In [34]:

```
#Sanket Badjate ...
import numpy as np
import pandas as pd
```

In [35]:

```
data=pd.read_csv("sales.csv")
data
```

Out[35]:

	Age	Income	Gender	MaritialStatus	Buys
0	<21	High	Male	Single	No
1	<21	High	Male	Married	No
2	21-35	High	Male	Single	Yes
3	>35	Medium	Male	Single	Yes
4	>35	Low	Female	Single	Yes
5	>35	Low	Female	Married	No
6	21-35	Low	Female	Married	Yes
7	<21	Medium	Male	Single	No
8	<21	Low	Female	Married	Yes
9	>35	Medium	Female	Single	Yes
10	<21	Medium	Female	Married	Yes
11	21-35	Medium	Male	Married	Yes
12	21-35	High	Female	Single	Yes
13	>35	Medium	Male	Married	No

In [36]:

```
data.describe()
```

Out[36]:

	Age	Income	Gender	MaritialStatus	Buys
count	14	14	14	14	14
unique	3	3	2	2	2
top	>35	Medium	Male	Married	Yes
freq	5	6	7	7	9

In [37]:

```
data['Buys'].value_counts()
```

Out[37]:

```
Yes    9
No     5
Name: Buys, dtype: int64
```

In [38]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder();
#data=data.apply(le.fit_transform)
x=data.iloc[:,:-1] #-1 means don't take last column

print(x)

x=x.apply(le.fit_transform)
print(x)
#find label with their encoded value
print("Age with encoded value :",list( zip(data.iloc[:,0], x.iloc[:,0])))
print("\nIncome with encoded value :",list( zip(data.iloc[:,1], x.iloc[:,1])))
print("\nGender with encoded value :",list( zip(data.iloc[:,2], x.iloc[:,2])))
print("\nmaritialStatus with encoded value :",list( zip(data.iloc[:,3], x.iloc[:,3]
```

```
      Age  Income  Gender MaritialStatus
0     <21    High    Male         Single
1     <21    High    Male        Married
2   21-35    High    Male         Single
3     >35  Medium    Male         Single
4     >35     Low  Female         Single
5     >35     Low  Female        Married
6   21-35     Low  Female        Married
7     <21  Medium    Male         Single
8     <21     Low  Female        Married
9     >35  Medium  Female         Single
10    <21  Medium  Female        Married
11  21-35  Medium    Male        Married
12  21-35    High  Female         Single
13    >35  Medium    Male        Married
    Age  Income  Gender  MaritialStatus
0     1       0       1               1
1     1       0       1               0
2     0       0       1               1
```

In [39]:

```
y=data.iloc[:,-1]
```

In [40]:

```
from sklearn.tree import DecisionTreeClassifier
classifier=DecisionTreeClassifier(criterion='entropy')
classifier.fit(x,y)
```

Out[40]:

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='en
tropy',
                       max_depth=None, max_features=None, max_leaf_nod
es=None,
                       min_impurity_decrease=0.0, min_impurity_split=N
one,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecat
ed',
                       random_state=None, splitter='best')
```

In [41]:

```
#Predict value for the given Expression
#[Age < 21, Income = Low,Gender = Female, Marital Status = Married]
test_x=np.array([1,1,0,0])
pred_y=classifier.predict([test_x])
print("Predicted class for input [Age < 21, Income = Low,Gender = Female, Marital S
```

```
Predicted class for input [Age < 21, Income = Low,Gender = Female, Mar
ital Status = Married]
 [1 1 0 0]  is  Yes
```

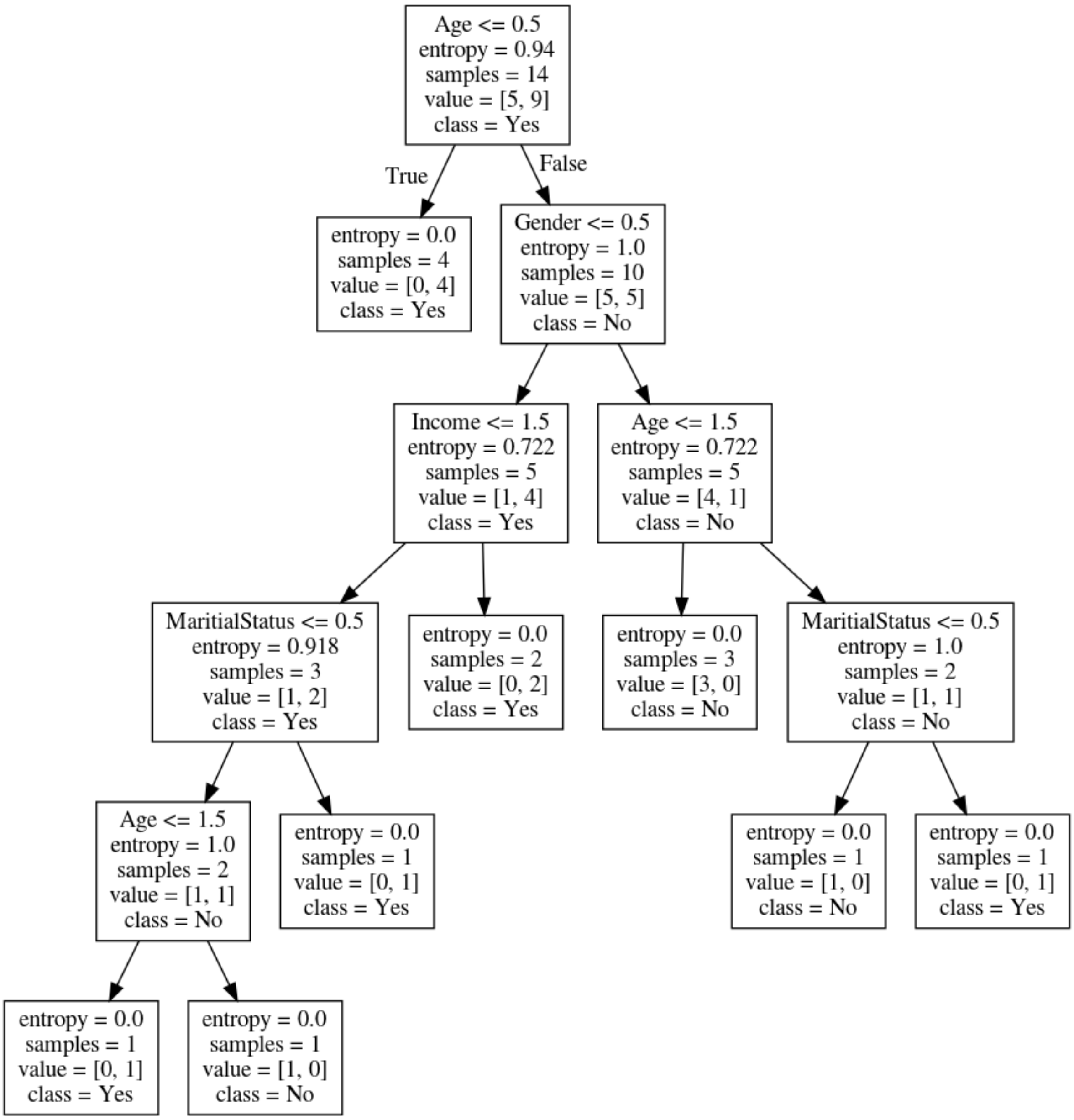
In [42]:

```
#method to generate graph p.s. needs dot utility installed in os
from sklearn.tree import export_graphviz
from IPython.display import Image
export_graphviz(classifier,out_file="data.dot",feature_names=x.columns,class_names=
#you need to install graphviz in fedora(IN LAB) for running below dor command
#yum install graphviz

#then go to terminal and cd to directory where you are saving jupyter notebook
# and execute below command
#    dot -Tpng data.dot -o tree.png


!dot -Tpng data.dot -o tree.png
Image("tree.png")
```

Out[42]:



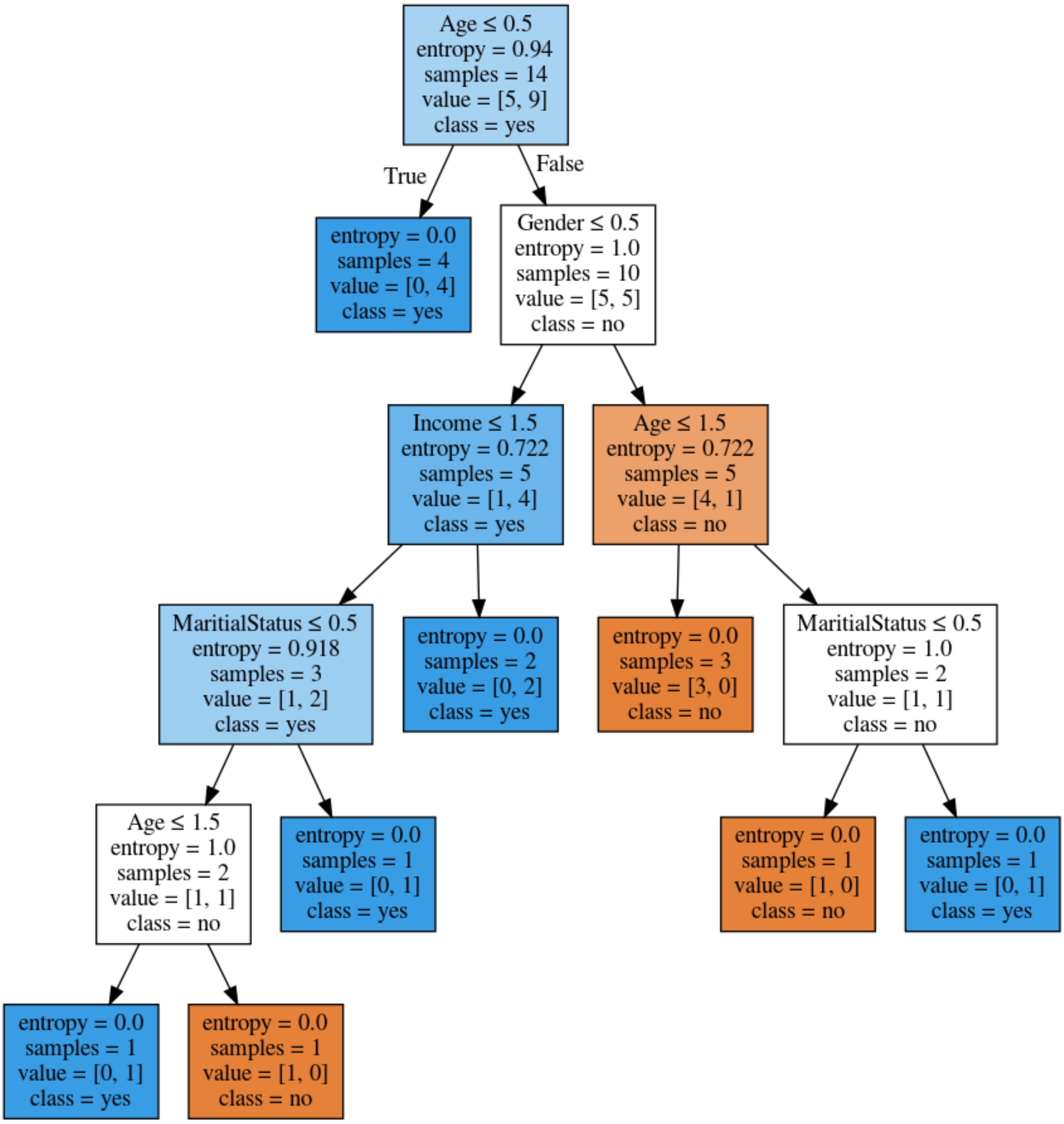
In [44]:

```
import pydotplus as pdd
from IPython.display import Image
dot_data = export_graphviz(classifier, out_file=None,feature_names=x.columns,class_

graph = pdd.graph_from_dot_data(dot_data)

Image(graph.create_png())
graph.write_png("dtree.png")
Image(graph.create_png())
```

Out[44]:



In [45]:

```
#No need to implement below code
#if you want to split into train test set
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
train,test=train_test_split(data.apply(le.fit_transform),test_size=0.14,random_stat
train_x=train.iloc[:,:-1]
train_y=train.iloc[:,-1]
test_x=test.iloc[:,:-1]
test_y=test.iloc[:,-1]
clf=DecisionTreeClassifier(criterion='entropy')
clf.fit(train_x,train_y)
pred_y=clf.predict(test_x)
accuracy=accuracy_score(test_y,pred_y)
accuracy*100
```

Out[45]:

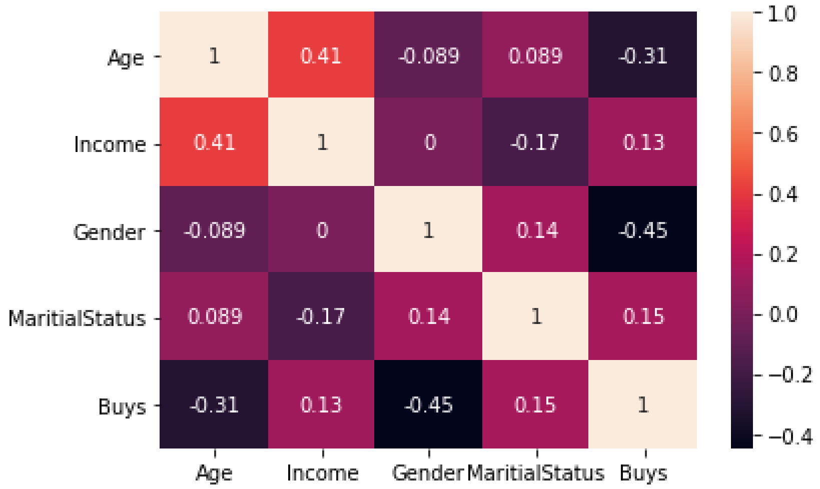
```
50.0
```

In [46]:

```
#just displaying correlation between fields
import seaborn as sns
corr=data.apply(le.fit_transform).corr();
sns.heatmap(corr,annot=True)
```

Out[46]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fddbe607c10>
```


Age
Income
Gender
MaritialStatus
Buys
Age
Income
Gender
MaritialStatus
Buys