NEAREST

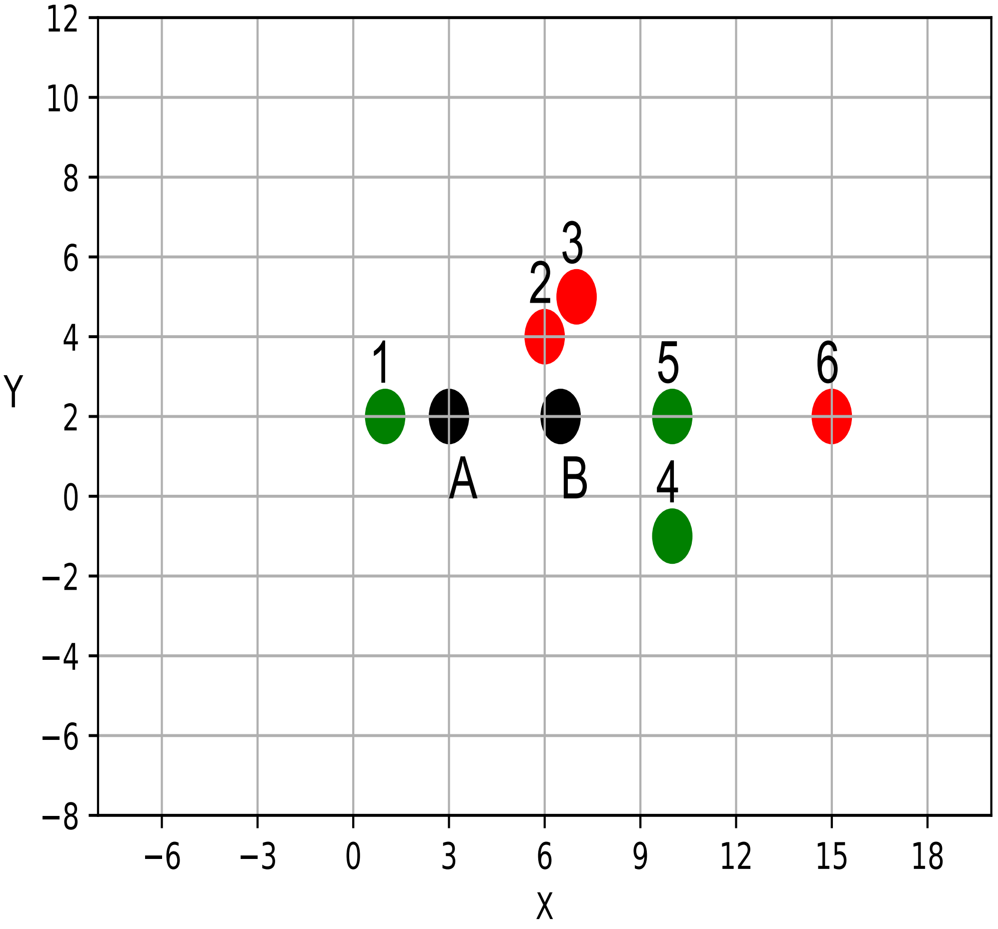
NEIGHBORS

CLASSIFICATION

General Idea

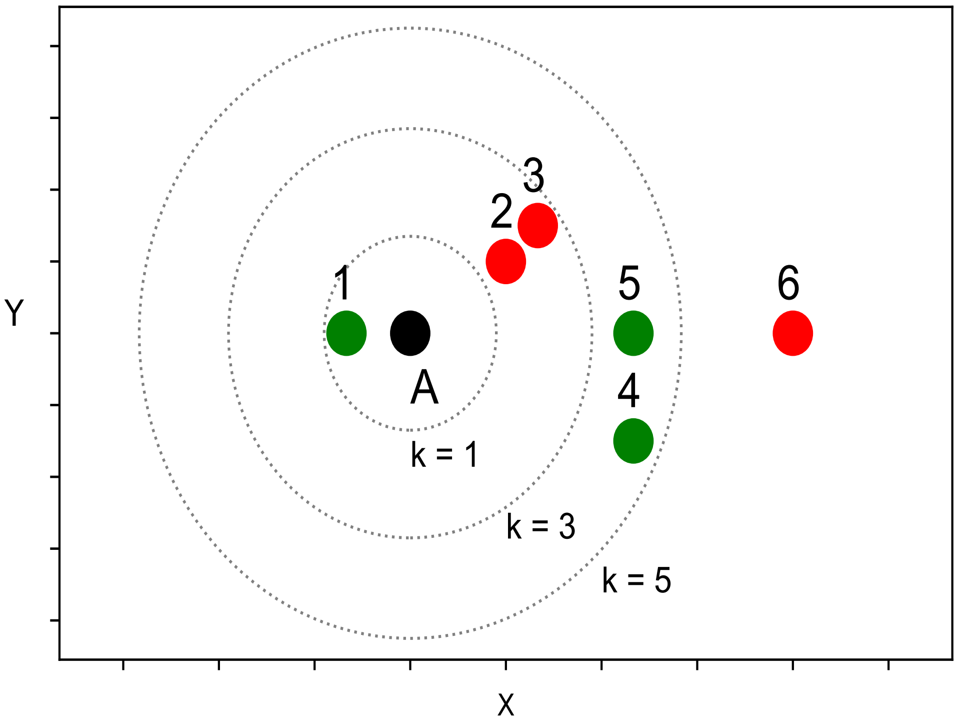
- points in the same class are ususally ”neighbors”
- assign class based on majority of neighbors
- need distance
- need to choose k - number of neighbors
- note: k must be odd for simple majority

Example of kNN



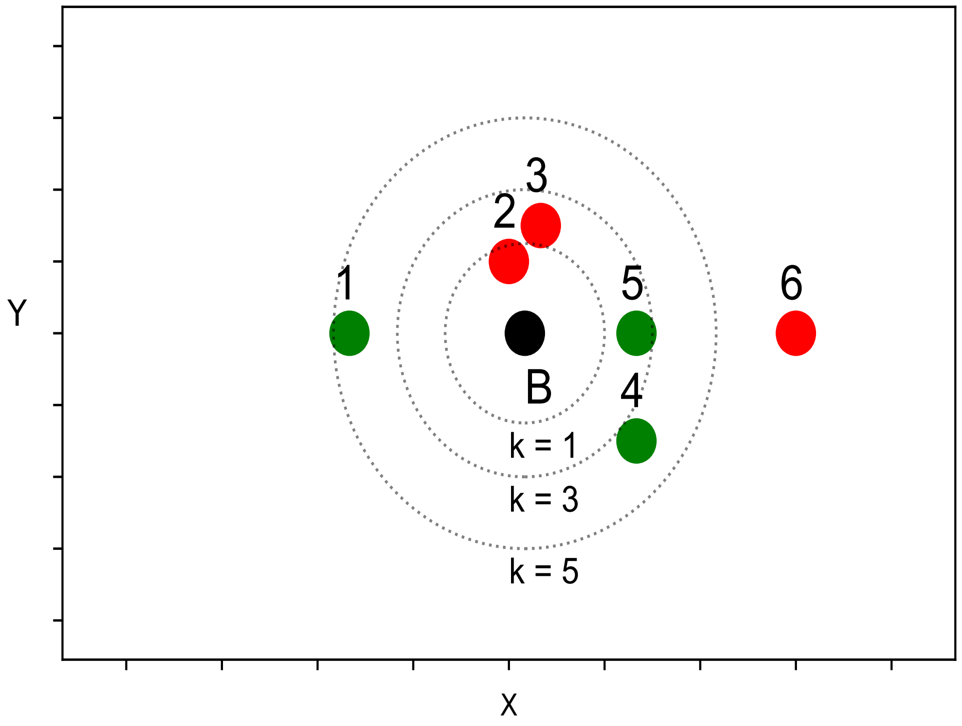
- what labels for A and B?

Assigning a Label for A



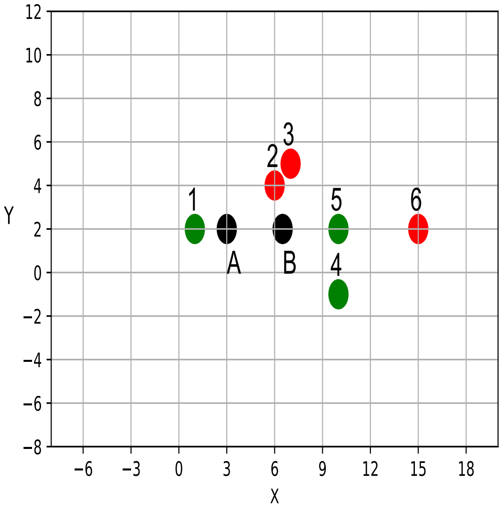
point	k	neighbors	majority
A	1	x_1	green
	3	x_1, x_2, x_3	red
	5	x_1, x_2, x_3, x_4, x_5	green

Assigning a Label for B



point	k	neighbors	majority
B	1	x_2	red
	3	x_2, x_3, x_5	red
	5	x_1, x_2, x_3, x_4, x_5	green

How to Choose k



point	k	neighbors	majority
A	1	x_1	green
	3	x_1, x_2, x_3	red
	5	x_1, x_2, x_3, x_4, x_5	green
B	1	x_2	red
	3	x_2, x_3, x_5	red
	5	x_1, x_2, x_3, x_4, x_5	green

Illustration in Python

```
import numpy as np
import pandas as pd
from sklearn.neighbors import \
               KNeighborsClassifier

data = pd.DataFrame(
      {"id": [ 1,2,3,4,5,6],
       "Label": ["green","red","red",
               "green","green","red"],
    "X": [1, 6, 7, 10, 10, 15],
    "Y": [2, 4, 5, -1, 2, 2 ]},
    columns = ["id", "Label", "X","Y"]}
X = data[["X","Y"]].values
Y = data[["Label"]].values
knn_classifier=KNeighborsClassifier(
                       n_neighbors=3)
knn_classifier.fit(X,Y)
new_instance = np.asmatrix([3, 2])
prediction = knn_classifier.predict(
                   new_instance)
```

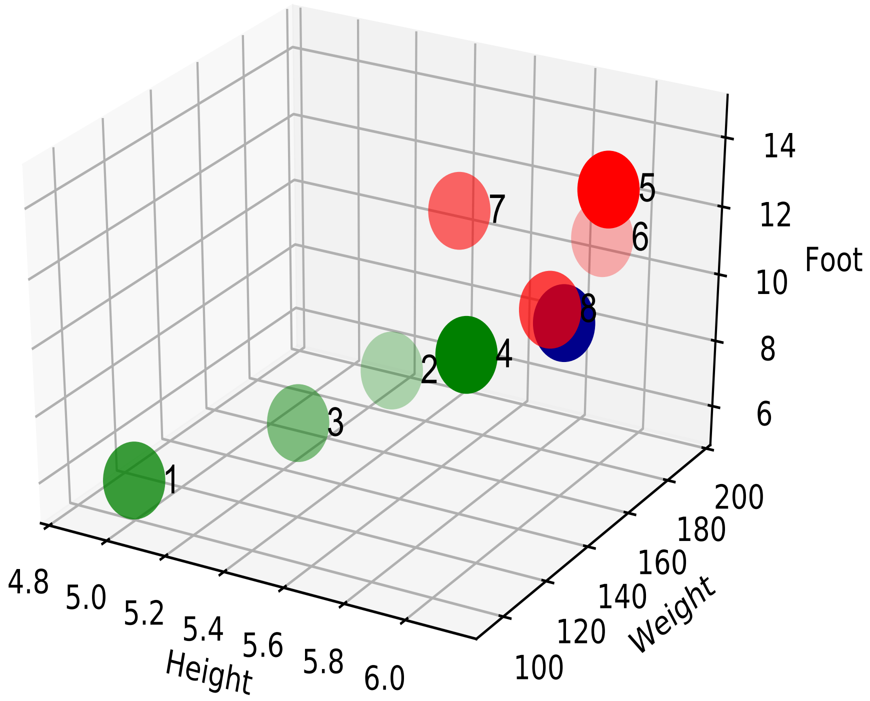
```
ipdb> prediction[0]
red
```

A Numerical Example

object x_i	Height (H)	Weight (W)	Foot (F)	Label (L)
x_1	5.00	100	6	green
x_2	5.50	150	8	green
x_3	5.33	130	7	green
x_4	5.75	150	9	green
x_5	6.00	180	13	red
x_6	5.92	190	11	red
x_7	5.58	170	12	red
x_8	5.92	165	10	red

- note different scales

What is the Label?



$$(\mathrm{H}=6,\ \mathrm{W}=160,\ \mathrm{F}=10) \mapsto\ ?$$

kNN in Python

```
import pandas as pd
data = pd.DataFrame(
 {"id":[ 1,2,3,4,5,6,7,8],
  "Label":["green","green","green","green",
                   "red","red","red","red"],
  "Height":[5,5.5,5.33,5.75,6.00,5.92,5.58,5.92],
  "Weight":[100,150,130,150,180,190,170,165],
  "Foot":[6, 8, 7, 9, 13, 11, 12, 10]},
  columns=["id","Height","Weight",
                         "Foot","Label"])

X = data[["Height","Weight","Foot"]].values
Y = data[["Label"]].values

scaler = StandardScaler().fit(X)
X = scaler.transform(X)

knn_classifier = KNeighborsClassifier(n_neighbors=3)
knn_classifier.fit(X,Y)

new_instance = np.asmatrix([6, 160, 10])
new_instance_scaled = scaler.transform(new_instance)
prediction = knn_classifier.predict(new_instance_scaled)
```

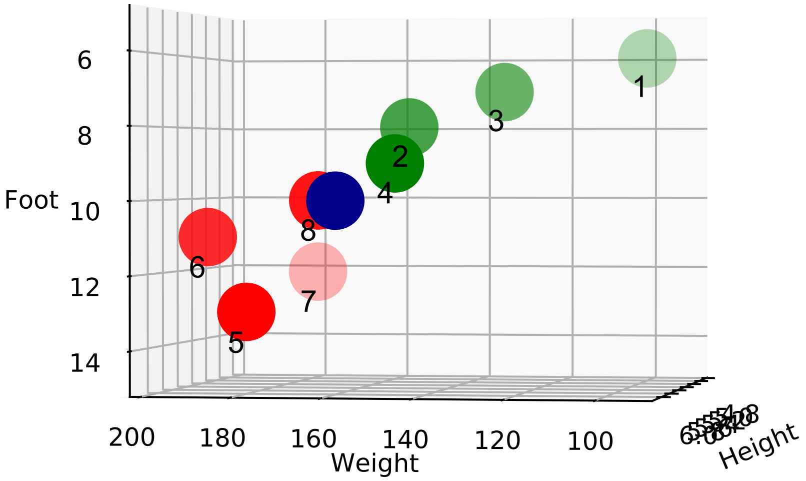
```
ipdb> prediction[0]

'red'
```

Result Without Scaling

```python
import pandas as pd
data = pd.DataFrame(
 {"id":[ 1,2,3,4,5,6,7,8],
  "Label":["green","green","green","green",
                   "red","red","red","red"],
  "Height":[5,5.5,5.33,5.75,6.00,5.92,5.58,5.92],
  "Weight":[100,150,130,150,180,190,170,165],
  "Foot":[6, 8, 7, 9, 13, 11, 12, 10]},
  columns=["id","Height","Weight",
                         "Foot","Label"])

X = data[["Height","Weight","Foot"]].values
Y = data[["Label"]].values

knn_classifier = KNeighborsClassifier(n_neighbors=3)
knn_classifier.fit(X,Y)

new_instance = np.asmatrix([6, 160, 10])
prediction = knn_classifier.predict(new_instance)
```

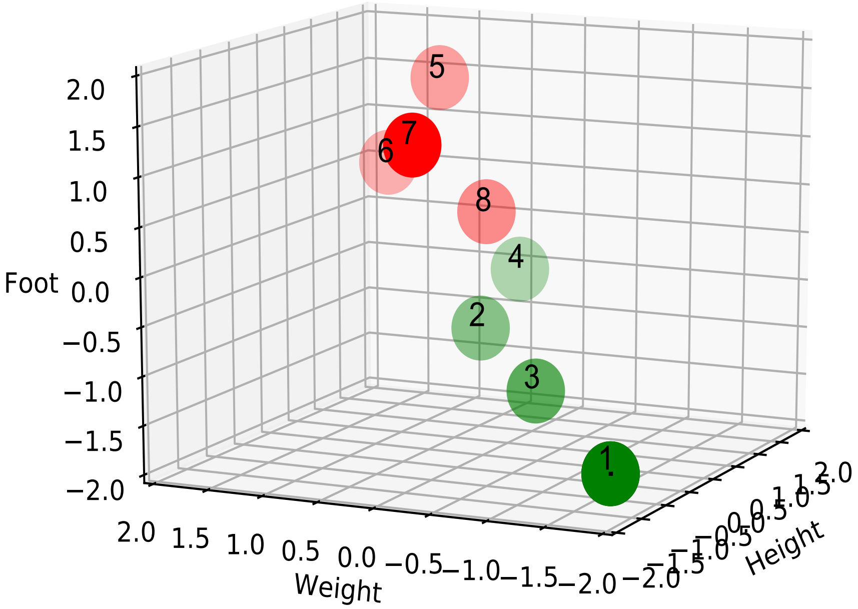
```
ipdb> prediction[0]

’red’
```

Why Scaling?



- (euclidean) distances $d(\cdot)$ dominated by one dimension

Effect of Scaling



- without scaling: $d(x_7, x_8) < d(x_4, x_8)$
- with scaling: $d(x_7, x_8) > d(x_4, x_8)$

Calculating k

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

import pandas as pd
data = pd.DataFrame(
 {"id":[ 1,2,3,4,5,6,7,8],
  "Label":["green","green","green","green",
                   "red","red","red","red"],
  "Height":[5,5.5,5.33,5.75,6.00,5.92,5.58,5.92],
  "Weight":[100,150,130,150,180,190,170,165],
  "Foot":[6, 8, 7, 9, 13, 11, 12, 10]},
  columns=["id","Height","Weight",
                         "Foot","Label"])

X = data[["Height","Weight","Foot"]].values
Y = data[["Label"]].values

scaler = StandardScaler().fit(X)
X = scaler.transform(X)
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,
                   test_size=0.5,random_state=0)
error_rate = []
for k in [1,3]:
    knn_classifier = KNeighborsClassifier(n_neighbors=k)
    knn_classifier.fit(X_train,Y_train)
    pred_k = knn_classifier.predict(X_test)
    error_rate.append(np.mean(pred_k != Y_test))
```

```
ipdb> error_rate

[0.5, 0.5]
```

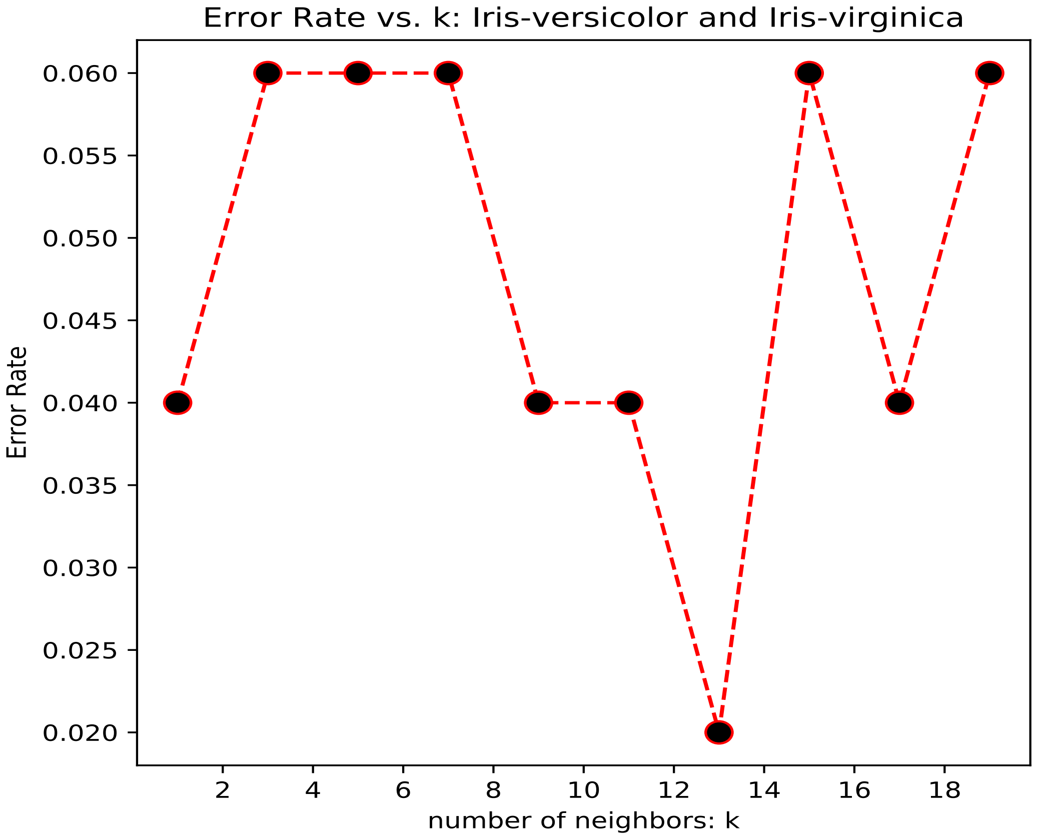
Calculating k for IRIS

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

url = r'https://archive.ics.uci.edu/ml/'  + \
           r'machine-learning-databases/iris/iris.data'

iris_feature_names = ['sepal-length', 'sepal-width',
                            'petal-length', 'petal-width']

data = pd.read_csv(url, names=['sepal-length', 'sepal-width',
                         'petal-length', 'petal-width', 'Class'])

class_labels = ['Iris-versicolor', 'Iris-virginica']
data = data[data['Class'].isin(class_labels)]

X = data[iris_feature_names].values
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)

le = LabelEncoder()
Y = le.fit_transform(data['Class'].values)
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.5,
                                                   random_state=3)
```

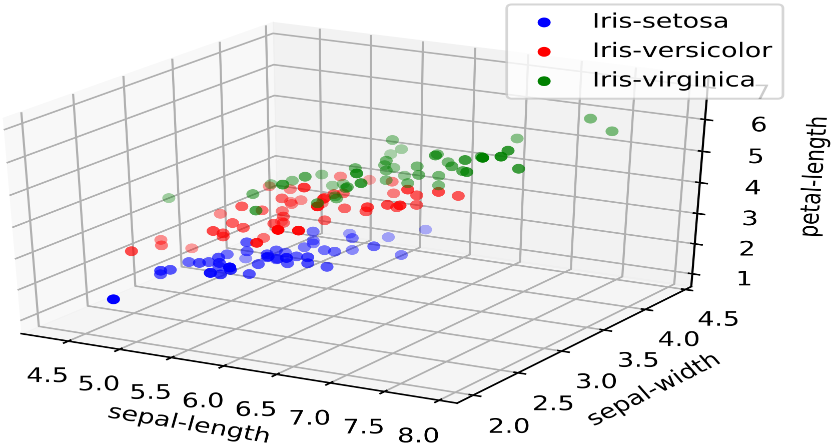
Calculating k for IRIS (cont’d)

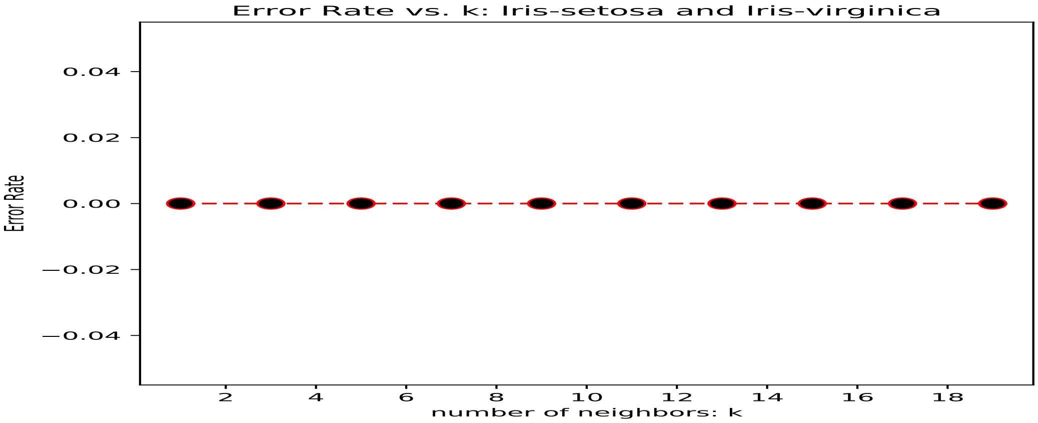
```
error_rate = []
for k in range(1,21,2):
    knn_classifier = KNeighborsClassifier(n_neighbors=k)
    knn_classifier.fit(X_train,Y_train)
    pred_k = knn_classifier.predict(X_test)
    error_rate.append(np.mean(pred_k != Y_test))

figure(figsize=(10,4))
ax = plt.gca()
ax.xaxis.set_major_locator(MaxNLocator(integer=True))
plt.plot(range(1,21,2), error_rate, color=’red’, linestyle=’dashed’,
                marker=’o’, markerfacecolor=’black’, markersize=10)
plt.title(’Error Rate vs. k for Iris Subset’)
plt.xlabel(’number of neighbors: k’)
plt.ylabel(’Error Rate’)
```

Calculating k for IRIS



k for IRIS


Iris-setosa
Iris-versicolor
Iris-virginica
sepal-length
4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0
sepal-width
2.0 2.5 3.0 3.5 4.0 4.5
petal-length
1 2 3 4 5 6 7


Error Rate vs. k: Iris-setosa and Iris-virginica
Error Rate
0.04
0.02
0.00
−0.02
−0.04
2 4 6 8 10 12 14 16 18
number of neighbors: k

A Categorical Dataset

Day	Weather	Temperature	Wind	Play
1	sunny	hot	low	**no**
2	rainy	mild	high	**yes**
3	sunny	cold	low	**yes**
4	rainy	cold	high	**no**
5	sunny	cold	high	**yes**
6	overcast	mild	low	**yes**
7	sunny	hot	low	**yes**
8	overcast	hot	high	**yes**
9	rainy	hot	high	**no**
10	rainy	mild	low	**yes**

- what label for x* = (sunny, cold, low)?

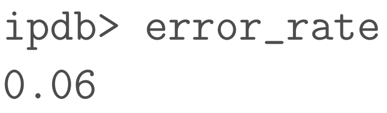
Python Code

```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder

data = pd.DataFrame(
        {’Day’: [1,2,3,4,5,6,7,8,9,10],
        ’Weather’:[’sunny’,’rainy’,’sunny’,’rainy’,’sunny’,’overcast’,
                        ’sunny’, ’overcast’, ’rainy’,’rainy’],
        ’Temperature’: [’hot’, ’mild’, ’cold’, ’cold’,’cold’,’mild’,
                        ’hot’, ’hot’, ’hot’, ’mild’],
        ’Wind’: [’low’, ’high’, ’low’, ’high’, ’high’, ’low’, ’low’,
                 ’high’, ’high’, ’low’],
        ’Play’: [’no’,’yes’, ’yes’, ’no’, ’yes’,’yes’,’yes’,
                 ’yes’,’no’,’yes’]},
        columns = [’Day’, ’Weather’, ’Temperature’, ’Wind’, ’Play’]
        )
input_data = data[[’Weather’, ’Temperature’, ’Wind’]]
dummies = [pd.get_dummies(data[c]) for c in input_data.columns]
binary_data = pd.concat(dummies, axis=1)

X = binary_data[0:10].values
le = LabelEncoder()
Y = le.fit_transform(data[’Play’].values)

knn_classifier = KNeighborsClassifier(n_neighbors=3)
knn_classifier.fit(X,Y)
new_instance = np.asmatrix([0,0,1,1,0,0,0,1])
prediction = knn_classifier.predict(new_instance)
```

```
ipdb> prediction

1
```

kNN: IRIS

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

url = r'https://archive.ics.uci.edu/ml/'  + \
           r'machine-learning-databases/iris/iris.data'

iris_feature_names = ['sepal-length', 'sepal-width',
                            'petal-length', 'petal-width']
data = pd.read_csv(url, names=['sepal-length', 'sepal-width',
                         'petal-length', 'petal-width', 'Class'])
class_labels = ['Iris-versicolor', 'Iris-virginica']
data = data[data['Class'].isin(class_labels)]

X = data[iris_feature_names].values
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)
le = LabelEncoder()
Y = le.fit_transform(data['Class'].values)

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,
                                     test_size=0.5,random_state=3)
knn_classifier = KNeighborsClassifier(n_neighbors=15)
knn_classifier.fit(X_train, Y_train)
prediction = knn_classifier.predict(X_test)
error_rate = np.mean(prediction != Y_test)
```



```
ipdb> error_rate
0.06
```

Concepts Check:

(a) distances and neighbors

(b) nearest neigbor intuition

(c) need for scaling

(d) how to choose k

(e) analyzing categorical data