دانشکده علوم ریاضی و آمار

پیش بینی بیماری قلبی

پروژه درس یادگیری ماشین

استاد

دکتر حسین حاجی ابولحسن

دانشجو

عارف عابد

تابستان ۱۴۰۰

چکیده

داده های قلب از سایت کگل اخذ شده است. در این پروژه هدف پیش بینی بیماری قلبی براساس اطلاعات موجود است.این گزارش در چهار بخش نوشته شده است:

- آماده سازی داده
- برازش مدل ها
- Ensemble
- مقایسه مدل ها و انتخاب بهترین مدل

داده دارای ۱۴ متغیر است.جدول زیر خلاصه ای از اطلاعات متغیرهایی که در داده قرار گرفته اند ارائه میدهد.

نام متغیر	نوع متغیر	توضیحات
age	گسسته	سن
cp	گسسته	نوع درد قفسه سینه
trestbps	پیوسته	فشارخون در حالت استراحت(در میلی متر جیوه)
chol	پیوسته	گلسترول خون
fbs	گسسته	قند خون درحالت ناشتا
restecg	گسسته	نتایج الکتروکاردیوگرافی درحالت استراحت
thalach	پیوسته	حداکثر ضربان قلب بدست آمده
exang	گسسته	آنژین ناشی از ورزش
oldpeak	پیوسته	افسردگی القا شده
slope	گسسته	شیب پیک ورزش
ca	گسسته	تعداد عروق قلبی
target	گسسته	به وجود بیماری قلبی اشاره می کند

فهرست مطالب

۱ آماده سازی داده

۱-۱ پاکسازی داده

در بررسی داده موجود ،به دلیل ناهمخوانی اطلاعات ستون thal با تعریف ستون ،

One Hot Encoder انجام دادیم.بنابراین ستون thal با سه ستون یک، دو، سه جایگزین شده است.

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	target	1	2	3
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1	0	0
1	37	1	2	130	250	0	1	187	0	3.5	0	0	1	0	1	0
2	41	0	1	130	204	0	0	172	0	1.4	2	0	1	0	1	0
3	56	1	1	120	236	0	1	178	0	0.8	2	0	1	0	1	0
4	57	0	0	120	354	0	1	163	1	0.6	2	0	1	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	0	0	0	1
299	45	1	3	110	264	0	1	132	0	1.2	1	0	0	0	0	1
300	68	1	0	144	193	1	1	141	0	3.4	1	2	0	0	0	1
301	57	1	0	130	131	0	1	115	1	1.2	1	1	0	0	0	1
302	57	0	1	130	236	0	0	174	0	0.0	1	1	0	0	1	0

۱-۲ بررسی داده های گمشده

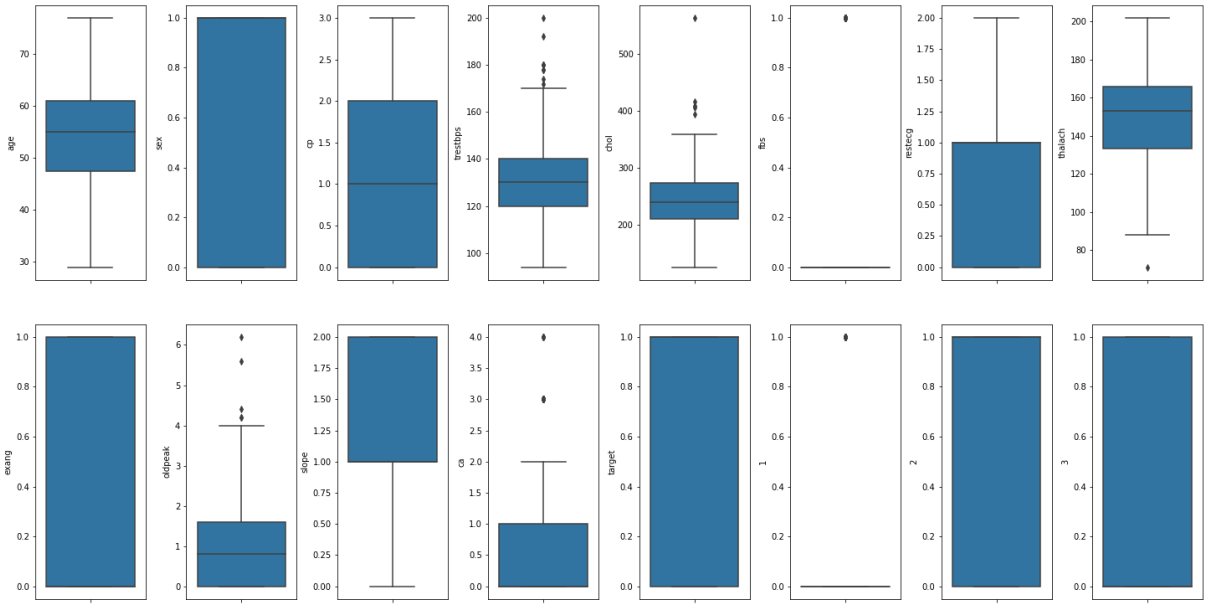
همه داده های گمشده در ستون ها را بررسی می کنیم:

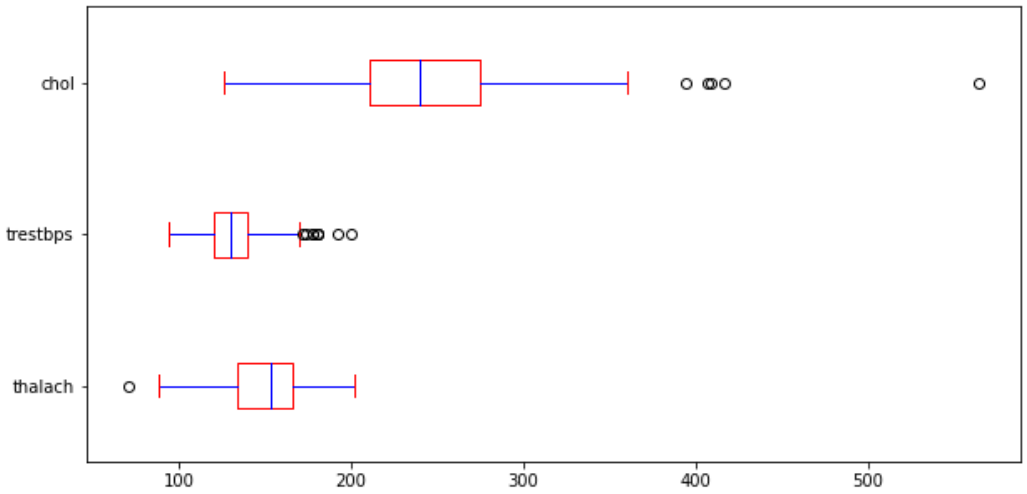
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
target      0
1           0
2           0
3           0
dtype: int64
```

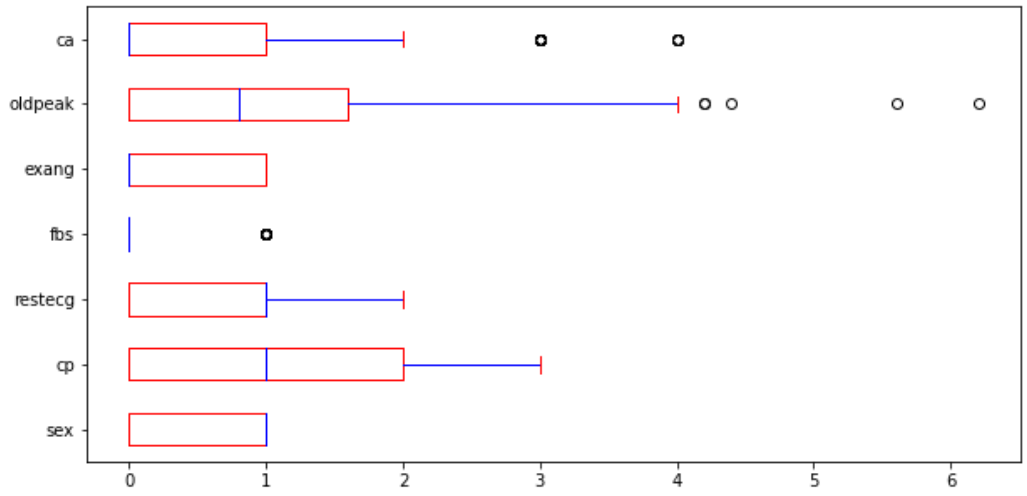
که متوجه میشویم هیچ داده گمشده ای در ستون ها وجود ندارد.

۱-۳ بررسی داده های پرت

همه ستون ها را در نمودار باکس پلات رسم می کنیم:



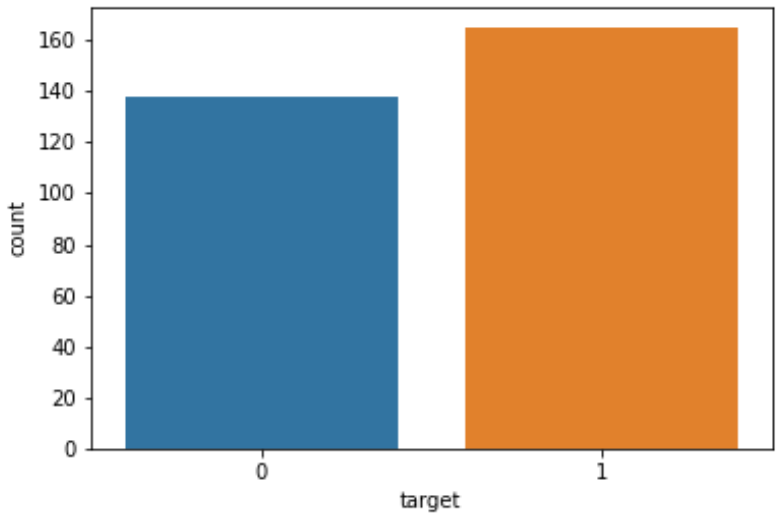




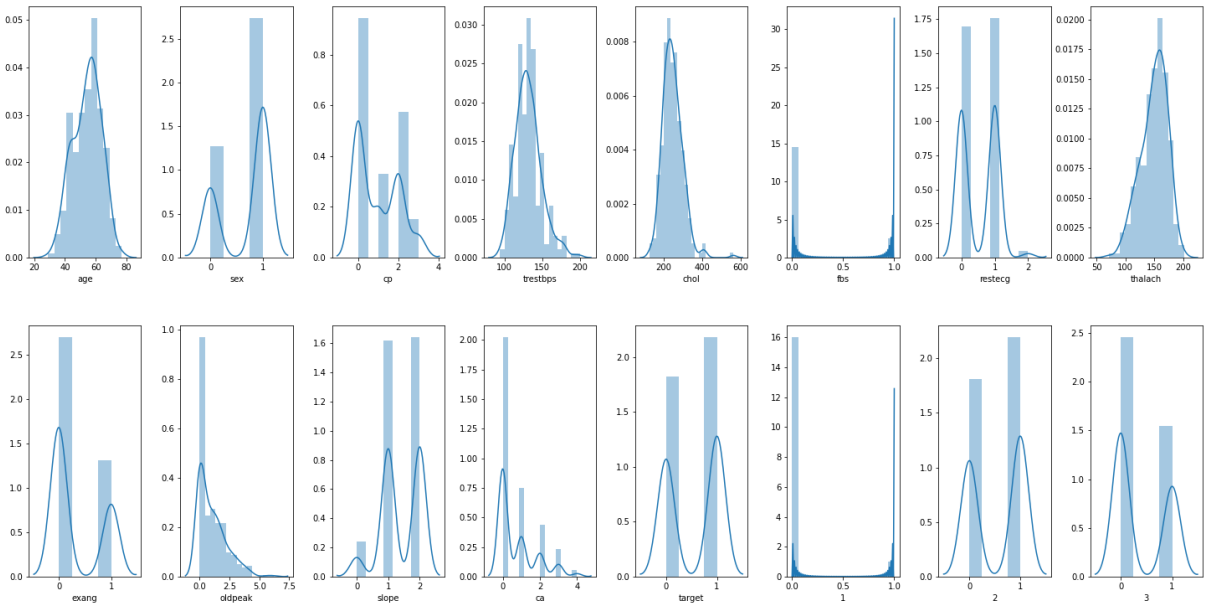
با توجه به بررسی اطلاعات موجود از هیچ داده ای صرف نظر نکردیم.

۱-۴ تصویری سازی داده

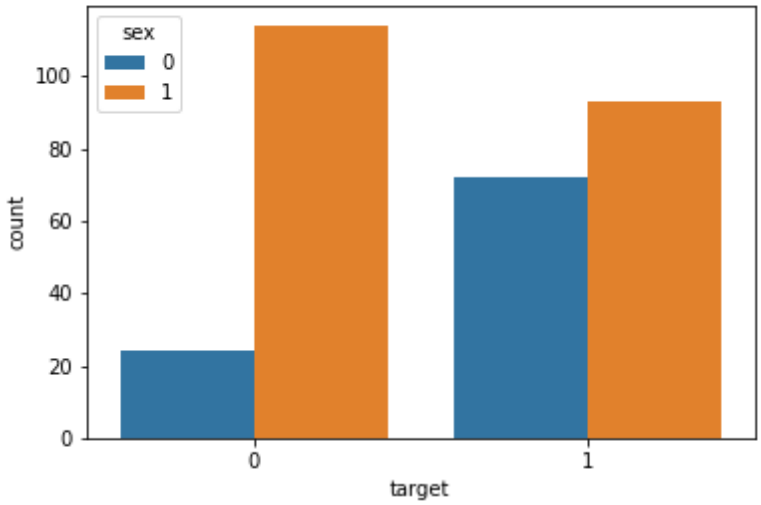
در بخش تصویری سازی،ابتدا نمودارهای تک متغیره سپس نمودارهای دو متغیره و سه متغیره بررسی می شوند.

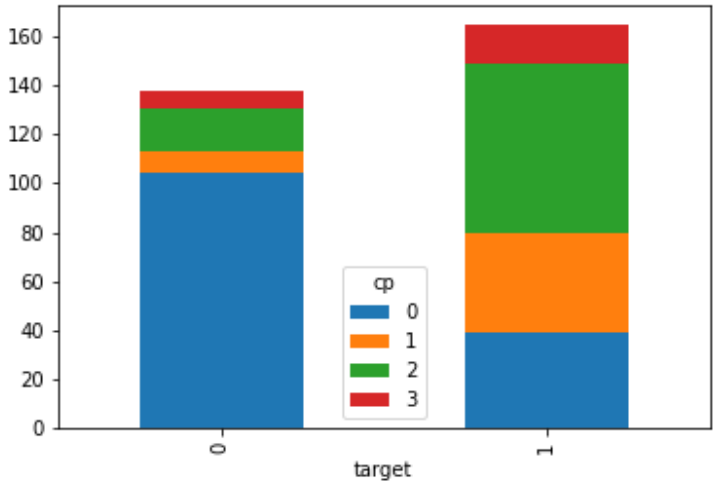


مشاهده می شود افرادی که دچار حمله قلبی شده اند بیشتر است.

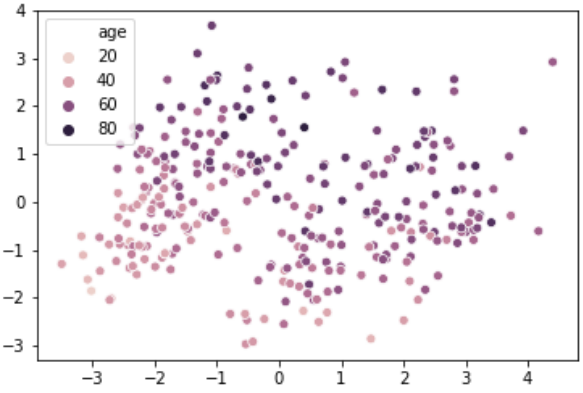


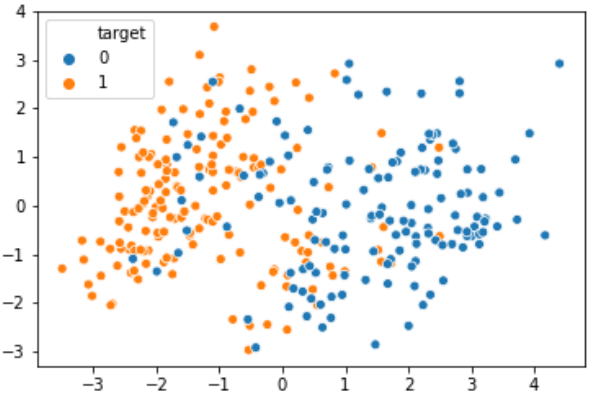
این نمودار به ما نشان می دهد که کدام ستون ها دارای توزیع نرمال هستند و کدام ستون ها توزیع شبه نرمال دارند.





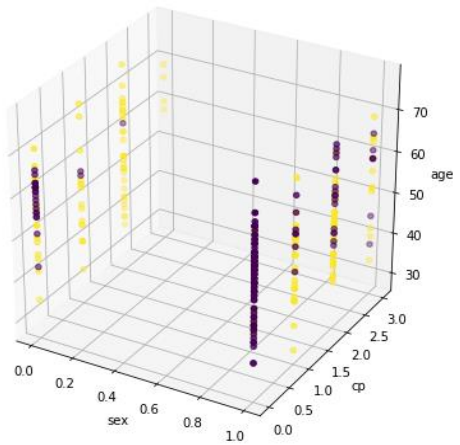
مشاهده می شود افرادی که درد قفسه سینه ندارند و مورد حمله قلبی قرار نگرفتن و افرادی که دارای درد قفسه سینه حالت دو هستند و دچار حمله قلبی شدند، از بقیه بیشتر است.



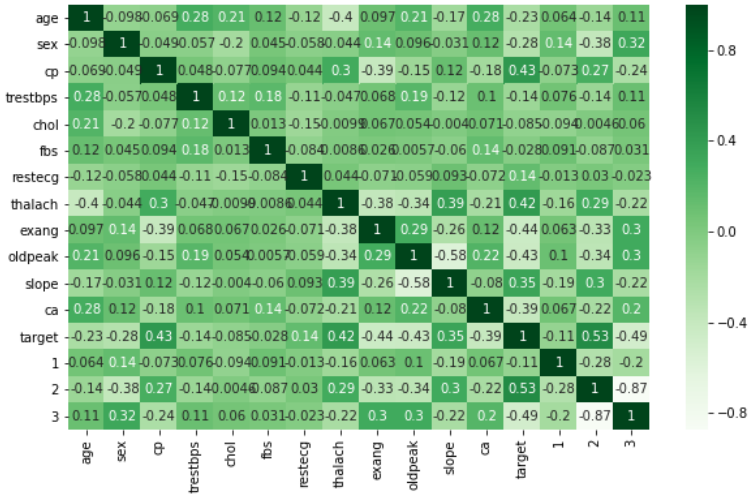


در این دو نمودار ابتدا pca انجام دادیم و داده ۱۳ بعدی را به ۲ بعد کاهش دادیم. رنگ آمیزی نمودار اول براساس age و نمودار دوم براساس target است.

در نمودار دوم هرچه به سمت راست میرویم افرادی که مورد حمله قلبی قرار نگرفتن بیشتر می شود.



این نمودار نشان میدهد آقایانی که درد قفسه سینه ندارند در تمامی سنین دچار حمله قلبی نشدند، خانم هایی که درد قفسه سینه آنها در حالت یک و دو قراردارد تقریبا در تمام سنین دچار حمله قلبی شده اند.



در نمودار correlation بالا ملاحضه می شود که ستون target با هیچ ستونی همبستگی خطی ندارد.

۲ برازش مدل ها بر داده

۲-۱ آماده سازی داده

اکنون داده ها را به دو دسته داده train و داده test تقسیم می کنیم و مدل ها را با داده train آموزش می دهیم.داده train شامل ۲۱۲ نمونه و داده test شامل ۹۱ نمونه می باشد.

ابتدا روی هر مدل GridSearchCV می زنیم تا بهترین پارامترهای مدل را پیدا کنیم سپس با پارامتر های بدست آمده مدل را اجرا می کنیم.

۲-۲ برازش مدل لجستیک رگرسیون

ابتدا نتایج GridSearchCV را روی مدل لجستیک رگرسیون می بینیم :

```
{'max_iter': 10000, 'penalty': 'none', 'solver': 'sag'}
Accuracy : 0.8588039867109634
```

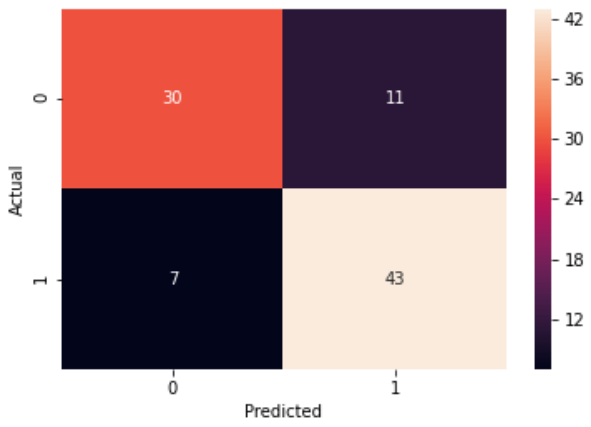
اکنون مدل را اجرا می کنیم :

```
Accuracy test  : 0.8021978021978022
Accuracy train : 0.8915094339622641
```

با دستور classification_report نتایج زیر حاصل می شود :

```
  precision    recall  f1-score   support

          0       0.81      0.73      0.77        41
          1       0.80      0.86      0.83        50

    accuracy                           0.80        91
   macro avg       0.80      0.80      0.80        91
weighted avg       0.80      0.80      0.80        91
```

در ادامه confusion_matrix مدل هم حساب می کنیم که در نمودار پایین آورده شده است :



ملاحضه می شود که از ۹۱ نمونه داده test، ۷۳ نمونه به درستی پیش بینی می شوند.

۳-۲ برازش مدل درخت تصمیم

ابتدا نتایج GridSearchCV را روی مدل درخت تصمیم می بینیم :

```
{'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 9,
'min_samples_split': 2}
Accuracy : 0.8403100775193799
```

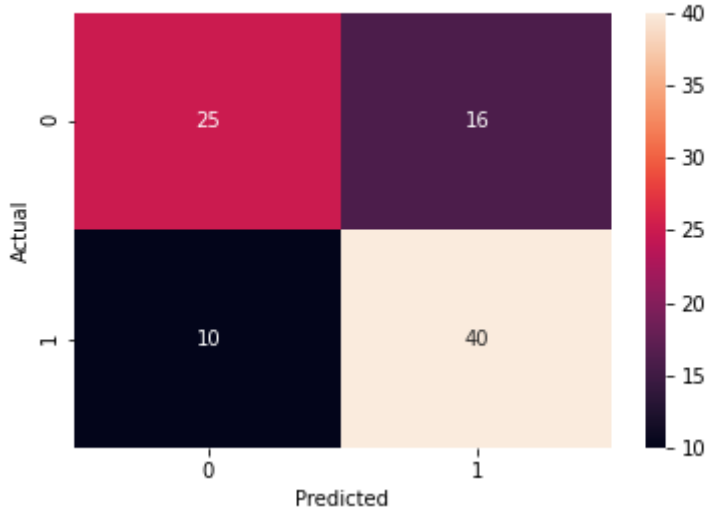
با پارامترهای بدست آمده و با پارامتر الفا که در ادامه توضیح می دهیم، مدل را اجرا می کنیم :

```
Accuracy test  : 0.7142857142857143
Accuracy train : 0.8584905660377359
```

با دستور classification_report نتایج زیر حاصل می شود :

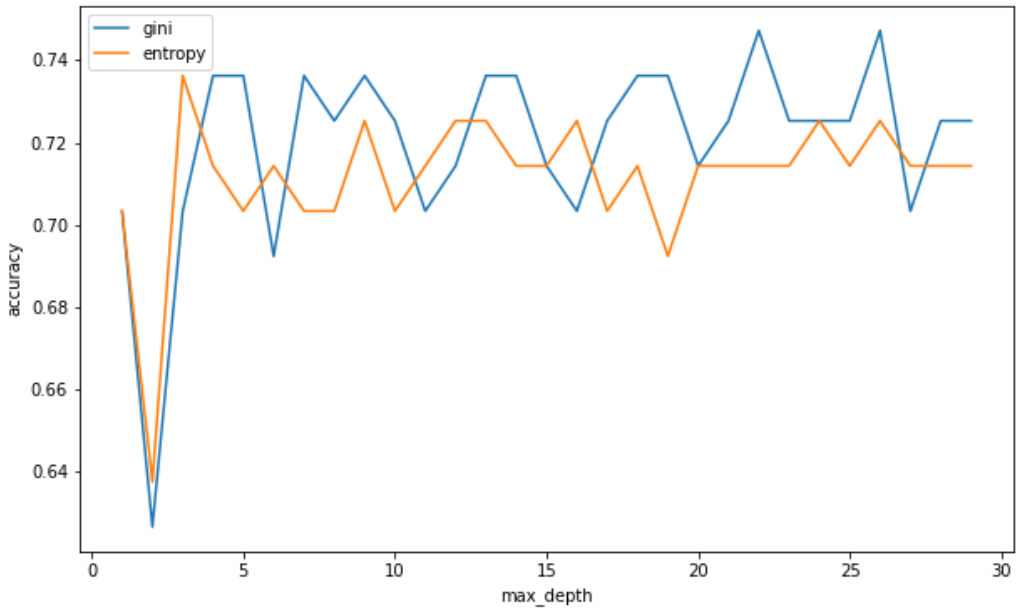
```
              precision    recall  f1-score   support

           0       0.71      0.61      0.66        41
           1       0.71      0.80      0.75        50

    accuracy                           0.71        91
   macro avg       0.71      0.70      0.71        91
weighted avg       0.71      0.71      0.71        91
```

در ادامه confusion_matrix مدل را حساب می کنیم که در نمودار پایین آورده شده است :

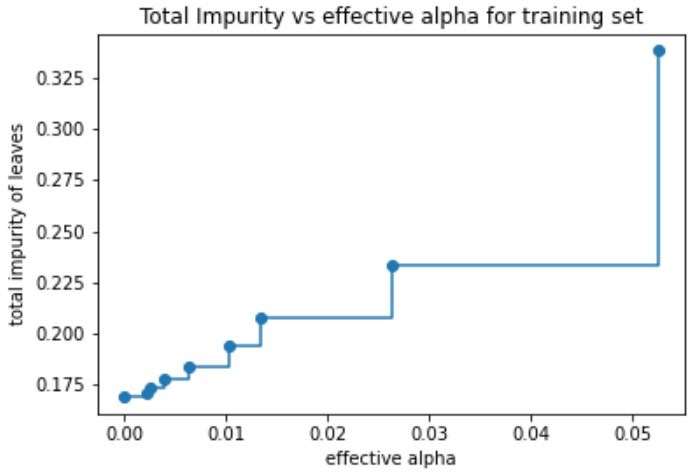


ملاحضه می شود که از ۹۱ نمونه داده test، ۶۵ نمونه به درستی پیش بینی می شوند.

نمودار max_Depth و Accuracy criterion های مدل درخت تصمیم را رسم می کنیم :

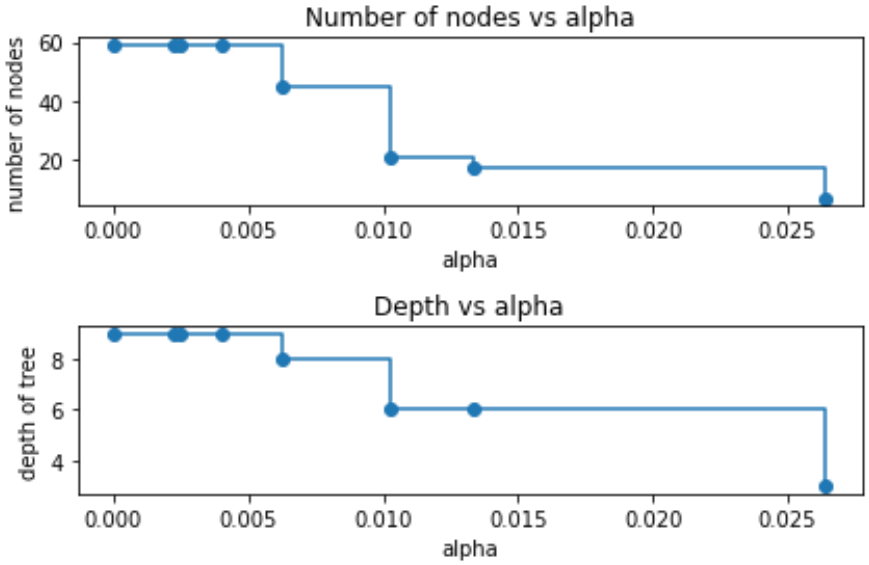


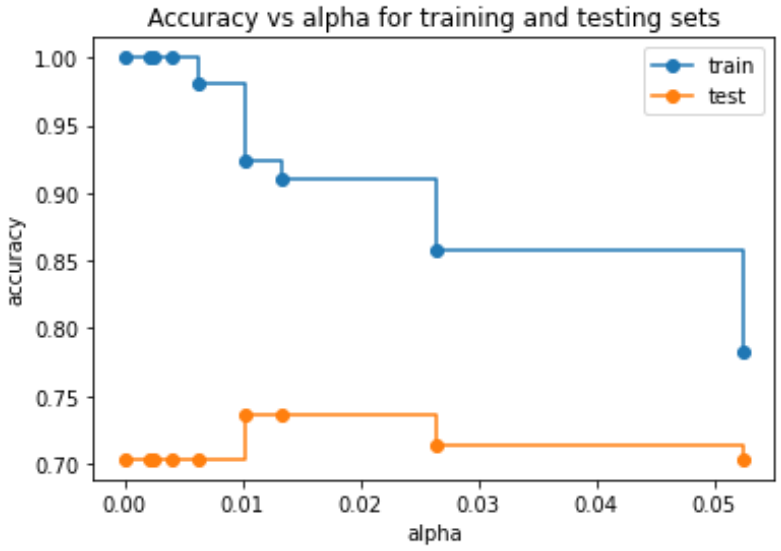
برای اینکه overfitting در مدل را کم کنیم از پارامتر الفا استفاده می کنیم و مدل را هرس می کنیم :



ملاحظه می شود که با افزایش الفا، ناخالصی در برگ ها افزایش میابد.

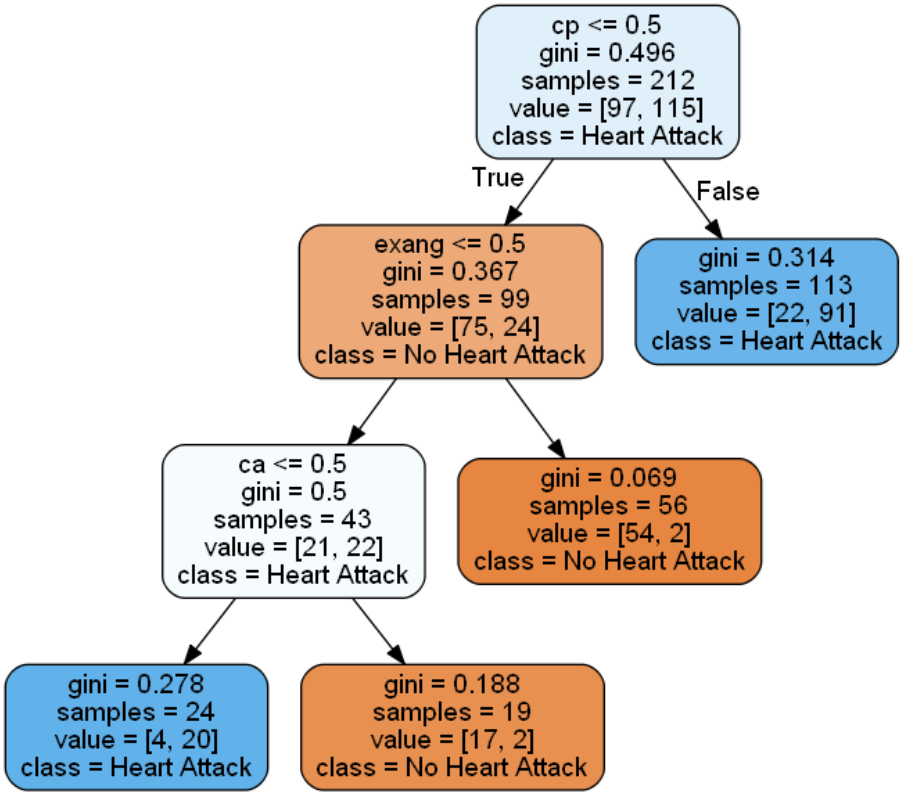
و در نمودارهای زیر مشاهده می کنیم که با افزایش الفا تعداد راس و عمق درخت کم می شود.





دقت داده train و داده test را با الفاهای مختلف میبینیم و بهترین الفا را انتخاب می کنیم.

تصویر مدل درخت تصمیم را می بینیم :



۲-۴ برازش مدل جنگل تصادفی

ابتدا نتایج GridSearchCV را روی مدل درخت تصمیم می بینیم :

```
{'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 2, 'min_sampl
es_split': 4, 'n_estimators': 21}
0.8781838316722037
```

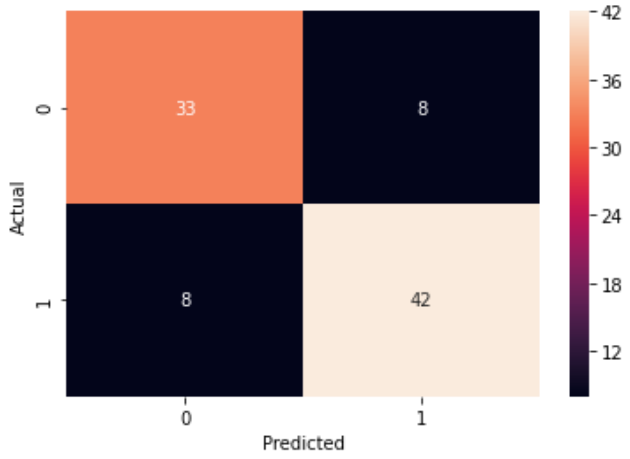
با پارامترهای بدست آمده ، مدل را اجرا می کنیم :

```
Accuracy test  : 0.8241758241758241
Accuracy train : 0.8632075471698113
```

با دستور classification_report نتایج زیر حاصل می شود :

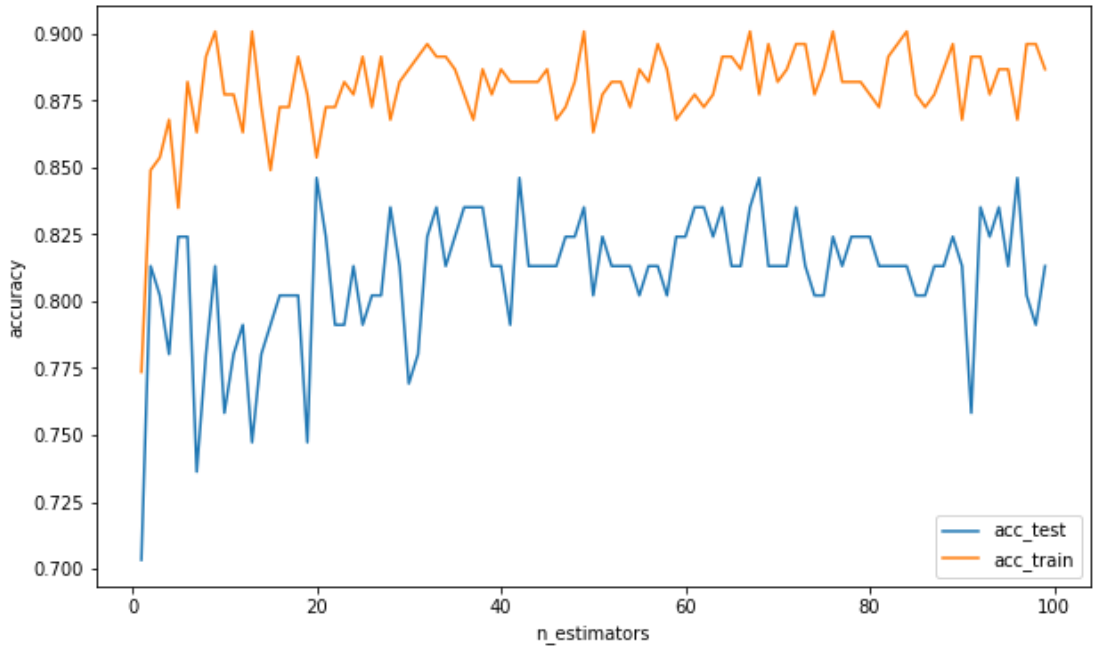
	precision	recall	f1-score	support
0	0.80	0.80	0.80	41
1	0.84	0.84	0.84	50
accuracy			0.82	91
macro avg	0.82	0.82	0.82	91
weighted avg	0.82	0.82	0.82	91

در ادامه confusion_matrix مدل را حساب می کنیم که در نمودار پایین آورده شده است :



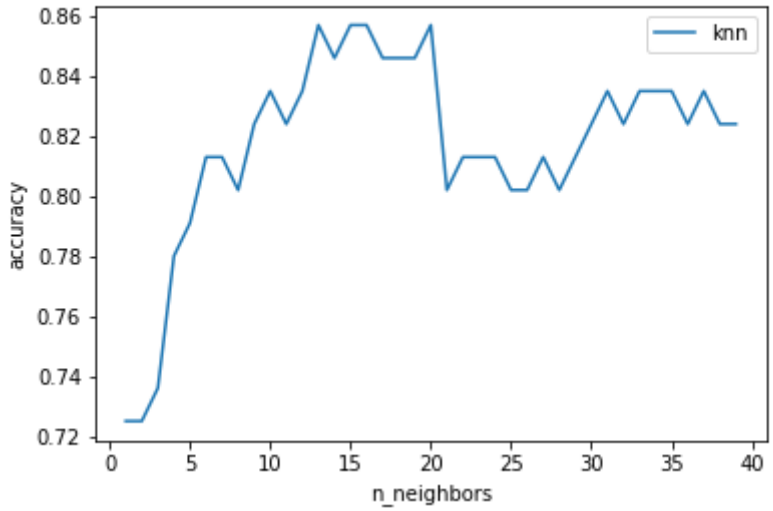
ملاحضه می شود که از ۹۱ نمونه داده test، ۷۵ نمونه به درستی پیش بینی می شوند.

با n_estimator های مختلف دقت های داده train و داده test را در نمودار زیر می بینیم :

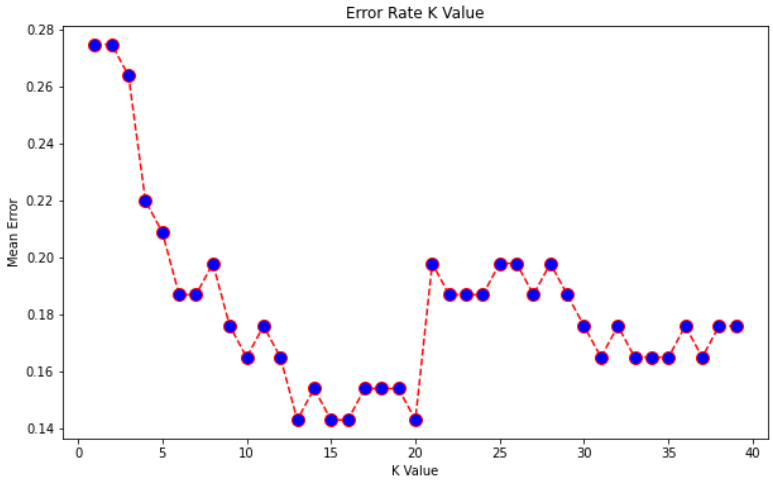


۵-۲ برازش مدل k نزدیک ترین همسایه

برای مدل Knn ابتدا داده ها را نرمالایز می کنیم سپس نمودار دقت به ازای n_neighbor های مختلف را رسم می کنیم :



نمودار Error Rate به ازای مقادیر K را رسم می کنیم :



با n_neighbor = ۱۵ ، مدل را اجرا می کنیم :

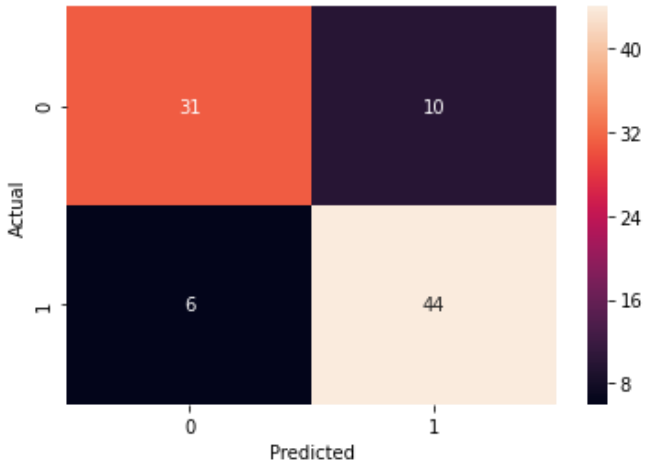
```
Accuracy test   : 0.8571428571428571
Accuracy train  : 0.8443396226415094
```

با دستور classification_report نتایج زیر حاصل می شود :

```
              precision    recall  f1-score   support

           0       0.84      0.76      0.79        41
           1       0.81      0.88      0.85        50

    accuracy                           0.82        91
```

```
   macro avg       0.83      0.82      0.82        91
weighted avg       0.83      0.82      0.82        91
```

در ادامه confusion_matrix مدل را حساب می کنیم که در نمودار پایین آورده شده است :



مشاهده می شود که مدل، ۷۵ نمونه را به درستی پیش بینی کرده است.

۲-۶ برازش مدل perceptrone

ابتدا نتایج GridSearchCV را روی مدل می بینیم :

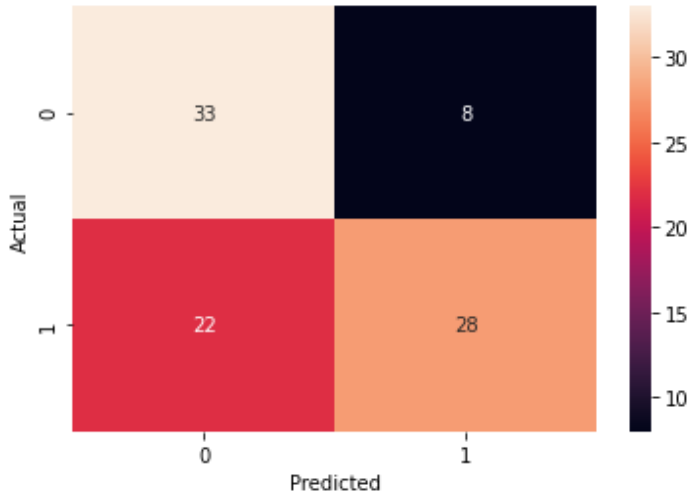
```
{'alpha': 0.01, 'early_stopping': False, 'penalty': 'l1'}
0.6282392026578073
```

با پارامترهای بدست آمده ، مدل را اجرا می کنیم :

```
Accuracy test   : 0.6703296703296703
Accuracy train  : 0.6886792452830188
```

confusion_matrix ,classification_report این مدل را مشاهده می کنیم :

	precision	recall	f1-score	support
0	0.60	0.80	0.69	41
1	0.78	0.56	0.65	50
accuracy			0.67	91
macro avg	0.69	0.68	0.67	91
weighted avg	0.70	0.67	0.67	91



مشاهده می شود که مدل، ۶۱ نمونه را به درستی پیش بینی کرده است.

۲-۷ برازش مدل شبکه عصبی

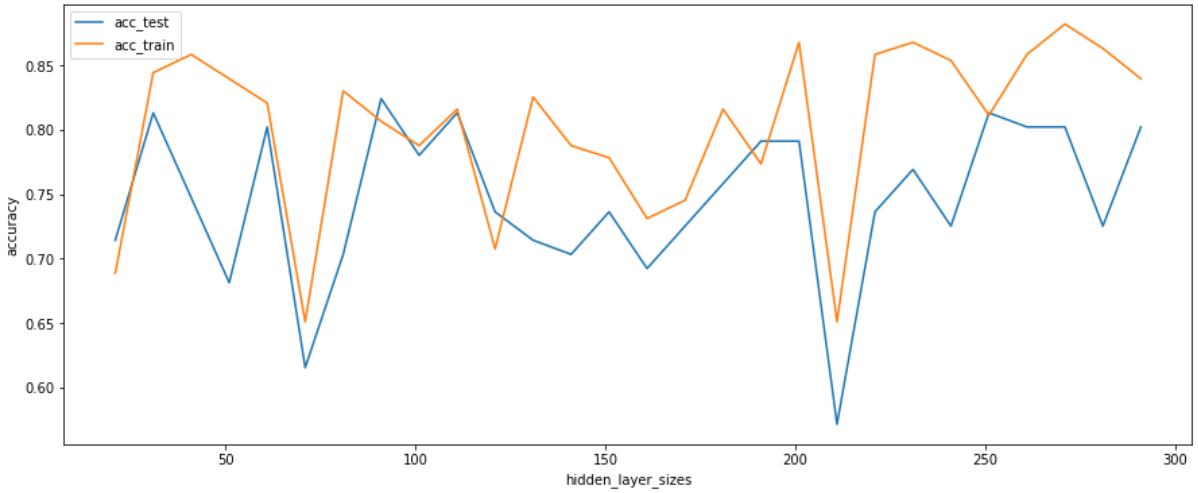
ابتدا نتایج GridSearchCV را روی مدل شبکه عصبی می بینیم :

```
{'activation': 'identity', 'hidden_layer_sizes': 252}
0.8589147286821707
```

با پارامترهای بدست آمده ، مدل را اجرا می کنیم :

```
Accuracy test   : 0.8131868131868132
Accuracy train  : 0.8113207547169812
```

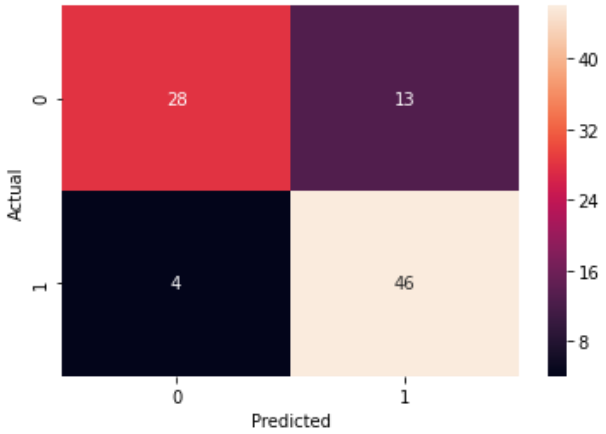
نمودار دقت test و train مدل، به ازای لایه های پنهان مختلف را رسم می کنیم :



مشاهده می شود که اگر لایه پنهان ۲۵۱ قرارگیرد،دقت train و test برابر می شود.

confusion_matrix ,classification_report این مدل را مشاهده می کنیم :

	precision	recall	f1-score	support
0	0.88	0.68	0.77	41
1	0.78	0.92	0.84	50
accuracy			0.81	91
macro avg	0.83	0.80	0.81	91
weighted avg	0.82	0.81	0.81	91



مشاهده می شود که مدل، ۷۴ نمونه را به درستی پیش بینی کرده است.

۸-۲ برازش مدل ماشین بردار پشتیبان

ابتدا نتایج GridSearchCV را روی مدل ماشین بردار پشتیبان می بینیم :

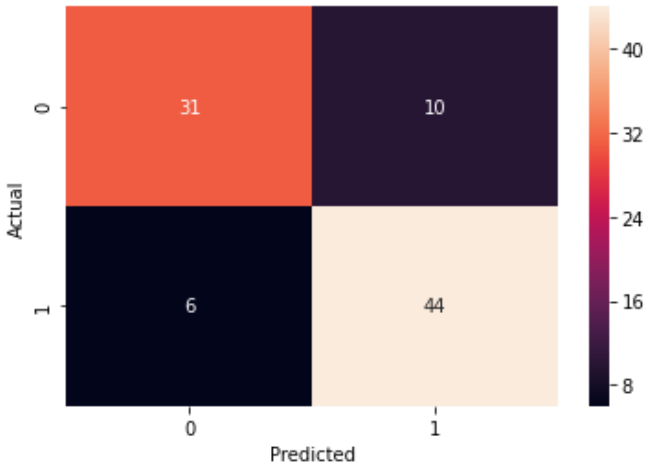
```
{'C': 2, 'degree': 1, 'kernel': 'linear'}
0.8495016611295683
```

با پارامترهای بدست آمده ، مدل را اجرا می کنیم :

```
Accuracy test   : 0.8241758241758241
Accuracy train  : 0.8726415094339622
```

confusion_matrix ,classification_report این مدل را مشاهده می کنیم :

```
              precision    recall  f1-score   support

           0       0.84      0.76      0.79        41
           1       0.81      0.88      0.85        50

    accuracy                           0.82        91
   macro avg       0.83      0.82      0.82        91
weighted avg       0.83      0.82      0.82        91
```



ملاحظه می شود که مدل، ۷۵ نمونه را به درستی پیش بینی کرده است.

۳ Ensemble

در این قسمت به مدل هایی که دقت پایینی دارند می پردازیم و آن مدل ها را با Ensemble های مختلف امتحان می کنیم.

۳-۱ AdaBoost

ابتدا روی AdaBoost ، GridSearchCV می زنیم و نتایج را در زیر می بینیم :

```
{'base_estimator': LogisticRegression(max_iter=1000, random_state=1), '
n_estimators': 11}
0.8398671096345517
```

مشاهده می شود که n_estimators ،۱۱ و base_estimator ، لجستیک رگرسیون می باشد.

در ادامه به ترتیب روی مدل های لجستیک رگرسیون،درخت تصمیم و پرسپترون AdaBoost می زنیم.

لجستیک رگرسیون :

```
Accuracy AdaBoost LR test  : 0.7912087912087912
Accuracy LR test           : 0.8021978021978022
Accuracy AdaBoost LR train : 0.8726415094339622
Accuracy LR train          : 0.8915094339622641
```

درخت تصمیم :

```
Accuracy AdaBoost DT test  : 0.7252747252747253
Accuracy DT test           : 0.7142857142857143
Accuracy AdaBoost DT train : 0.8632075471698113
Accuracy DT train          : 0.8584905660377359
```

پرسپترون :

```
Accuracy AdaBoost PRC test  : 0.6923076923076923
Accuracy PRC test           : 0.6703296703296703
Accuracy AdaBoost PRC train : 0.7169811320754716
Accuracy PRC train          : 0.6886792452830188
```

مشاهده می کنیم که AdaBoost روی دقت مدل ها تاثیر زیادی نداشته است.

۳-۲ Bagging

ابتدا روی Bagging ، GridSearchCV می زنیم و نتایج را در زیر می بینیم :

```
{'n_estimators': 61}
0.8730897009966778
```

ملاحظه می شود که n_stimators مطلوب، ۶۱ می باشد.

در ادامه به ترتیب روی مدل های درخت تصمیم و پرسپترون Bagging می زنیم.

درخت تصمیم :

```
Accuracy Bagging DT test  : 0.7032967032967034
Accuracy DT test          : 0.7142857142857143
Accuracy Bagging DT train : 0.8490566037735849
Accuracy DT train         : 0.8584905660377359
```

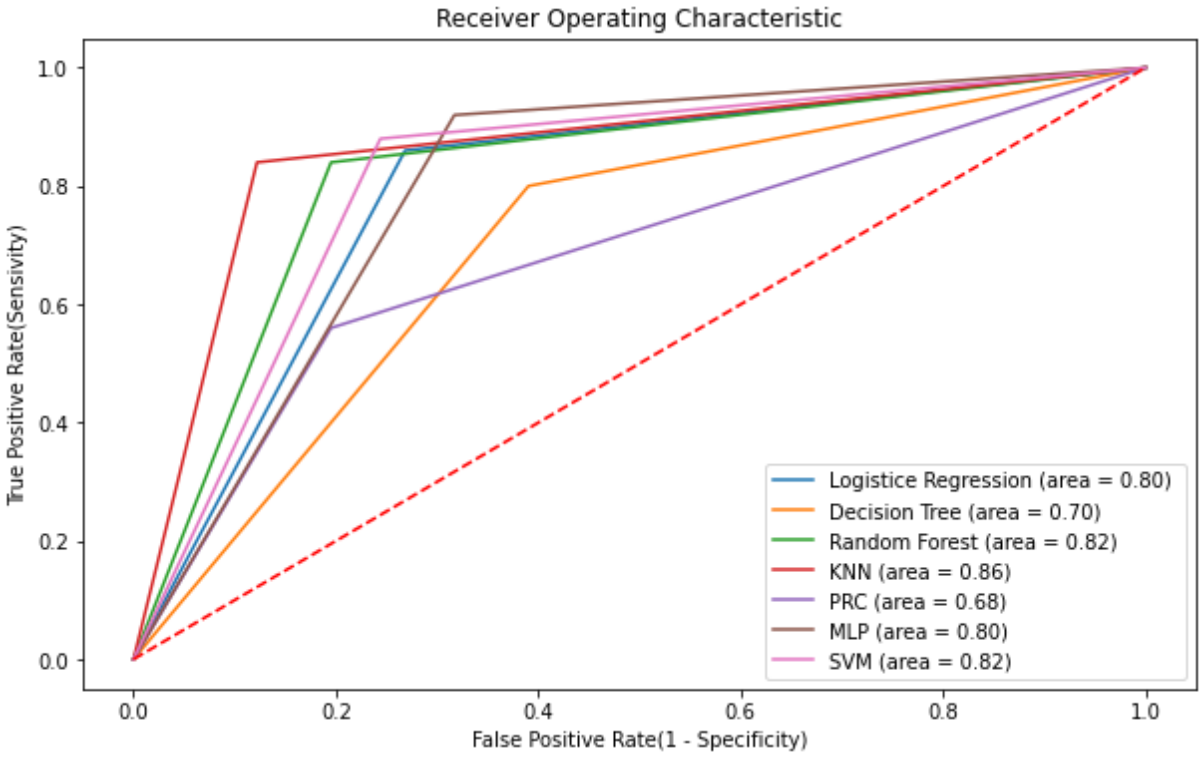
پرسپترون :

```
Accuracy Bagging PRC test  : 0.6813186813186813
Accuracy PRC test          : 0.6703296703296703
Accuracy Bagging PRC train : 0.6933962264150944
Accuracy PRC train         : 0.6886792452830188
```

مشاهده می کنیم که Bagging روی دقت مدل ها تاثیر زیادی نداشته است.

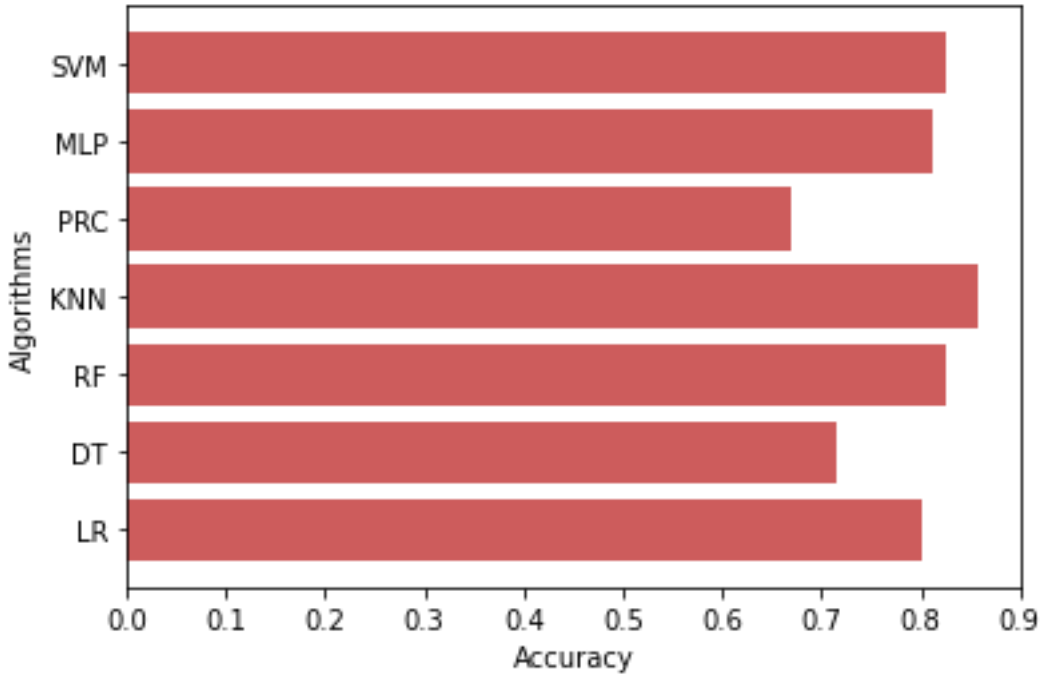
۴ مقایسه مدل ها و انتخاب بهترین مدل

۴-۱ نمودار ROC مدل ها



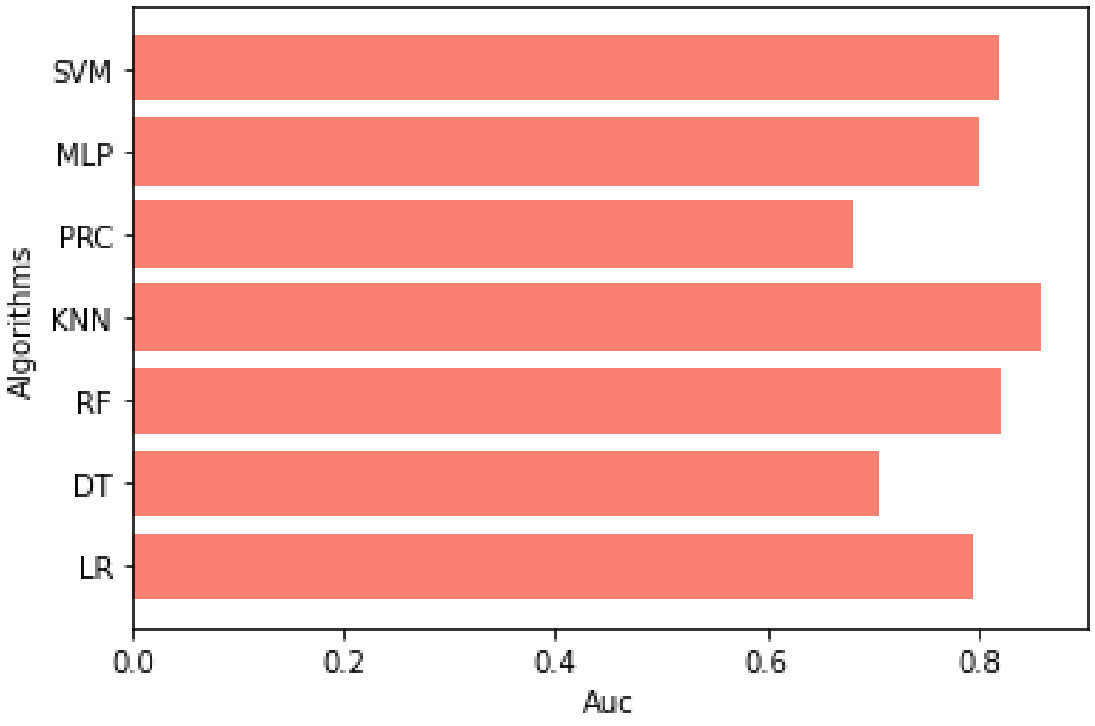
مشاهده می شود که K نزدیک ترین همسایه بیشترین مساحت را داراست.

۴-۲ نمودار دقت مدل ها



در این نمودار مشاهده می شود که دقت مدل k نزدیک ترین همسایه از دیگر مدل ها بیشتر است.

۴-۳ نمودار AUC مدل ها



در این نمودار ملاحظه می شود که مساحت زیر نمودار ROC، K نزدیک ترین همسایه از دیگر مدل ها بیشتر است.

۴-۴ نتیجه گیری

	Algorithms	Accuracy	Auc
0	LR	0.802198	0.795854
1	DT	0.714286	0.704878
2	RF	0.824176	0.822439
3	KNN	0.857143	0.859024
4	PRC	0.670330	0.682439
5	MLP	0.813187	0.801463
6	SVM	0.824176	0.818049

طبق دیتا فریم بالا، K نزدیک ترین همسایه از دیگر مدل ها دارای دقت و Auc بالاتری می باشد.