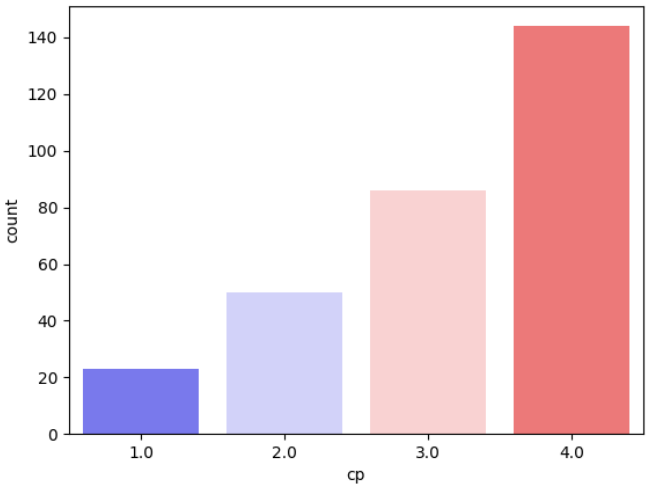
گزارش کار تمرین سوم یادگیری ماشین

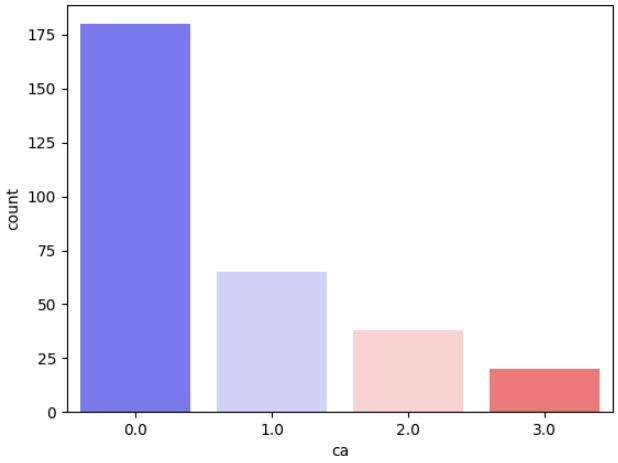
بیتا آذری جو ۸۳۰۵۹۸۰۰۷

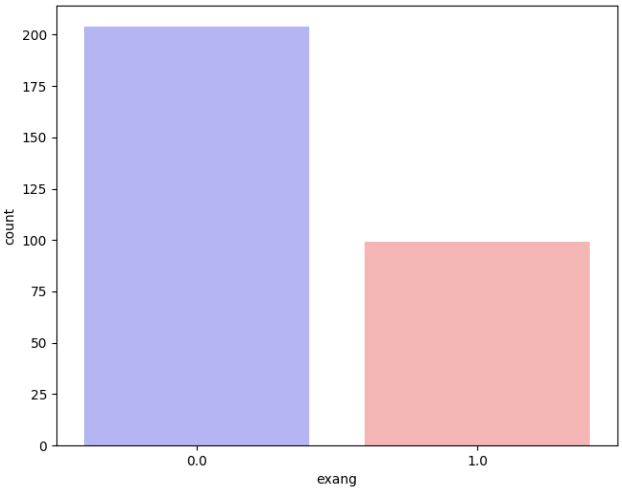
قسمت اول : رسم توزیع ویژگی ها

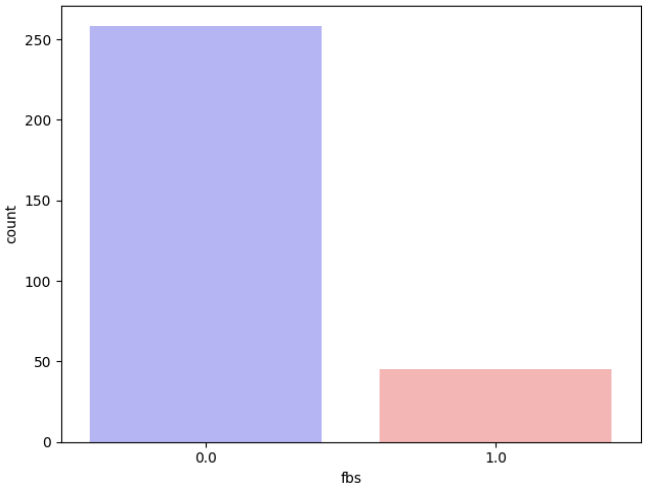
برای هر ویژگی در صورتی که گسسته با بازه ی عددی کم بود نمودار فراوانی رسم شده و احتمال وقوع هر کدام از متغیر تصادفی موجود در فضای نمونه برابر با تعداد نمونه هایی که آن مقدار متغیر تصادفی را دارند تقسیم بر کل نمونه ها.

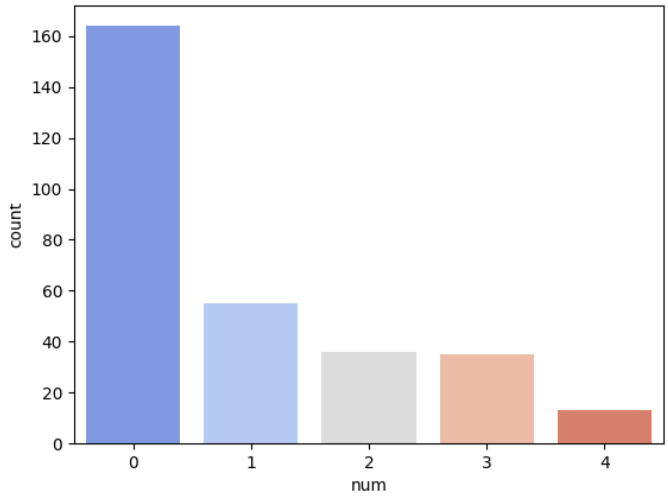
برای سایرویژگی ها نمودار توزیع روی هیستوگرام fit شده است.

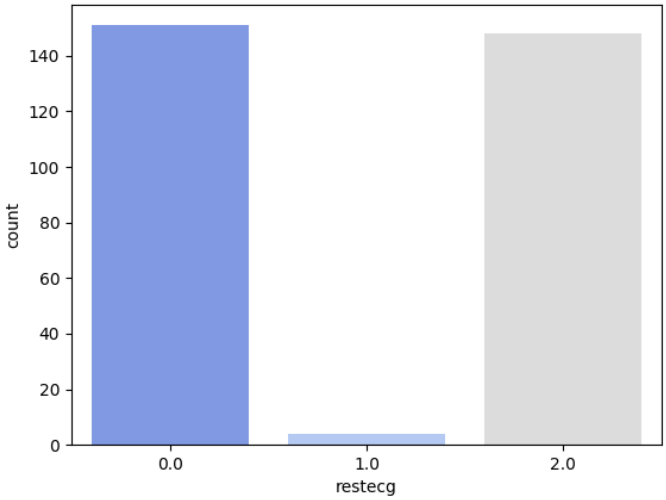


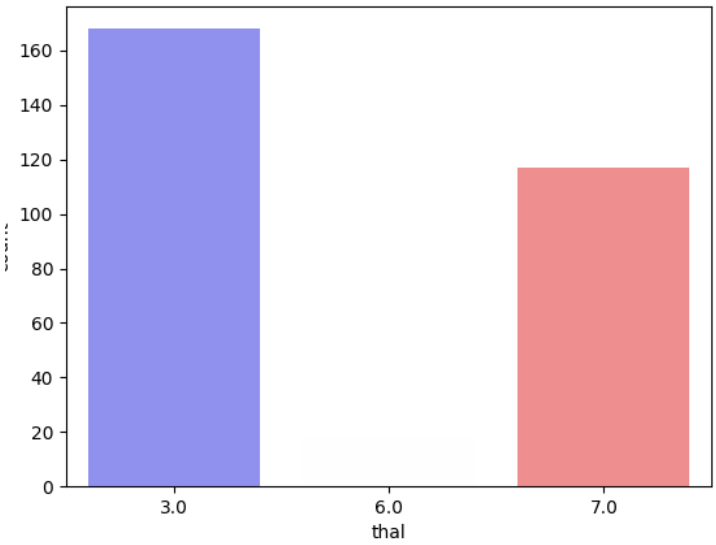


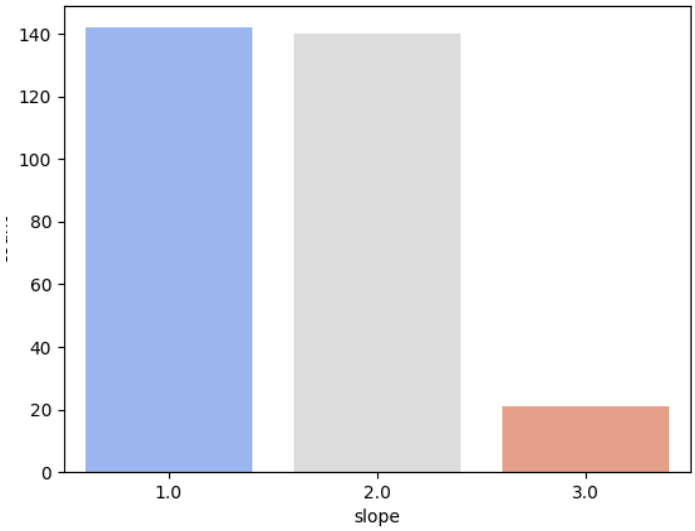


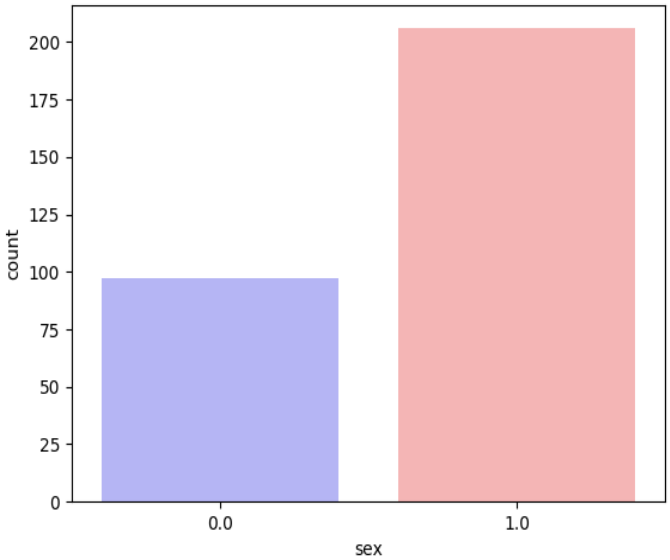


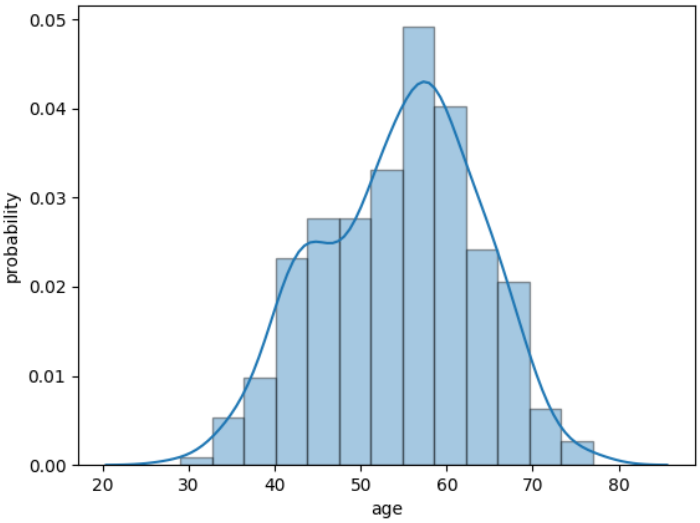

count
num
0
1
2
3
4

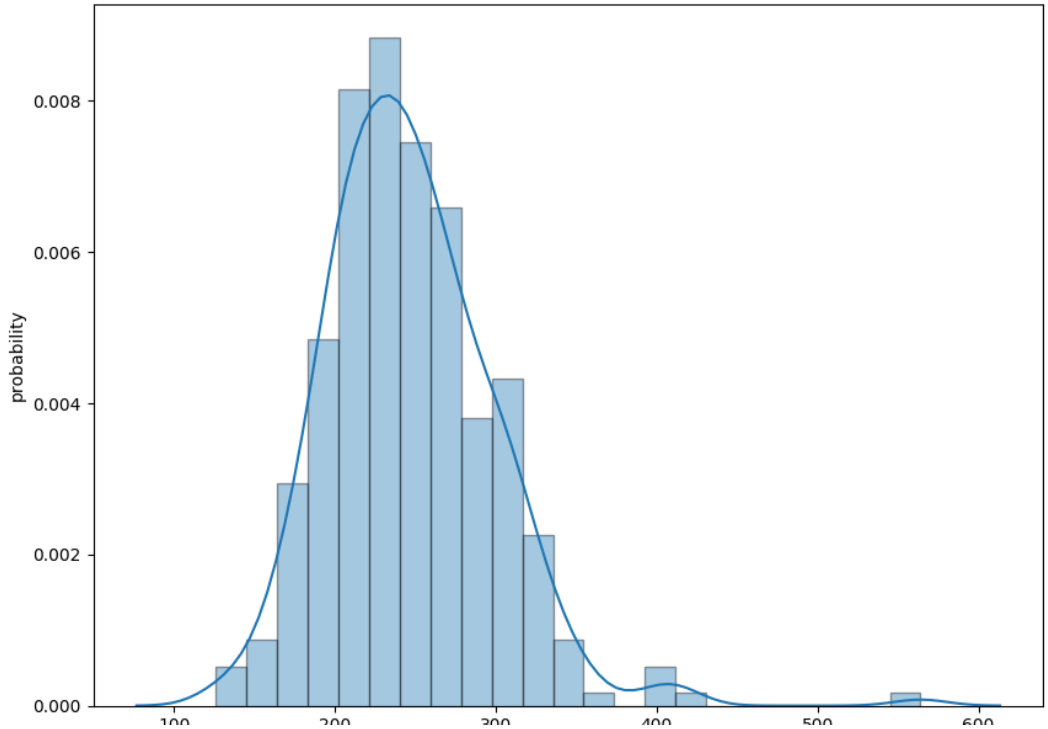

count
restecg
0.0
1.0
2.0

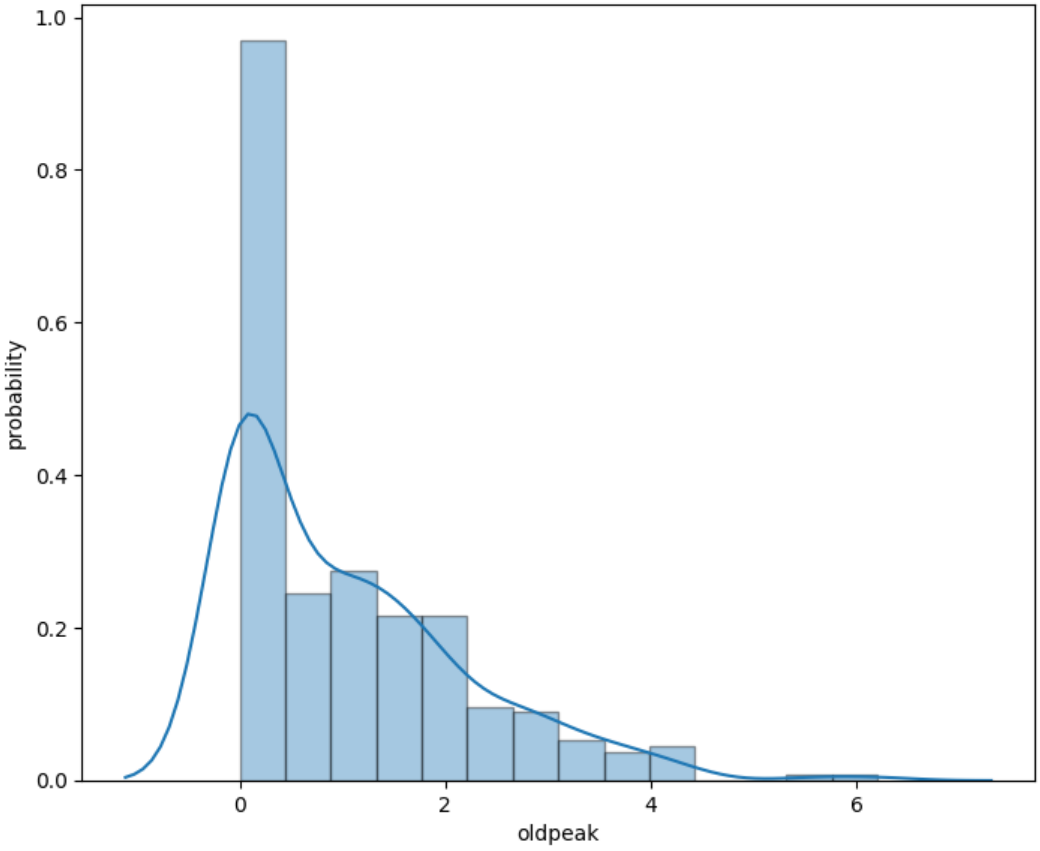

thal
3.0
6.0
7.0

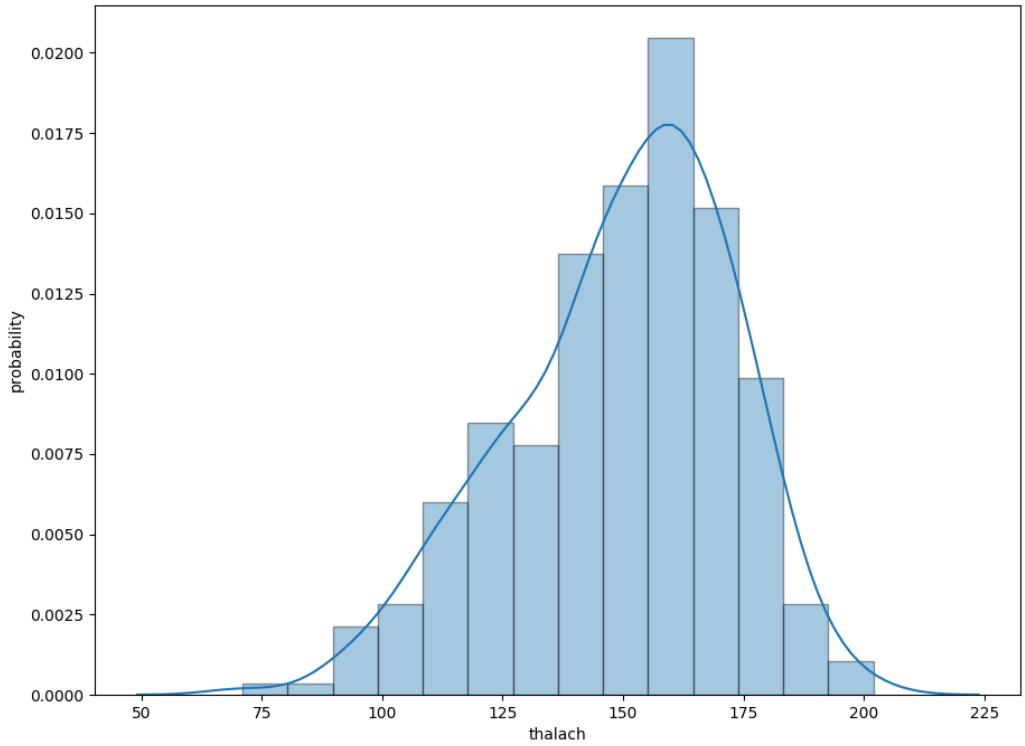

slope
1.0
2.0
3.0

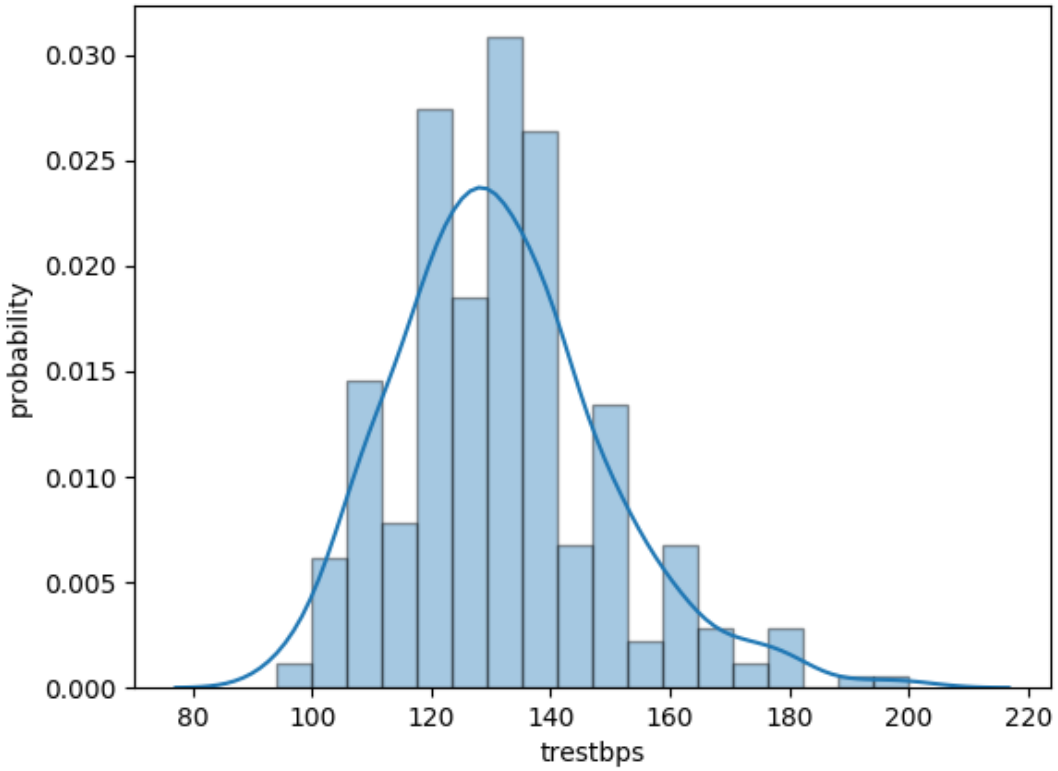

count
sex
0.0
1.0


probability
age


probability
0.008
0.006
0.004
0.002
0.000
100
200
300
400
500
600


probability
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
oldpeak


probability
0.0200
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
50
75
100
125
150
175
200
225
thalach


probability
0.030
0.025
0.020
0.015
0.010
0.005
0.000
80
100
120
140
160
180
200
220
trestbps

قسمت دوم : پیاده سازی الگوریتم های CART و ID3 با معیار جداسازی gini index و entropy

در این بخش نتایج حاصل از الگوریتم های ID3 و CART نشان داده شده است که هم به طور دستی و با پیاده سازی کامل الگوریتم به دست آمده هم با استفاده از کتابخانه ی scikit learn برای درست سنجی الگوریتم. برچسب داده ها هم به دو بخش ۰ به معنای بیماری قلبی ندارد و ۱ به معنای داشتن بیماری قلبی تقسیم شد.

برای گسسته سازی داده ها به روش جداسازی با آنتروپی، با استفاده از الگوریتمFayyad-Irani بهترین آستانه برای جداسازی داده ها برای هر ویژگی به دست آوردیم طوری که آستانه ی جداسازی بیشترین information gain را بدهد.این را در یک دیکشنری از feature و آستانه ی تعیین شده به بیشترین gain به دست آمده برای آن فیچر نگاشت کردیم .برای هر ویژگی این کار انجام شد.سپس ماکزیمم gain بین تمام فیچر ها به دست آمد و فیچری برای جدا سازی انتخاب شد که بیشترین information gain را بین تمام فیچر ها بدهد. درخت باینری ای که به دست آمد با درخت حاصل از الگوریتم scikit learn بسیار نزدیک بود.

الگوریتم CART هم برای جداسازی داده های گسسته به کار می رود هم پیوسته. بنابراین در این مرحله نیازی به استفاده از الگوریتم Fayyad-Irani نبود. و فقط لازم بود مینیمم gini index ها به صورت وزن دار گرفته شود.

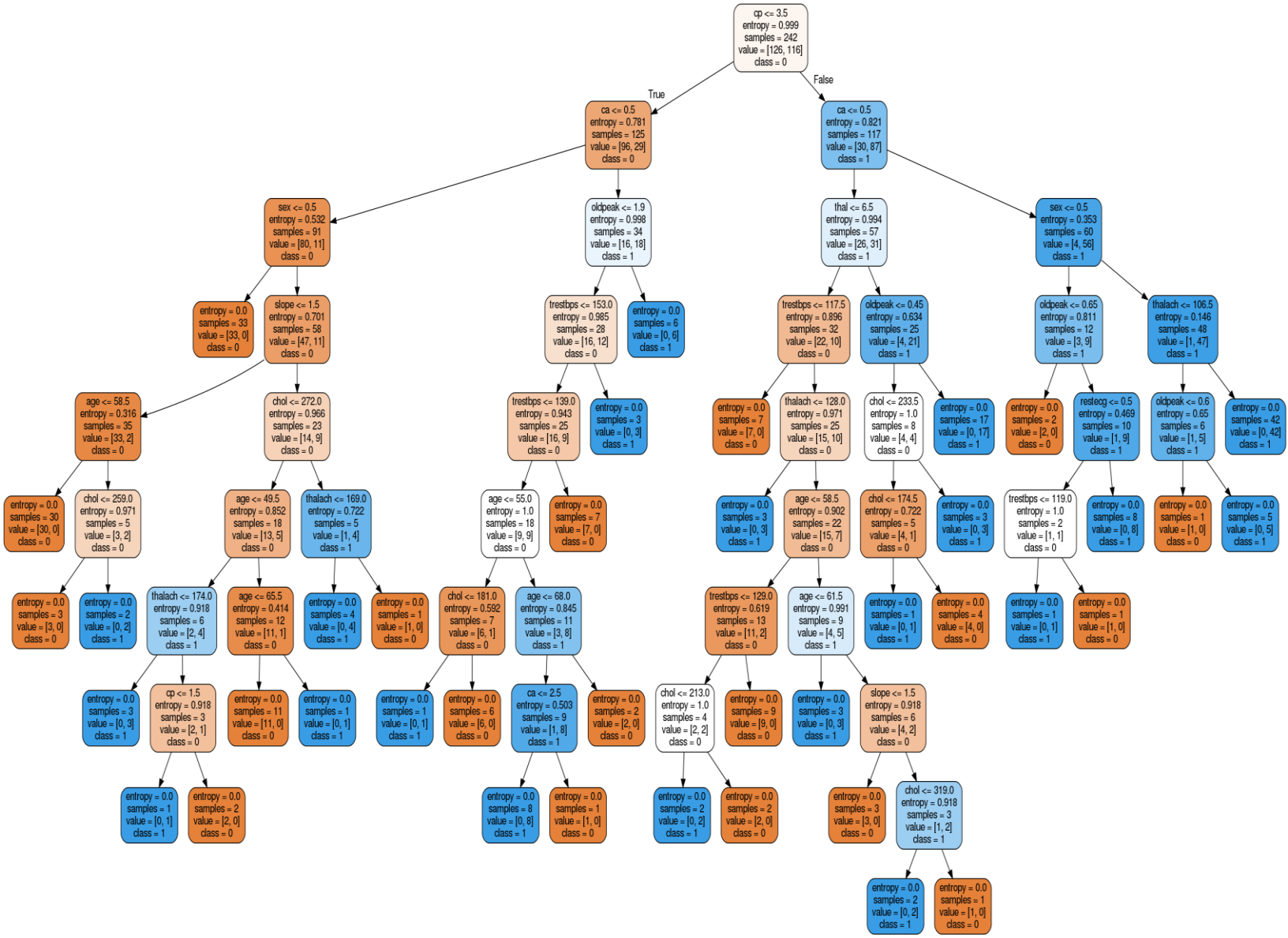
```
def cal_entropy(self,taret_col):
    elements,counts=np.unique(taret_col,return_counts=True)
    entropy=np.sum(-counts[i]/np.sum(counts) * np.log2(counts[i]/np.sum(counts)) for i in range(len(elements)))
    return entropy

def cal_information_gain(self,data,entropy,splitting_index,target_name='num'):
    partial_entropy_sum=(splitting_index* self.cal_entropy(data.iloc[:splitting_index][target_name])+
                         (len(data)-splitting_index)*self.cal_entropy(data.iloc[splitting_index:][target_name]))/len(data)
    return entropy-partial_entropy_sum

def cal_gini_index(self,taret_col):
    elements,counts=np.unique(taret_col,return_counts=True)
    gini=1-np.sum((counts[i]/np.sum(counts))**2 for i in range(len(elements)))
    return gini

def find_best_feature_gini(self,gini_costs):
    best_feature=None
    best_threshold=None
    min_gini=1
    for key,val in gini_costs.items():
        if val[0]<min_gini:
            min_gini=val[0]
            best_threshold=val[1]
            best_feature=key
    return best_feature,best_threshold,min_gini
```

```
def find_best_feature_entropy(self,data,attributes,entropy):
    attribute_threshold=dict()

    for attribute in attributes:
        sorted_data=data.sort_values(by=attribute).reset_index(drop=True)
        index_gain_dict=dict()
        for i in  range(1,len(sorted_data)):   # target labels of data
            if sorted_data['num'][i-1] != sorted_data['num'][i]:
                info_gain=self.cal_information_gain(sorted_data,entropy,i)
                index_gain_dict[i]=info_gain

        #caculating maximum information gain of that attribute and correspoint separating index
        split_index,info_gain=max(index_gain_dict.items(),key=operator.itemgetter(1))
        threshold=(sorted_data[attribute][split_index]+sorted_data[attribute][split_index-1])/2
        attribute_threshold[attribute,threshold]=info_gain
    #return best attribute with best threshold
    best_feature=max(attribute_threshold.items(),key=operator.itemgetter(1))[0]
    feature,threshold=best_feature[0],best_feature[1]
    return feature,threshold
```

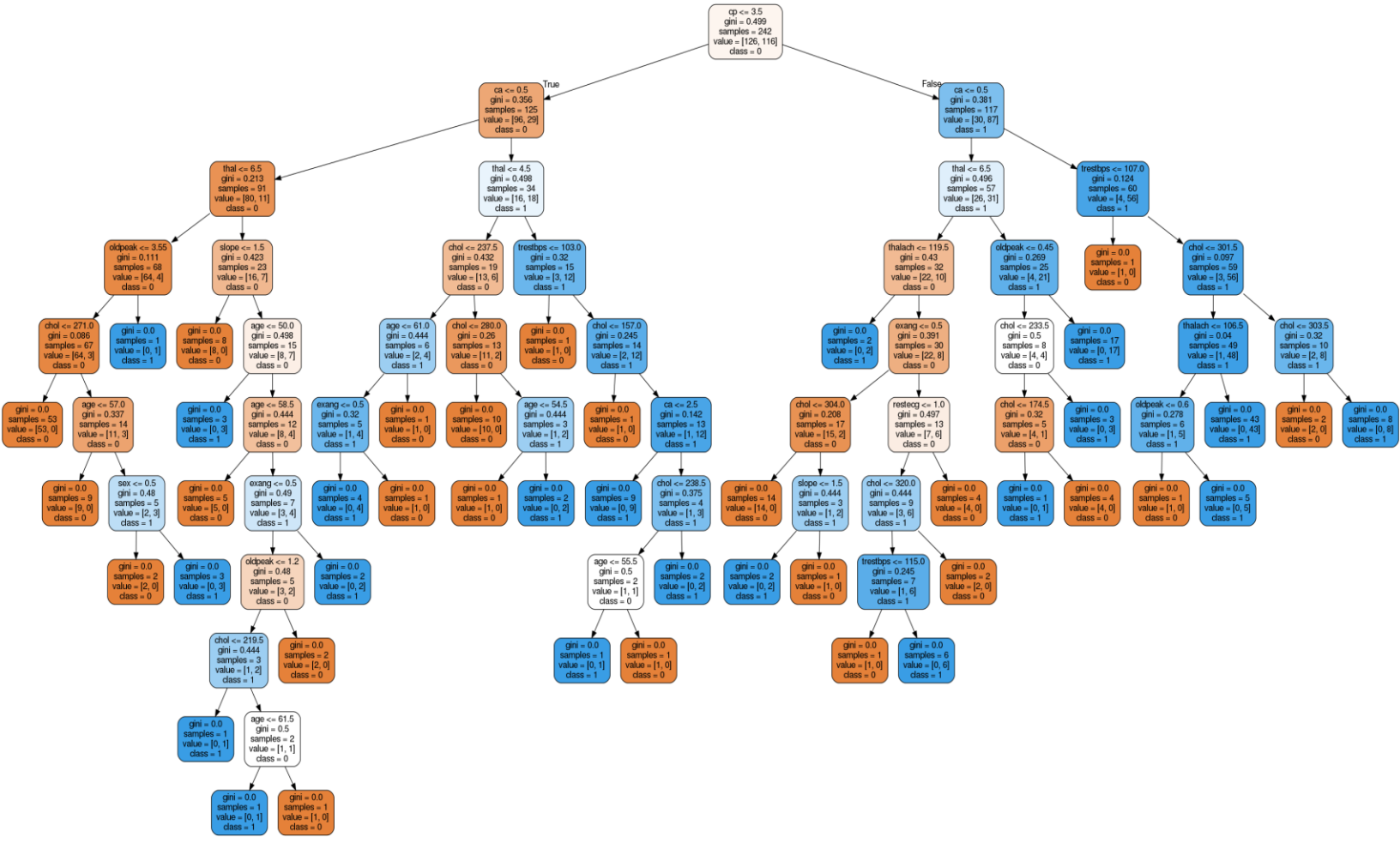
برای بخش هرس کردن، یک max depth تعریف کردم که عمق درخت از یک حدی بیشتر نشود تا مشکل overfitting به وجود نیاید. مشاهده شد که این کار دقت دسته بندی را حدود ۱۵ درصد بیشتر کرد.

۱. بدون هیچ محدودیتی بر روی ماکزیمم عمق

درخت حاصل از آنتروپی :



درخت حاصل از الگوریتم CART با جداساز gini index :



مشاهده می شود درخت حاصل از جداساز gini index عمیق تر بوده چون بر روی داده های پیوسته کار می کند.

دقت حاصل از این دو الگوریتم بر روی دیتاست تست (درصورتی که درخت هرس نشود)

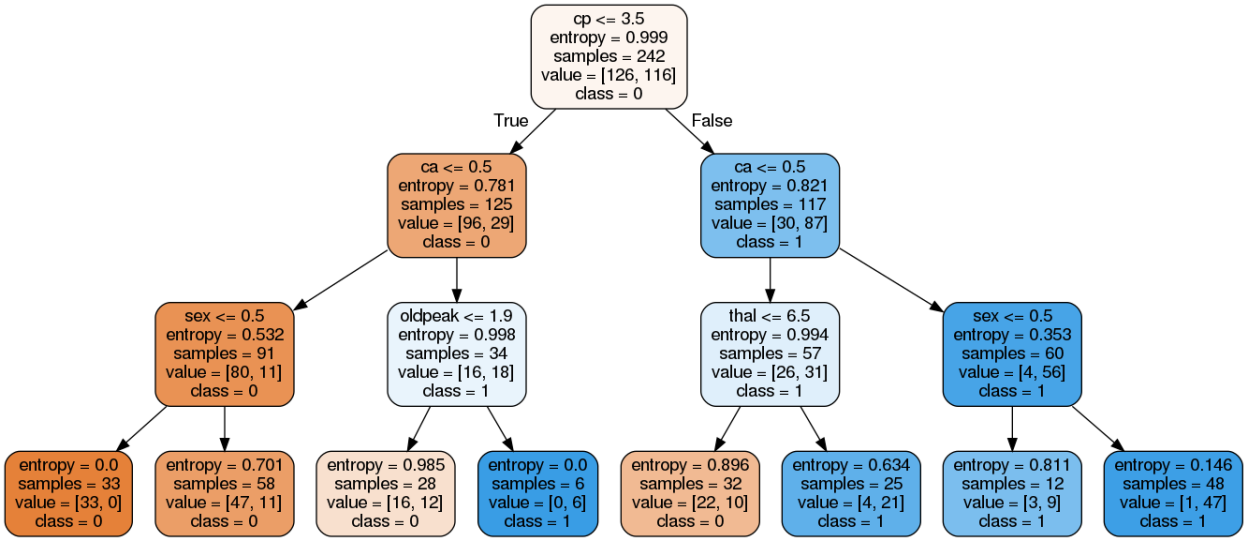
```
bita@bita-K401UQK:~/programming/Decision Tree$ python dt.py
Manual  Acuracy  with gini index criteria :  0.74
Manual  Acuracy  with entropy criteria :  0.74

scikit learn confusion metric with  gini  criteria :
              precision    recall  f1-score   support

           0       0.78      0.66      0.71        38
           1       0.55      0.70      0.62        23

    accuracy                           0.67        61
   macro avg       0.67      0.68      0.66        61
weighted avg       0.69      0.67      0.68        61

The prediction accuracy is:  67.21311475409836 %
scikit learn confusion metric with  entropy  criteria :
              precision    recall  f1-score   support

           0       0.84      0.71      0.77        38
           1       0.62      0.78      0.69        23

    accuracy                           0.74        61
   macro avg       0.73      0.75      0.73        61
weighted avg       0.76      0.74      0.74        61

The prediction accuracy is:  73.77049180327869 %
```

مشاهده می شود نتایج حاصل از الگوریتم های پیاده سازی شده با scikit learn بسیار شبیه هستند.

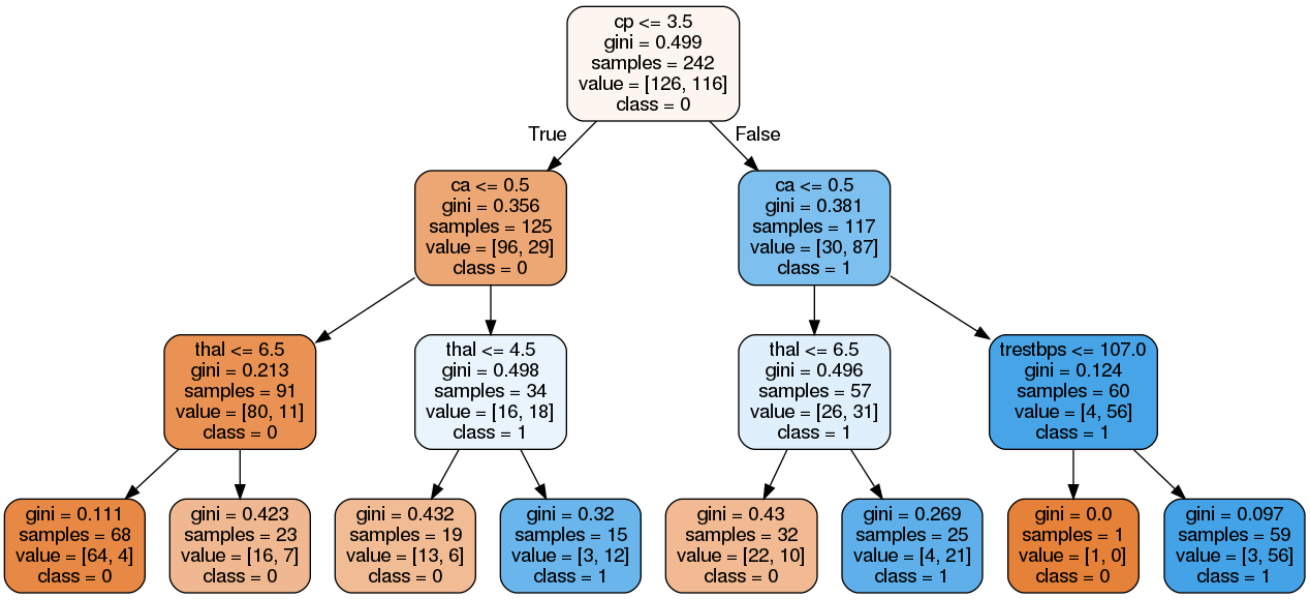
۲. در صورتی که روی عمق درخت محدودیت تعریف کنیم:

با این کار از overfitting جلوگیری می شود و دقت بیشتر می شود.

درخت حاصل از الگوریتم با جدا ساز آنتروپی :



درخت حاصل از الگوریتم با جداساز gini index :



درخت ها خیلی خلوت تر ساخته می شوند. پیمایش آن ها زمان کمتری می برد . دقت هم بالاتر می رود.

```
bita@bita-K401UQK:~/programming/Decision Tree$ python dt.py
Manual  Acuracy  with gini index criteria :  85%
Manual  Acuracy  with entropy criteria :  85%

scikit learn confusion metric with  gini  criteria :
              precision    recall  f1-score   support

           0       0.85      0.92      0.89        38
           1       0.85      0.74      0.79        23

    accuracy                           0.85        61
   macro avg       0.85      0.83      0.84        61
weighted avg       0.85      0.85      0.85        61

The prediction accuracy is:  85.24590163934425 %
scikit learn confusion metric with  entropy  criteria :
              precision    recall  f1-score   support

           0       0.90      0.95      0.92        38
           1       0.90      0.83      0.86        23

    accuracy                           0.90        61
   macro avg       0.90      0.89      0.89        61
weighted avg       0.90      0.90      0.90        61

The prediction accuracy is:  90.1639344262295 %
```

می بینیم که دقت بیشتر از ۱۰ درصد افزایش داشته است.