Data Science & ML Course

Lesson #20 Clustering

Ivanovitch Silva

December, 2018

Update from repository

```
git clone https://github.com/ivanovitchm/datascience2machinelearning.git
```

Or ....

```
git pull
```

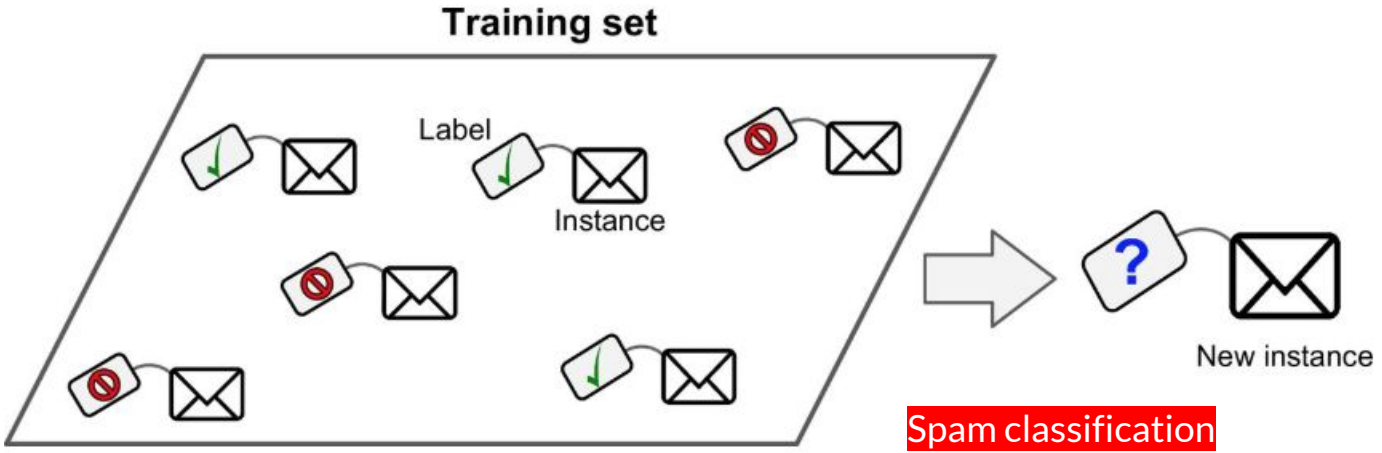
Agenda

1. Clustering Basic
2. K-Means
3. Case study: senators votes, nba


Got Google's machine learning code
Now need 1000000000000
training samples
TheBigLead

Supervised Learning

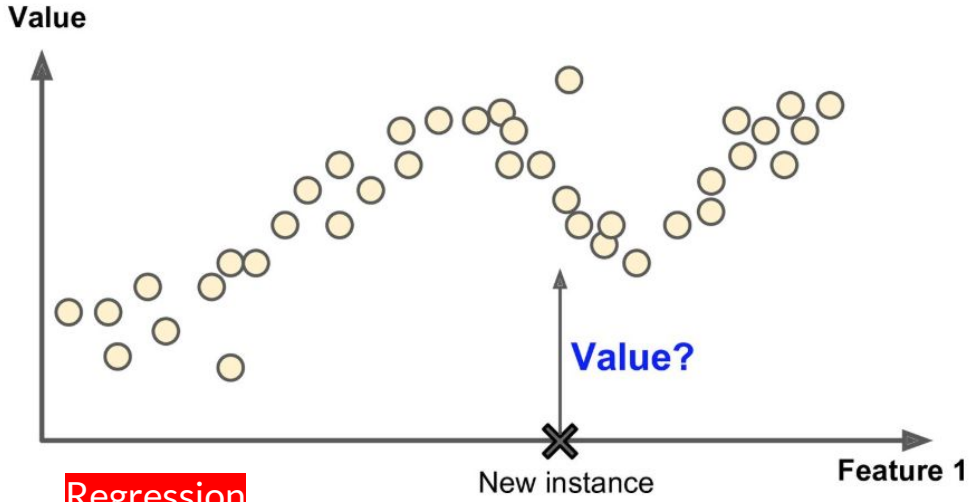
In supervised learning, the training data you feed to the algorithm includes the desired solutions, called labels.



Spam classification

Supervised Learning

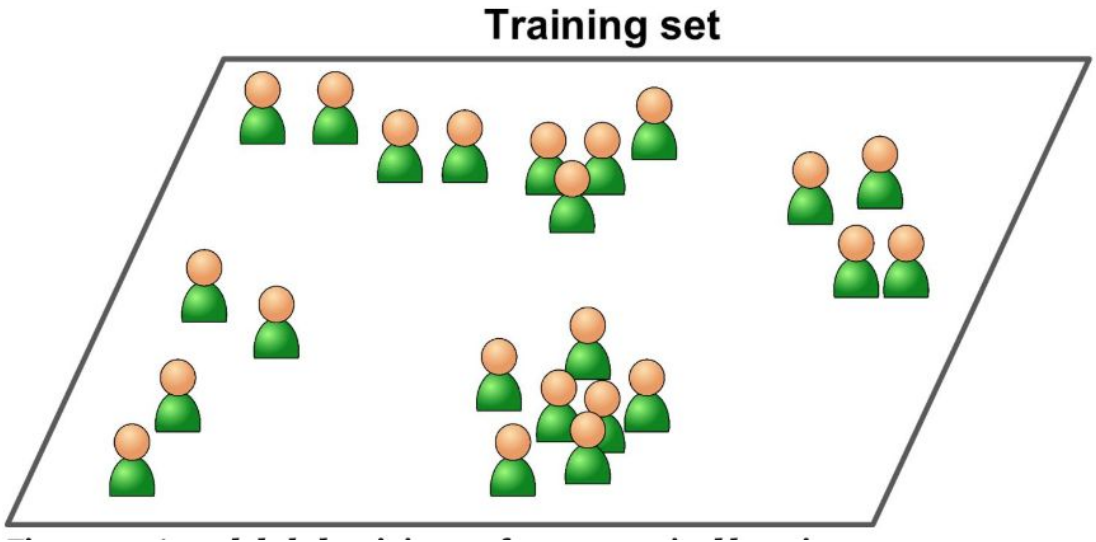
Another typical task is to predict a target numeric value, such as the price of a car, given a set of features (mileage, age, brand, etc) called predictors.



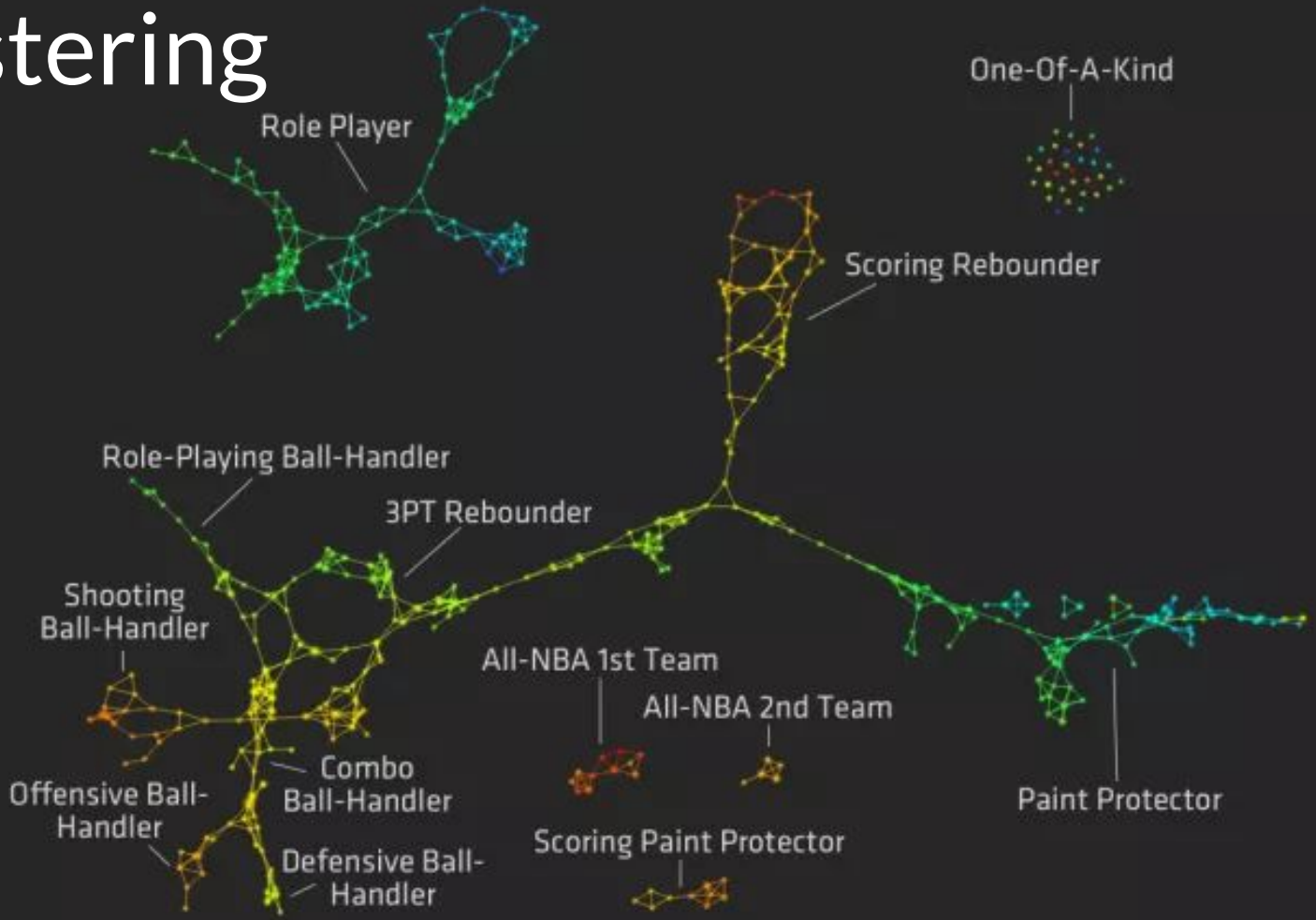
Regression

Unsupervised Learning

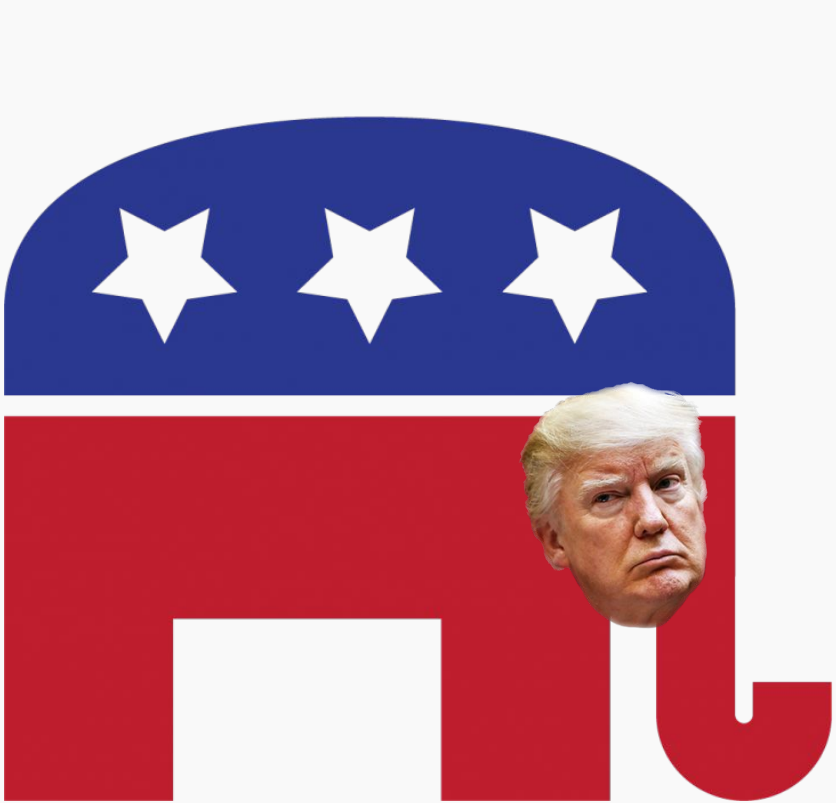
In unsupervised learning, as you might guess, the training data is unlabeled. The system tries to learn without a teacher.



Clustering


One-Of-A-Kind
Role Player
Scoring Rebounder
Role-Playing Ball-Handler
3PT Rebounder
Shooting
Ball-Handler
All-NBA 1st Team
All-NBA 2nd Team
Combo
Ball-Handler
Offensive Ball-
Handler
Defensive Ball-
Handler
Scoring Paint Protector
Paint Protector

Republican vs. Democrat



114th US Congress

January 3, 2015, to January 3, 2017

Senator voted No : 0
Senator voted Yes: 1
Senator abstained: 0.5

	name	party	state	00001	00004	00005	00006	00007	00008	00009
0	Alexander	R	TN	0.0	1.0	1.0	1.0	1.0	0.0	0.0
1	Ayotte	R	NH	0.0	1.0	1.0	1.0	1.0	0.0	0.0
2	Baldwin	D	WI	1.0	0.0	0.0	1.0	0.0	1.0	0.0
3	Barrasso	R	WY	0.0	1.0	1.0	1.0	1.0	0.0	1.0
4	Bennet	D	CO	0.0	0.0	0.0	1.0	0.0	1.0	0.0

```
Name: party, dtype: int64
00001    0.325
00004    0.575
00005    0.535
00006    0.945
00007    0.545
00008    0.415
00009    0.545
00010    0.985
00020    0.525
00026    0.545
00032    0.410
00038    0.480
00039    0.510
00044    0.460
00047    0.370
dtype: float64
```

Exploring Data

R	54
D	44
I	2

Distance between Senators

```
00001,00004,00005,00006,00007,00008,00009,00010,00020,00026,00032,00038,00039,00044,00047
0,1,1,1,1,0,0,1,1,1,0,0,0,0,0
0,1,1,1,1,0,0,1,0,1,0,1,0,1,0
```

$$d = \sqrt{(0-0)^2 + (1-1)^2 + (1-1)^2 + (1-1)^2 + (1-1)^2 + (0-0)^2 + \ldots + (0-0)^2}$$

```
from sklearn.metrics.pairwise import euclidean_distances

euclidean_distances(votes.iloc[0,3:].values.reshape(1,-1),
                    votes.iloc[1,3:].values.reshape(1,-1))
```

Initial Clustering

```
import pandas as pd
from sklearn.cluster import KMeans

kmeans_model = KMeans(n_clusters=2, random_state=1)
senator_distances = kmeans_model.fit_transform(votes.iloc[:, 3:])
```

```
array([[3.12141628, 1.3134775 ],
       [2.6146248 , 2.05339992],
       [0.33960656, 3.41651746],
       [3.42004795, 0.24198446],
       [1.43833966, 2.96866004],
       [0.33960656, 3.41651746],
       [3.42004795, 0.24198446],
       [0.33960656, 3.41651746],
```

Exploring the Clusters

```
import numpy as np
is_smoker =        [0,1,1,0,0,1]
has_lung_cancer = [1,0,1,0,1,0]

pd.crosstab(np.array(has_lung_cancer),
            np.array(is_smoker),
            colnames=["has_lung_cancer"],
            rownames=["is_smoker"])
```

has_lung_cancer	0	1
is_smoker		
0	1	2
1	2	1

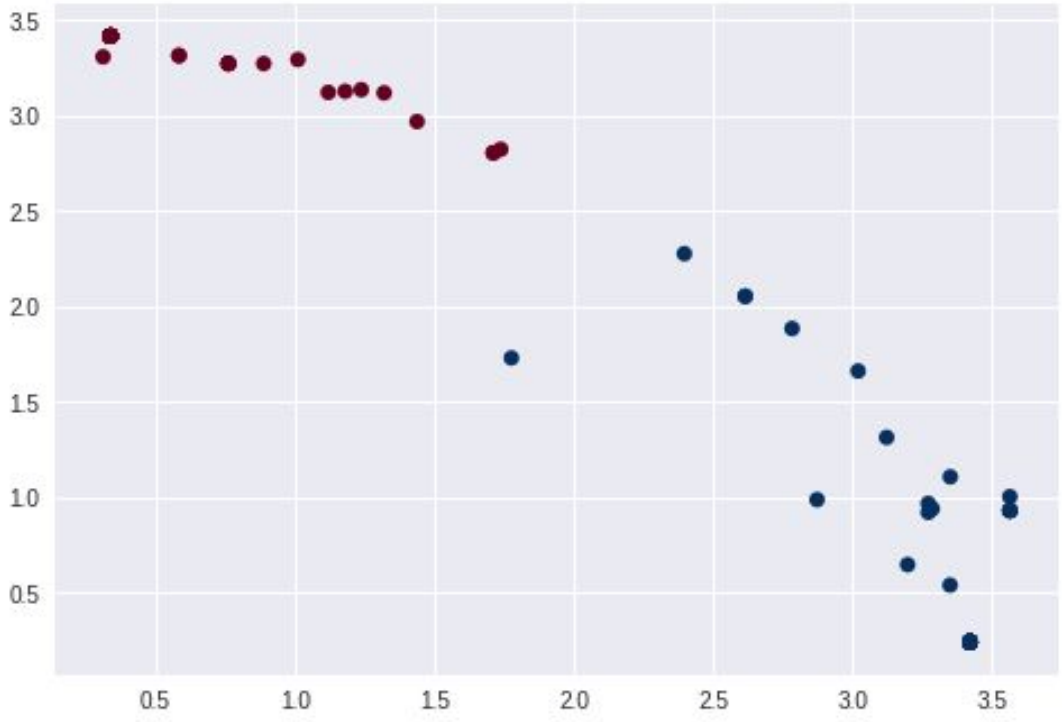
Exploring the Clusters

```
1 kmeans_model.labels_
```

```
array([1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0], dtype=int32)
```

party	D	I	R
row_0			
0	41	2	0
1	3	0	54

Plotting out the clusters



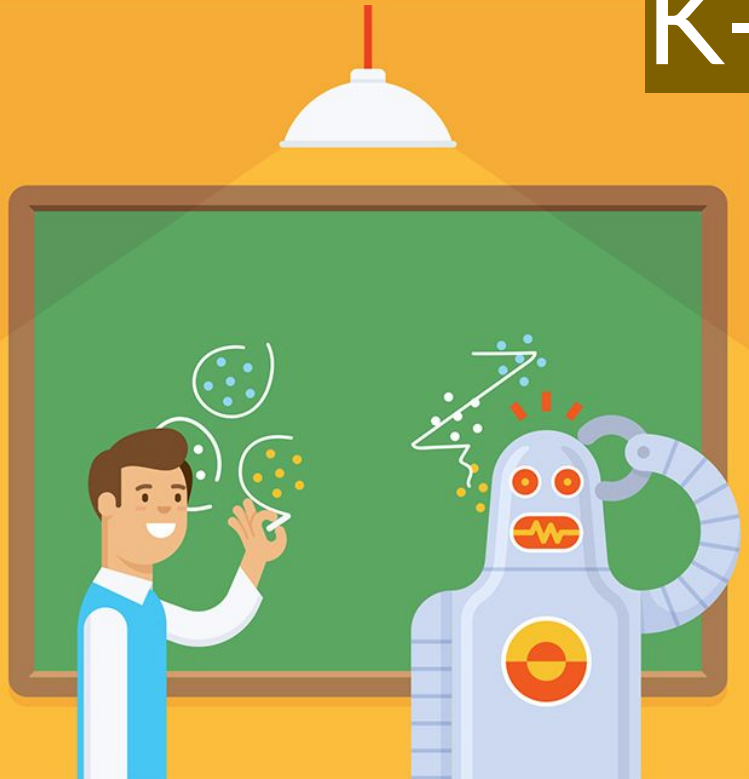
Finding the most extreme

```
[
    [ 3.12141628,  1.3134775 ], # Slightly moderate, far from cluster 1, close to cluster 2.
    [ 2.6146248 ,  2.05339992], # Moderate, far from cluster 1, far from cluster 2.
    [ 0.33960656,  3.41651746], # Somewhat extreme, very close to cluster 1, very far from cluster 2.
    [ 3.42004795,  0.24198446], # Fairly extreme, very far from cluster 1, very close to cluster 2.
    ...
]
```

$$3.12 + 1.31 = 4.43$$
$$2.61 + 2.05 = 4.66$$
$$0.34 + 3.41 = 3.75$$
$$3.42 + 0.24 = 3.66$$

$$3.12^3 + 1.31^3 = 32.62$$
$$2.61^3 + 2.05^3 = 26.39$$
$$0.34^3 + 3.41^3 = 39.69$$
$$3.42^3 + 0.24^3 = 40.01$$

K-Means Clustering

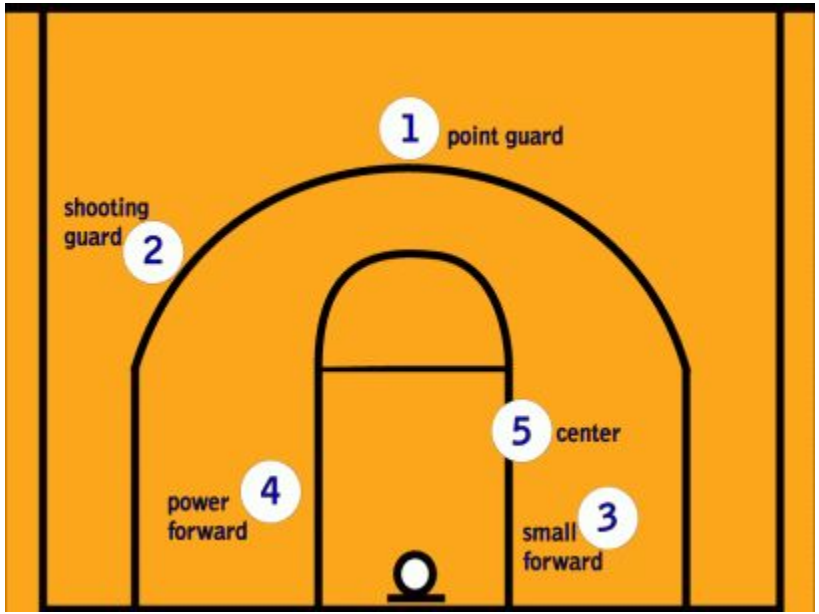



30
WARRIORS
10
23
6
JACK
2
Budweiser

Clustering NBA Players - dataset

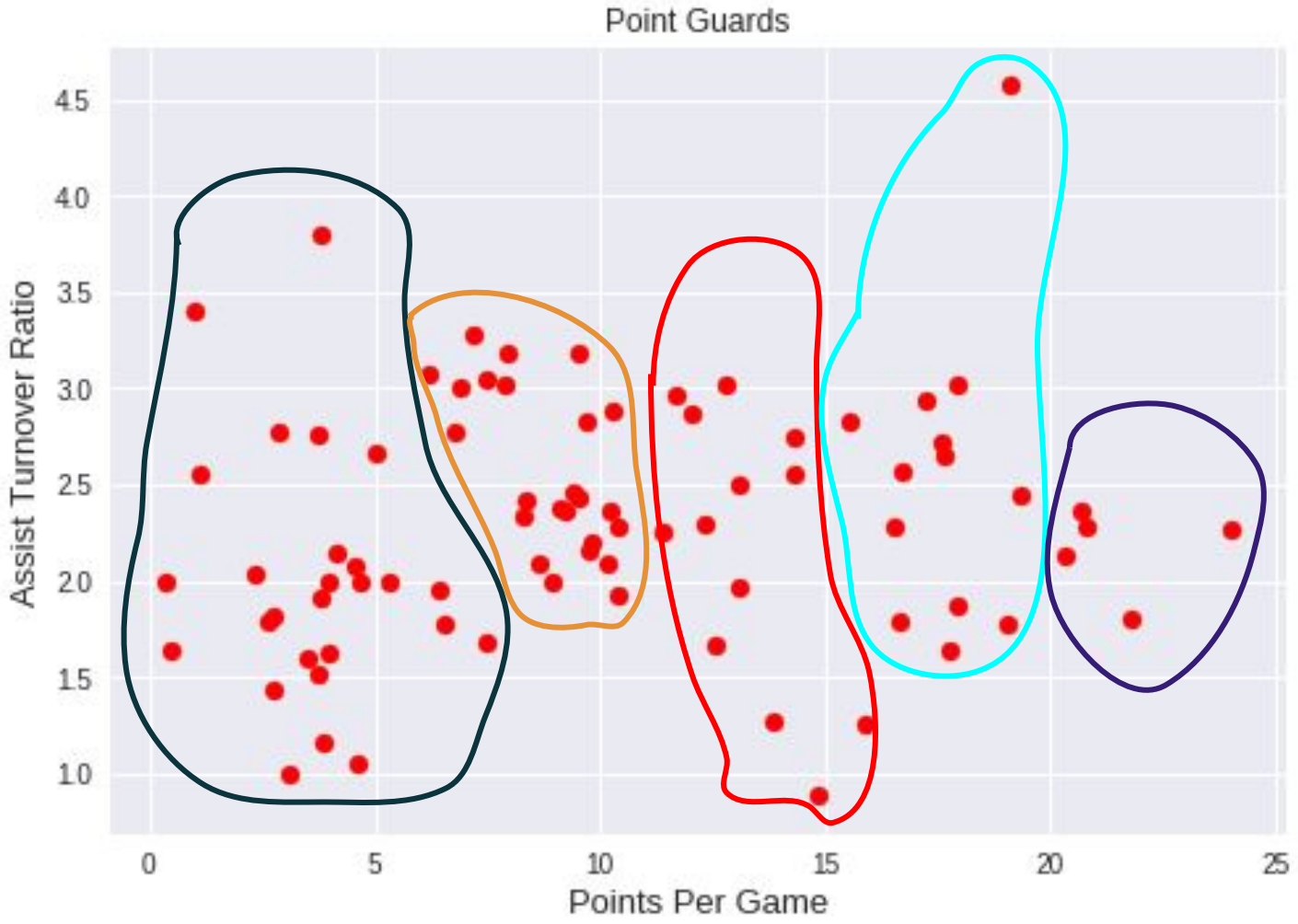
player	pos	g	pts	fg.	ft.	ast	tov
Kevin Durant	SF	81	2593	0.503	0.873	445	285
Carmelo Anthony	PF	77	2112	0.452	0.848	242	198
LeBron James	PF	77	2089	0.567	0.750	488	270
Kevin Love	PF	77	2010	0.457	0.821	341	196
Blake Griffin	PF	80	1930	0.528	0.715	309	224
Stephen Curry	PG	78	1873	0.471	0.885	666	294

Clustering NBA Players - point guard



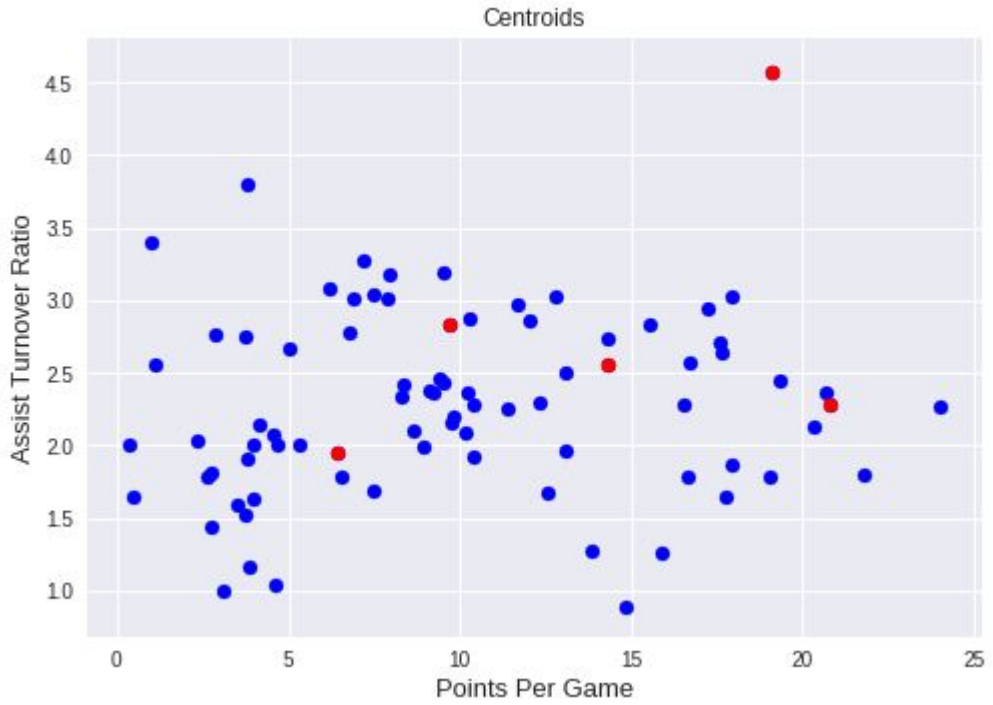
$$\text{points per game (ppg)} = \frac{pts}{g}$$

$$\text{assist turnover ratio (atr)} = \frac{ast}{tov}$$


Point Guards
Assist Turnover Ratio
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
5
10
15
20
25
Points Per Game

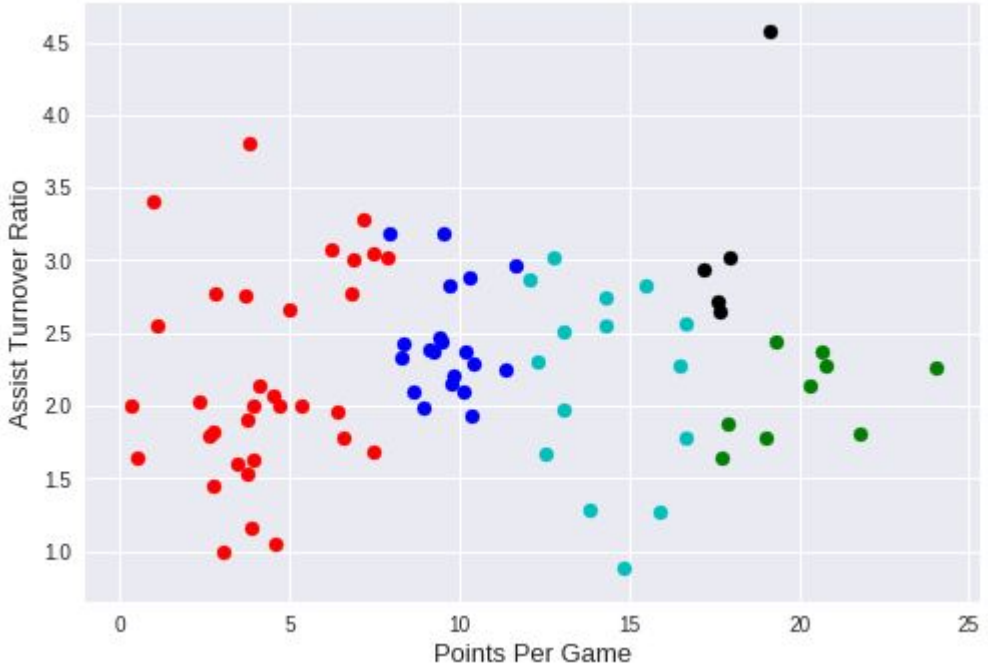
K-Means Algorithm (Step 1)

Assign Points to Cluster



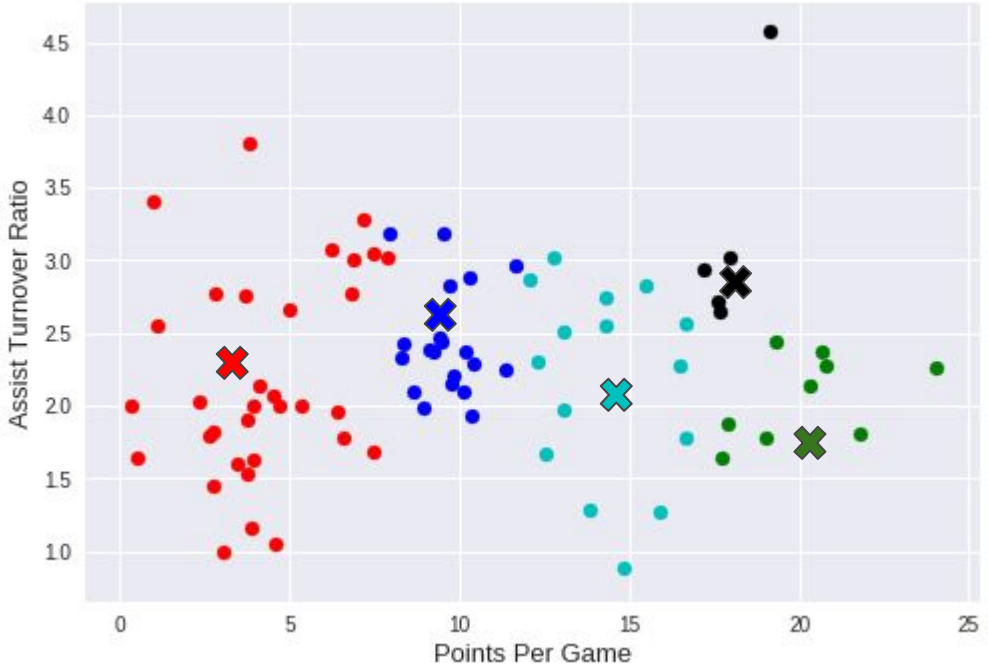
K-Means Algorithm (Step 1)

Euclidean Distance

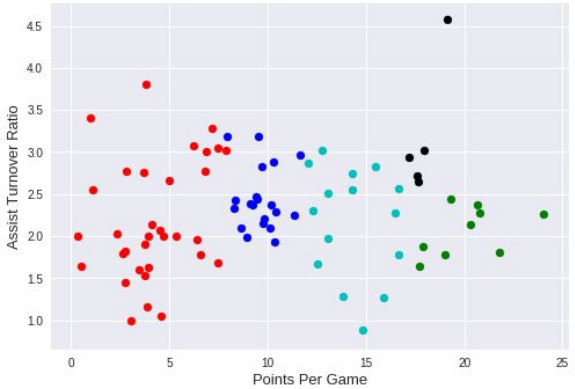


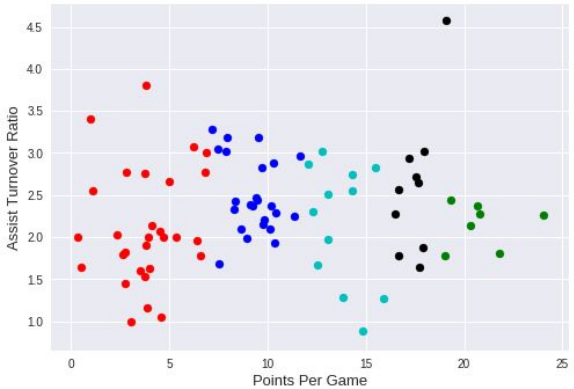
K-Means Algorithm (Step 2)

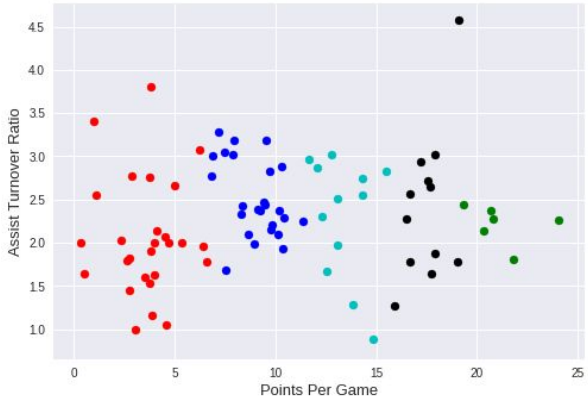
Recalculate the centroids



Repeat Steps 1 and 2 until to converge







Challenges of K-Means

- K-Means doesn't cause massive changes in the makeup of clusters between iterations, meaning that it will always converge and become stable
- Because K-Means is conservative between iterations, where we pick the initial centroids and how we assign the players to clusters initially matters a lot
- Scikit counteract!!!!