# Usama Arif Roll No 14

In [1]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier,plot_tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer
```

In [2]:

```
data=pd.read_csv('Social_Network.csv')
data.head()

```

Out[2]:

| | User ID | Gender | Age | EstimatedSalary | Purchased |
|---|---|---|---|---|---|
| **0** | 15624510 | Male | 19 | 19000 | 0 |
| **1** | 15810944 | Male | 35 | 20000 | 0 |
| **2** | 15668575 | Female | 26 | 43000 | 0 |
| **3** | 15603246 | Female | 27 | 57000 | 0 |
| **4** | 15804002 | Male | 19 | 76000 | 0 |

# seprating the data

In [3]:

```
data=data.drop(['User ID'],axis=1)
data=data.replace(['Male','Female'],[1,2])
```

In [4]:

```
scaler=StandardScaler()
trans_d=['Age','EstimatedSalary']
transformer=ColumnTransformer([('trans_d',scaler,trans_d)],remainder='passthrough')
x_transformed=transformer.fit_transform(data)
```

In [5]:

```
data[['Age','EstimatedSalary']]=x_transformed[:,:2]
data #replaced and standardized data
```

Out[5]:

| | Gender | Age | EstimatedSalary | Purchased |
|---|---|---|---|---|
| **0** | 1 | -1.781797 | -1.490046 | 0 |
| **1** | 1 | -0.253587 | -1.460681 | 0 |
| **2** | 2 | -1.113206 | -0.785290 | 0 |
| **3** | 2 | -1.017692 | -0.374182 | 0 |
| **4** | 1 | -1.781797 | 0.183751 | 0 |
| **...** | ... | ... | ... | ... |
| **395** | 2 | 0.797057 | -0.844019 | 1 |
| **396** | 1 | 1.274623 | -1.372587 | 1 |
| **397** | 2 | 1.179110 | -1.460681 | 1 |
| **398** | 1 | -0.158074 | -1.078938 | 0 |
| **399** | 2 | 1.083596 | -0.990844 | 1 |

400 rows × 4 columns

In [6]:

```
x=data.drop('Purchased',axis=1)
y=data['Purchased']
```

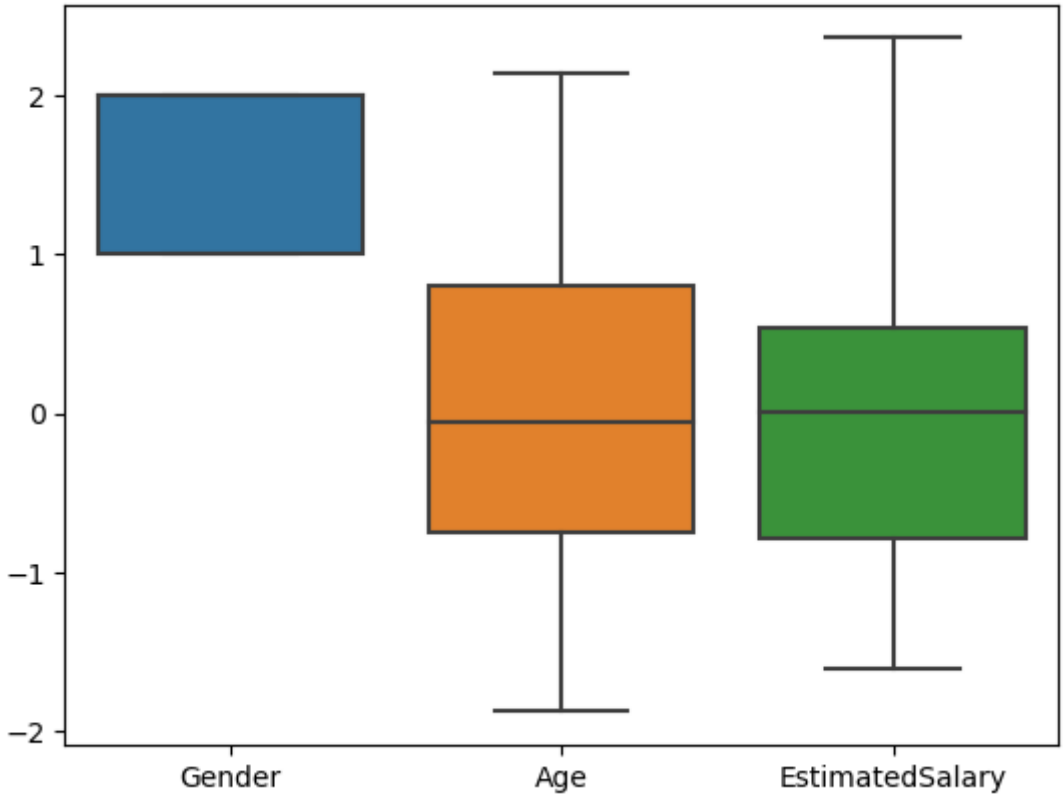
# Data Visualization through box plot

In [7]:

```
1 sns.boxplot(x)
```

Out[7]:

<Axes: >




# DecisionTree implimentation

In [8]:

```
1 model=DecisionTreeClassifier(criterion='gini',splitter='best')
```

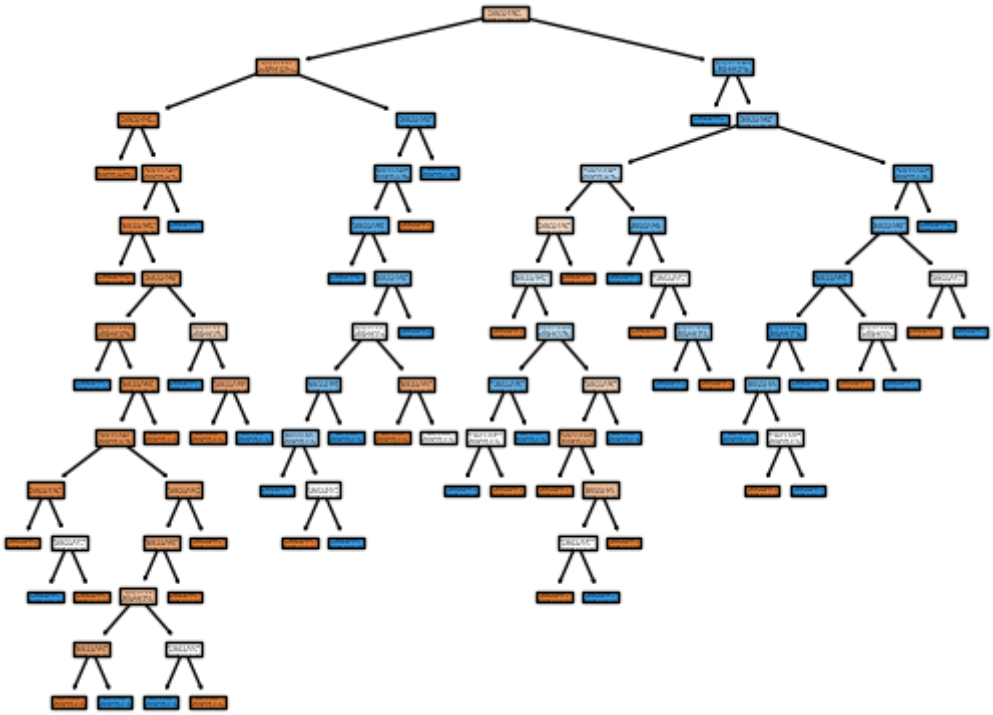
In [9]:

```
1 x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
2 model.fit(x_train,y_train);
```

# DecisionTree visualization

In [10]:

```
1 plot_tree(model,filled=True);
```



# prediction

In [11]:

```
1 y_preds=model.predict(x_test)
2 y_preds
```

Out[11]:

```
array([1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0], dtype=int64)
```

# model Evaluation

In [12]:

```
1 model.score(x_test,y_test)
```

Out[12]:

```
0.8375
```

In [13]:

```
conf_m=confusion_matrix(y_test,y_preds)
conf_m
```

Out[13]:

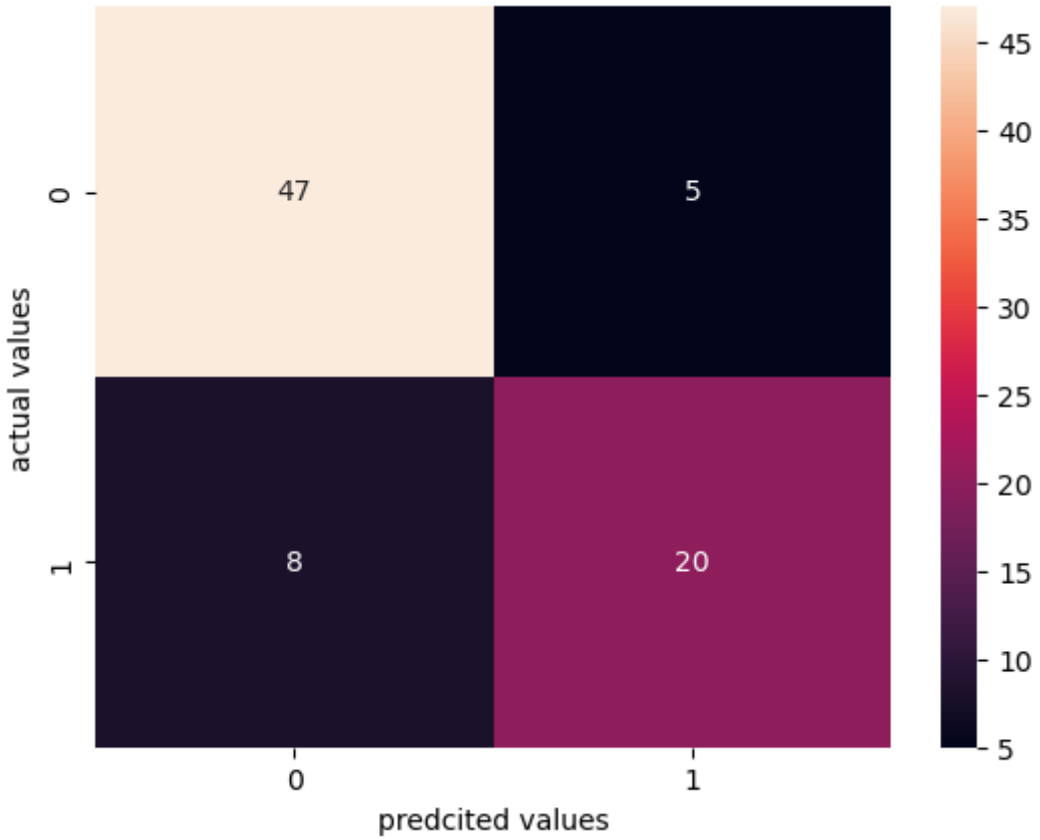
```
array([[47,  5],
       [ 8, 20]], dtype=int64)
```

In [14]:

```
sns.heatmap(conf_m,annot=True)
plt.xlabel('predcited values')
plt.ylabel('actual values')
```

Out[14]:

```
Text(50.722222222222214, 0.5, 'actual values')
```




# manual prediction

In [17]:

```
Gender = float(input("Gender ")) #put replaced values as above
Age = float(input("Age: "))
EstimatedSalary = float(input("Estimated Salary: "))

df=pd.DataFrame({
    'Gender':[Gender],
    'Age':[Age],
    'EstimatedSalary':[EstimatedSalary],

})
y_pred=model.predict(df)
print(f'predicted class is {y_pred[0]}')
```

```
Gender 0
Age: 1
Estimated Salary: 0
predicted class is 1
```

# RandomForestClassifier

In [48]:

```
clf=RandomForestClassifier(n_estimators=500)
```

In [49]:

```
np.random.seed(2)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
clf.fit(x_train,y_train);
```

In [50]:

```
y__preds=clf.predict(x_test)
y__preds
```

Out[50]:

```
array([0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0], dtype=int64)
```

# Model Evaluation

In [51]:

```
model.score(x_test,y_test)
```

Out[51]:

```
0.95
```

In [52]:

```
confusion_matrix=confusion_matrix(y_test,y__preds)
confusion_matrix
```

Out[52]:

```
array([[44,  4],
       [ 6, 26]], dtype=int64)
```

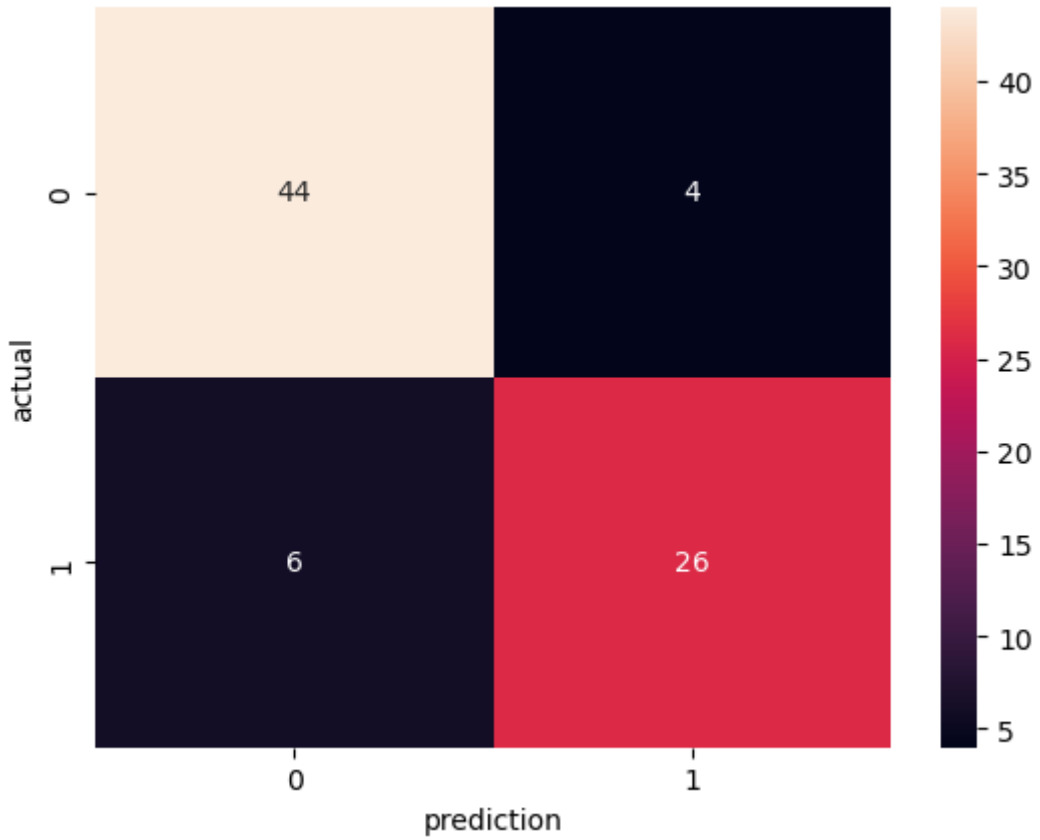
# Confusion Matrix visualization

In [54]:

```
sns.heatmap(confusion_matrix,annot=True)
plt.xlabel('prediction')
plt.ylabel('actual')
```

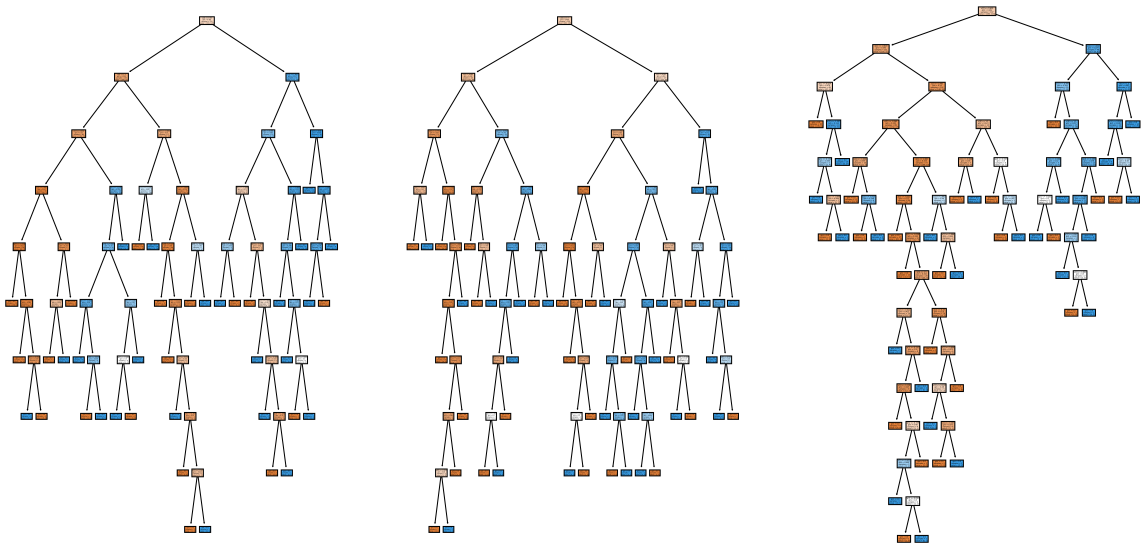
Out[54]:

```
Text(50.722222222222214, 0.5, 'actual')
```



In [61]:

```
fig,axes = plt.subplots(nrows = 1,ncols = 3,figsize = (20,10), dpi=900)
for i in range(0,3):
    plot_tree(clf.estimators_[i],
              filled=True,ax=axes[i])
```



# MANUAL PREDICTION

In [64]:

```
Gender = float(input("Gender ")) #put replaced values as above
Age = float(input("Age: "))
EstimatedSalary = float(input("Estimated Salary: "))

df=pd.DataFrame({
    'Gender':[Gender],
    'Age':[Age],
    'EstimatedSalary':[EstimatedSalary],

})
y_pred=model.predict(df)

print(f'predicted class is {y_pred}')
```

```
Gender 0
Age: 0
Estimated Salary: 0
predicted class is [1]
```

In [ ]:

```

```