In [238]:
```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt
```

In [239]:
```
bank = pd.read_csv("bank.csv")
#Returns the first 10 entries of the Dataframe -Bank
#bank.head()
```

In [240]:
```
#pd.set_option('mode.chained_assignment', None)
```

In [241]:
```
#Lists the Column Names
bank.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11162 entries, 0 to 11161
Data columns (total 17 columns):
age          11162 non-null int64
job          11162 non-null object
marital      11162 non-null object
education    11162 non-null object
default      11162 non-null object
balance      11162 non-null int64
housing      11162 non-null object
loan         11162 non-null object
contact      11162 non-null object
day          11162 non-null int64
month        11162 non-null object
duration     11162 non-null int64
campaign     11162 non-null int64
pdays        11162 non-null int64
previous     11162 non-null int64
poutcome     11162 non-null object
deposit      11162 non-null object
dtypes: int64(7), object(10)
memory usage: 1.4+ MB
```

In [242]:
```
#Returns No of Rows and Columns (Rows,Columns)
bank.shape
```

Out[242]:
```
(11162, 17)
```

In [243]:
```
#Lists the various measures like Mean,Count,Standard Deviation of given
 Dataset
bank.describe()
```

Out[243]:

	age	balance	day	duration	campaign	pdays	
count	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000	111(
mean	41.231948	1528.538524	15.658036	371.993818	2.508421	51.330407	
std	11.913369	3225.413326	8.420740	347.128386	2.722077	108.758282	
min	18.000000	-6847.000000	1.000000	2.000000	1.000000	-1.000000	
25%	32.000000	122.000000	8.000000	138.000000	1.000000	-1.000000	
50%	39.000000	550.000000	15.000000	255.000000	2.000000	-1.000000	
75%	49.000000	1708.000000	22.000000	496.000000	3.000000	20.750000	
max	95.000000	81204.000000	31.000000	3881.000000	63.000000	854.000000	!

In [244]:
```
y=bank['deposit']
y=y.to_frame()
y.head()
```

Out[244]:

	deposit
0	yes
1	yes
2	yes
3	yes
4	yes

In [245]:
```
X=bank
```

In [246]:
```
X.head()
```

Out[246]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	du
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	
1	56	admin.	married	secondary	no	45	no	no	unknown	5	may	
2	41	technician	married	secondary	no	1270	yes	no	unknown	5	may	
3	55	services	married	secondary	no	2476	yes	no	unknown	5	may	
4	54	admin.	married	tertiary	no	184	no	no	unknown	5	may	

In [247]:
```
#Applying Train,Test Split 1st time
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.4,random_
state=100)
```

In [248]:
```
#Print only columns of Training Data
X_train.columns
```

Out[248]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'hou
sing',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pday
s',
       'previous', 'poutcome', 'deposit'],
      dtype='object')
```

In [249]:
```
#Display first five records from X_train
X_train.head()
```

Out[249]:

	age	job	marital	education	default	balance	housing	loan	contact	day	m
5541	45	housemaid	married	primary	no	317	yes	no	cellular	8	
9240	53	management	married	tertiary	no	232	yes	no	telephone	29	
10826	24	student	single	secondary	no	493	yes	no	cellular	13	
1089	26	unemployed	single	tertiary	no	814	no	no	cellular	28	
9168	55	technician	married	unknown	no	1393	yes	yes	telephone	21	

In [250]:
```
y_train.head()
```

Out[250]:

	deposit
5541	no
9240	no
10826	no
1089	yes
9168	no

In [251]:
```
#The following code is added to convert all the string type data into nu
merical data
combine_=[y_train,y_test]
depositmapping={'yes':1,'no': 0}
for dt in combine_:
   dt['deposit']=bank['deposit'].map(depositmapping)
```

In [252]:
```
#y_train.head()
```

In [253]:
```
#X_train.head()
```

In [254]:
```
#assigning 1 to yes and 0 to no in default (X_train)
combine=[X_train,X_test]
depositmapping={'yes':1,'no': 0}
for dt in combine:
   dt['deposit']=bank['deposit'].map(depositmapping)

X_train.head()
```

Out[254]:

	age	job	marital	education	default	balance	housing	loan	contact	day	m
5541	45	housemaid	married	primary	no	317	yes	no	cellular	8	
9240	53	management	married	tertiary	no	232	yes	no	telephone	29	
10826	24	student	single	secondary	no	493	yes	no	cellular	13	
1089	26	unemployed	single	tertiary	no	814	no	no	cellular	28	
9168	55	technician	married	unknown	no	1393	yes	yes	telephone	21	

In [255]:
```
#assigning 1 to yes and 0 to no in default (X_train)
combine=[X_train,X_test]
defaultmapping={'yes':1,'no': 0}
for dt in combine:
   dt['default']=bank['default'].map(defaultmapping)
```

In [256]:
```
X_train.head()
```

Out[256]:

	age	job	marital	education	default	balance	housing	loan	contact	day	m
5541	45	housemaid	married	primary	0	317	yes	no	cellular	8	
9240	53	management	married	tertiary	0	232	yes	no	telephone	29	
10826	24	student	single	secondary	0	493	yes	no	cellular	13	
1089	26	unemployed	single	tertiary	0	814	no	no	cellular	28	
9168	55	technician	married	unknown	0	1393	yes	yes	telephone	21	

In [257]:
```
#assigning 1 to yes and 0 to no in housing (X_train)
combine=[X_train,X_test]
housingmapping={'yes':1,'no': 0}
for dt in combine:
   dt['housing']=bank['housing'].map(housingmapping)
```

In [258]:
```
X_train.head()
```

Out[258]:

	age	job	marital	education	default	balance	housing	loan	contact	day	m
5541	45	housemaid	married	primary	0	317	1	no	cellular	8	
9240	53	management	married	tertiary	0	232	1	no	telephone	29	
10826	24	student	single	secondary	0	493	1	no	cellular	13	
1089	26	unemployed	single	tertiary	0	814	0	no	cellular	28	
9168	55	technician	married	unknown	0	1393	1	yes	telephone	21	

In [259]:
```
#assigning 1 to yes and 0 to no in loan (X_train)
combine=[X_train,X_test]
loanmapping={'yes':1,'no': 0}
for dt in combine:
   dt['loan']=bank['loan'].map(loanmapping)
```

In [260]:
```
#X_train.head()
```

In [261]:
```
#Analysing Education variable to assign them numerical values
X_train[['education','deposit']].groupby('education',as_index=False).mea
n().sort_values('deposit',ascending=False)
```

Out[261]:

	education	deposit
2	tertiary	0.541836
3	unknown	0.504886
1	secondary	0.458766
0	primary	0.390055

In [262]:
```
 educationmapping={'primary':1,'secondary':2,'tertiary':3,'unknown':0}
for df in combine:
    df['education']=df['education'].map(educationmapping)
```

In [263]:
```
X_train.head()
```

Out[263]:

	age	job	marital	education	default	balance	housing	loan	contact	day	m
5541	45	housemaid	married	1	0	317	1	0	cellular	8	
9240	53	management	married	3	0	232	1	0	telephone	29	
10826	24	student	single	2	0	493	1	0	cellular	13	
1089	26	unemployed	single	3	0	814	0	0	cellular	28	
9168	55	technician	married	0	0	1393	1	1	telephone	21	

In [264]:
```
#Analysing Marital variable to assign them numerical values
X_train[['marital','deposit']].groupby('marital',as_index=False).mean().
sort_values('deposit',ascending=False)
```

Out[264]:

	marital	deposit
2	single	0.553462
0	divorced	0.485640
1	married	0.436187

In [265]:
```
#analysing the variable 'month'
X_train[['month','deposit']].groupby('month',as_index=False).mean().sort
_values('deposit',ascending=False)
```

Out[265]:

	month	deposit
2	dec	0.945205
7	mar	0.890805
11	sep	0.862069
10	oct	0.828000
0	apr	0.617699
3	feb	0.534934
9	nov	0.441230
1	aug	0.439377
6	jun	0.438375
4	jan	0.432836
5	jul	0.423529
8	may	0.342703

In [266]:
```
for df in combine:
    df['month']=df['month'].replace(['mar','dec','sep','oct'],2,regex=Tr
ue)
    df['month']=df['month'].replace(['apr','feb','aug','jun'],1,regex=Tr
ue)
    df['month']=df['month'].replace(['nov','jul','jan','may'],0,regex=Tr
ue)
```

In [267]:
```
X_train[['poutcome','deposit']].groupby('poutcome',as_index=False).mean
().sort_values('deposit',ascending=False)
```

Out[267]:

	poutcome	deposit
2	success	0.914157
1	other	0.569182
0	failure	0.512295
3	unknown	0.410395

In [268]:
```
poutcomemapping={'success':2,'other':1,'failure':0,'unknown':0}

for df in combine:
    df['poutcome']=df['poutcome'].map(poutcomemapping)
```

In [269]:
```
for df in combine:
    df['job']=df['job'].replace(['management','technician','unknown','ad
min.','housemaid','self-employed','services',
                                'blue-collar','entrepreneur'],'rare',reg
ex=True)
jobmapping={'student':1,'retired':2,'unemployed':3,'rare':0}
for df in combine:
    df['job']=df['job'].map(jobmapping)
```

In [270]:
```
X_final_train=X_train[['job','loan','month','poutcome']]
X_final_test=X_test[['job','loan','month','poutcome']]
```

In [271]:
```
X_train.columns
```

Out[271]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'hou
sing',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pday
s',
       'previous', 'poutcome', 'deposit'],
      dtype='object')
```

In [272]:
```
from sklearn.tree import  DecisionTreeClassifier
dt=DecisionTreeClassifier(random_state=101)
dt.fit(X_final_train,y_train)
predict=dt.predict(X_final_test)
accuracy_test=round(dt.score(X_final_test,y_test)*100,2)
accuracy_train=round(dt.score(X_final_train,y_train)*100,2)
print('train accuracy of decision tree classifier',accuracy_train)
print('test accuracy of decision tree classifier',accuracy_test)
```

```
train accuracy of decision tree classifier 67.28
test accuracy of decision tree classifier 66.81
```

In [273]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, precision_
recall_fscore_support
cm = confusion_matrix(y_test, predict)
```

In [274]:
```
cm
```

Out[274]:
```
array([[2140,  244],
       [1238,  843]])
```

In [275]:
```
cm_df = pd.DataFrame(cm,
                     index = ['Deposit','Others'],
                     columns = ['Deposit','Others'])
```

In [276]:
```
plt.figure(figsize=(5.5,4))
sns.heatmap(cm_df, annot=True)
plt.title(' Decision Tree \nAccuracy:{0:.3f}'.format(accuracy_test))
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```