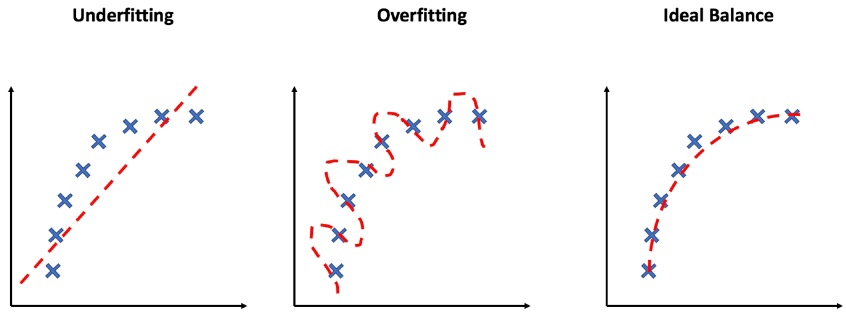
|  |  |  |
| --- | --- | --- |
|  | به نام خدا |  |
| **دانشگاه تهران**  **دانشکده‌ مهندسی مکانیک**  **هوش مصنوعی**  **تمرین 2** | | |

|  |  |
| --- | --- |
| محمد مشرقی | نام و نام خانوادگی |
|  | شماره‌ دانشجویی |
|  | تاریخ ارسال گزارش |

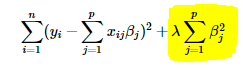
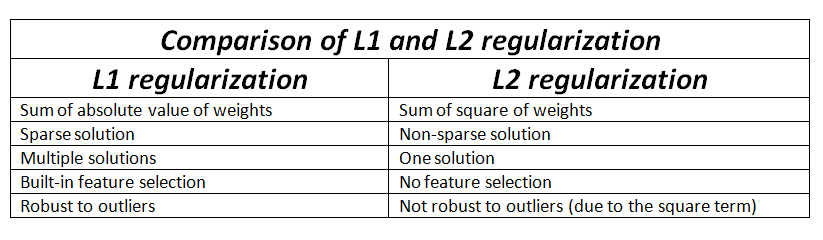
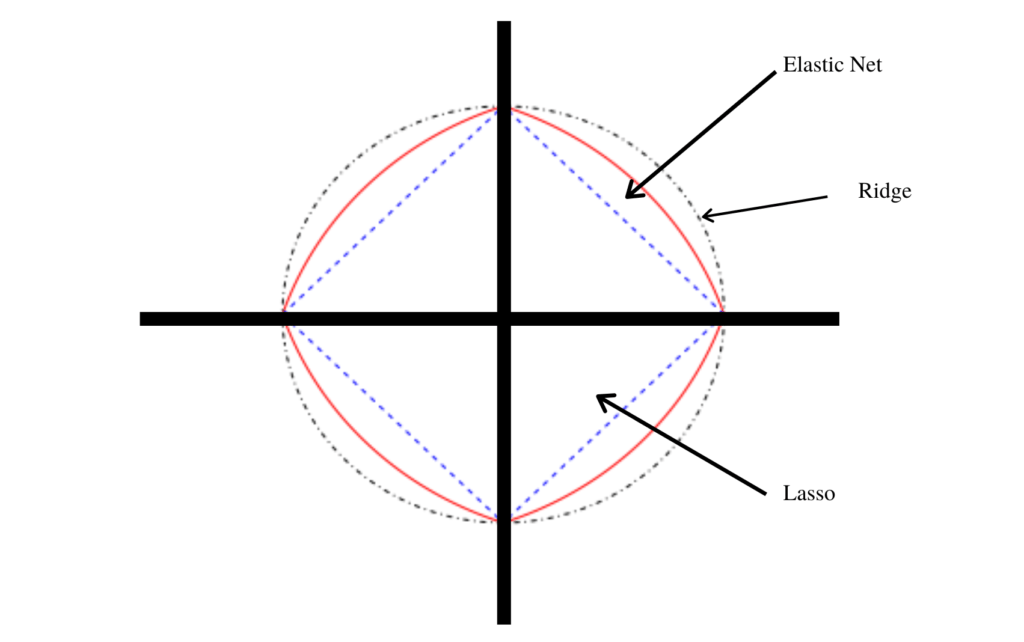
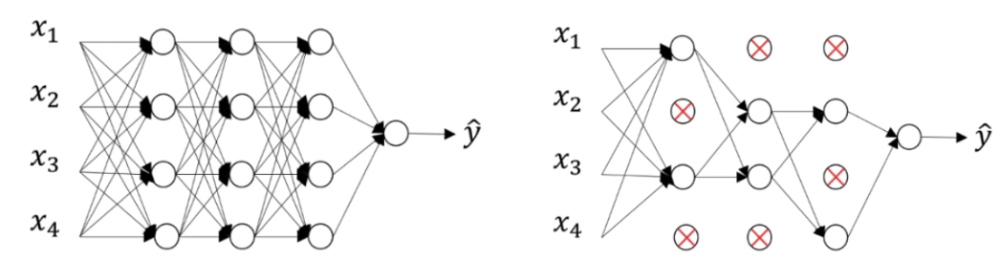
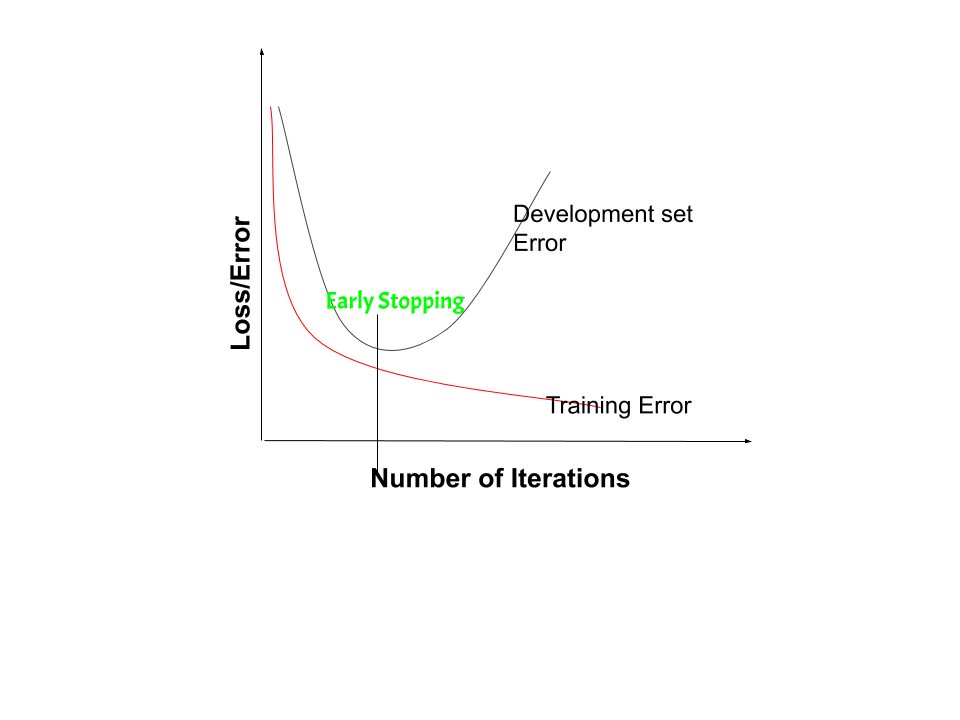
# 1-Regularization

**تعریف** : در ماشین لرنینگ و دیپ لرنینگ برای اینکه از

1. پیچیده تر شدن مدل(Overfitting).
2. و نتایج بسیار نزدیک به داده های تمرینی نزدیک شود که در نهایت ممکن است باعث شود در داده های استفاده نشده (جدید) خطا زیاد شود.

 جلوگیری شود از Regularization استفاده می شود.

انواع آن :

* **L1 Regularization (Lasso Regression):** (Least Absolute Shrinkage and Selection Operator)   
  در اینجا با انتخاب مناسب lambda می توانیم برای هر ضریب به مقداری مناسب برسیم و آن را به کم کنیم(و نهایتا وزن به صفر متمایل کنیم)  
  این ویژگی وقتی ضریب های زیادی داشته باشیم و بخواهیم وزن ضریب های کم مهم صفر شوند بدرد می خورد.  
    
  
* **L2 Regularization(Ridge regression):  
  در اینجا هم مانند L1 عمل می کند اما در انتهای تمرین نتایج فقط به صفر میل می کنن و صفر نمی شوند.  
  وقتی استفاده می کنیم که** ویژگی های هم خطی و همبستگی داشته باشیم.  
    
    
    
    
    
  
* **Elastic net  
  این روش ترکیبی بین L1و L2 هستش.**
* Dropout  
  در دیپ لرنینگ گاهی وقت ها با حذف بعضی نود ها از overfitting جلوگیری می کنیم.
* Early Stopping  
   برای روش هایی در تعیین تعداد دسته استفاده می شود. 

# 2-

در این قسمت ابتدا داده ها و سپس برای تمرین و تست تقسیم می کنیم.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

سپس به درجه 7 می بریم.

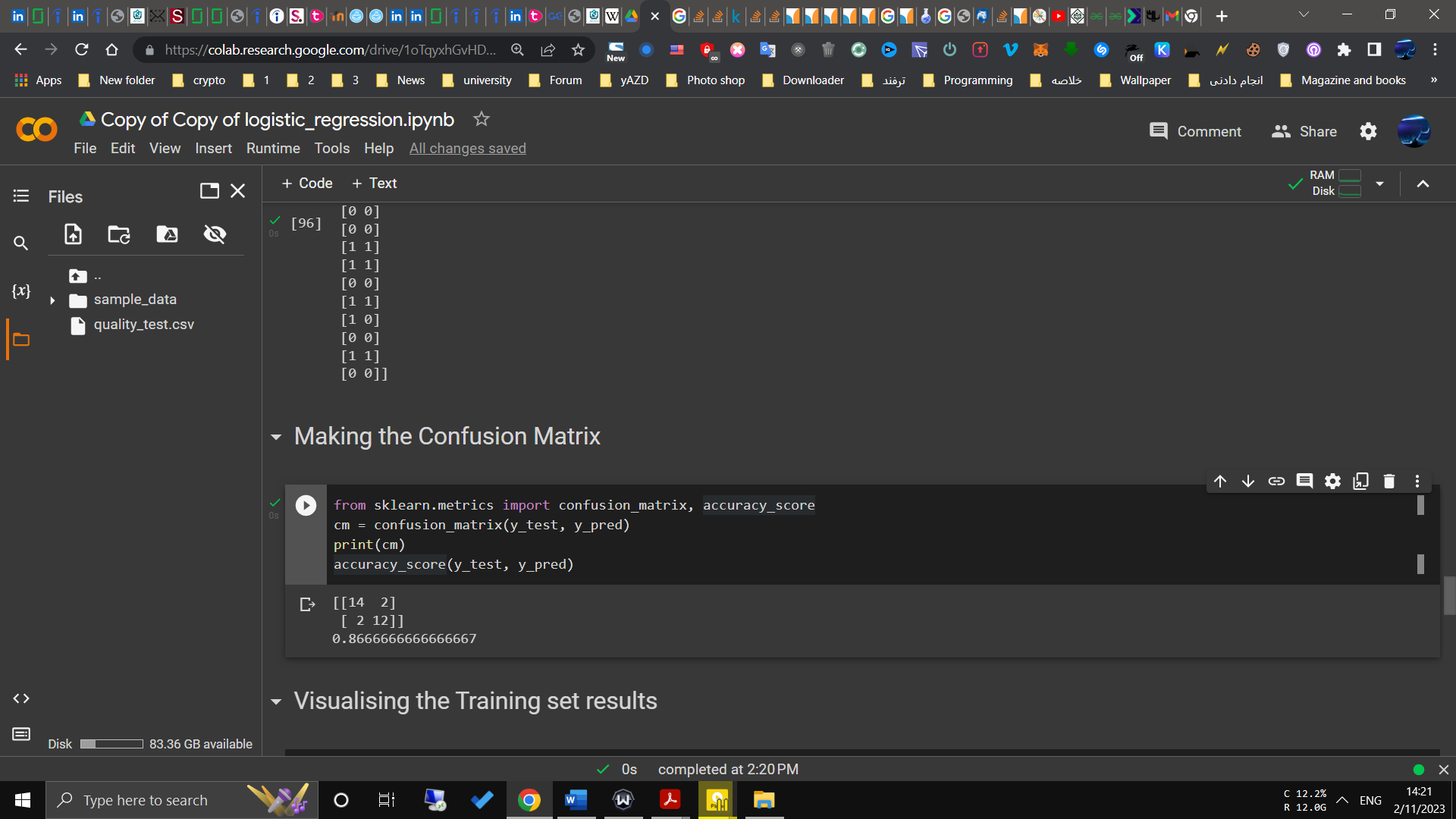
poly = PolynomialFeatures(degree = 7)

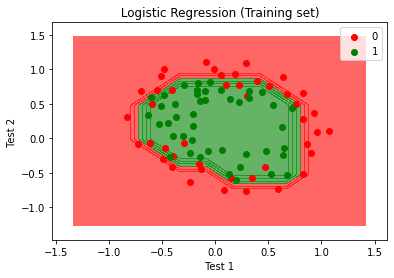
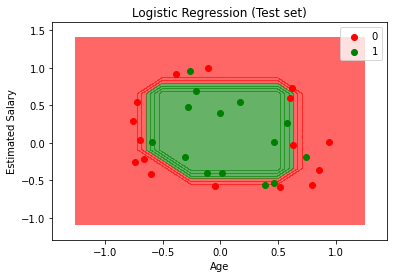
X\_poly\_train = poly.fit\_transform(X\_train)

حال به سراغ اموزش می رویم و regularization رو l2 تنظیم می کنیم c=0.01

classifier = LogisticRegression  
(C=0.01, penalty='l2',random\_state =43)

classifier.fit(X\_poly\_train, y\_train)

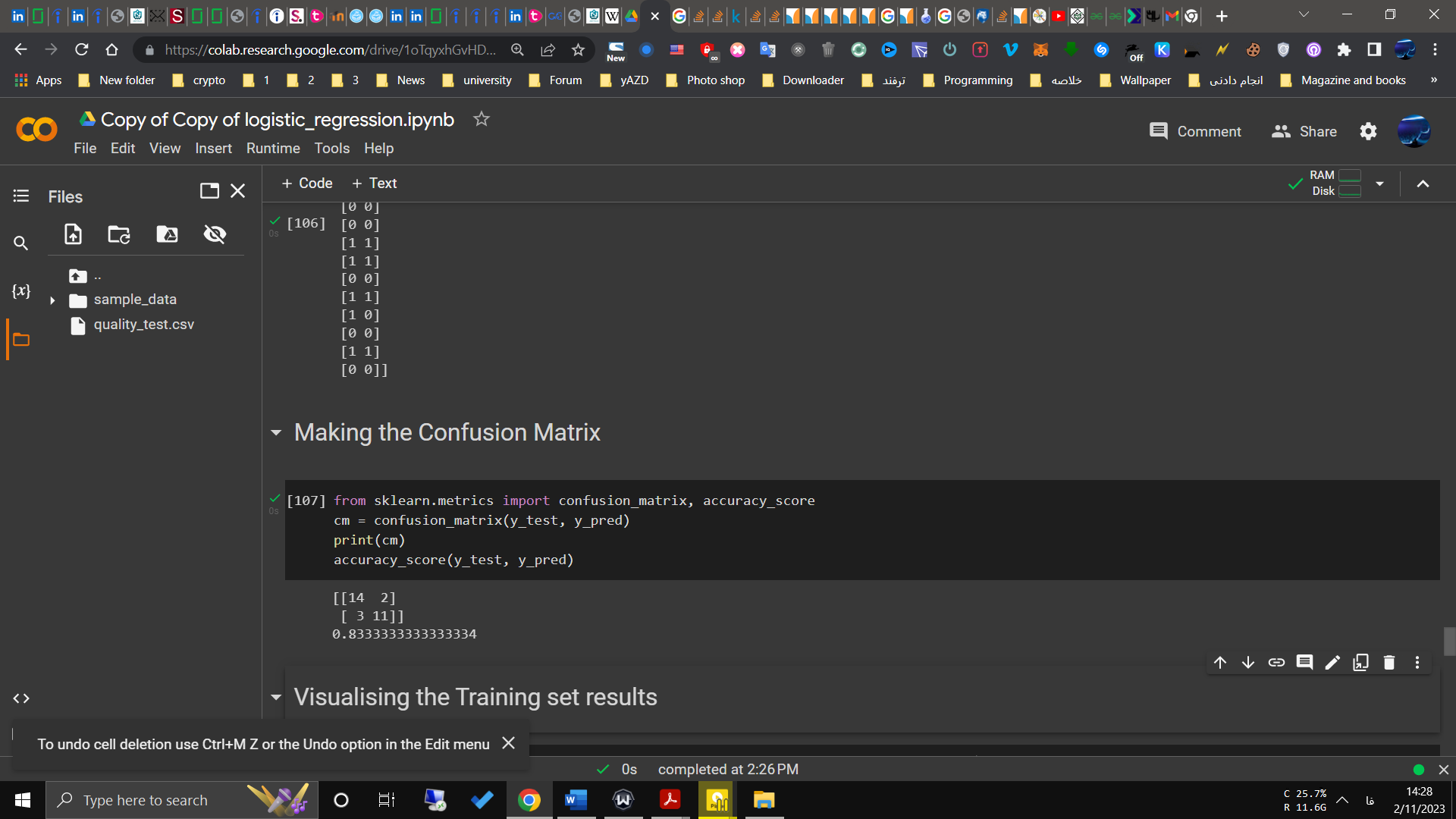
نتایج:

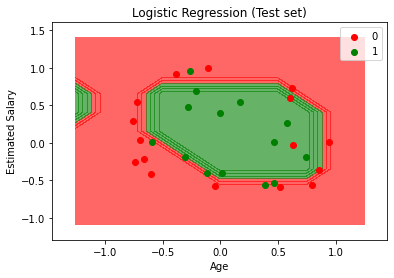
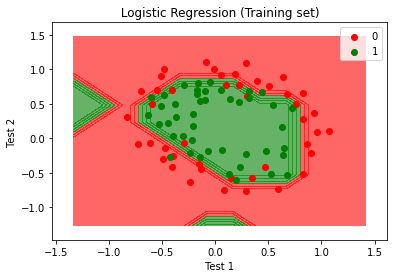


حال اگر c=1 داریم:

classifier = LogisticRegression(C=1, penalty='l2',random\_state =43)

classifier.fit(X\_poly\_train, y\_train)

نتایج:

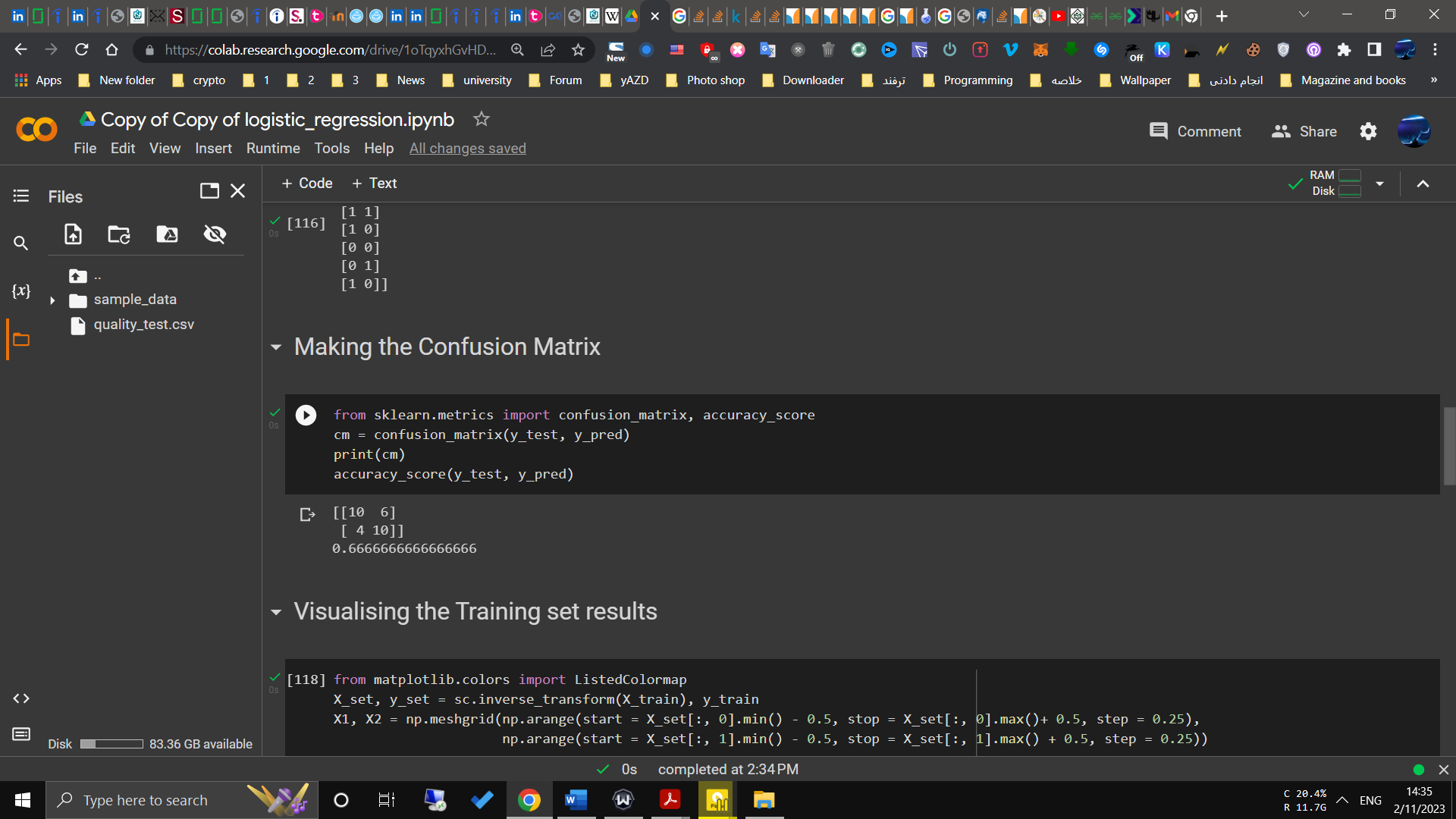


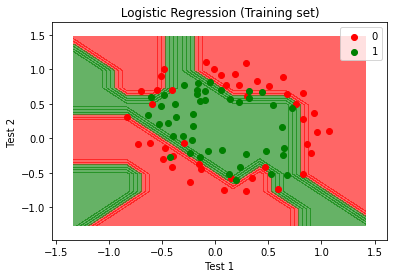
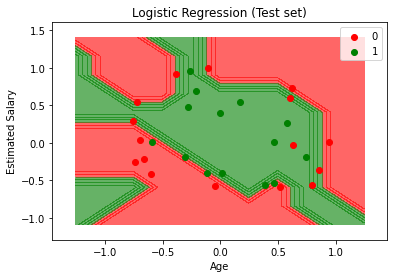
حال اگر c=1000 داریم:

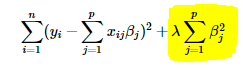
classifier = LogisticRegression  
(C=10000, penalty='l2',random\_state =43)

classifier.fit(X\_poly\_train, y\_train)

نتایج:





با توجه به نتایج می فهمیم با بزرگتر شدن C دقت اموزش پایین می آید و با کوچکتر شدن آن دقت بیشتر می شود. و

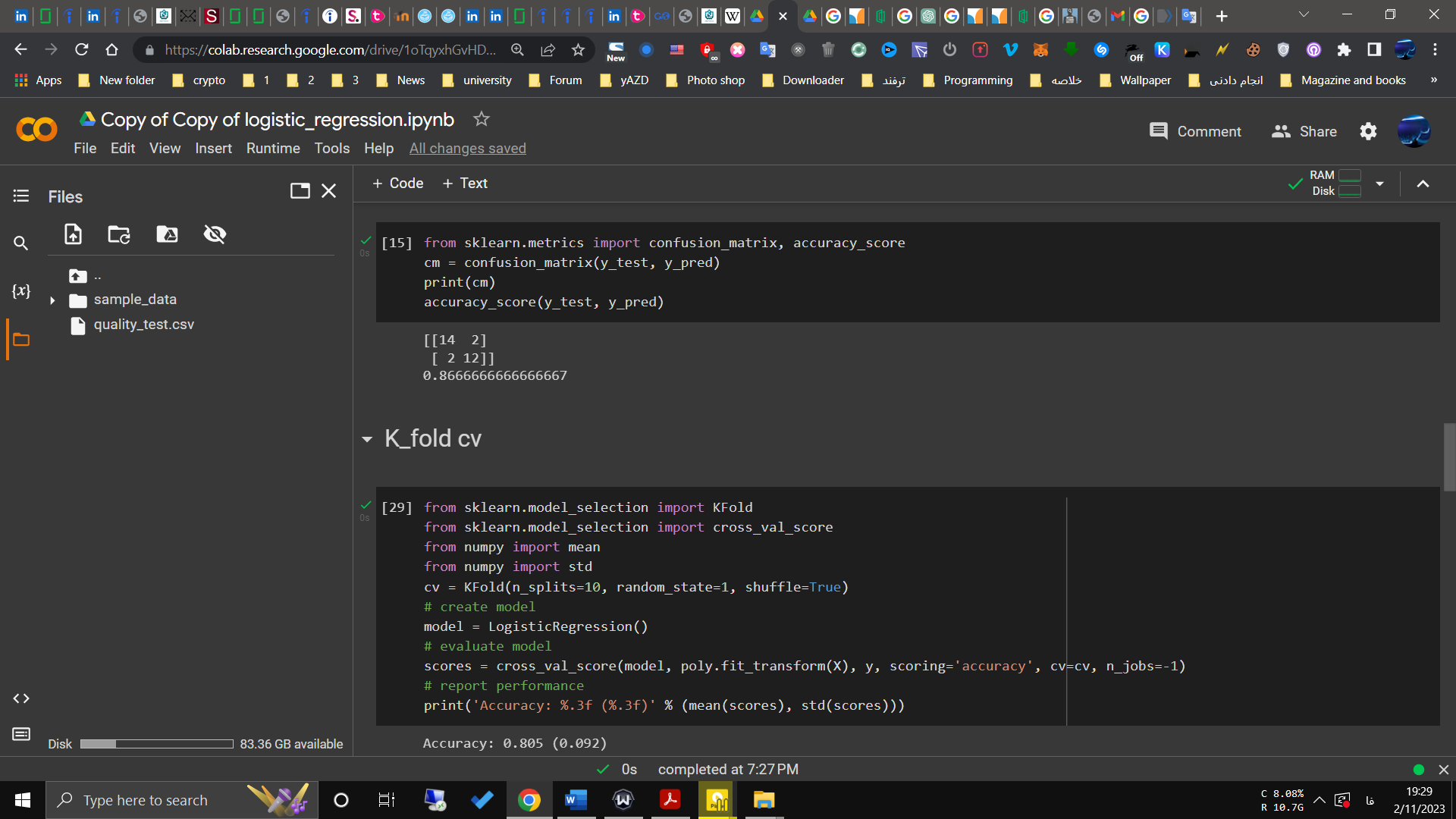
با بزرگ شدن c ،جریمه کوچکتر می شود و ممکن است overfit رخ دهد یا نتایج غیر قابل قبول باشند و اینکه با توجه به سوال یک مدل بیش از حد در مورد ویژگی های داده های اموزشی یاد می گیرند و نمی توانند با داده های جدید تعمیم داد.  
با کوچک شدن c ، جریمه بزرگتر می شود و در نهایت overfit رخ نمی دهد و دقت آن نیز بیشتر است اما از یه حدی به بعد مدل بسیار ساده خواهد بود و مدل ممکن است با عدم تناسب داده مواجهه شود و نمی تواند به خوبی یاد بگیرد .

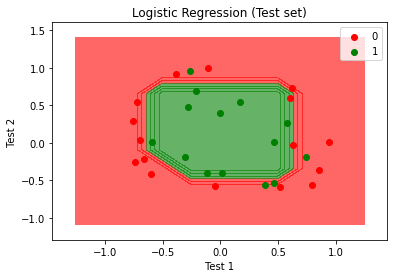
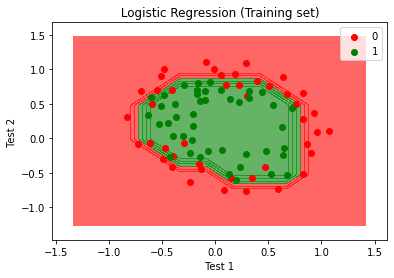
مقدار ایده آل C برای هر مدل بستگی به داده های آن دارد و نمی توان همیشه یه مقدار ثابت گرفت.  
حال در این سوال ما c=0.01 فرض می کنیم.

# 3-

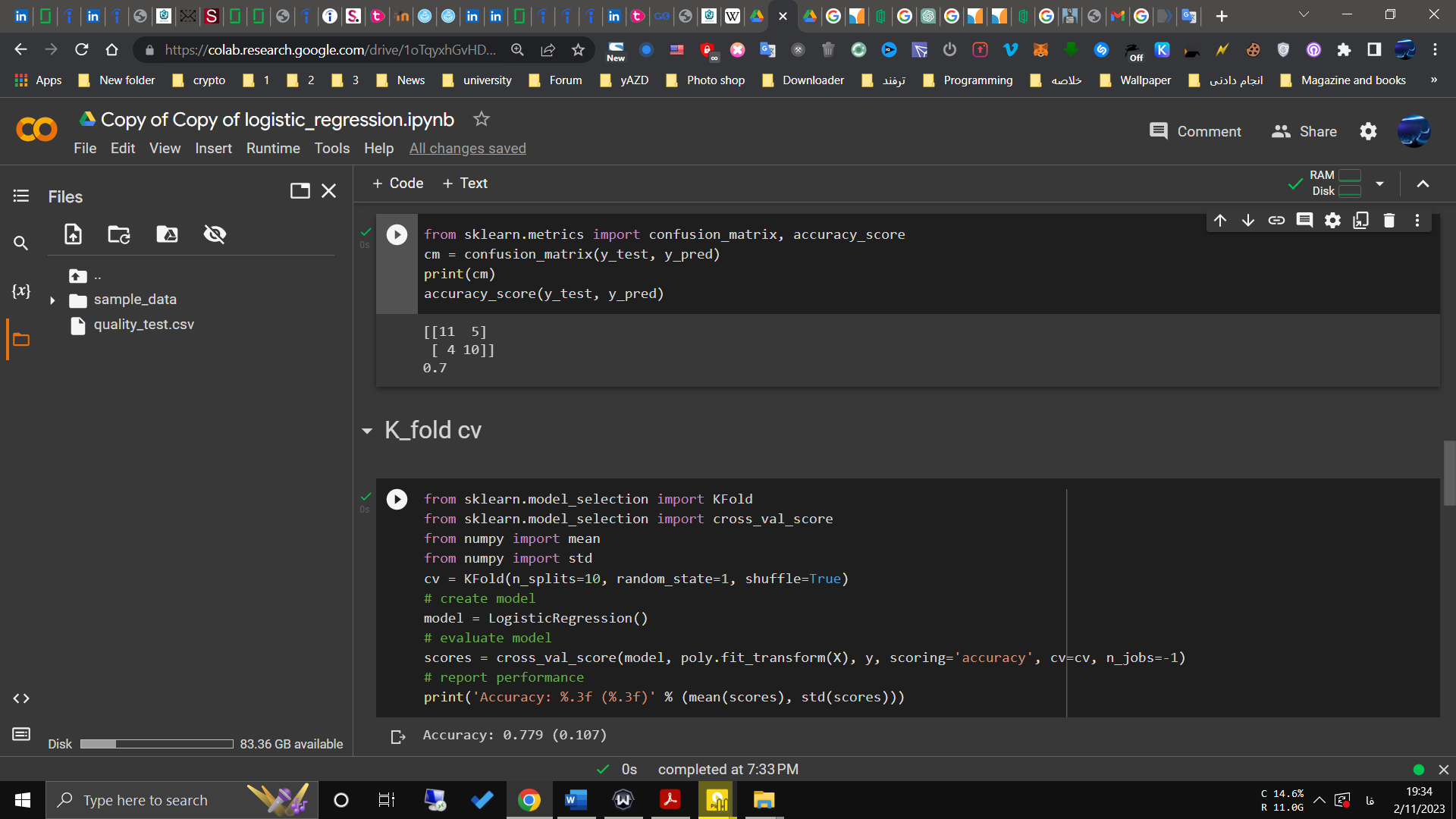
اعتبار سنجی متقابل یک روش برای ارزیابی یک مدل در ماشین لرنینگ و همچنین آزمایش نحوه عملکرد آن است. از CV اصولا در فعالیت های کاربردی یادگیری ماشین استفاده می شود. این کار درمقایسه و انتخاب یک مدل مناسب برای مسئله مدل‌سازی پیش‌بینی‌کننده خاص کمک می‌کند.

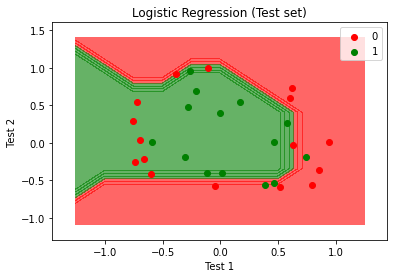
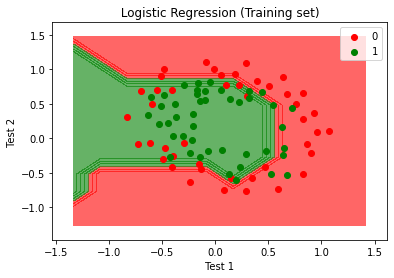
حال تست می کنیم و مقدار k را برابر 10 قرار می دهیم.

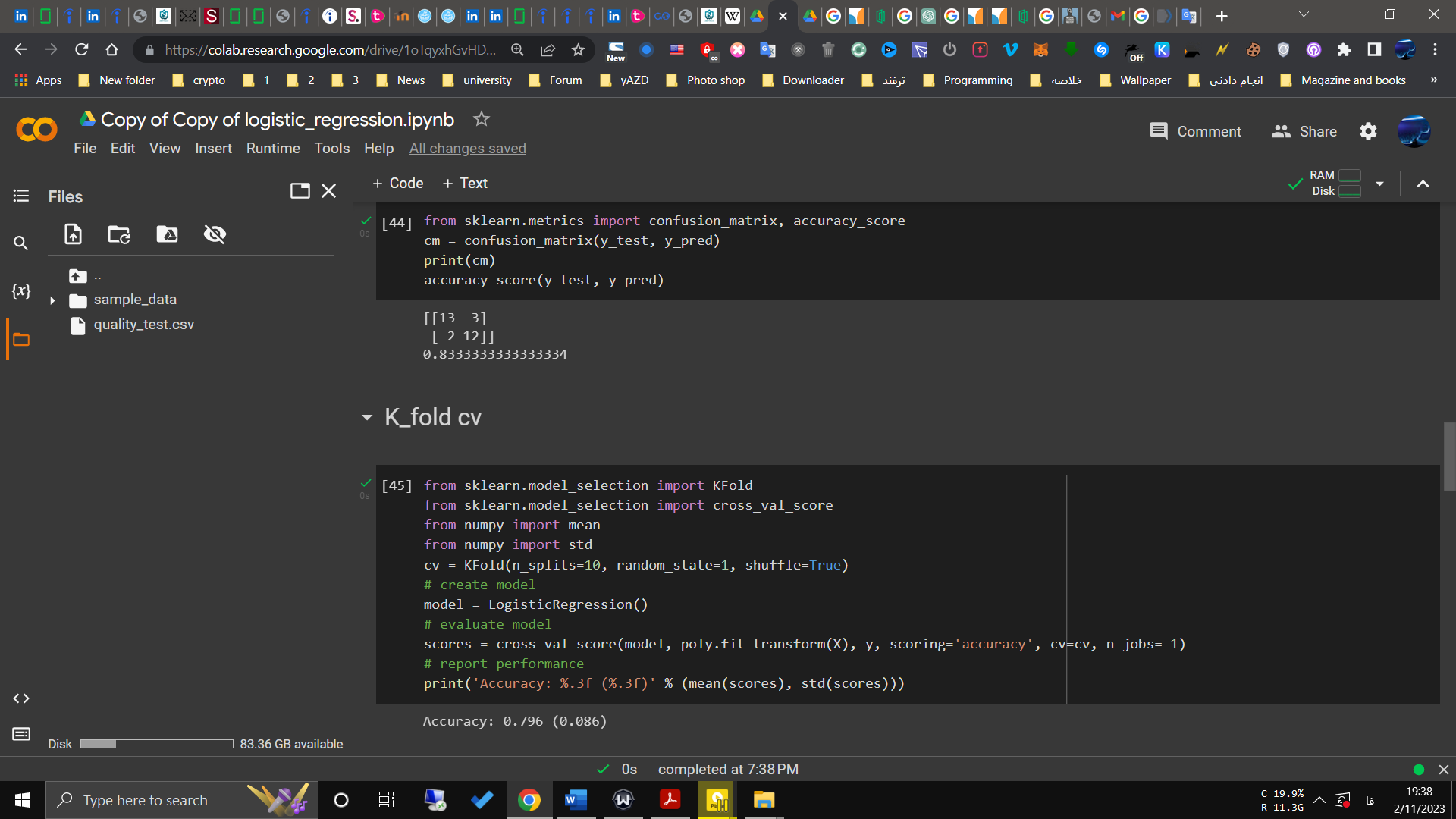
اگر درجه 7 باشد نتایج داریم:

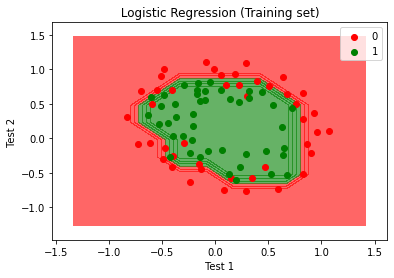
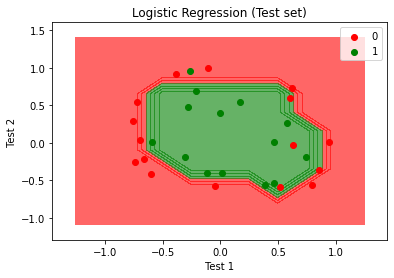


## اگر درجه 3 باشد نتایج داریم:





درجه 10 باشد داریم :



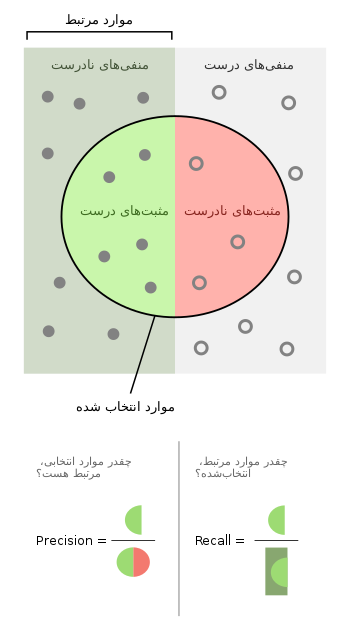
## نتیجه :

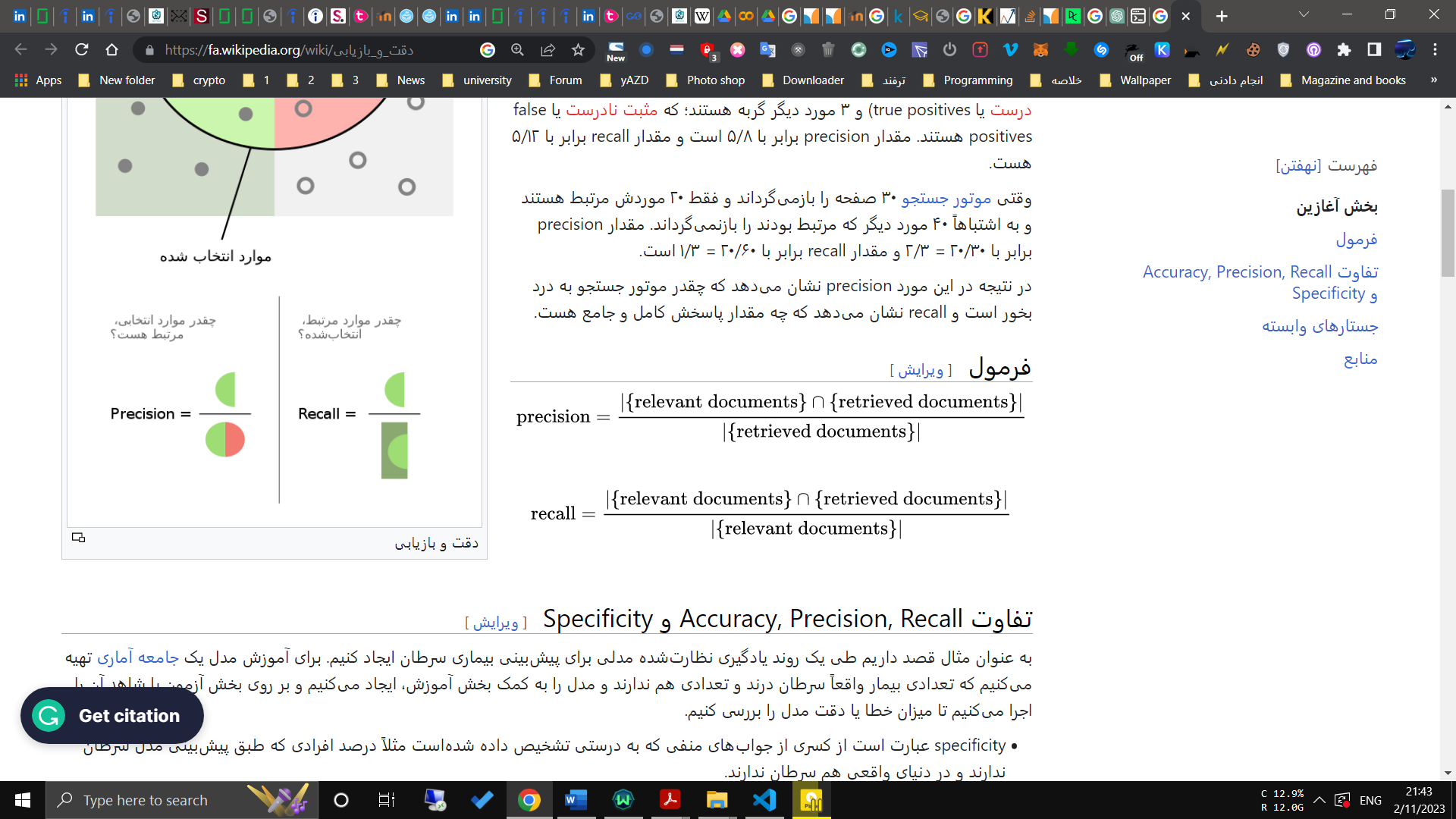
با توجه به تست k fold cv ، mean(score) هر چقدر به یک نزدیک تر شود اون درجه بهتر است که با توجه به نتایج درجه برابر 7 بهترین حالت است.

# 4-

تعاریف :

Recallبرابر است با تقسیم تعداد مواردی که توسط مدل درست تشخیص داده‌اند شده بر تعداد کل مواردی که توسط مدل ایجاد شده‌اند و Precision برابر است با تقسیم تعداد مواردی که توسط مدل درست تشخیص داده شده‌است بر تعداد مواردی که واقعاً درست هستند، درست تشخیص داده شده‌اند.





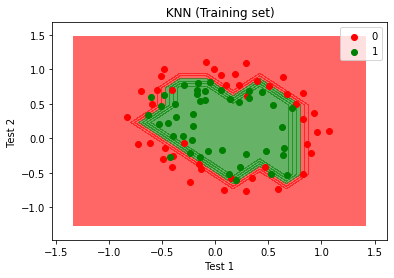
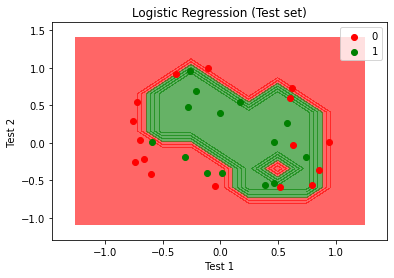
K=1 :

classifier = KNeighborsClassifier(n\_neighbors = 1 , p = 2)

نتایج:

accuracy\_score = 0.8

precision\_score = 0.7857142857142857

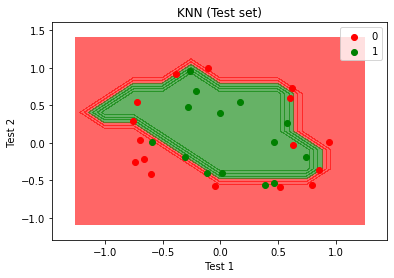
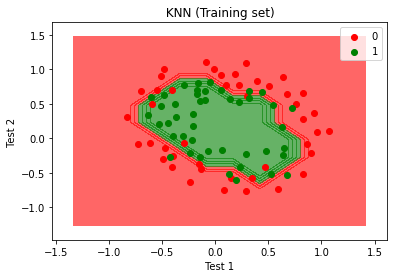
recall\_score = 0.7857142857142857

## K=2 :

classifier = KNeighborsClassifier(n\_neighbors = 7 , p = 2)

نتایج:

accuracy\_score = 0.8  
precision\_score = 0.75  
recall\_score = 0.8571428571428571



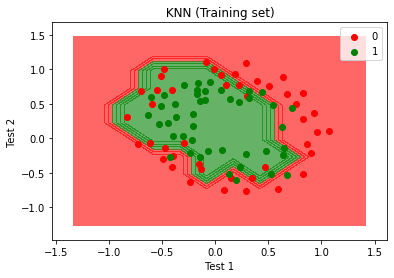
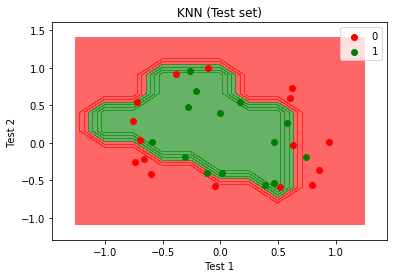
K=13 :

classifier = KNeighborsClassifier(n\_neighbors = 13 , p = 2)

نتایج :

accuracy\_score = 0.7

precision\_score = 0.631578947368421

recall\_score = 0.8571428571428571

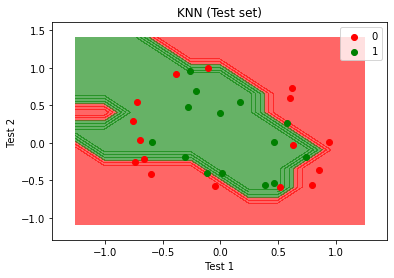
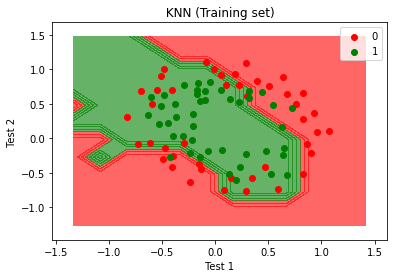
## K=19 :

classifier = KNeighborsClassifier(n\_neighbors = 19 , p = 2)

نتایج :

accuracy\_score = 0.6666666666666666

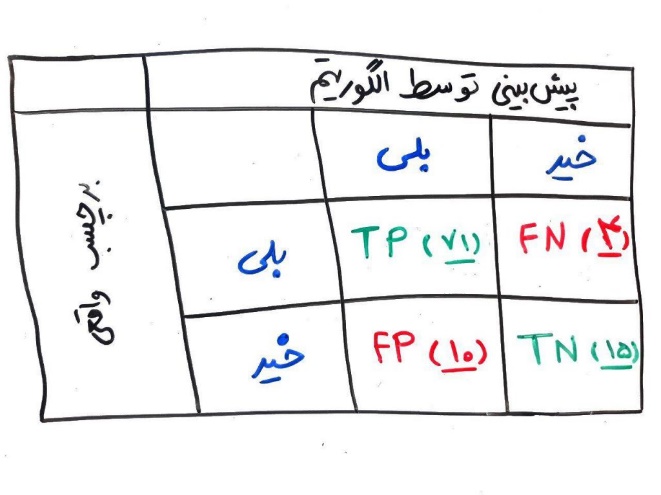
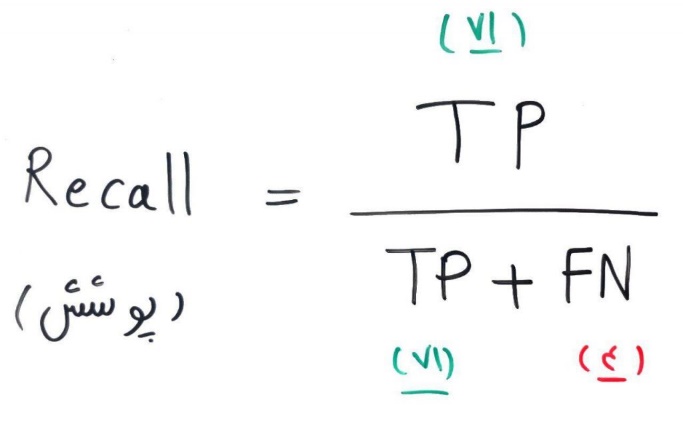
precision\_score = 0.6111111111111112

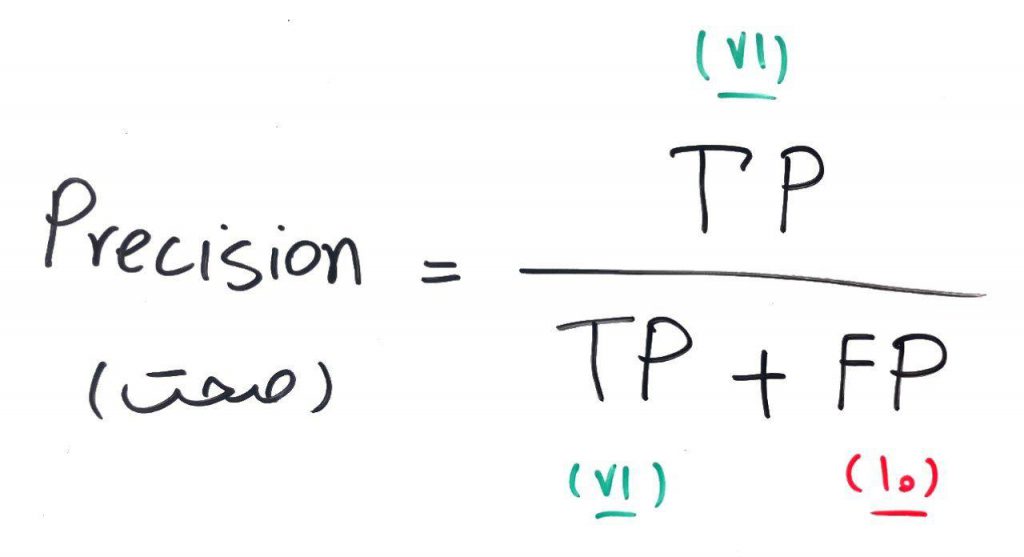
recall\_score = 0.7857142857142857

## نتیجه:

می دانیم که در معیار Accuracy نمی**‌تواند تفاوتی بین خطای False Negative و خطای False Positive داشته باشد.**  
برای همین از پوشش (recall) و صحت (**Precision**) استفاده می کنیم.  
اگر با دقت به فرمول نگاه کرده باشید، متوجه می‌شوید که تمرکز اصلی این معیار، بر روی **درستیِ تشخیص‌های «بلی» توسط الگوریتم**است. در واقع معیار صحت (Precision) معیاری است که به ما می‌گوید الگوریتم چند درصدِ «بلی»هایش درست بوده.  
همان‌طور که مشاهده می‌کنید، تمرکز اصلی معیار پوشش (Recall) بر خلاف معیار صحت (Precision) **بر روی داده‌هایی است که واقعاً «بلی» بوده‌اند.**

**مثال یکی از سایت ها :**

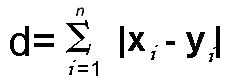
 



حال با توجه به نتایج و نیازمان از تحلیل دنبال معیار مورد نظر می گردیم و طبق اون دنبال k مورد نظر می رویم.  
حال اگر فرض کنیم که قطعه مورد نظر اگر خراب باشد و داخل دستگاه قرار گیرد باعث می شود بقیه قطعات دستگاه نیز خراب شوند پس اینجا باید سعی کنیم تا می توانیم از ورود دستگاه خراب جلوگیری کنیم پس از روش پوشش Recall را مهمتر از روش صحت فرض می کنیم که طبق این قاعده K = 7 انتخاب می شود.

# 5-

برای بدست اوردن بهترین مقدار K با استفاده از روش فاصله منهتن باید جای p =2 ، باید **یک** بگذاریم داریم:



فرمول فاصله منهتن

classifier = KNeighborsClassifier(n\_neighbors = 1 , **p = 1**)

حال این بار از روشی دیگر استفاده می کنیم و با یک for کار را تموم می کنیم:

for i in range(1 ,20):

  classifier = KNeighborsClassifier(n\_neighbors = i , p = 1)

  classifier.fit(X\_train, y\_train)

  y\_pred = classifier.predict(X\_test)

  print( "k = ",i, ' accuracy\_score =  %.3f    precision\_score =

  %.3f   recall\_score =   %.3f '%(accuracy\_score(y\_test, y\_pred),prec ision\_score(y\_test, y\_pred),recall\_score(y\_test, y\_pred)))

نتیجه :  
k = 1 accuracy\_score = 0.800 precision\_score = 0.833 recall\_score = 0.714

k = 2 accuracy\_score = 0.767 precision\_score = 0.889 recall\_score = 0.571

k = 3 accuracy\_score = 0.633 precision\_score = 0.615 recall\_score = 0.571

k = 4 accuracy\_score = 0.700 precision\_score = 0.857 recall\_score = 0.429

k = 5 accuracy\_score = 0.767 precision\_score = 0.769 recall\_score = 0.714

k = 6 accuracy\_score = 0.733 precision\_score = 0.875 recall\_score = 0.500

k = 7 accuracy\_score = 0.767 precision\_score = 0.769 recall\_score = 0.714

k = 8 accuracy\_score = 0.833 precision\_score = 0.909 recall\_score = 0.714

k = 9 accuracy\_score = 0.767 precision\_score = 0.684 recall\_score = 0.929

k = 10 accuracy\_score = 0.767 precision\_score = 0.733 recall\_score = 0.786

k = 11 accuracy\_score = 0.800 precision\_score = 0.722 recall\_score = 0.929

k = 12 accuracy\_score = 0.833 precision\_score = 0.846 recall\_score = 0.786

k = 13 accuracy\_score = 0.767 precision\_score = 0.684 recall\_score = 0.929

k = 14 accuracy\_score = 0.767 precision\_score = 0.733 recall\_score = 0.786

k = 15 accuracy\_score = 0.700 precision\_score = 0.609 recall\_score = 1.000

k = 16 accuracy\_score = 0.567 precision\_score = 0.533 recall\_score = 0.571

k = 17 accuracy\_score = 0.667 precision\_score = 0.600 recall\_score = 0.857

k = 18 accuracy\_score = 0.600 precision\_score = 0.600 recall\_score = 0.429

k = 19 accuracy\_score = 0.733 precision\_score = 0.667 recall\_score = 0.857

اگر بخواهیم مثل قبل عمل کنیم k=15 بهترین دسته بندی است چون معیار پوشش آن صد درصد است.