DS-210: Programming for Data Science

Lecture 3: Decision trees (continued).

New students

- We're using Piazza (not Blackboard)
- Links from https://onak.pl/ds210
- Fill out the survey and send it to me

Everyone

- Jupyter Notebook / Jupyter Lab
- You can use it to open lecture slides
- Homework 1 out today (due next Wed)
- Vedaant's office hours: Mondays 3:45-5:45pm @ MCS B51

New students

- We're using Piazza (not Blackboard)
- Links from https://onak.pl/ds210
- Fill out the survey and send it to me

Everyone

- Jupyter Notebook / Jupyter Lab
- You can use it to open lecture slides
- Homework 1 out today (due next Wed)
- Vedaant's office hours: Mondays 3:45-5:45pm @ MCS B51

Last time

- Supervised vs. unsupervised learning
- Decision trees

Function arguments in Python: via position or name

In [1]:
```
# simplest function definition
def foo(x, y, z):
    return x + 10 * y + 100 * z

print(foo(1,2,3))
```

```
321
```

Function arguments in Python: via position or name

In [1]:
```python
# simplest function definition
def foo(x, y, z):
    return x + 10 * y + 100 * z

print(foo(1,2,3))
```

```
321
```

In [2]:
```python
# add default values
def moo(x, y = 0, z = 0):
    return x + 10 * y + 100 * z

# only one argument is mandatory
print(moo(1),moo(1,2),moo(1,2,3))
```

```
1 21 321
```

Function arguments in Python: via position or name

In [1]:
```
# simplest function definition
def foo(x, y, z):
    return x + 10 * y + 100 * z

print(foo(1,2,3))
```

```
321
```

In [2]:
```
# add default values
def moo(x, y = 0, z = 0):
    return x + 10 * y + 100 * z

# only one argument is mandatory
print(moo(1),moo(1,2),moo(1,2,3))
```

```
1 21 321
```

In [3]:
```
# can refer to all or some arguments via variable names
print(foo(z = 3, y = 2, x = 1))
print(foo(1, z = 3, y = 2))
#won't work: print(foo(z = 3, y = 2, 1))
```

```
321
321
```

Function arguments in Python: via position or name

In [1]:
```
# simplest function definition
def foo(x, y, z):
    return x + 10 * y + 100 * z

print(foo(1,2,3))
```

```
321
```

In [2]:
```
# add default values
def moo(x, y = 0, z = 0):
    return x + 10 * y + 100 * z

# only one argument is mandatory
print(moo(1),moo(1,2),moo(1,2,3))
```

```
1 21 321
```

In [3]:
```
# can refer to all or some arguments via variable names
print(foo(z = 3, y = 2, x = 1))
print(foo(1, z = 3, y = 2))
#won't work: print(foo(z = 3, y = 2, 1))
```

```
321
321
```

In [4]:
```
# can arbitrarily skip over arguments
print(moo(1, z = 3))
```

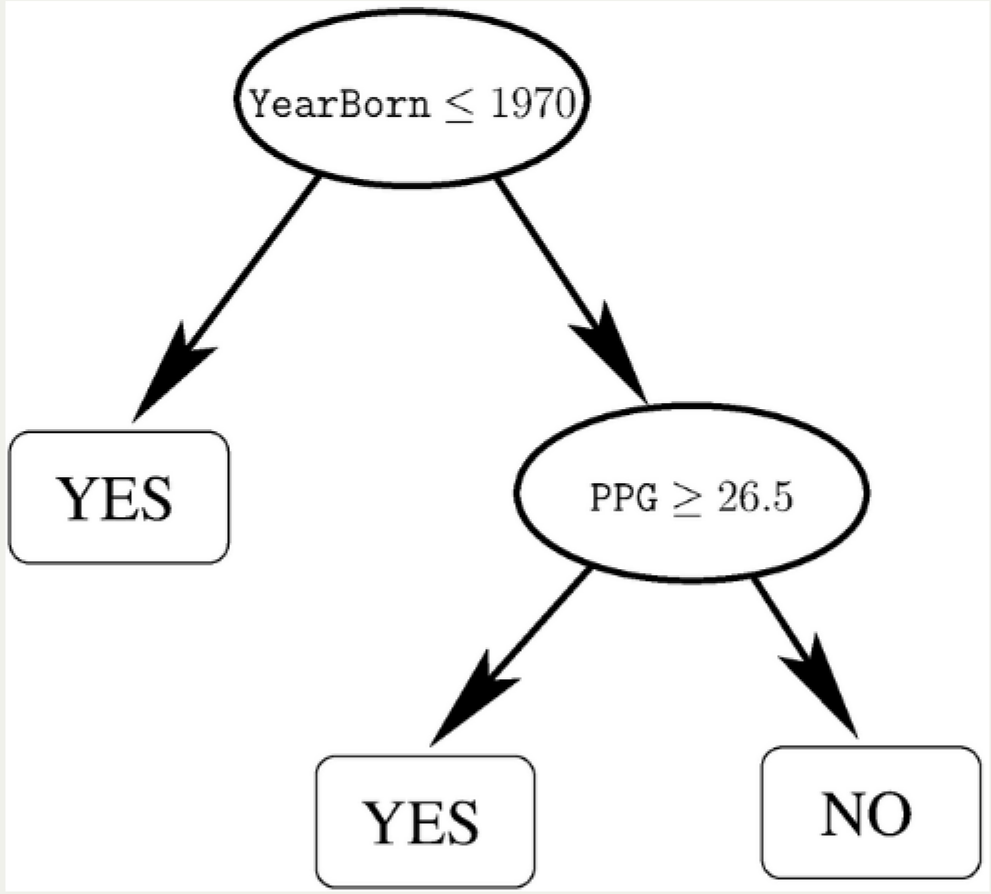
```
301
```

Decision trees

Popular machine learning tool for predictive data analysis:

- start at the root and keep going down
- every internal node labeled with a condition
    - if satisfied, go left
    - if not satisfied, go right
- leafs labeled with predicted labels

Does a player like bluegrass?

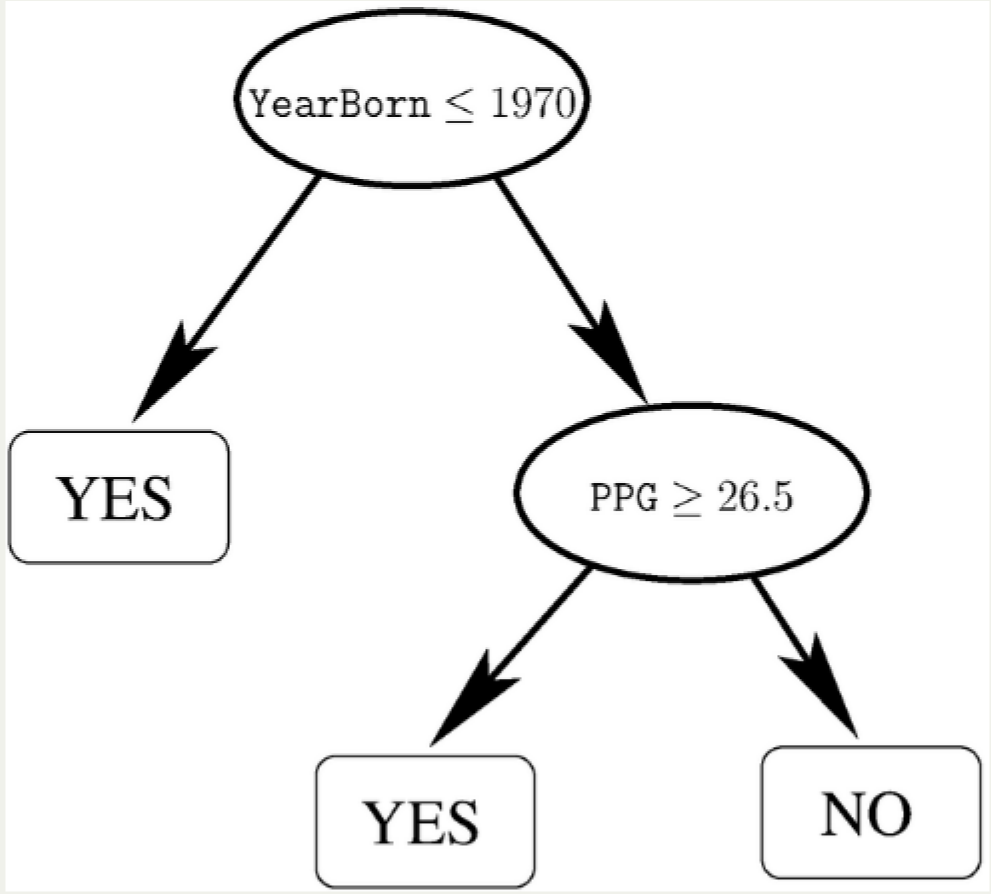


Decision trees

Popular machine learning tool for predictive data analysis:

- start at the root and keep going down
- every internal node labeled with a condition
    - if satisfied, go left
    - if not satisfied, go right
- leafs labeled with predicted labels

Does a player like bluegrass?



Big challenge: finding a decision tree that matches data!

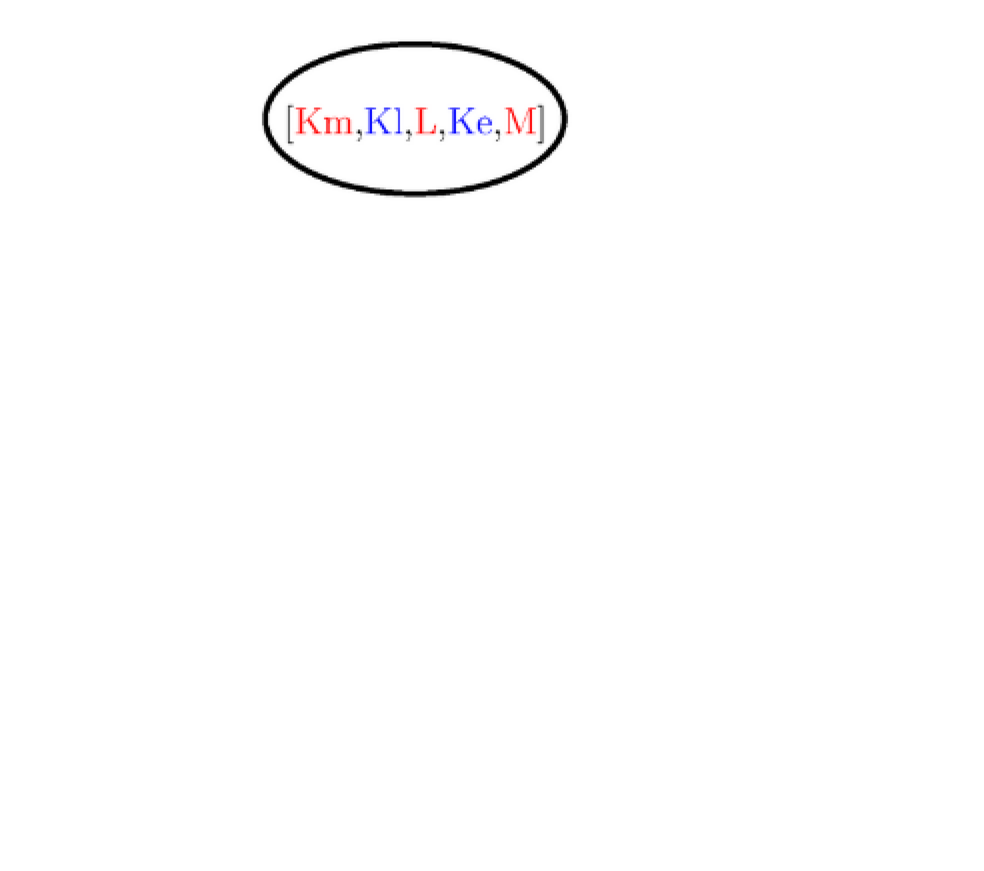
Heuristics for constructing decision trees

- Start from a single node with all samples
- Iterate:
  - select a node
  - use the samples in the node to split it into children
  - pass each sample to respective child
- Label leafs

Heuristics for constructing decision trees

- Start from a single node with all samples
- Iterate:
    - select a node
    - use the samples in the node to split it into children
    - pass each sample to respective child
- Label leafs

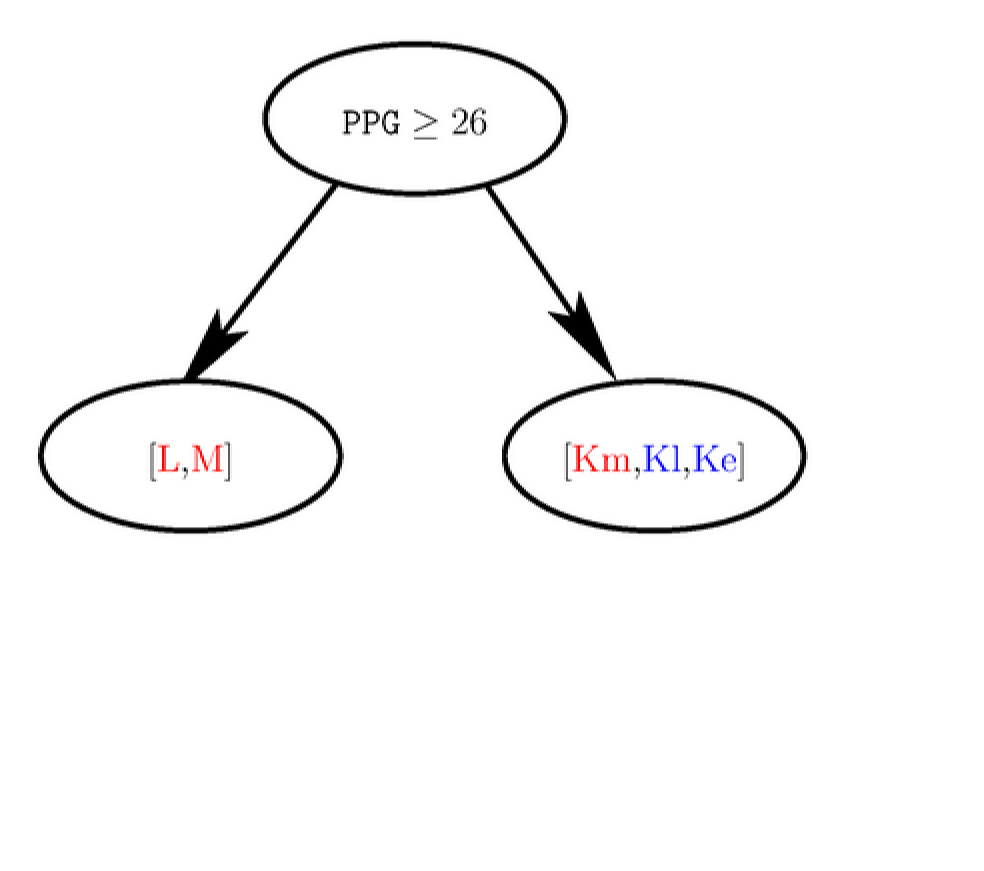
Favorite color?



Heuristics for constructing decision trees

- Start from a single node with all samples
- Iterate:
  - select a node
  - use the samples in the node to split it into children
  - pass each sample to respective child
- Label leafs

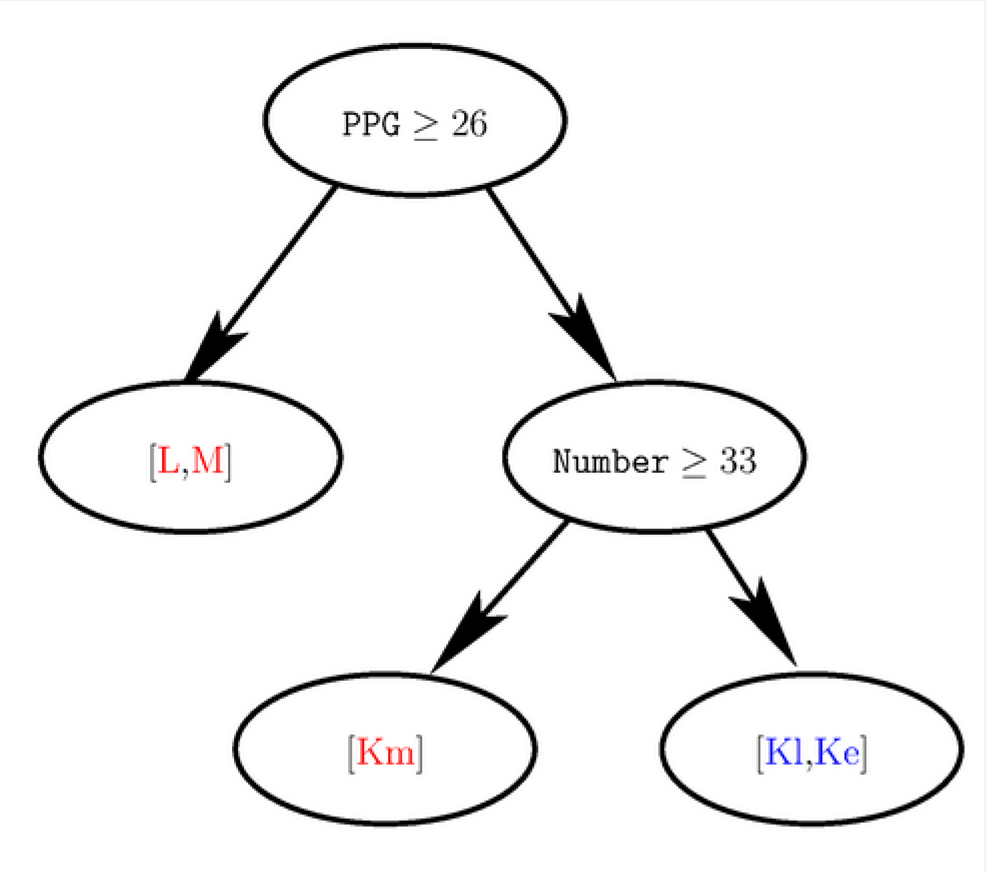
Favorite color?



Heuristics for constructing decision trees

- Start from a single node with all samples
- Iterate:
    - select a node
    - use the samples in the node to split it into children
    - pass each sample to respective child
- Label leafs

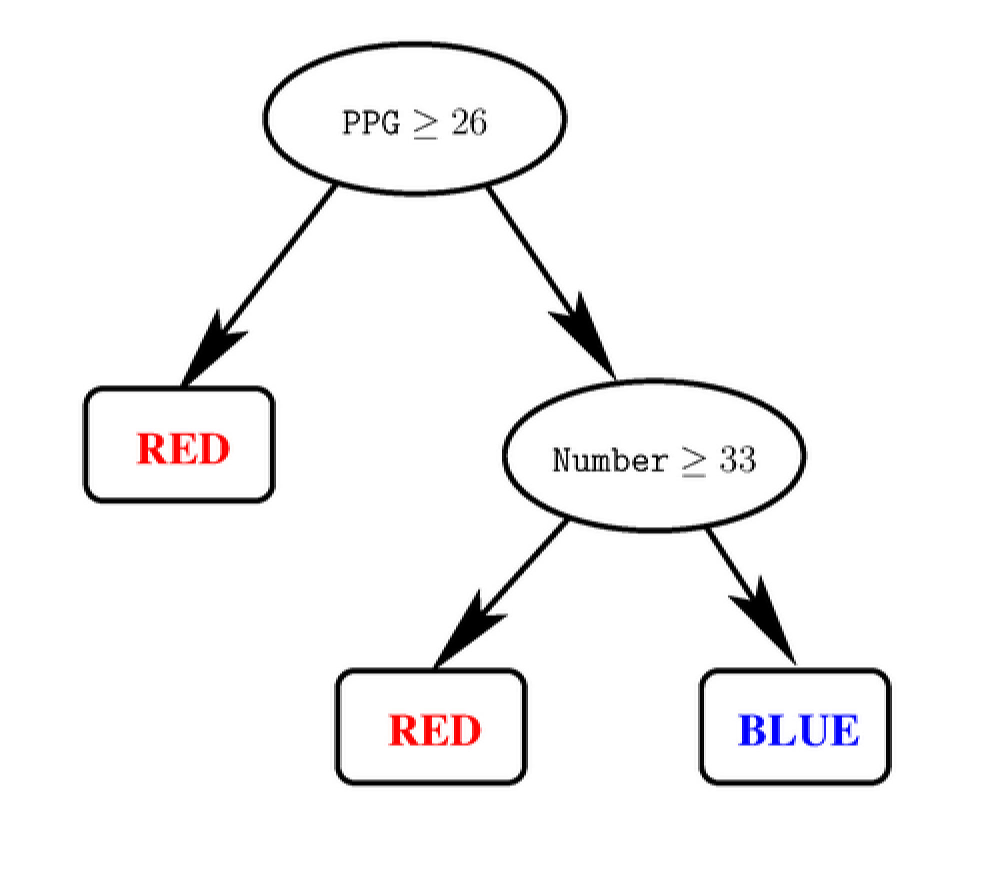
Favorite color?



Heuristics for constructing decision trees

- Start from a single node with all samples
- Iterate:
    - select a node
    - use the samples in the node to split it into children
    - pass each sample to respective child
- Label leafs

Favorite color?



Split selection

- Typical heuristic: select a split that improves classification most
- Various measures of "goodness" or "badness":
  - Information gain
  - Ginni impurity
  - Variance

Split selection

- Typical heuristic: select a split that improves classification most
- Various measures of "goodness" or "badness":
  - Information gain
  - Ginni impurity
  - Variance

Popular algorithms

- ID3
- C4.5
- C5.0
- CART (used by `scikit-learn`)

(feature several additional ideas)

Advantages and disadvantages of decision trees

Advantages and disadvantages of decision trees

Advantages:

- easy to interpret
- not much data preparation needed
- categorical and numerical data
- relatively fast

Advantages and disadvantages of decision trees

Advantages:

- easy to interpret
- not much data preparation needed
- categorical and numerical data
- relatively fast

Disadvantages:

- can be very sensitive to data changes
- can create an overcomplicated tree that matches the sample, but not the underlying problem
- hard to find an optimal tree

Decision tree construction using `scikit-learn`

Note: ignore machine learning context for now

First, we read our sample data and add information who likes pizza

Decision tree construction using `scikit-learn`

Note: ignore machine learning context for now

First, we read our sample data and add information who likes pizza

In [5]:
```
# Let's read our sample data
import pandas as pd
data = pd.read_csv('players.csv')
data
```

Out[5]:

	Name	Number	PPG	YearBorn	TotalPoints
0	Kareem	33	24.6	1947	38387
1	Karl	32	25.0	1963	36928
2	LeBron	23	27.0	1984	36381
3	Kobe	24	25.0	1978	33643
4	Michael	23	30.1	1963	32292

Decision tree construction using `scikit-learn`

Note: ignore machine learning context for now

First, we read our sample data and add information who likes pizza

In [5]:
```python
# Let's read our sample data
import pandas as pd
data = pd.read_csv('players.csv')
data
```

Out[5]:

	Name	Number	PPG	YearBorn	TotalPoints
0	Kareem	33	24.6	1947	38387
1	Karl	32	25.0	1963	36928
2	LeBron	23	27.0	1984	36381
3	Kobe	24	25.0	1978	33643
4	Michael	23	30.1	1963	32292

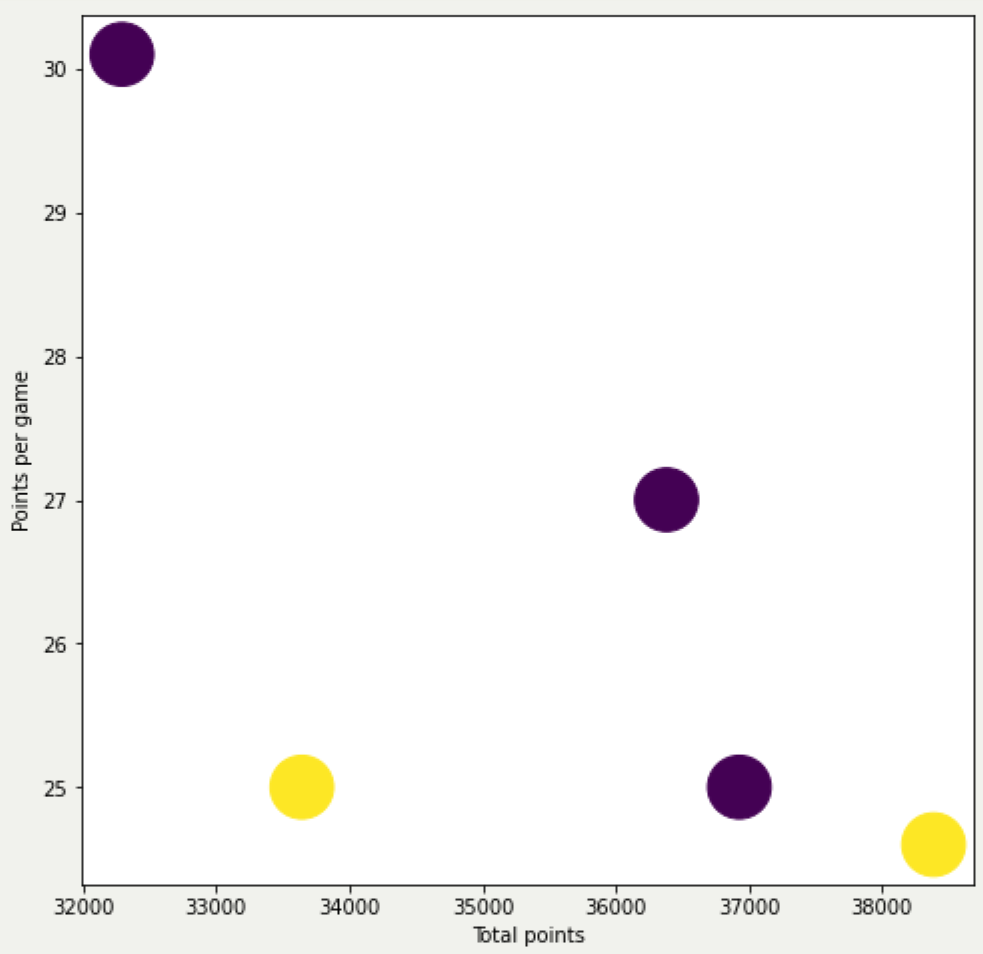
In [6]:
```python
likes_pizza = [1,0,0,1,0]
data['LikesPizza'] = likes_pizza
data
```

Out[6]:

	Name	Number	PPG	YearBorn	TotalPoints	LikesPizza
0	Kareem	33	24.6	1947	38387	1
1	Karl	32	25.0	1963	36928	0
2	LeBron	23	27.0	1984	36381	0
3	Kobe	24	25.0	1978	33643	1
4	Michael	23	30.1	1963	32292	0

Visualized

In [7]:
```
import matplotlib.pyplot as plt
data['radius'] = [1000 for x in data['PPG']]
fig,ax = plt.subplots(figsize=(8,8))
ax.set_xlabel('Total points')
ax.set_ylabel('Points per game')
ax.scatter('TotalPoints','PPG','radius','LikesPizza',data=data);
```



Data selection

- set of inputs: `X`
- set of desired outputs: `y`

In [8]:
```
data
```

Out[8]:

	Name	Number	PPG	YearBorn	TotalPoints	LikesPizza	radius
0	Kareem	33	24.6	1947	38387	1	1000
1	Karl	32	25.0	1963	36928	0	1000
2	LeBron	23	27.0	1984	36381	0	1000
3	Kobe	24	25.0	1978	33643	1	1000
4	Michael	23	30.1	1963	32292	0	1000

Data selection

- set of inputs: `X`
- set of desired outputs: `y`

In [8]:
```
data
```

Out[8]:

	Name	Number	PPG	YearBorn	TotalPoints	LikesPizza	radius
0	Kareem	33	24.6	1947	38387	1	1000
1	Karl	32	25.0	1963	36928	0	1000
2	LeBron	23	27.0	1984	36381	0	1000
3	Kobe	24	25.0	1978	33643	1	1000
4	Michael	23	30.1	1963	32292	0	1000

In [9]:
```
features = ['PPG','YearBorn','TotalPoints']
X = data[features]
y = data['LikesPizza']
print(X,y,sep='\n\n')
```

```
    PPG  YearBorn  TotalPoints
0  24.6      1947        38387
1  25.0      1963        36928
2  27.0      1984        36381
3  25.0      1978        33643
4  30.1      1963        32292

0    1
1    0
2    0
3    1
4    0
Name: LikesPizza, dtype: int64
```

Decision tree construction

In [10]:
```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(max_leaf_nodes = 3,\
                             random_state = 0)
clf = clf.fit(X,y)
```

Visualizing the outcome

In [11]:
```
from sklearn import tree
text = tree.export_text(clf,feature_names = features)
print(text)
```

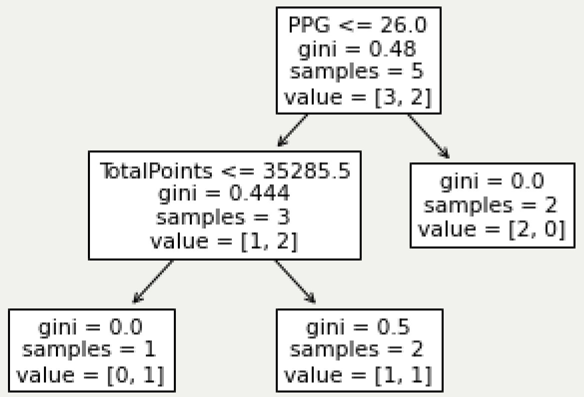
```
|--- PPG <= 26.00
|   |--- TotalPoints <= 35285.50
|   |   |--- class: 1
|   |--- TotalPoints >  35285.50
|   |   |--- class: 0
|--- PPG >  26.00
|   |--- class: 0
```

Visualizing the outcome

In [11]:
```
from sklearn import tree
text = tree.export_text(clf,feature_names = features)
print(text)
```

```
|--- PPG <= 26.00
|   |--- TotalPoints <= 35285.50
|   |   |--- class: 1
|   |--- TotalPoints >  35285.50
|   |   |--- class: 0
|--- PPG >  26.00
|   |--- class: 0
```

In [12]:
```
tree.plot_tree(clf,feature_names = features);
```



Closing remarks

- **Suggested reading:** https://scikit-learn.org/stable/modules/tree.html
- **Next time:** using this in the context of a data science pipeline
- Homework 1 out tonight (announcement to be posted on Piazza)