DS-210: Programming for Data Science

Lecture 10: Overfitting and underfitting. Bias and variance.

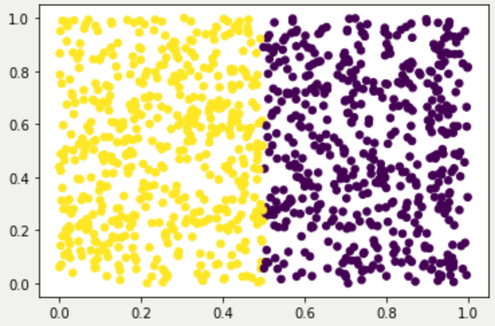
DS-210: Programming for Data Science

Lecture 10: Overfitting and underfitting. Bias and variance.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

from sklearn import tree
from sklearn.tree import DecisionTreeClassifier

from decision_area import draw_decision_area
```

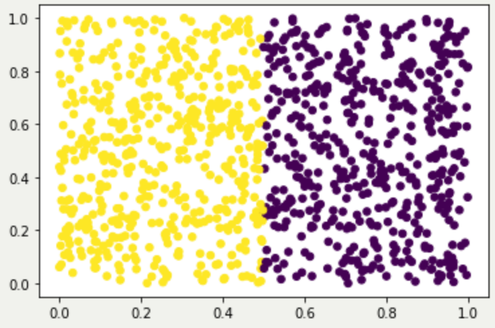
Noisy classification

In [5]:
```
DIMENSIONS = 2

def noise(x, flip_prob=.25):
    if np.random.uniform() < flip_prob:
        return not x
    return x

f = lambda x : x[0] <= 0.5
#f = lambda x : noise(x[0] <= 0.5)

SAMPLES = 1000
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```



Noisy classification

In [5]:
```
DIMENSIONS = 2

def noise(x, flip_prob=.25):
    if np.random.uniform() < flip_prob:
        return not x
    return x

f = lambda x : x[0] <= 0.5
#f = lambda x : noise(x[0] <= 0.5)

SAMPLES = 1000
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```

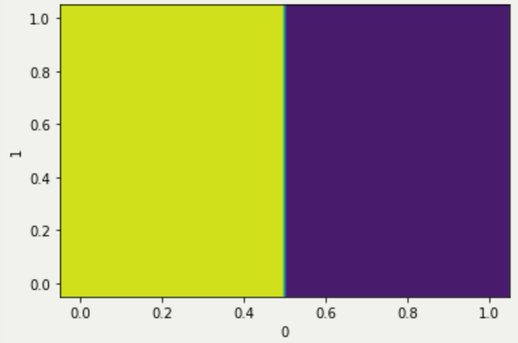


In [8]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=2)
clf.fit(X,Y)
print(tree.export_text(clf))
```

```
|--- feature_0 <= 0.50
|   |--- class: True
|--- feature_0 >  0.50
|   |--- class: False
```

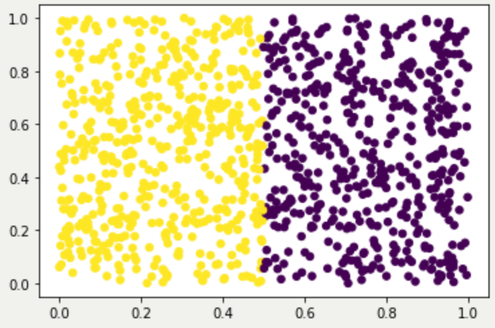
In [9]:
```
draw_decision_area(clf,X,0,1)
print(clf.score(X2,Y2))
```

```
0.999
```



Noisy classification

In [5]:
```
DIMENSIONS = 2

def noise(x, flip_prob=.25):
    if np.random.uniform() < flip_prob:
        return not x
    return x

f = lambda x : x[0] <= 0.5
#f = lambda x : noise(x[0] <= 0.5)

SAMPLES = 1000
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```

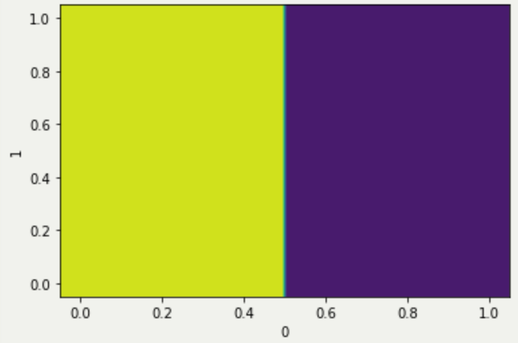


In [10]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=20)
clf.fit(X,Y)
print(tree.export_text(clf))
```

```
|--- feature_0 <= 0.50
|   |--- class: True
|--- feature_0 >  0.50
|   |--- class: False
```

In [11]:
```
draw_decision_area(clf,X,0,1)
print(clf.score(X2,Y2))
```

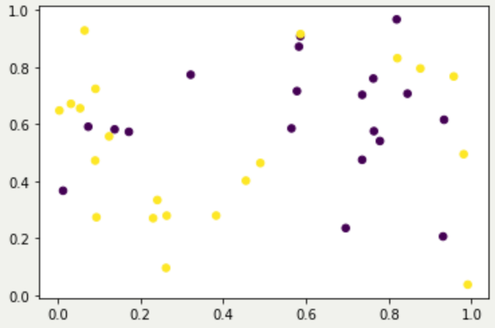
```
0.999
```



Noisy classification

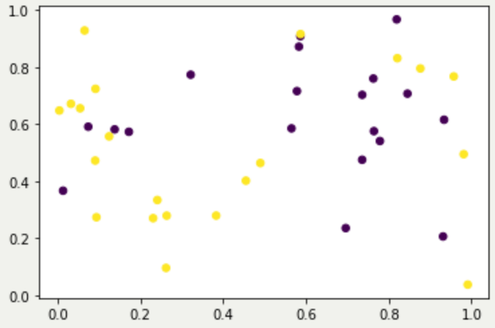
In [17]:

```
DIMENSIONS = 2

def noise(x, flip_prob=.25):
    if np.random.uniform() < flip_prob:
        return not x
    return x

#f = lambda x : x[0] <= 0.5
f = lambda x : noise(x[0] <= 0.5)

SAMPLES = 40
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```



Noisy classification

In [17]:
```
DIMENSIONS = 2

def noise(x, flip_prob=.25):
    if np.random.uniform() < flip_prob:
        return not x
    return x

#f = lambda x : x[0] <= 0.5
f = lambda x : noise(x[0] <= 0.5)

SAMPLES = 40
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```

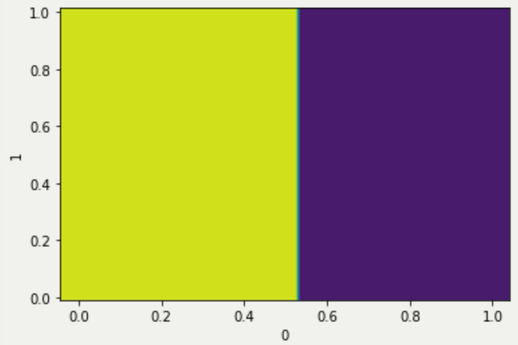


In [18]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=2)
clf.fit(X,Y)
print(tree.export_text(clf))
```

```
|--- feature_0 <= 0.53
|   |--- class: True
|--- feature_0 >  0.53
|   |--- class: False
```

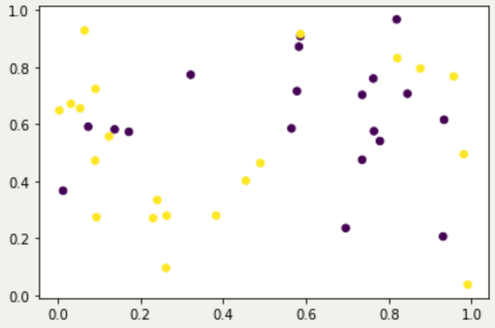
In [19]:
```
draw_decision_area(clf,X,0,1)
print(clf.score(X2,Y2))
```

```
0.748
```



Noisy classification

In [17]:
```
DIMENSIONS = 2

def noise(x, flip_prob=.25):
    if np.random.uniform() < flip_prob:
        return not x
    return x

#f = lambda x : x[0] <= 0.5
f = lambda x : noise(x[0] <= 0.5)

SAMPLES = 40
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```

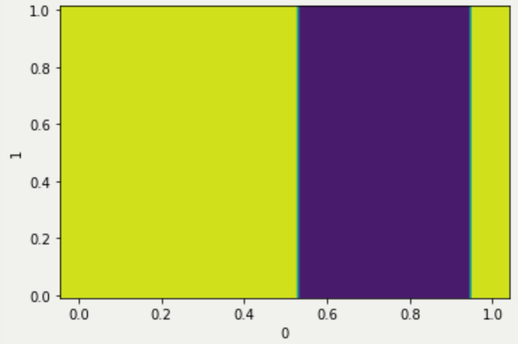


In [20]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=3)
clf.fit(X,Y)
print(tree.export_text(clf))
```

```
|--- feature_0 <= 0.53
|   |--- class: True
|--- feature_0 >  0.53
|   |--- feature_0 <= 0.95
|   |   |--- class: False
|   |--- feature_0 >  0.95
|   |   |--- class: True
```

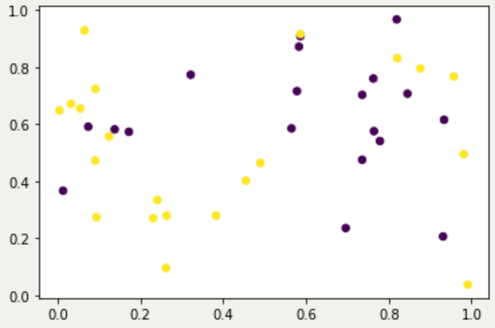
In [21]:
```
draw_decision_area(clf,X,0,1)
print(clf.score(X2,Y2))
```

```
0.72
```



Noisy classification

In [17]:
```
DIMENSIONS = 2

def noise(x, flip_prob=.25):
    if np.random.uniform() < flip_prob:
        return not x
    return x

#f = lambda x : x[0] <= 0.5
f = lambda x : noise(x[0] <= 0.5)

SAMPLES = 40
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```

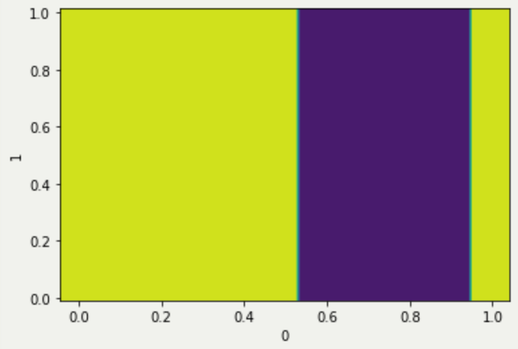


In [23]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=4)
clf.fit(X,Y)
print(tree.export_text(clf))
```

```
|--- feature_0 <= 0.53
|   |--- class: True
|--- feature_0 >  0.53
|   |--- feature_0 <= 0.95
|   |   |--- feature_1 <= 0.78
|   |   |   |--- class: False
|   |   |--- feature_1 >  0.78
|   |   |   |--- class: False
|   |--- feature_0 >  0.95
|   |   |--- class: True
```

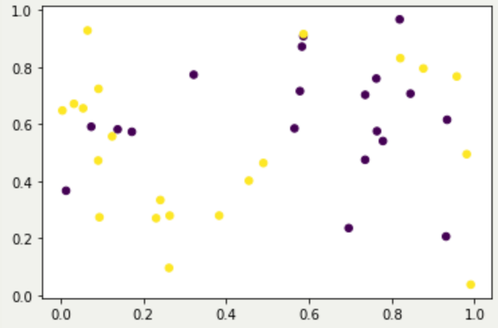
In [24]:
```
draw_decision_area(clf,X,0,1)
print(clf.score(X2,Y2))
```

```
0.72
```



Noisy classification

In [17]:
```
DIMENSIONS = 2

def noise(x, flip_prob=.25):
    if np.random.uniform() < flip_prob:
        return not x
    return x

#f = lambda x : x[0] <= 0.5
f = lambda x : noise(x[0] <= 0.5)

SAMPLES = 40
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```

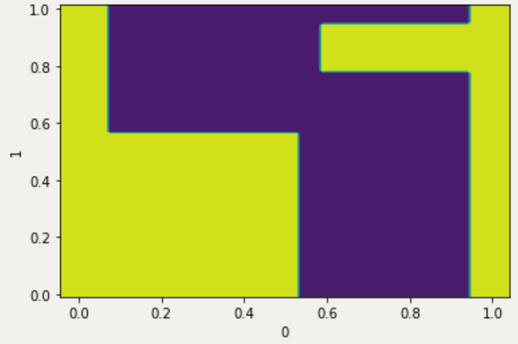


In [28]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=8)
clf.fit(X,Y)
#print(tree.export_text(clf))
```

Out[28]: DecisionTreeClassifier(max_leaf_nodes=8)

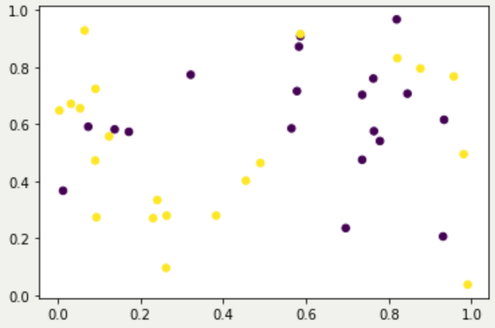
In [29]:
```
draw_decision_area(clf,X,0,1)
print(clf.score(X2,Y2))
```

0.589



Noisy classification

In [17]:
```
DIMENSIONS = 2

def noise(x, flip_prob=.25):
    if np.random.uniform() < flip_prob:
        return not x
    return x

#f = lambda x : x[0] <= 0.5
f = lambda x : noise(x[0] <= 0.5)

SAMPLES = 40
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```

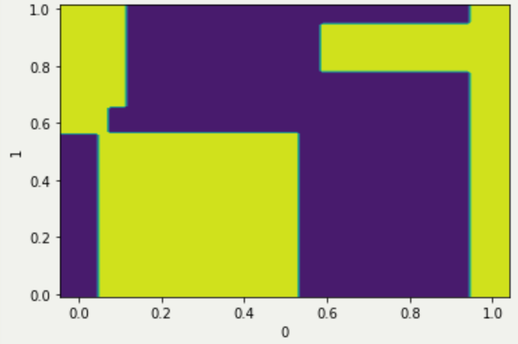


In [30]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=16)
clf.fit(X,Y)
#print(tree.export_text(clf))
```

Out[30]: DecisionTreeClassifier(max_leaf_nodes=16)

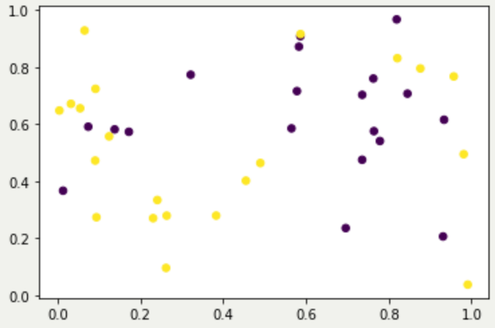
In [31]:
```
draw_decision_area(clf,X,0,1)
print(clf.score(X2,Y2))
```

0.589



Noisy classification

In [17]:
```
DIMENSIONS = 2

def noise(x, flip_prob=.25):
    if np.random.uniform() < flip_prob:
        return not x
    return x

#f = lambda x : x[0] <= 0.5
f = lambda x : noise(x[0] <= 0.5)

SAMPLES = 40
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```

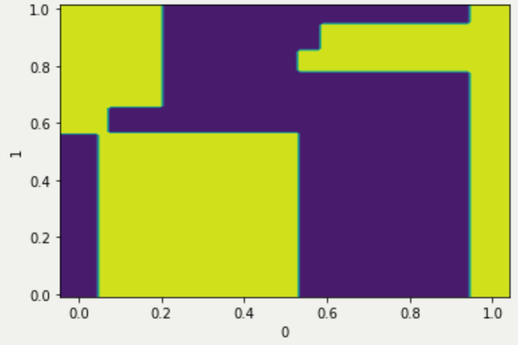


In [32]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=25)
clf.fit(X,Y)
#print(tree.export_text(clf))
```

Out[32]: DecisionTreeClassifier(max_leaf_nodes=25)

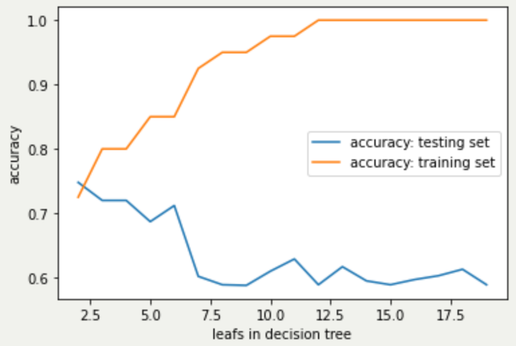
In [33]:
```
draw_decision_area(clf,X,0,1)
print(clf.score(X2,Y2))
```

0.613



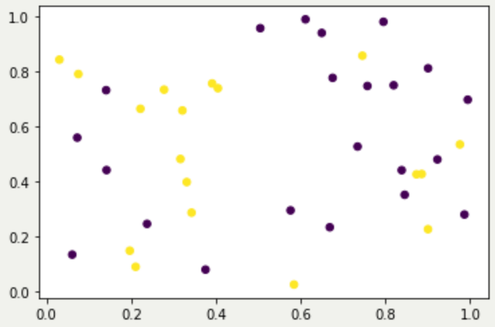
Noisy classification

In [39]:
```
xs,ys,zs = [],[],[]

for leafs in range(2,20):
    clf = DecisionTreeClassifier(max_leaf_nodes=leafs)
    clf.fit(X,Y)

    xs.append(leafs)
    ys.append(clf.score(X2,Y2))
    zs.append(clf.score(X,Y))

plt.plot(xs,ys);
plt.plot(xs,zs);
plt.legend(["accuracy: testing set","accuracy: training set"])
plt.xlabel('leafs in decision tree')
plt.ylabel("accuracy");
```



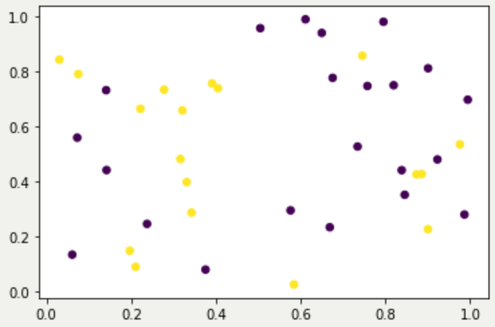
High–dimensional example

In [108]:
```
DIMENSIONS = 50

SAMPLES = 40
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y);

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```



High–dimensional example

In [108]:
```
DIMENSIONS = 50

SAMPLES = 40
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y);

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```

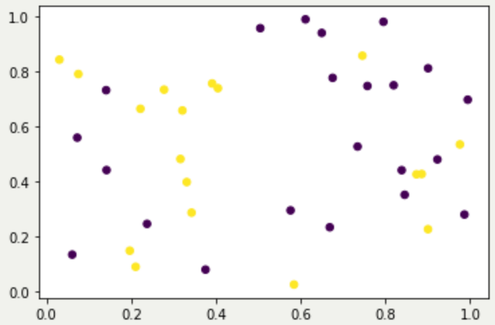


In [113]:
```
dimension = 2
clf=DecisionTreeClassifier(max_leaf_nodes=2)
clf.fit(X[:,:dimension],Y)
print(tree.export_text(clf))
print(clf.score(X2[:,:dimension],Y2))
```

```
|--- feature_0 <= 0.45
|   |--- class: True
|--- feature_0 >  0.45
|   |--- class: False

0.727
```

High–dimensional example

In [108]:
```
DIMENSIONS = 50

SAMPLES = 40
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y);

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```

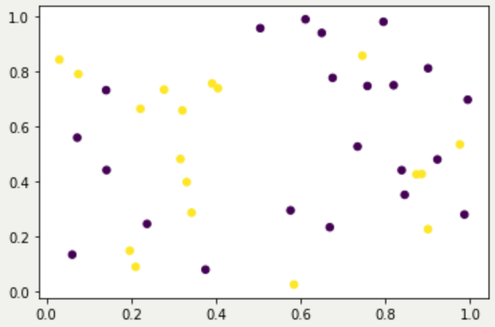


In [114]:
```
dimension = 4
clf=DecisionTreeClassifier(max_leaf_nodes=2)
clf.fit(X[:,:dimension],Y)
print(tree.export_text(clf))
print(clf.score(X2[:,:dimension],Y2))
```

```
|--- feature_0 <= 0.45
|   |--- class: True
|--- feature_0 >  0.45
|   |--- class: False

0.727
```

High–dimensional example

In [108]:
```
DIMENSIONS = 50

SAMPLES = 40
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y);

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```

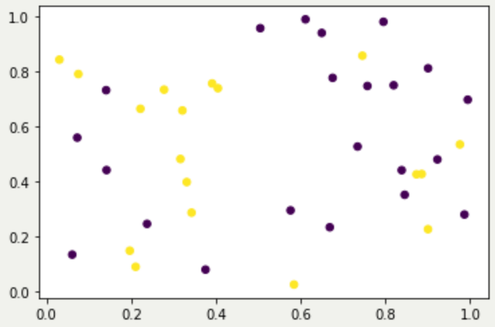


In [115]:
```
dimension = 8
clf=DecisionTreeClassifier(max_leaf_nodes=2)
clf.fit(X[:,:dimension],Y)
print(tree.export_text(clf))
print(clf.score(X2[:,:dimension],Y2))
```

```
|--- feature_0 <= 0.45
|   |--- class: True
|--- feature_0 >  0.45
|   |--- class: False

0.727
```

High–dimensional example

In [108]:
```
DIMENSIONS = 50

SAMPLES = 40
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y);

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```

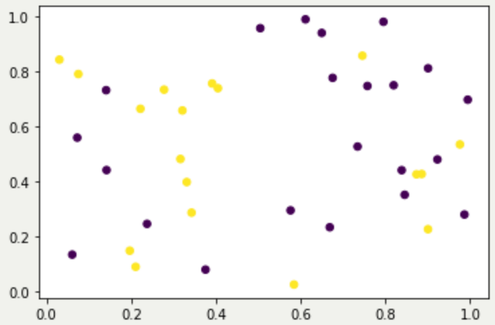


In [116]:
```
dimension = 16
clf=DecisionTreeClassifier(max_leaf_nodes=2)
clf.fit(X[:,:dimension],Y)
print(tree.export_text(clf))
print(clf.score(X2[:,:dimension],Y2))
```

```
|--- feature_12 <= 0.88
|   |--- class: False
|--- feature_12 >  0.88
|   |--- class: True

0.494
```

High–dimensional example

In [108]:
```
DIMENSIONS = 50

SAMPLES = 40
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y);

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```

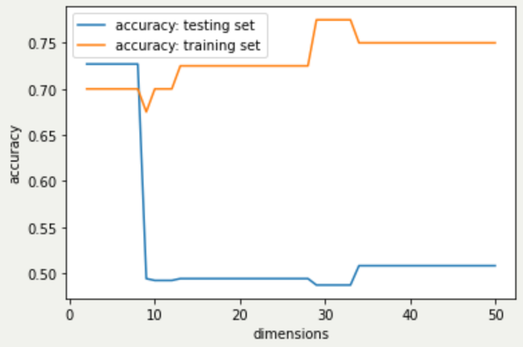


In [117]:
```
dimension = 32
clf=DecisionTreeClassifier(max_leaf_nodes=2)
clf.fit(X[:,:dimension],Y)
print(tree.export_text(clf))
print(clf.score(X2[:,:dimension],Y2))
```

```
|--- feature_28 <= 0.50
|   |--- class: False
|--- feature_28 >  0.50
|   |--- class: True

0.487
```

High–dimensional example

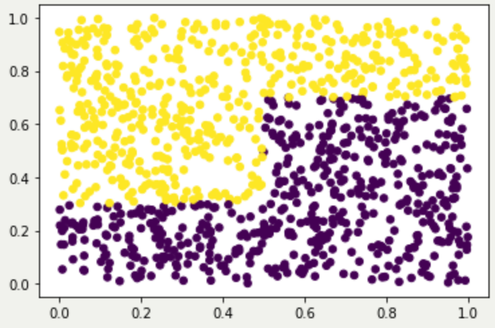
In [122]:
```
xs,ys,zs = [],[],[]

LEAFS = 2
for d in range(2,DIMENSIONS+1):
    clf = DecisionTreeClassifier(
        max_leaf_nodes=LEAFS)
    clf.fit(X[:,:d],Y)

    xs.append(d)
    ys.append(clf.score(X2[:,:d],Y2))
    zs.append(clf.score(X[:,:d],Y))

plt.plot(xs,ys)
plt.plot(xs,zs)
plt.legend(["accuracy: testing set","accuracy: training set"])
plt.xlabel("dimensions")
plt.ylabel("accuracy");
```



More complicated function

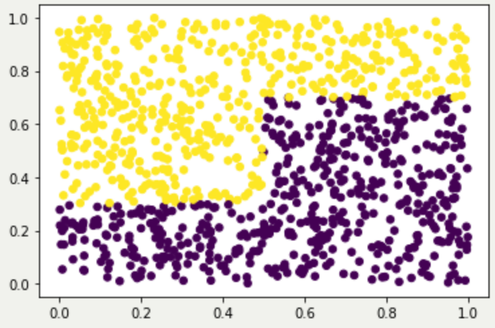
In [126]:

```
DIMENSIONS = 2

def f(x):
    if x[0] < 0.5:
        return x[1] > 0.3
    else:
        return x[1] > 0.7

g = lambda x : noise(f(x))

SAMPLES = 1000
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```



More complicated function

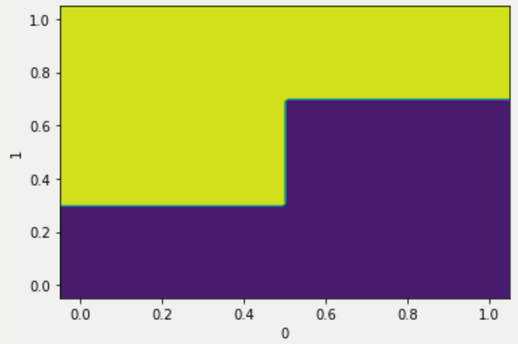
In [126]:
```
DIMENSIONS = 2

def f(x):
    if x[0] < 0.5:
        return x[1] > 0.3
    else:
        return x[1] > 0.7

g = lambda x : noise(f(x))

SAMPLES = 1000
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```



In [128]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=10)
clf.fit(X,Y)
print(tree.export_text(clf))
print(clf.score(X2,Y2))
```

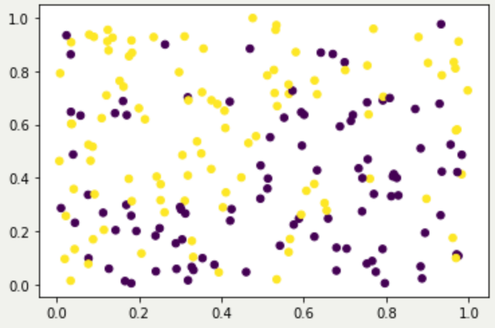
```
|--- feature_1 <= 0.30
|   |--- class: False
|--- feature_1 >  0.30
|   |--- feature_0 <= 0.50
|   |   |--- class: True
|   |--- feature_0 >  0.50
|   |   |--- feature_1 <= 0.70
|   |   |   |--- class: False
|   |   |--- feature_1 >  0.70
|   |   |   |--- class: True

0.998
```

In [129]:
```
draw_decision_area(clf,X,0,1)
```



More complicated function

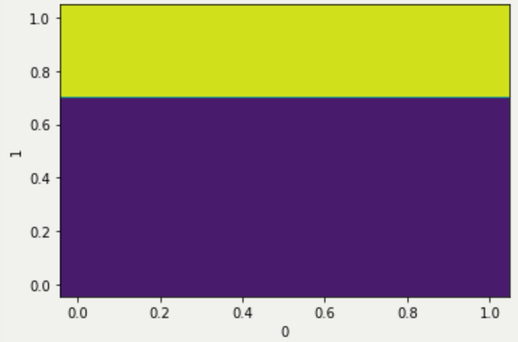
In [136]:
```
DIMENSIONS = 2

def f(x):
    if x[0] < 0.5:
        return x[1] > 0.3
    else:
        return x[1] > 0.7

g = lambda x : noise(f(x))

SAMPLES = 200
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [g(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [g(x) for x in X2]
```



In [137]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=2)
clf.fit(X,Y)
print(tree.export_text(clf))
print(clf.score(X2,Y2))
```

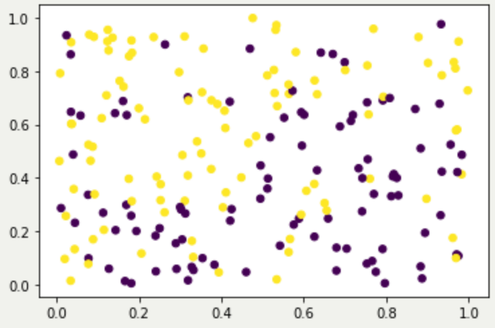
```
|--- feature_1 <= 0.70
|   |--- class: False
|--- feature_1 >  0.70
|   |--- class: True

0.629
```

In [138]:
```
draw_decision_area(clf,X,0,1)
```



More complicated function

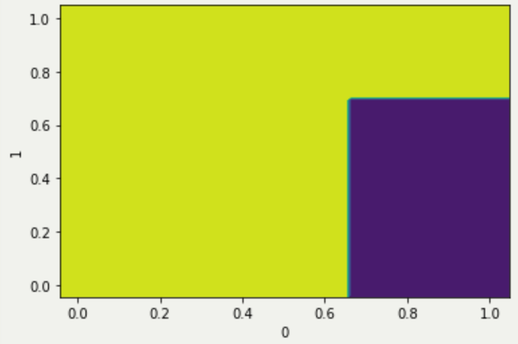
In [136]:
```
DIMENSIONS = 2

def f(x):
    if x[0] < 0.5:
        return x[1] > 0.3
    else:
        return x[1] > 0.7

g = lambda x : noise(f(x))

SAMPLES = 200
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [g(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [g(x) for x in X2]
```



In [139]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=3)
clf.fit(X,Y)
print(tree.export_text(clf))
print(clf.score(X2,Y2))
```

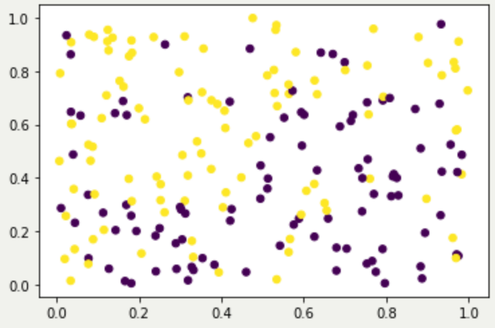
```
|--- feature_1 <= 0.70
|   |--- feature_0 <= 0.66
|   |   |--- class: True
|   |--- feature_0 >  0.66
|   |   |--- class: False
|--- feature_1 >  0.70
|   |--- class: True

0.61
```

In [140]:
```
draw_decision_area(clf,X,0,1)
```



More complicated function

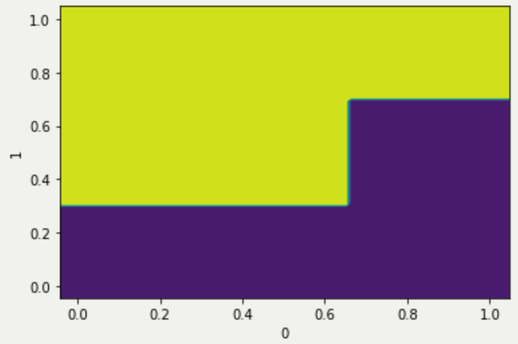
In [136]:
```
DIMENSIONS = 2

def f(x):
    if x[0] < 0.5:
        return x[1] > 0.3
    else:
        return x[1] > 0.7

g = lambda x : noise(f(x))

SAMPLES = 200
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [g(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [g(x) for x in X2]
```



In [141]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=4)
clf.fit(X,Y)
print(tree.export_text(clf))
print(clf.score(X2,Y2))
```

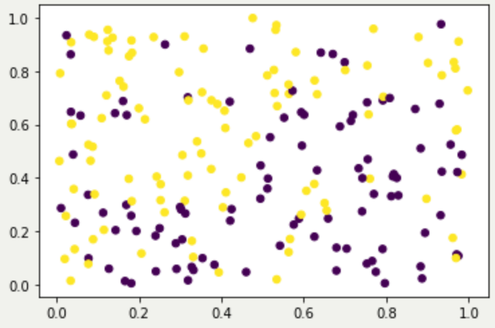
```
|--- feature_1 <= 0.70
|   |--- feature_0 <= 0.66
|   |   |--- feature_1 <= 0.30
|   |   |   |--- class: False
|   |   |--- feature_1 >  0.30
|   |   |   |--- class: True
|   |--- feature_0 >  0.66
|   |   |--- class: False
|--- feature_1 >  0.70
|   |--- class: True

0.708
```

In [142]:
```
draw_decision_area(clf,X,0,1)
```



More complicated function

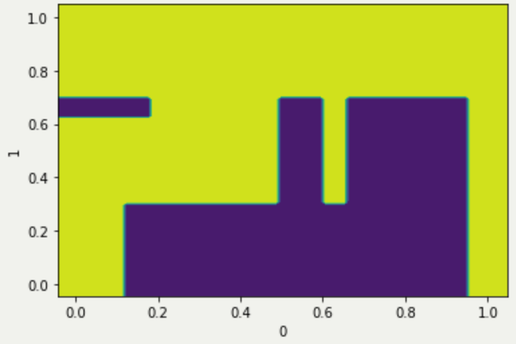
In [136]:
```
DIMENSIONS = 2

def f(x):
    if x[0] < 0.5:
        return x[1] > 0.3
    else:
        return x[1] > 0.7

g = lambda x : noise(f(x))

SAMPLES = 200
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [g(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [g(x) for x in X2]
```



In [146]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=10)
clf.fit(X,Y)
#print(tree.export_text(clf))
print(clf.score(X2,Y2))
```

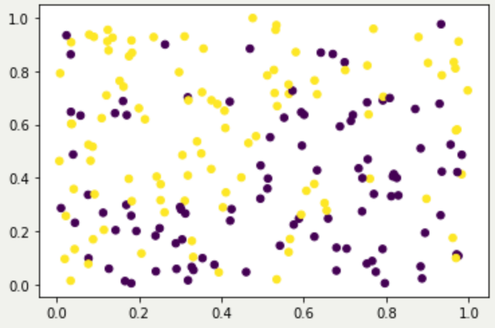
```
0.69
```

In [147]:
```
draw_decision_area(clf,X,0,1)
```



More complicated function

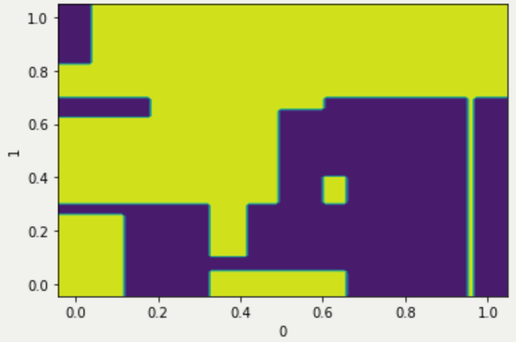
In [136]:
```
DIMENSIONS = 2

def f(x):
    if x[0] < 0.5:
        return x[1] > 0.3
    else:
        return x[1] > 0.7

g = lambda x : noise(f(x))

SAMPLES = 200
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [g(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 1000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [g(x) for x in X2]
```



In [148]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=20)
clf.fit(X,Y)
#print(tree.export_text(clf))
print(clf.score(X2,Y2))
```

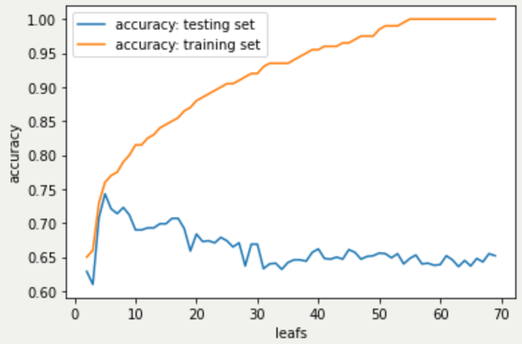
0.685

In [149]:
```
draw_decision_area(clf,X,0,1)
```



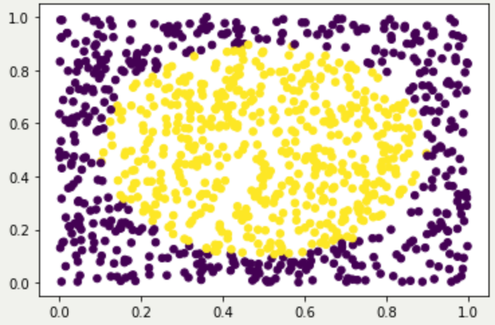
In [152]:
```
xs,ys,zs = [],[],[]

for leafs in range(2,70):
    clf = DecisionTreeClassifier(max_leaf_nodes=leafs)
    clf.fit(X,Y)

    xs.append(leafs)
    ys.append(clf.score(X2,Y2))
    zs.append(clf.score(X,Y))

plt.plot(xs,ys);
plt.plot(xs,zs);
plt.legend(["accuracy: testing set","accuracy: training set"])
plt.xlabel("leafs")
plt.ylabel("accuracy")
```

Out[152]:
```
Text(0, 0.5, 'accuracy')
```



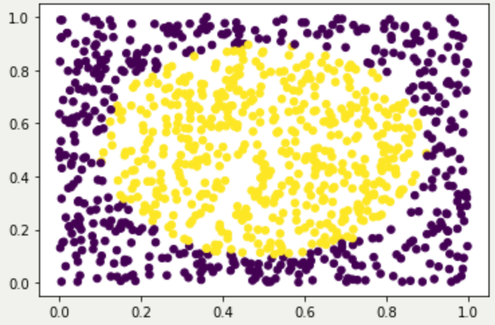
Another example

In [153]:
```
DIMENSIONS = 2

f = lambda x : (x[0]-0.5)**2 + (x[1]-0.5)**2 < 0.16

g = lambda x : noise(f(x),.15)

SAMPLES = 1000
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 10000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```



Another example

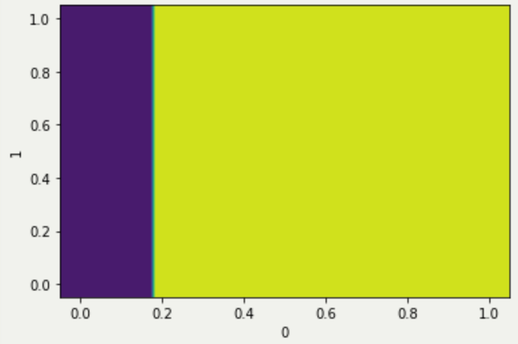
In [153]:
```
DIMENSIONS = 2

f = lambda x : (x[0]-0.5)**2 + (x[1]-0.5)**2 < 0.16

g = lambda x : noise(f(x),.15)

SAMPLES = 1000
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 10000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```



In [155]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=2)
clf.fit(X,Y)
#print(tree.export_text(clf))
print(clf.score(X2,Y2))
```

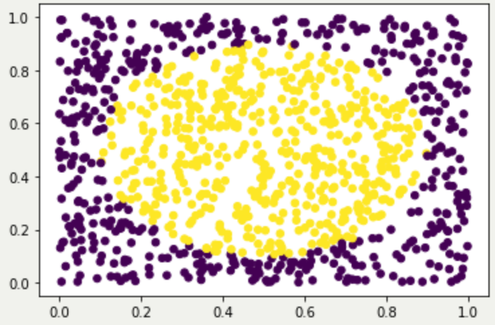
```
0.6346
```

In [156]:
```
draw_decision_area(clf,X,0,1)
```



Another example

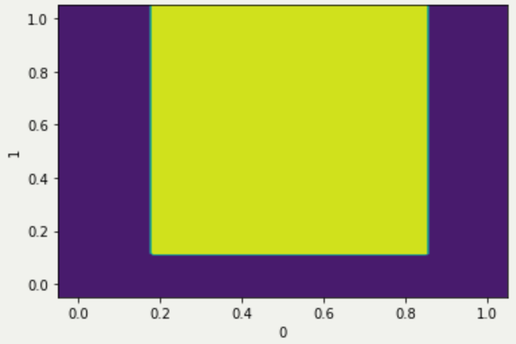
In [153]:
```
DIMENSIONS = 2

f = lambda x : (x[0]-0.5)**2 + (x[1]-0.5)**2 < 0.16

g = lambda x : noise(f(x),.15)

SAMPLES = 1000
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 10000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```



In [157]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=4)
clf.fit(X,Y)
#print(tree.export_text(clf))
print(clf.score(X2,Y2))
```

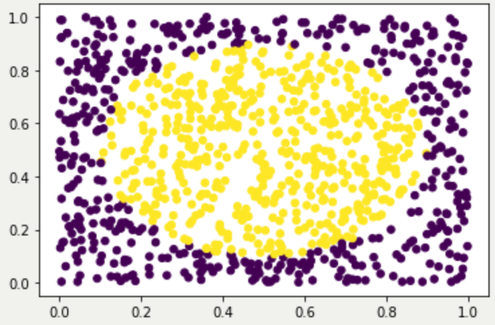
```
0.8246
```

In [158]:
```
draw_decision_area(clf,X,0,1)
```



Another example

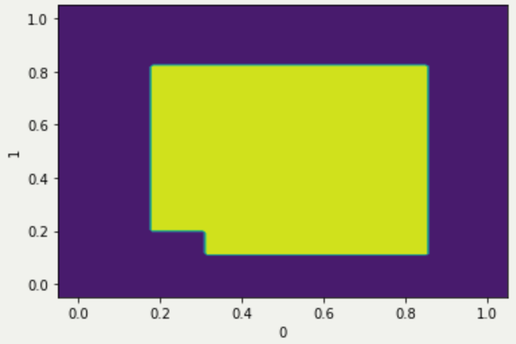
In [153]:
```
DIMENSIONS = 2

f = lambda x : (x[0]-0.5)**2 + (x[1]-0.5)**2 < 0.16

g = lambda x : noise(f(x),.15)

SAMPLES = 1000
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 10000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```



In [159]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=8)
clf.fit(X,Y)
#print(tree.export_text(clf))
print(clf.score(X2,Y2))
```

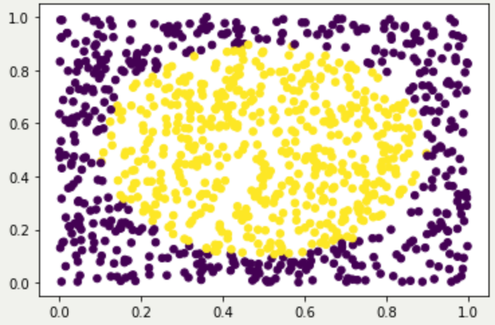
```
0.9039
```

In [160]:
```
draw_decision_area(clf,X,0,1)
```



Another example

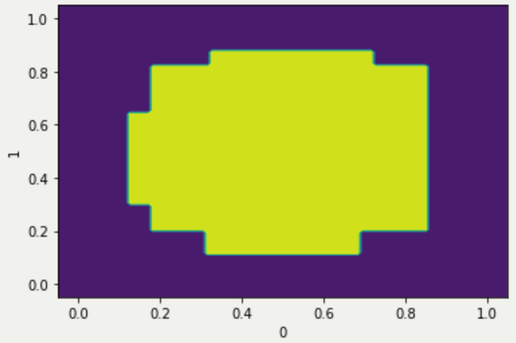
In [153]:
```
DIMENSIONS = 2

f = lambda x : (x[0]-0.5)**2 + (x[1]-0.5)**2 < 0.16

g = lambda x : noise(f(x),.15)

SAMPLES = 1000
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 10000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```



In [161]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=16)
clf.fit(X,Y)
#print(tree.export_text(clf))
print(clf.score(X2,Y2))
```

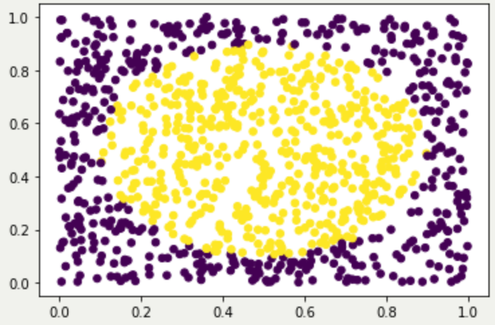
0.9465

In [162]:
```
draw_decision_area(clf,X,0,1)
```



Another example

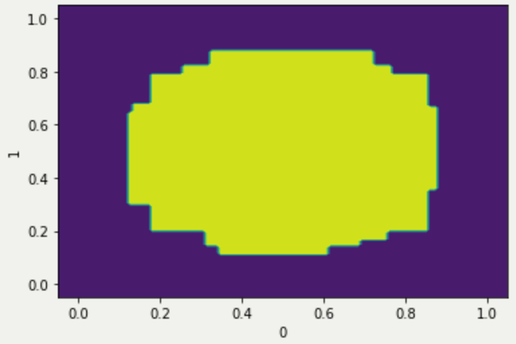
In [153]:
```
DIMENSIONS = 2

f = lambda x : (x[0]-0.5)**2 + (x[1]-0.5)**2 < 0.16

g = lambda x : noise(f(x),.15)

SAMPLES = 1000
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [f(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 10000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [f(x) for x in X2]
```



In [163]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=32)
clf.fit(X,Y)
#print(tree.export_text(clf))
print(clf.score(X2,Y2))
```

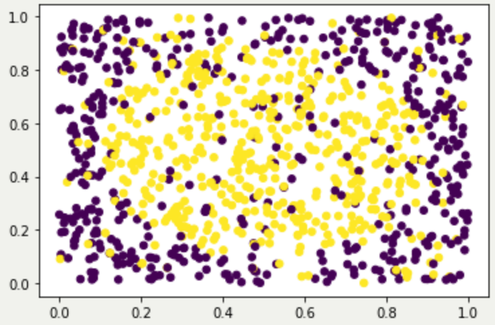
```
0.9619
```

In [164]:
```
draw_decision_area(clf,X,0,1)
```



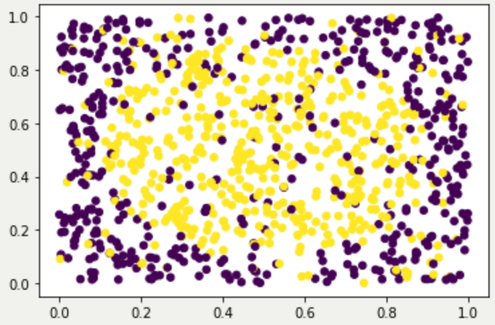
Another example

In [165]:
```
DIMENSIONS = 2

f = lambda x : (x[0]-0.5)**2 + (x[1]-0.5)**2 < 0.16

g = lambda x : noise(f(x),.15)

SAMPLES = 1000
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [g(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 10000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [g(x) for x in X2]
```



Another example

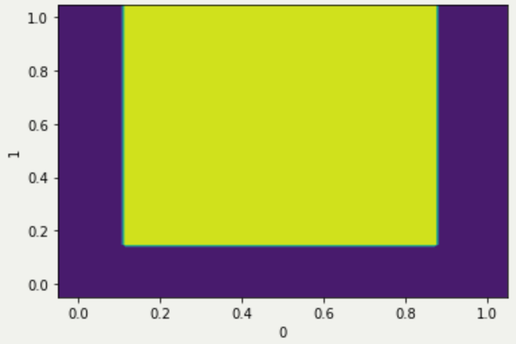
In [165]:
```
DIMENSIONS = 2

f = lambda x : (x[0]-0.5)**2 + (x[1]-0.5)**2 < 0.16

g = lambda x : noise(f(x),.15)

SAMPLES = 1000
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [g(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 10000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [g(x) for x in X2]
```



In [166]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=4)
clf.fit(X,Y)
#print(tree.export_text(clf))
print(clf.score(X2,Y2))
```

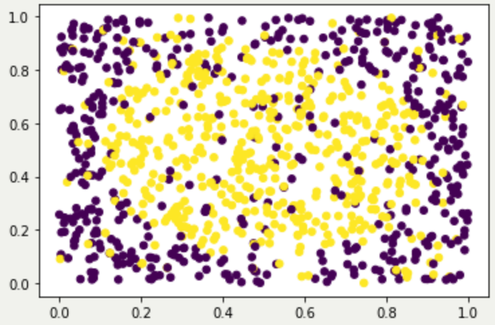
```
0.719
```

In [167]:
```
draw_decision_area(clf,X,0,1)
```



Another example

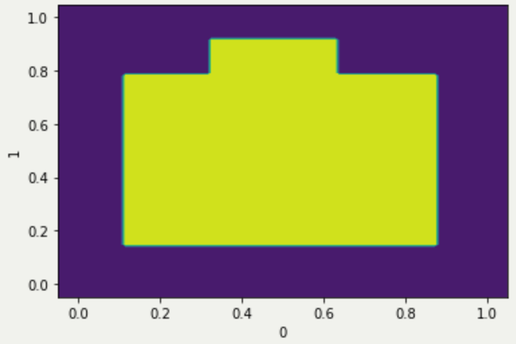
In [165]:
```
DIMENSIONS = 2

f = lambda x : (x[0]-0.5)**2 + (x[1]-0.5)**2 < 0.16

g = lambda x : noise(f(x),.15)

SAMPLES = 1000
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [g(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 10000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [g(x) for x in X2]
```



In [168]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=8)
clf.fit(X,Y)
#print(tree.export_text(clf))
print(clf.score(X2,Y2))
```

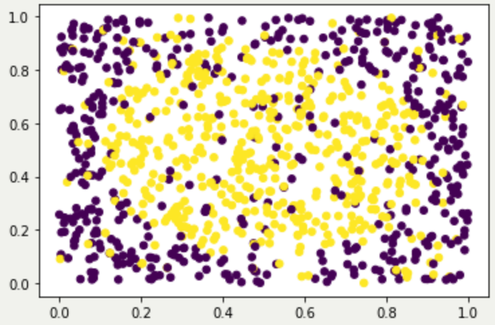
```
0.7917
```

In [169]:
```
draw_decision_area(clf,X,0,1)
```



Another example

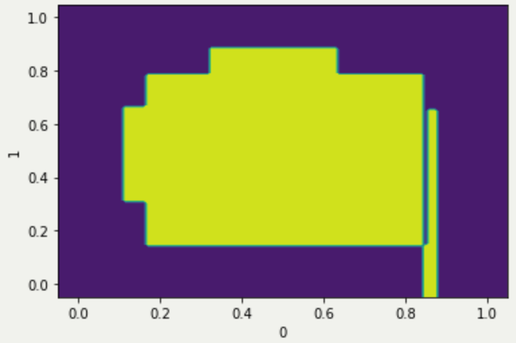
In [165]:
```
DIMENSIONS = 2

f = lambda x : (x[0]-0.5)**2 + (x[1]-0.5)**2 < 0.16

g = lambda x : noise(f(x),.15)

SAMPLES = 1000
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [g(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 10000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [g(x) for x in X2]
```



In [170]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=16)
clf.fit(X,Y)
#print(tree.export_text(clf))
print(clf.score(X2,Y2))
```

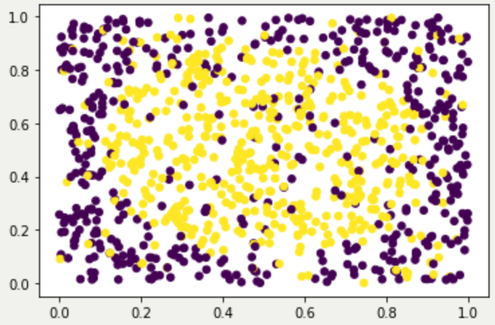
0.806

In [171]:
```
draw_decision_area(clf,X,0,1)
```



Another example

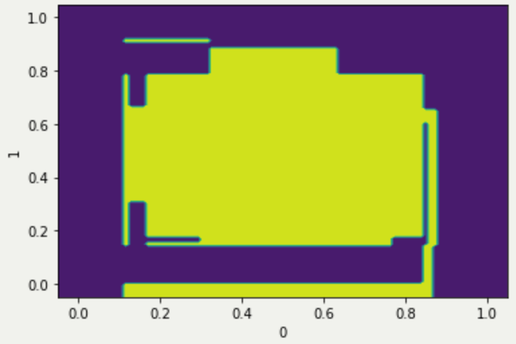
In [165]:
```
DIMENSIONS = 2

f = lambda x : (x[0]-0.5)**2 + (x[1]-0.5)**2 < 0.16

g = lambda x : noise(f(x),.15)

SAMPLES = 1000
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [g(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 10000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [g(x) for x in X2]
```



In [172]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=32)
clf.fit(X,Y)
#print(tree.export_text(clf))
print(clf.score(X2,Y2))
```

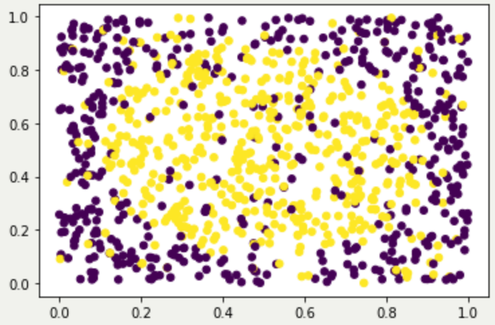
```
0.8052
```

In [173]:
```
draw_decision_area(clf,X,0,1)
```



Another example

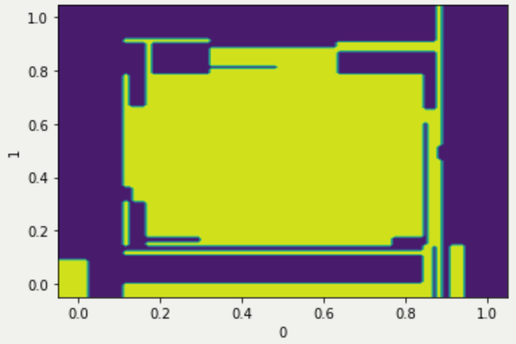
In [165]:
```
DIMENSIONS = 2

f = lambda x : (x[0]-0.5)**2 + (x[1]-0.5)**2 < 0.16

g = lambda x : noise(f(x),.15)

SAMPLES = 1000
X = np.random.uniform(size=(SAMPLES,DIMENSIONS))
Y = [g(x) for x in X]
plt.scatter(X[:,0],X[:,1],30,c = Y)

TEST_SAMPLES = 10000
X2 = np.random.uniform(size=(TEST_SAMPLES,DIMENSIONS))
Y2 = [g(x) for x in X2]
```



In [174]:
```
clf = DecisionTreeClassifier(max_leaf_nodes=64)
clf.fit(X,Y)
#print(tree.export_text(clf))
print(clf.score(X2,Y2))
```

```
0.7811
```

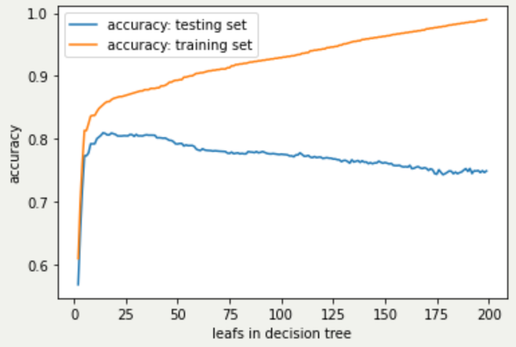
In [175]:
```
draw_decision_area(clf,X,0,1)
```



Another example

In [177]:
```
xs,ys,zs = [],[],[]

for leafs in range(2,200):
    clf = DecisionTreeClassifier(max_leaf_nodes=leafs)
    clf.fit(X,Y)

    xs.append(leafs)
    ys.append(clf.score(X2,Y2))
    zs.append(clf.score(X,Y))

plt.plot(xs,ys);
plt.plot(xs,zs);
plt.legend(["accuracy: testing set","accuracy: training set"])
plt.xlabel("leafs in decision tree")
plt.ylabel("accuracy")
```

Out[177]:
```
Text(0, 0.5, 'accuracy')
```



Underfitting

- Model not expressive enough to capture the problem
- Or a solution found does not match the problem

Possible solutions:

- Try harder to find a better solution
- Add more parameters
- Try a different model that captures the solution

Overfitting

- Predictions adjusted too well to training data
- Error on test data ⋙ error on training data

Possible solutions:

- Don't optimize the model on the training data too much
- Remove features that are too noisy
- Add more training data
- Reduce model complexity

Bias and variance

$$\text{Total learning error} = \text{Bias} + \text{Variance} + \text{noise}$$

Bias: error due to model unable to match the complexity of the problem

Variance: how much the prediction will vary in response to data points

Overfitting: high variance, low bias

Underfitting: high bias, low variance

Important in practice:

- detecting the source of problems: variance vs. bias, overfitting vs. underfitting
- navigating the trade-off and finding the sweet spot