به نام خدا

محمدامین جمشیدی گوهرریزی

Project_1:

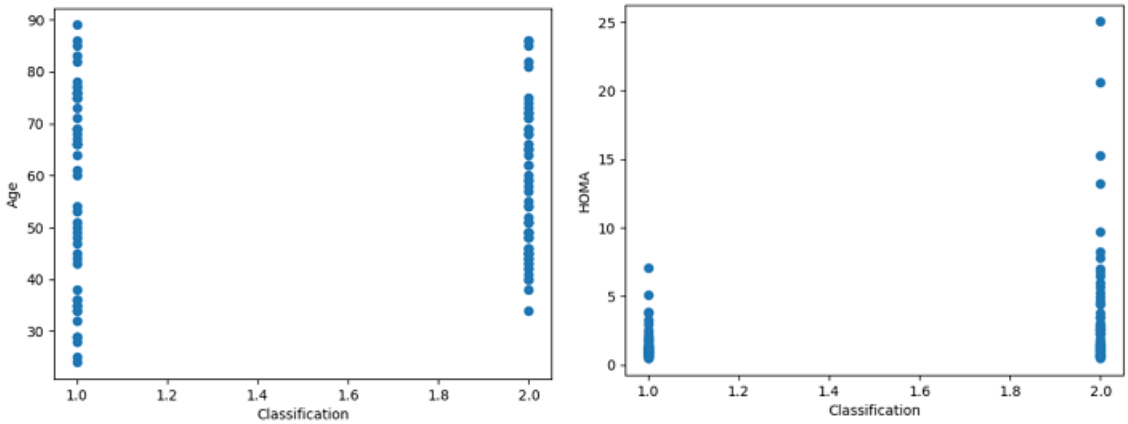
در مرحله اول قبل از اینکه اقدام به آموزش مدل ها کنیم با فرایند هایی اقدام به بهینه سازی و استاندارد سازی مقدار و تعداد پارامتر ها می کنیم:

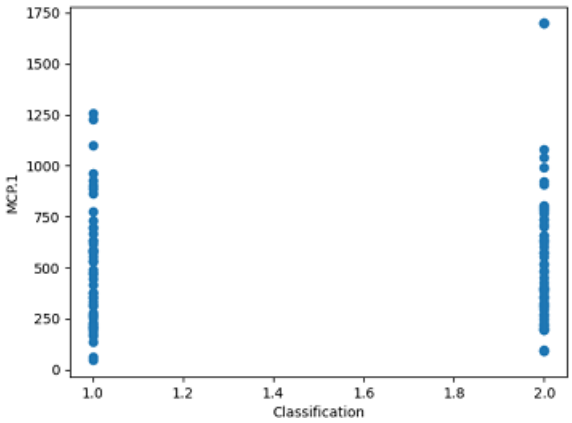
این مراحلی که در پایین ذکر شده اند را در نوتبوک feature_selection_method شده است:

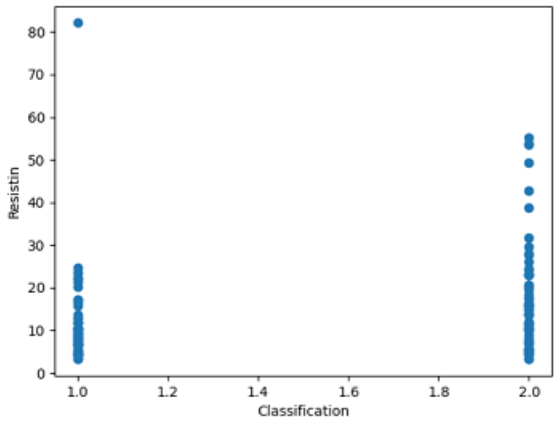
رسم نمودار های تک فیچر:

از رسم این نمودار ها می فهمیم که داده ها به صورت مستقیم از یک فیچر تاثیر نمی پذیرند تمام و نمی توان با تعیین یک threshold دسته بندی را انجام داد چون که بازه هر تک فیچر برای هر کدام از دسته همپوشانی قابل توجهی دارند .

به این صورت:

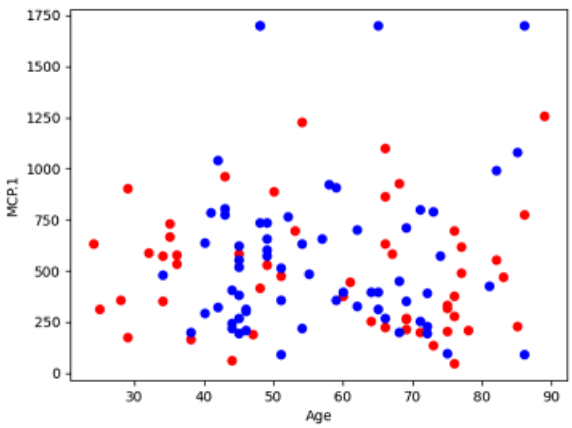


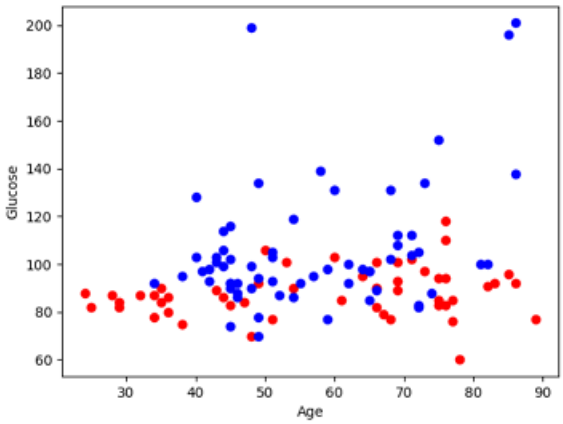


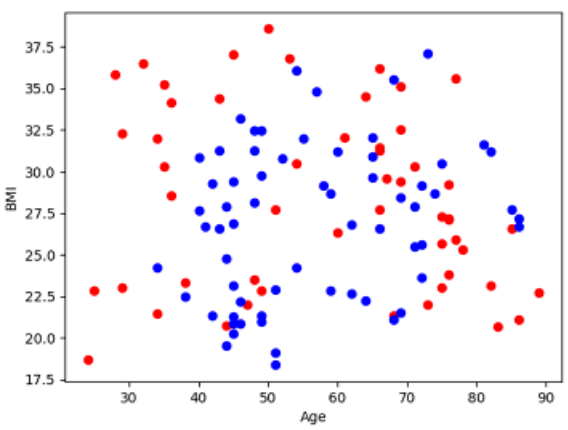


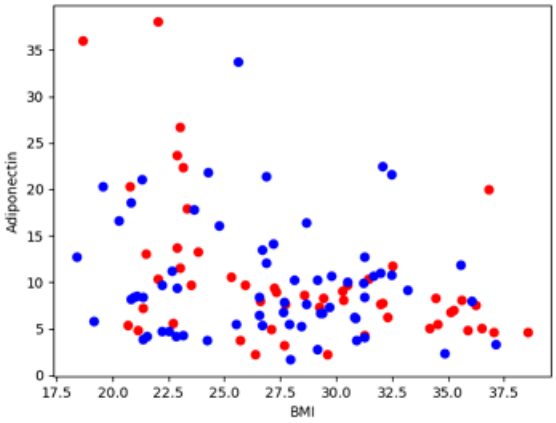
که همینطور مشاهده می شود هیچکدام از دسته ها به وسیله یک خط افقی از هم جدا نمی شوند و در بعضی موار شاید از مقدار یک فیچر از یک threshold خاص بیشتر باشد بتوان تعیین کرد که به نمونه به چه دسته ای مربوط است ولی برای مقادیر کمتر از threshold نمی توان پیش بینی کرد که مربوط به چه دسته ای است برای مثال برای فیچر HOMA شاید بتوان گفت که برای مقادیر بیشتر از 8 مورد مربوط به دسته دوم است ولی برای مقادیر زیر 8 نمی توان نتیجه خاصی گرفت .

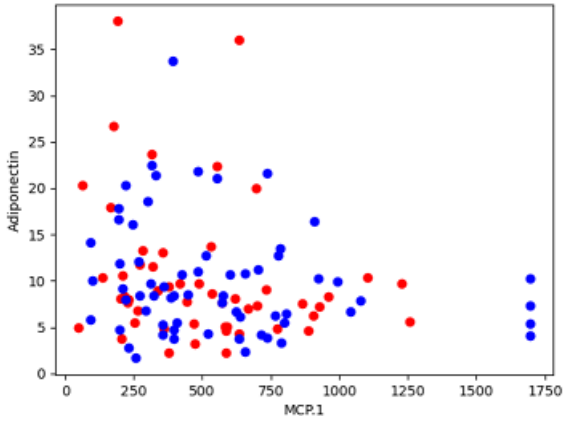
حال نمودار ها را با 2 فیچر بررسی می کنیم:

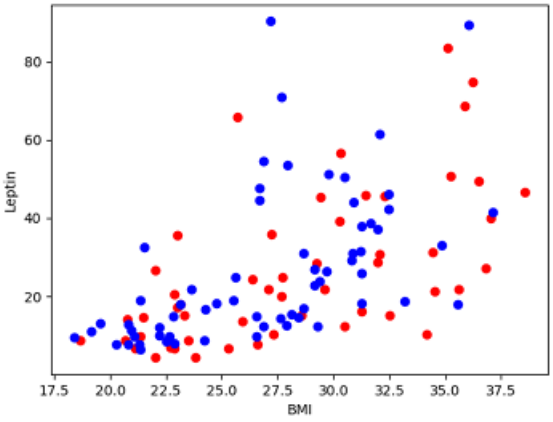


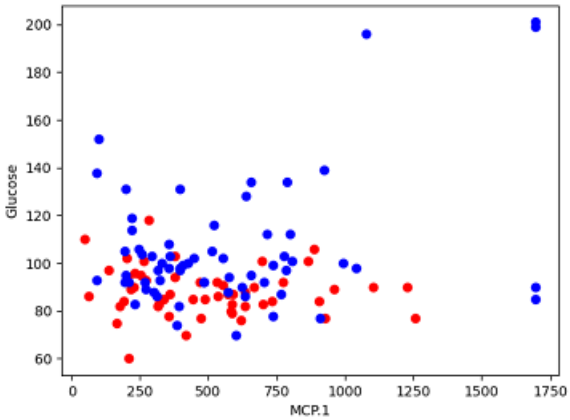


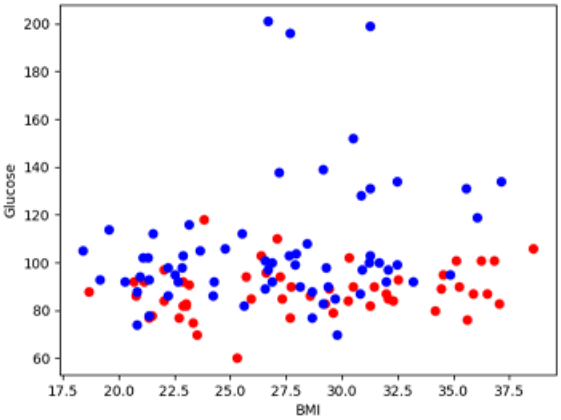

BMI
Age

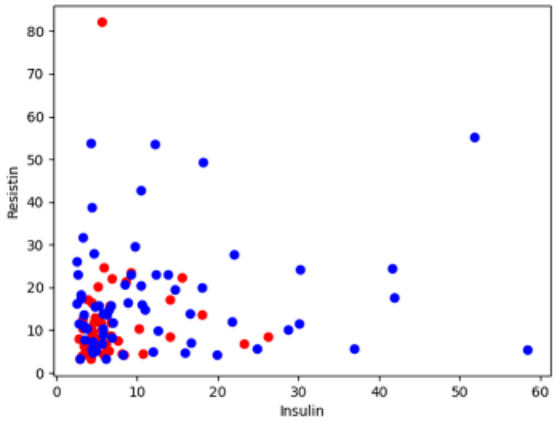

Adiponectin
BMI

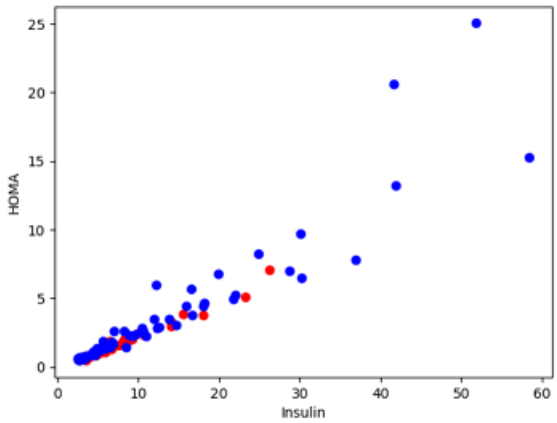

Adiponectin
MCP.1

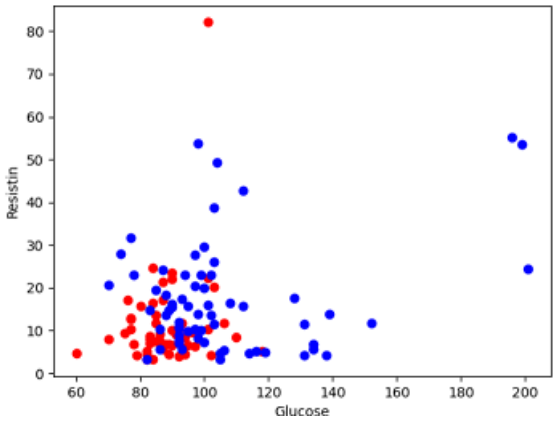

Leptin
BMI

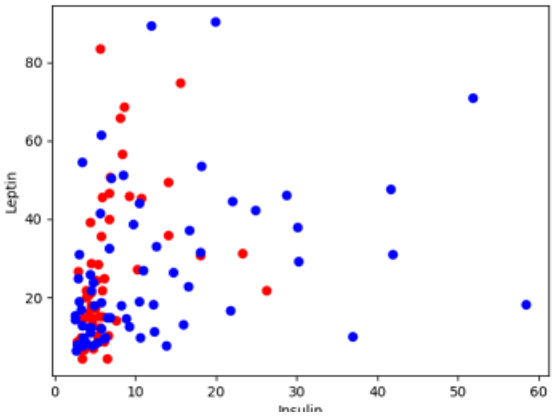

Glucose
MCP.1


Glucose
BMI









که با بررسی این نمودار ها هم به نتایج خاصی نرسیدیم چون که همانند نمونه های بالا هیچکدام از دسته توسط نه تنها یک خط راست بلکه به وسیله یک منحنی هم از یکدیگر جدا نمی شوند و همچنان ما باید با افزایش تعداد فیچر ها به یک نتیجه مطلوب تر برسیم
درضمن شایان ذکر است که به علت نزدیکی زیاد دسته ها به هم حتی الگوریتم svm با کرنل rbf هم نمی تواند نتیجه مطلوبی به وسیله دو فیچر بگیرد.

حال باز برای انتخاب فیچر های مناسب برای مدل از تکنیک های دیگری استفاده می کنیم:

1: با استفاده از میانگین و واریانس فیچر ها :

Age	57.301724
BMI	27.582111
Glucose	97.793103
Insulin	10.012086
HOMA	2.694988
Leptin	26.615080
Adiponectin	10.180874
Resistin	14.725966
MCP.1	534.647000
Classification	1.551724

میانگین

Age	259.621214
BMI	25.201763
Glucose	507.382909
Insulin	101.359945
HOMA	13.264479
Leptin	367.998771
Adiponectin	46.831322
Resistin	153.528100
MCP.1	119655.570601
Classification	0.249475

واریانس

با بررسی مقدار واریانس هر فیچر نسبت به میانگین آن می توان فهمید که فیچر مورد نظر تغییرات زیادی داشته یا نه

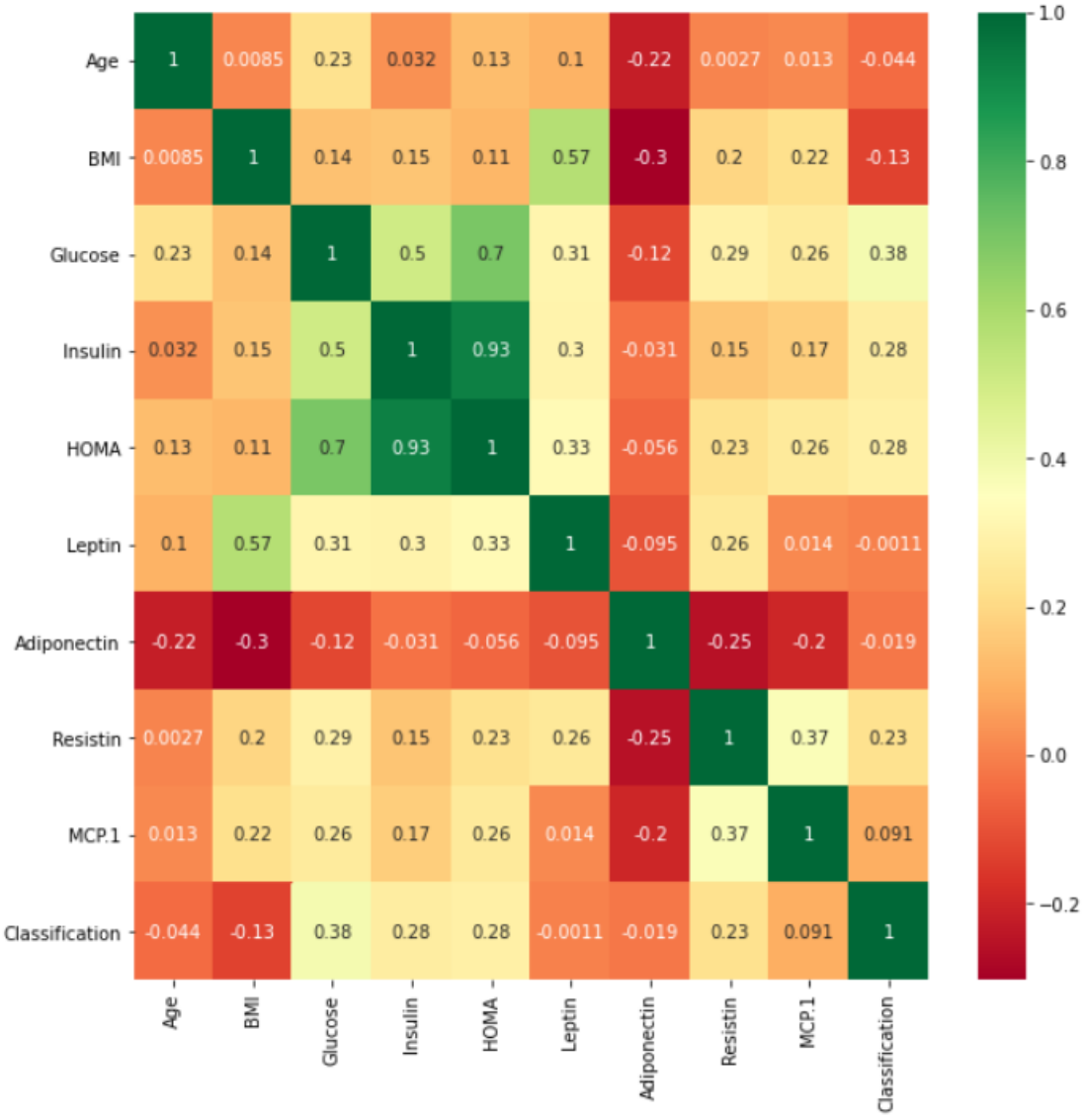
که اگر تغییرات یک پارامتر نسبت به میانگین آن کم باشد می توان نتیجه گرفت که بود و نبود آن پارامتر اهمیتی ندارد و می توان آن پارامتر را حذف کرد ولی با توجه به جدول های بالا مشاهده می شود که تمامی نسبت به میانگین خود تغییراتی زیادی دارند پس برای همین از این روش هم نمی توان داده های خاصی را انتخاب کرد.

2:استفاده از heatmap یا همان correlationمیان داده ها:

با استفاده از کتابخانه :

```
import seaborn as sns
```

heatmap داده ها را رسم می کنیم که به صورت زیر است:



که همانطور که مشاهده می شود در سطر مربوط به classification فیچر های

Age,BMI,glucose,HOMA,insulin,ressitin,MPC.1 ارتباط بیشتری با classification دارند.

ولی چون که در کل مقادیر corolation میان فیچر ها با classifiction خیلی زیاد نیست از روش های دیگر نیز استفاده می کنیم تا به دید بهتری نسبت به فیچر ها برسیم.

3:با استفاده از importace فیچرها که با استفاده از

from sklearn.ensemble import ExtraTreesClassifier

این مقادیر را بدست می آوریم

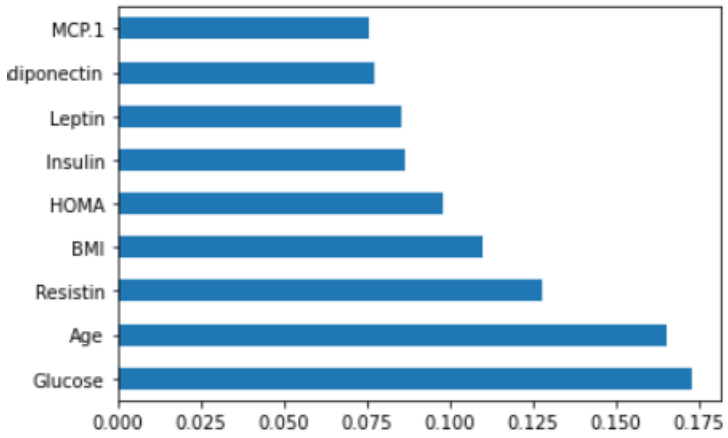
به صورت زیر:

```
In [6]: from sklearn.ensemble import ExtraTreesClassifier
        import matplotlib.pyplot as plt


        model=ExtraTreesClassifier()
        model.fit(x,y)

Out[6]: ExtraTreesClassifier()
```

```
In [7]: print(model.feature_importances_)

        [0.1655715  0.10980203 0.17319647 0.08679701 0.09825615 0.08557977
         0.07761674 0.12767257 0.07550776]
```

```
In [8]: feat_importances=pd.Series(model.feature_importances_, index = x.columns)
        feat_importances.nlargest(9).plot(kind='barh')
        plt.show()
```

که نمودار میله ای زیر حاصل قطعه کد بالا است:



که در این حالت هم فیچر هایی هستند که اهمیت خیلی زیادی دارند ولی در کل سایر فیچر ها هم از اهمیت قابل توجهی برخوردارند.

از این نمودار میتوان درک کرد که به نوعی تمام فیچر ها در دسته بندی تاثیر دارند وحذف بعضی از فیچر ها در میزان صحت مدل اهمیت خاصی نمی گذارد

4:حال از کلاس SelectKBest در کتاب خانه sklearn استفاده می کنیم که تا با استفاده از attribute score_ بتوان به نوعی دیگر فیچر ها را ارزیابی کرد.

که داریم:

```
In [2]: bestfeature=SelectKBest(score_func=chi2,k=6)
        fit=bestfeature.fit(x,y)

In [3]: dfscore=pd.DataFrame(fit.scores_)
        dfcolumns=pd.DataFrame(x.columns)

In [4]: featureScores=pd.concat([dfcolumns,dfscore],axis=1)
        featureScores.columns=['specs','score']
```

|:

	specs	score
0	Age	0.988417
1	BMI	1.847119
2	Glucose	88.125373
3	Insulin	89.203820
4	HOMA	45.656784
5	Leptin	0.001849
6	Adiponectin	0.200949
7	Resistin	61.949833
8	MCP.1	214.917039

حال در اینجا یک رتبه بندی دیگر برای فیچر ها داریم و می بنیم که بر این اساس فیچر های

MPC.1,Insulin.Glucose,HOMA بیشترین مقدار را دارند می توان بر این اساس همین 4 فیچر را انتخاب کرد .

البته در مقالاتی که لینکشان همراه دیتا (در سایت uci) بود ذکر شده بود که 4 فیچر

Age,BMI, ,Insulin,Glucose بهترین انتخاب از میان فیچر ها است.

البته یک را دیگر هم بود که همان استفاده از pca است که استفاده از این روش نتیجه خاص را به همراه نداشت .

چون که روش های بالا هیچکدام نتیجه خاصی را در پی نداشته اند برای همین یک روش کاملا غیر حرفه ای را دنبال کرده که آن هم امتحان کرده تمامی حالات ممکن از 9 فیچر است و برای هر الگوریتم تمامی حالات را امتحان کرده و هر کدام که نتیجه مطلوب تری داد را به عنوان فیچر های آن الگوریتم انتخاب کردم که حال برای تک تک الگوریتم ها روند را توضیح داده :

در ضمن ذکر شود که برای تعیین ترکیب مناسب برای هر فیچر به این صورت عمل شد که تمام ترکیب های ممکن از فیچر ها را (همه 511 حالت) هر یک را یک بار به عنوان X در نظر گرفته و مدل را بر این اساس فیت کرده و با روش k-fold صحت مدل را بدست آورده و چون که شافل کردن داده ها در مقدار صحت تاثیر گذار است پس برای بدست آوردن مقادیر پایدار 40 بار این روند را طی کرده و بعد از صحت این 40 مرحله میانگین گرفته تا به مقادیر نسبتا پایدار بتوان رسید.

توجه شود که در هر یک از نوتبوک های مدل ها چون که شافل کردن داده ها مقدار صحت را تغییر می دهد برای اینکه به مقدار پایداری برسیم برای هر مدل در یک حلقه 50 تایی هر بار داده ها را شافل کرده و براساس آن صحت مدل را با روش k-fold بدست آورده و صحت نهایی مدل را میانگین صحت در این 50 دفعه اعلام می کنیم

و همینطور در هر نوتبوک یک بخش مربوط به pca است که با اعمال pca مدل را ترین کرده که در هیچدکدام از حالات نتیجه مطلوبی از pcaنتیجه مطلوبی حاصل نشد.

در ضمن کد های مربوط به این فرایند در

نوتبوک embedded_feature_selection اجرا شده است

Logistic regression:

نوتبوک :logisticRegression_model

با استفاده از روش بالا بهترین فیچر های این الگوریتم به عبارت اند از :

'Age', 'Glucose', 'BMI', 'Resistin', 'HOMA'

که مدل هم به صورت زیر تعریف شده است:

logisticRegression_model=LogisticRegression(C=10)

که با این مقادیر به

این accuracy می رسیم:

```
acc:
0.7856723381809165(acc_train)
0.754996479854442(acc_test)
```

که به ترتیب اولی مربوط به داده های train و دومی مربوط به داده های test اند.

همینطور برای معیار f1 داریم:

```
f1
0.7976674461613084(f1_train)
0.7646592481451449(f1_test)
```

که در مورد مقدار C که ضریبregularization است

با تنظیم و دست کاری این پارامتر بهترین نتیجه برای مقدار 10 حاصل شده است.

البته pca را هم به هر 9 فیچر و هم 5 فیچر انتخاب شده اعمال کرده و نتیجه مطلوبی نگرفتم

SVM _with linear kernel:

نوتبوک:linear_svm_model.

برای این الگوریتم بهترین ترکیب فیچر ها به روش ذکر شده عبارت است از:

'Age', 'Glucose', 'BMI', 'Resistin', 'Insulin', 'HOMA'

که مدل هم به صورت زیر تعریف شده است:

linear_svm_model=svm.SVC(kernel='linear',C=5)

که با این مقادیر به

این accuracy می رسیم:

```
acc:
0.7853358220736889(acc_trian)
0.7550185786403091(acc_test)


f1
0.7946389467355844(f1_train)
0.7582259244780252(f1_test)
```

در اینجا هم بهترین مقدار C برابر با 5 است

SVM _with RBF kernel:

نوتبوک:rbf_svm_model

فیچر ها عبارت اند از:

'Age', 'Glucose', 'Resistin'

که مدل هم به صورت زیر تعریف شده است:

rbf_svm_model=svm.SVC(kernel='rbf',C=5)

که با این مقادیر به

این accuracy می رسیم:

```
acc:
0.8694163788406672(acc_trian)
0.8177426631905075(acc_test)

f1
0.8882297429014956(f1_trian)
0.8358510050084886(f1_test)
```

در اینجا هم بهترین مقدار C برابر با 5 است

decision tree:

نوتبوک:.tree_model

فیچر ها عبارت اند از:

'Age’, ‘Glucose’, ‘BMI’, ‘Resistin'

که مدل هم به صورت زیر تعریف شده است:

tree_model=tree.DecisionTreeClassifier(criterion='entropy')

که با این مقادیر به

این accuracy می رسیم:

```
acc:
1.0(acc_train)
0.7299578829026977(acc_test)

f1
1.0(f1_train)
0.7437491462845169(f1_test)
```

در این مدل criterion دو حالت دارد که که یکی gini است و دیگری entropy که Entropy مقدار تست بیشتری نسبت به gini دارد
و با max_depth=none هم پ acc_train برابر با 1 می شود

KNN:

نوتبوک:knn_model

فیچر ها عبارت اند از:

'Age', 'Glucose', 'BMI', 'Resistin'

که مدل هم به صورت زیر تعریف شده است:

knn_model=KNeighborsClassifier(n_neighbors=5,weights='distance')

که با این مقادیر به

این accuracy می رسیم:

```
acc:
1.0(acc_train)
0.8010751664484228(acc_test)


f1
1.0(f1_train)
```

0.8283964767173628(f1_test)

که چون که تعداد دسته های خروجی مدل 2 تا (زوج) است پس بهتر است که برای n_neighbor یک مقدار فرد انتخاب کنیم و مقدار 5 بهترین مقدار ممکن برای این مدل است(بعد امتحان کردن های مختلف) و برای weights هم به ازای مقدار distance به جای uniform مقدار acc_train بیشتری دارد نسبت به uniform