16.1) K-Nearest Neighbors

Vitor Kamada

December 2019

Reference

Tables, Graphics, and Figures from

Computational and Inferential Thinking: The Foundations of Data Science

Adhikari & DeNero (2019): Ch 17.1 Nearest Neighbors

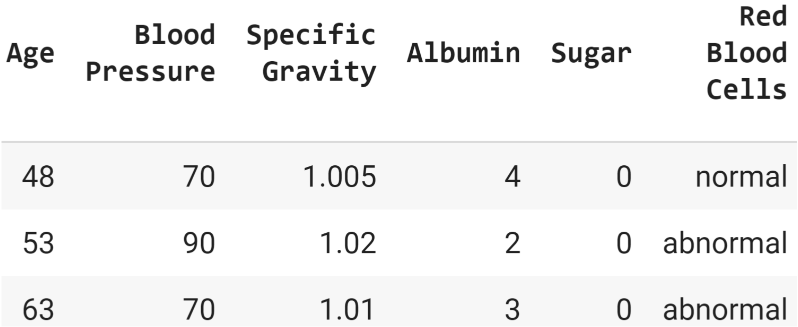
17.2 Training and Testing

17.3 Rows of Tables

https://www.inferentialthinking.com/

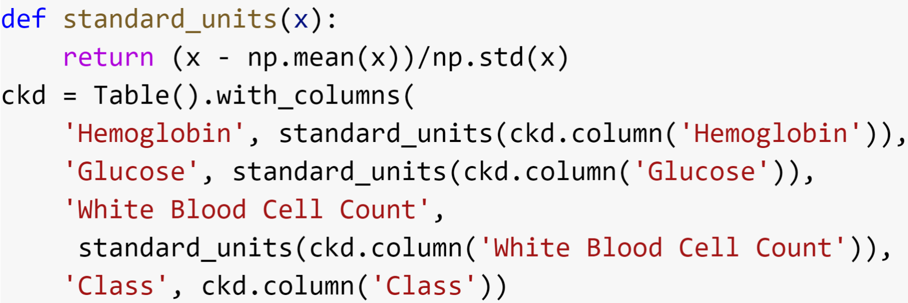
Chronic Kidney Disease (CKD)

```python
import numpy as np
from datascience import *
path_data = 'https://github.com/data-8/textbook/raw/gh-pages/data/'
data = Table.read_table(path_data + 'ckd.csv')
ckd = data.relabeled('Blood Glucose Random', 'Glucose')
```

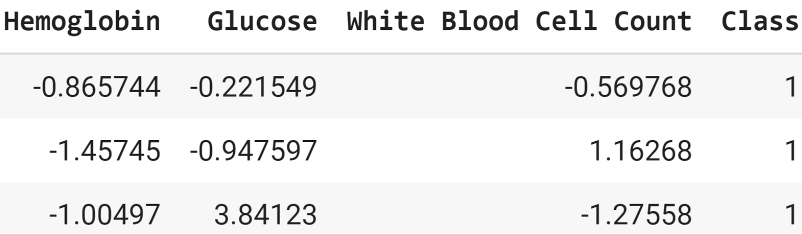


Age	Blood Pressure	Specific Gravity	Albumin	Sugar	Red Blood Cells
48	70	1.005	4	0	normal
53	90	1.02	2	0	abnormal
63	70	1.01	3	0	abnormal

1 = Chronic Kidney Disease (CKD)

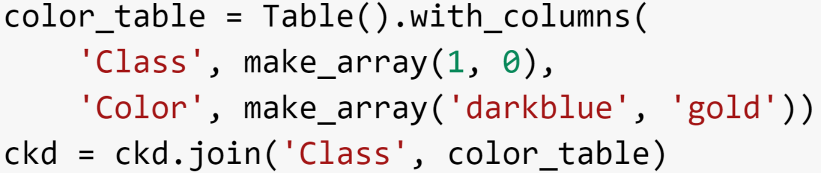


```
def standard_units(x):
    return (x - np.mean(x))/np.std(x)
ckd = Table().with_columns(
    'Hemoglobin', standard_units(ckd.column('Hemoglobin')),
    'Glucose', standard_units(ckd.column('Glucose')),
    'White Blood Cell Count',
     standard_units(ckd.column('White Blood Cell Count')),
    'Class', ckd.column('Class'))
```

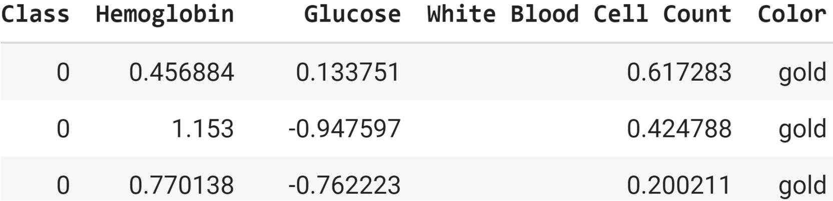


Hemoglobin	Glucose	White Blood Cell Count	Class
-0.865744	-0.221549	-0.569768	1
-1.45745	-0.947597	1.16268	1
-1.00497	3.84123	-1.27558	1

Blue dots are patients with CKD

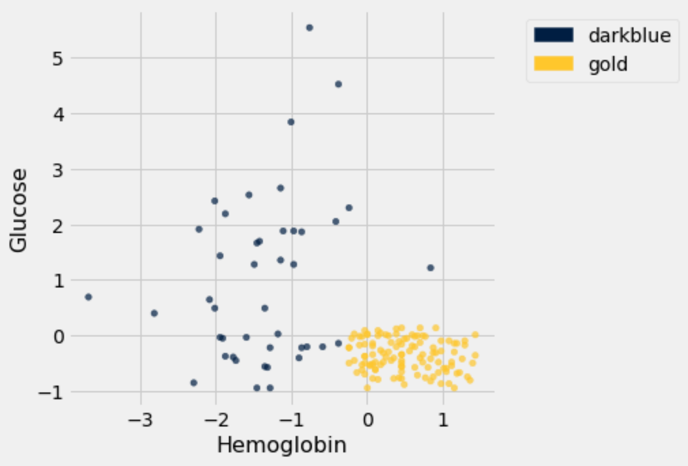


```
color_table = Table().with_columns(
    'Class', make_array(1, 0),
    'Color', make_array('darkblue', 'gold'))
ckd = ckd.join('Class', color_table)
```

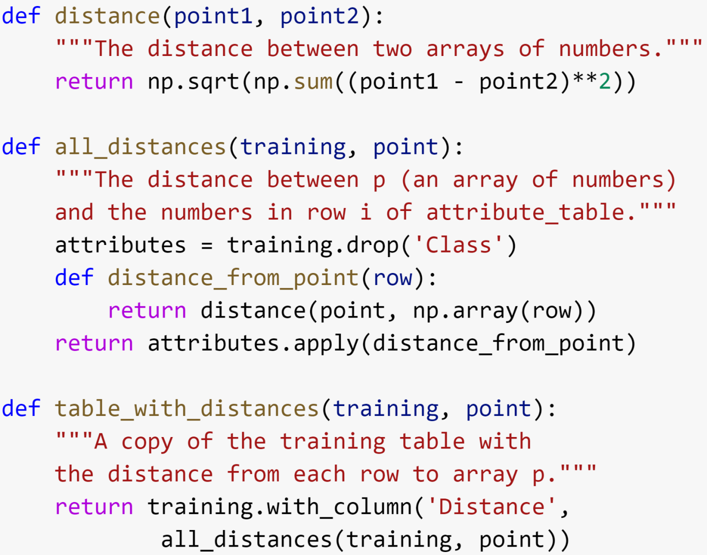


Class	Hemoglobin	Glucose	White Blood Cell Count	Color
0	0.456884	0.133751	0.617283	gold
0	1.153	-0.947597	0.424788	gold
0	0.770138	-0.762223	0.200211	gold

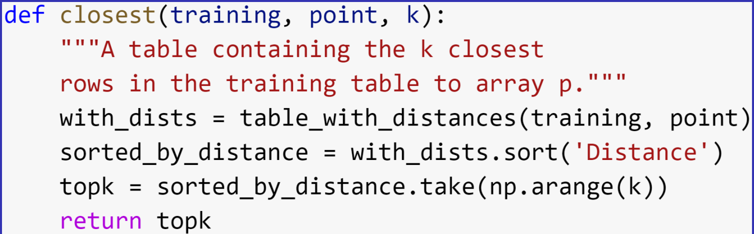
```
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
ckd.scatter('Hemoglobin', 'Glucose', colors='Color')
```



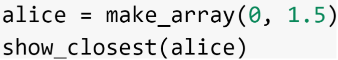
Functions to Calculate Distances



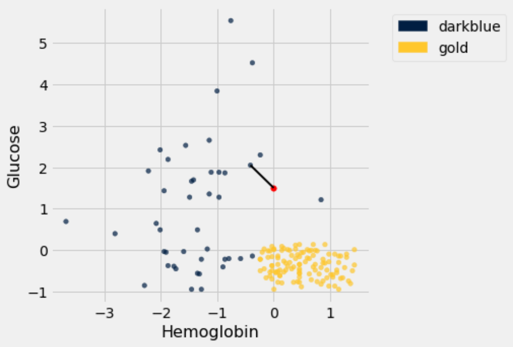
```
def distance(point1, point2):
    """The distance between two arrays of numbers."""
    return np.sqrt(np.sum((point1 - point2)**2))

def all_distances(training, point):
    """The distance between p (an array of numbers)
    and the numbers in row i of attribute_table."""
    attributes = training.drop('Class')
    def distance_from_point(row):
        return distance(point, np.array(row))
    return attributes.apply(distance_from_point)

def table_with_distances(training, point):
    """A copy of the training table with
    the distance from each row to array p."""
    return training.with_column('Distance',
              all_distances(training, point))
```

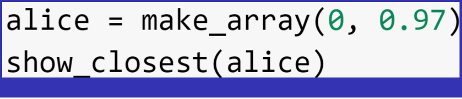


```python
def closest(training, point, k):
    """A table containing the k closest
    rows in the training table to array p."""
    with_dists = table_with_distances(training, point)
    sorted_by_distance = with_dists.sort('Distance')
    topk = sorted_by_distance.take(np.arange(k))
    return topk
```

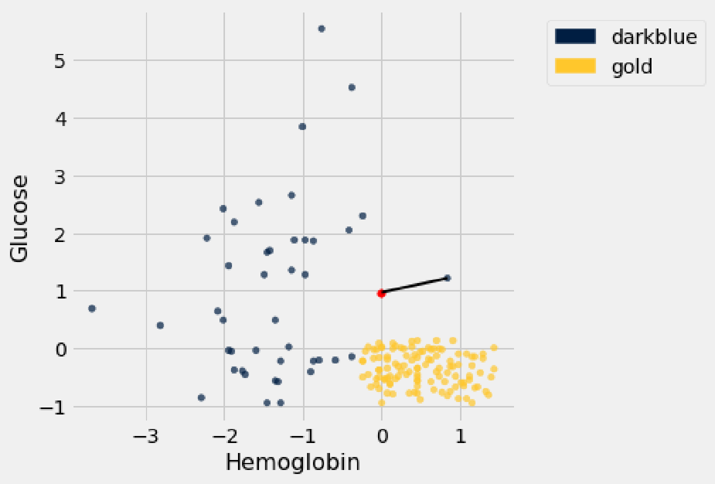


```python
alice = make_array(0, 1.5)
show_closest(alice)
```

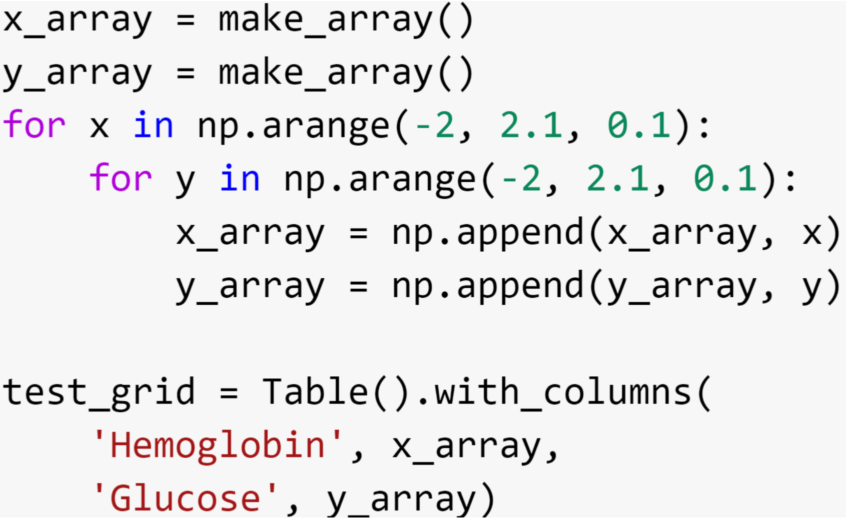




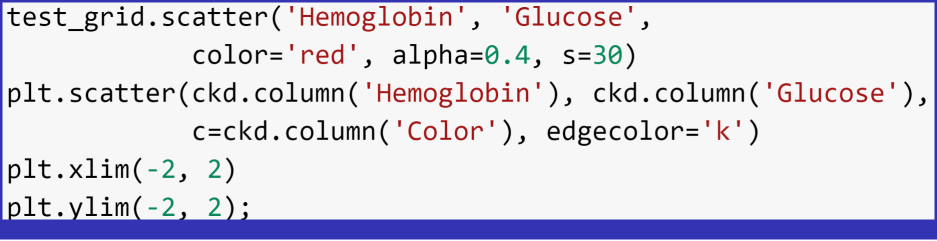
```
alice = make_array(0, 0.97)
show_closest(alice)
```



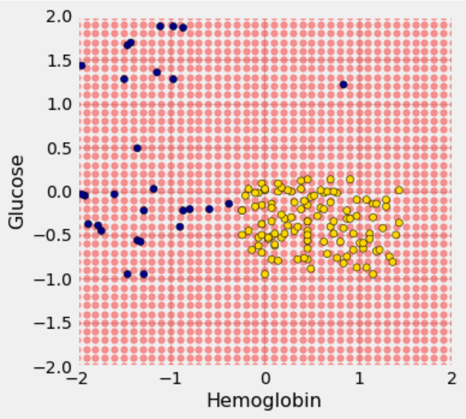
Create a Grid



```python
x_array = make_array()
y_array = make_array()
for x in np.arange(-2, 2.1, 0.1):
    for y in np.arange(-2, 2.1, 0.1):
        x_array = np.append(x_array, x)
        y_array = np.append(y_array, y)

test_grid = Table().with_columns(
    'Hemoglobin', x_array,
    'Glucose', y_array)
```



```
test_grid.scatter('Hemoglobin', 'Glucose',
            color='red', alpha=0.4, s=30)
plt.scatter(ckd.column('Hemoglobin'), ckd.column('Glucose'),
            c=ckd.column('Color'), edgecolor='k')
plt.xlim(-2, 2)
plt.ylim(-2, 2);
```

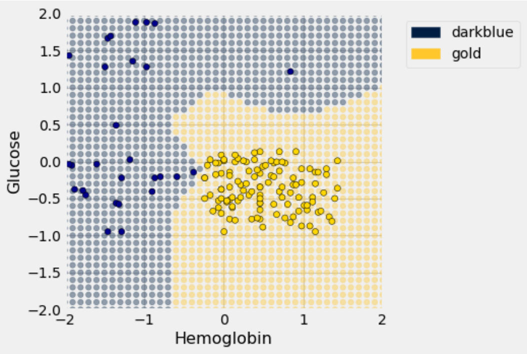


Nearest Neighbors Method

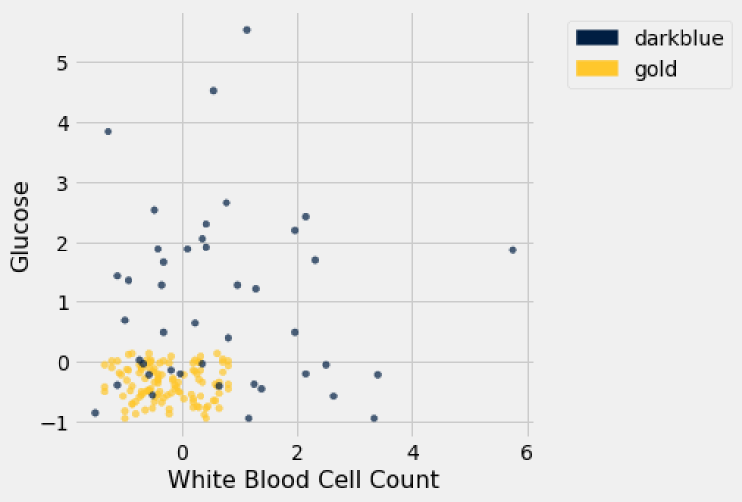
```python
def majority(topkclasses):
    """1 if the majority of the "Class" column is 1s, and 0 otherwise."""
    ones = topkclasses.where('Class', are.equal_to(1)).num_rows
    zeros = topkclasses.where('Class', are.equal_to(0)).num_rows
    if ones > zeros:
        return 1
    else:
        return 0

def classify(training, p, k):
    """Classify an example with attributes p using k-nearest
     neighbor classification with the given training table."""
    closestk = closest(training, p, k)
    topkclasses = closestk.select('Class')
    return majority(topkclasses)
```

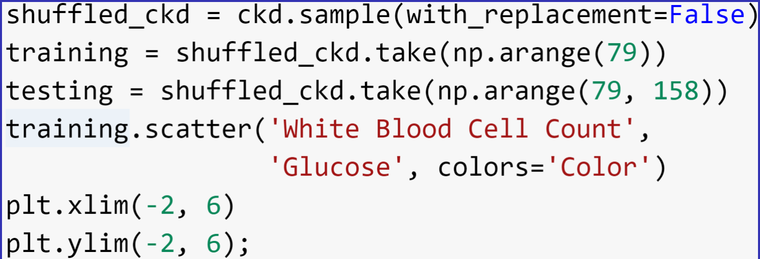
```python
def classify_grid(training, test, k):
    c = make_array()
    for i in range(test.num_rows):
        c = np.append(c, classify(training,
              make_array(test.row(i)), k))
    return c
```

```
c = classify_grid(ckd.drop('White Blood Cell Count',
                           'Color'), test_grid, 1)
test_grid = test_grid.with_column('Class',
              c).join('Class', color_table)
test_grid.scatter('Hemoglobin', 'Glucose',
          colors='Color', alpha=0.4, s=30)
plt.scatter(ckd.column('Hemoglobin'), ckd.column('Glucose'),
            c=ckd.column('Color'), edgecolor='k')
plt.xlim(-2, 2)
plt.ylim(-2, 2);
```

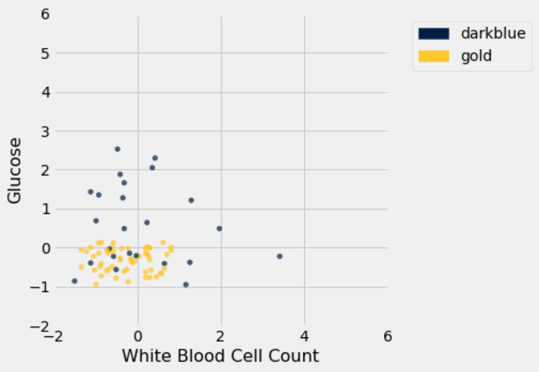


```
ckd.scatter('White Blood Cell Count',
            'Glucose', colors='Color')
```

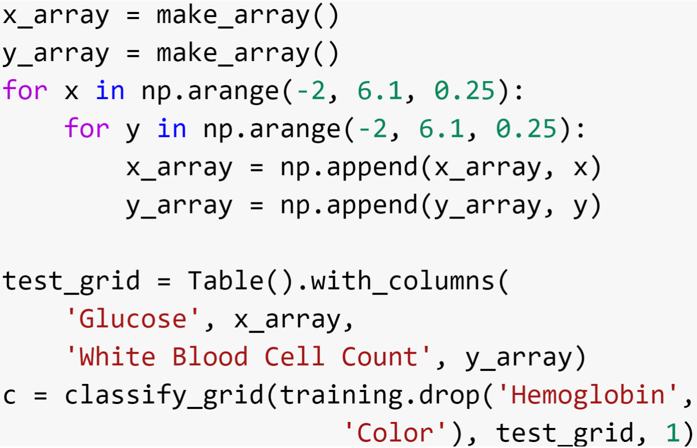




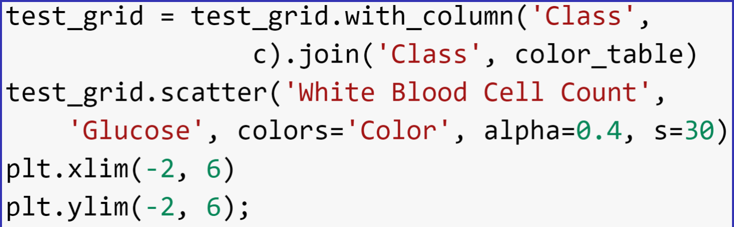
```
shuffled_ckd = ckd.sample(with_replacement=False)
training = shuffled_ckd.take(np.arange(79))
testing = shuffled_ckd.take(np.arange(79, 158))
training.scatter('White Blood Cell Count',
                 'Glucose', colors='Color')
plt.xlim(-2, 6)
plt.ylim(-2, 6);
```



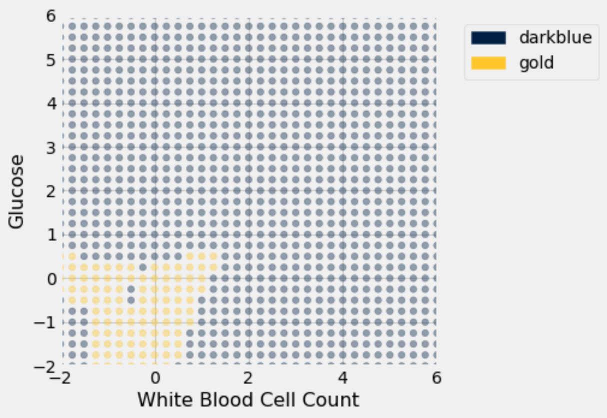
Run k-Nearest Neighbors

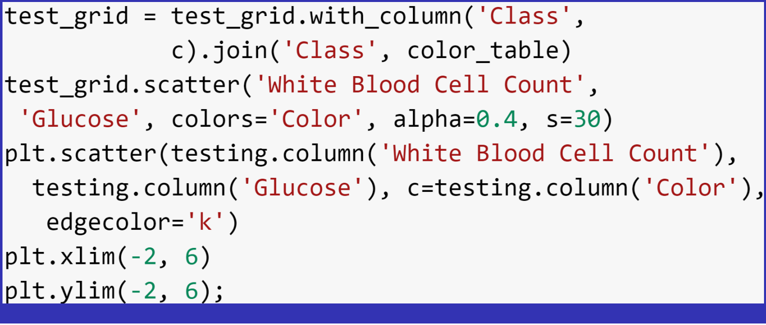


```
x_array = make_array()
y_array = make_array()
for x in np.arange(-2, 6.1, 0.25):
    for y in np.arange(-2, 6.1, 0.25):
        x_array = np.append(x_array, x)
        y_array = np.append(y_array, y)

test_grid = Table().with_columns(
    'Glucose', x_array,
    'White Blood Cell Count', y_array)
c = classify_grid(training.drop('Hemoglobin',
                      'Color'), test_grid, 1)
```

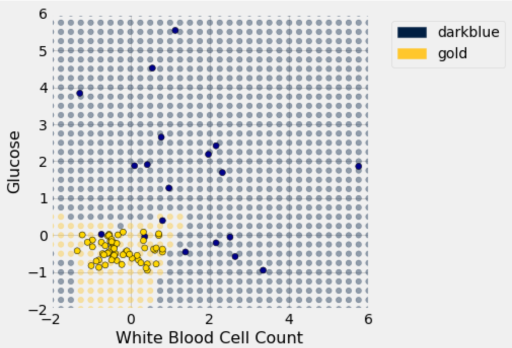


```
test_grid = test_grid.with_column('Class',
               c).join('Class', color_table)
test_grid.scatter('White Blood Cell Count',
    'Glucose', colors='Color', alpha=0.4, s=30)
plt.xlim(-2, 6)
plt.ylim(-2, 6);
```





```python
test_grid = test_grid.with_column('Class',
            c).join('Class', color_table)
test_grid.scatter('White Blood Cell Count',
 'Glucose', colors='Color', alpha=0.4, s=30)
plt.scatter(testing.column('White Blood Cell Count'),
  testing.column('Glucose'), c=testing.column('Color'),
   edgecolor='k')
plt.xlim(-2, 6)
plt.ylim(-2, 6);
```



ckd.row(0)

```
Row(Class=0, Hemoglobin=0.456883701715849, Glucose=0.133750854517
```

```
ckd.row(0).item(1)
```

```
0.456883701715849
```

```
ckd_attributes = ckd.select('Hemoglobin', 'Glucose')
ckd_attributes.row(3)
```

```
Row(Hemoglobin=0.5961076648232668, Glucose=-0.19065363034327
```

```
patient3 = np.array(ckd_attributes.row(3))
alice = make_array(0, 1.1)
alice, patient3
```

```
(array([0. , 1.1]), array([ 0.59610766, -0.19065363]))
```

t = ckd_attributes.take(np.arange(5))

Hemoglobin	Glucose
0.456884	0.133751
1.153	-0.947597
0.770138	-0.762223
0.596108	-0.190654
-0.239236	-0.49961

```python
def max_abs(row):
    return np.max(np.abs(np.array(row)))
max_abs(t.row(4))
```

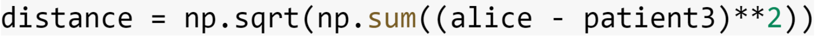
```
0.4996102825918697
```

```python
t.apply(max_abs)
```

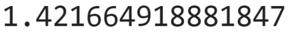
```
array([0.4568837 , 1.15300352, 0.77013762, 0.59610766, 0.49961028])
```

Distance between Alice and another point

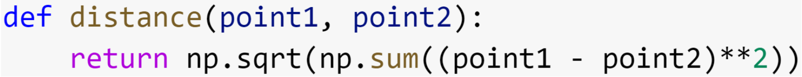
$$D = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}$$



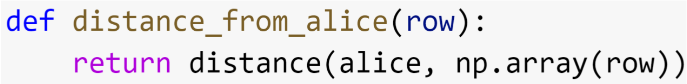
```
distance = np.sqrt(np.sum((alice - patient3)**2))
```



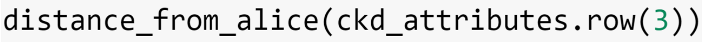
```
1.421664918881847
```



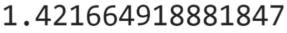
```
def distance(point1, point2):
    return np.sqrt(np.sum((point1 - point2)**2))
```



```
def distance_from_alice(row):
    return distance(alice, np.array(row))
```

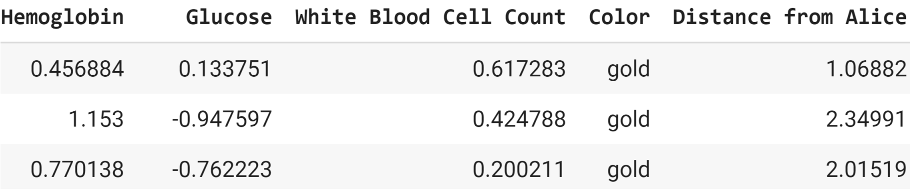


```
distance_from_alice(ckd_attributes.row(3))
```



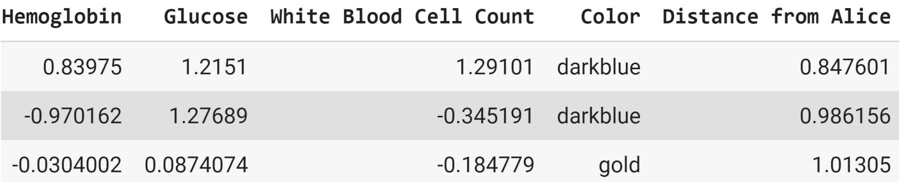
```
1.421664918881847
```

```
distances = ckd_attributes.apply(distance_from_alice)
ckd_with_distances = ckd.with_column('Distance from Alice', distances)
```

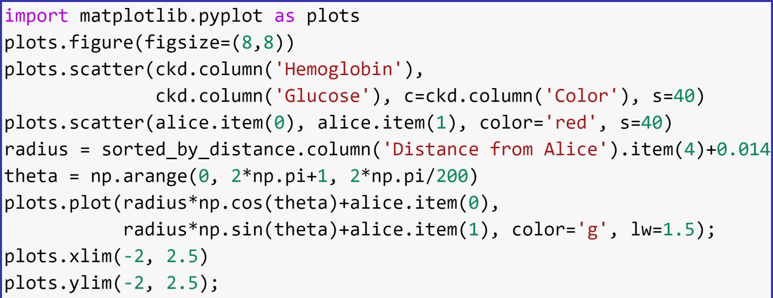


Hemoglobin	Glucose	White Blood Cell Count	Color	Distance from Alice
0.456884	0.133751	0.617283	gold	1.06882
1.153	-0.947597	0.424788	gold	2.34991
0.770138	-0.762223	0.200211	gold	2.01519

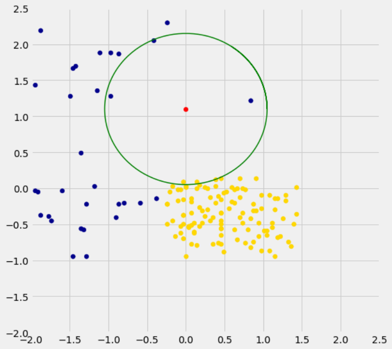
```
sorted_by_distance = ckd_with_distances.sort('Distance from Alice')
```



Hemoglobin	Glucose	White Blood Cell Count	Color	Distance from Alice
0.83975	1.2151	1.29101	darkblue	0.847601
-0.970162	1.27689	-0.345191	darkblue	0.986156
-0.0304002	0.0874074	-0.184779	gold	1.01305



```
import matplotlib.pyplot as plots
plots.figure(figsize=(8,8))
plots.scatter(ckd.column('Hemoglobin'),
              ckd.column('Glucose'), c=ckd.column('Color'), s=40)
plots.scatter(alice.item(0), alice.item(1), color='red', s=40)
radius = sorted_by_distance.column('Distance from Alice').item(4)+0.014
theta = np.arange(0, 2*np.pi+1, 2*np.pi/200)
plots.plot(radius*np.cos(theta)+alice.item(0),
           radius*np.sin(theta)+alice.item(1), color='g', lw=1.5);
plots.xlim(-2, 2.5)
plots.ylim(-2, 2.5);
```


2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5