوانین انجمنی برای تیم های کشور آمریکا با minsup = 0.2 و minconf = 0.6

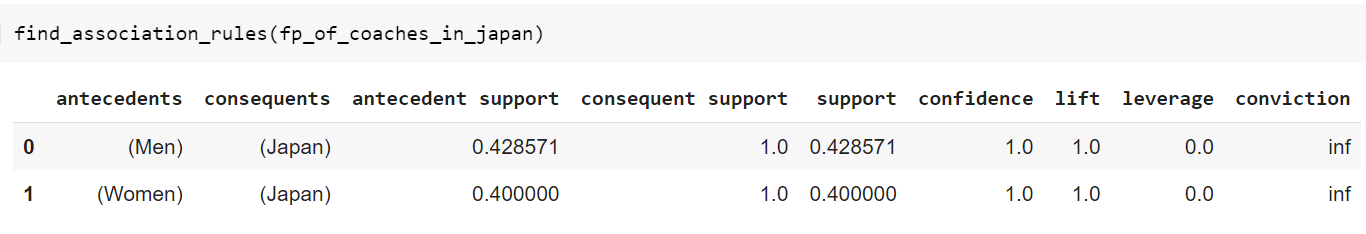
برای قوانین دو جدول بالا معیار همبستگی lift برابر 1 می باشد در نتیجه میتوان گفت که این قوانین از لحاظ آماری نمیتواند در دنیای واقعی استفاده شوند.



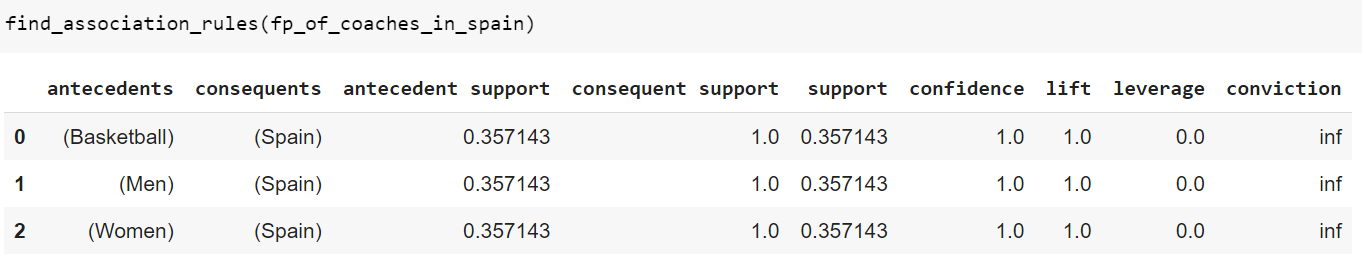
تصویر 53 - قوانین انجمنی برای تیم های کشور ژاپن و آمریکا با minsup = 0.2 و minconf = 0.6

در تصویر 53 مقدار معیار همبستگی lift برای قوانین بیشتر از 1 می باشد، در نتیجه میتوان گفت که این قوانین، قوی هستند و همبستگی مثبت وجود دارد.

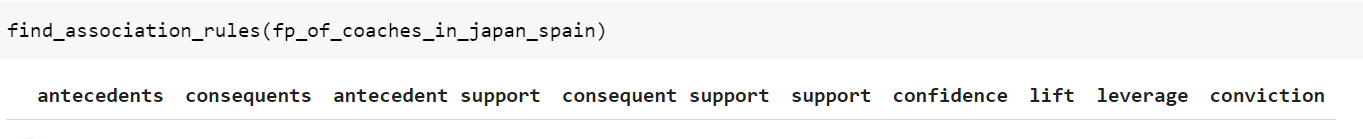
برای جداول بعد نیز همانطور که مشاهده میکنید مقدار lift برابر 1 می باشد که نشان میدهد که این قوانین، قوی نیستند و در دنیای واقعی کاربرد ندارند.



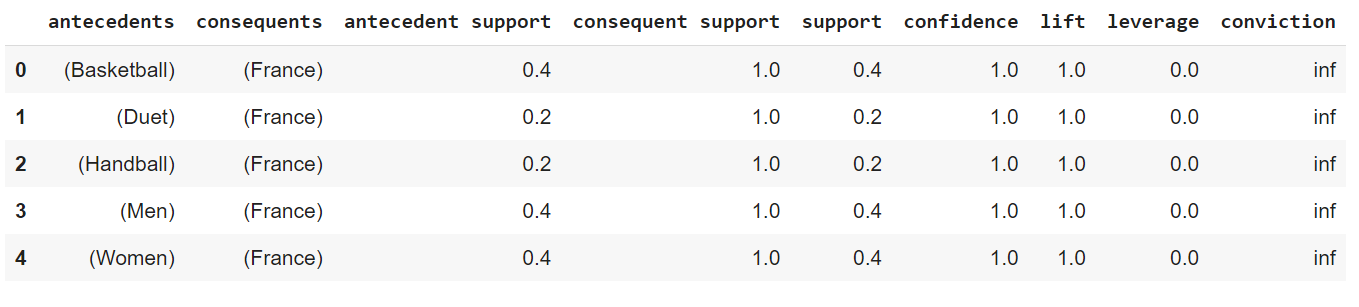
تصویر 54 - قوانین انجمنی برای مربیان کشور ژاپن با minsup = 0.4 و minconf = 0.6



تصویر 55- قوانین انجمنی برای مربیان کشور اسپانیا با minsup = 0.4 و minconf = 0.6



تصویر 56 - قوانین انجمنی برای مربیان کشور ژاپن و اسپانیا با minsup = 0.3 و minconf = 0.6



تصویر 57 - قوانین انجمنی برای مربیان کشور فرانسه با minsup = 0.2 و minconf = 0.6



تصویر 58 - قوانین انجمنی برای ورزشکاران کشور آمریکا با minsup = 0.2 و minconf = 0.6



تصویر 59 -قوانین انجمنی برای ورزشکاران کشور ژاپن با minsup = 0.1 و minconf = 0.6



تصویر 60 - قوانین انجمنی برای ورزشکاران کشور استرالیا با minsup = 0.1 و minconf = 0.6

# نتیجه گیری

ما در این پروژه سعی کردیم تا فاز های مختلف داده کاوی را روی مجموعه داده خود انجام دهیم. مجموعه داده ما بیشتر حالت گزارشی از المپیک 2021 داشتند و بیشتر برای تحلیل و بصری سازی داده ها مناسب هستند و هدف این مجموعه داده ساخت و آموزش و پیش‌بینی مدل روی داده ها نمی‌باشد، اما با این حال سعی کردیم تا با تحقیق در مورد الگوریتم ها و پارامتر های آنها دقت مدل ها را به بیشترین مقدار ممکن برسانیم.

# منابع و مراجع

[1] <https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn>

[2] <https://towardsdatascience.com/bbn-bayesian-belief-networks-how-to-build-them-effectively-in-python-6b7f93435bba>

[3] <https://mljar.com/blog/visualize-decision-tree/>

[4] <https://scikit-learn.org/stable/modules/tree.html>

[5] <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier>

[6] <https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html>

[7] <https://www.analyticsvidhya.com/blog/2020/10/cost-complexity-pruning-decision-trees/>

[8] <https://www.tensorflow.org/tutorials/keras/classification>

[9] <https://scikit-learn.org/stable/modules/ensemble.html>

[10] <https://www.kaggle.com/faressayah/ensemble-ml-algorithms-bagging-boosting-voting>

[11] <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html>

[12] <https://towardsdatascience.com/k-means-vs-dbscan-clustering-49f8e627de27>

1. پست الکترونیکی: mwaghei78@gmail.com [↑](#footnote-ref-1)
2. مجموعه داده ی المپیک 2021 در سایت kaggle در دسترس می باشد. [↑](#footnote-ref-2)
3. https://www.kaggle.com/arjunprasadsarkhel/2021-olympics-in-tokyo [↑](#footnote-ref-3)