Lab 4. Regression and Clustering

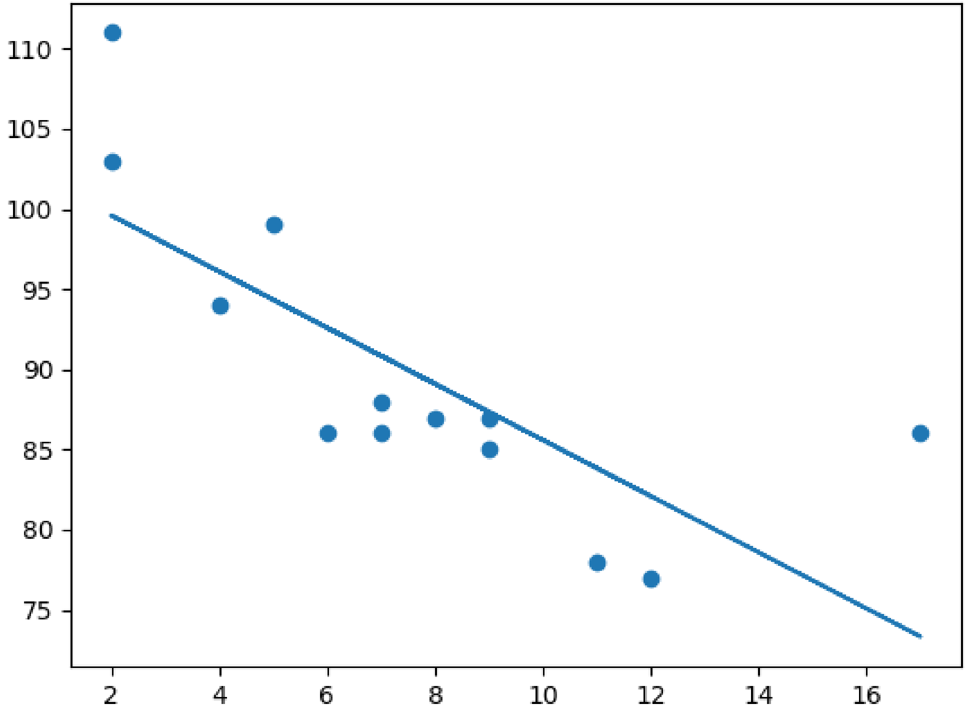
Yongjae Yoo, Ph. D.
Assistant Professor
Department of Artificial Intelligence
Hanyang University ERICA

Yesterday So Far,

- We have leant draw a scatter plot from dataset.
- Today, we’re going to learn how AI
  - Finds a trend from existing data, and
  - How to handles new inputs.

Regression

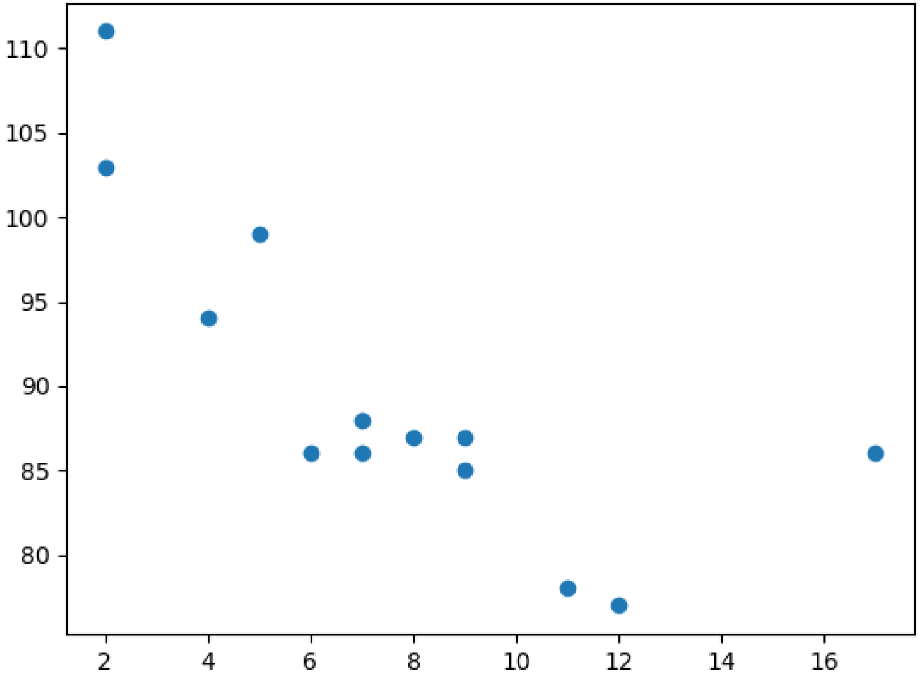
- Regression: to find the relationship between variables.
- Easiest way: find a linear trend (straight line).



Starting Point

- Drawing a scatter plot from data first.
- Can you see trend(s)?

```
import matplotlib.pyplot as plt

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

plt.scatter(x, y)
plt.show()
```



Middle School Math Revisited

- Generalized, straight line function: $y = ax + b$

- Where, a = slope, b = intercept.

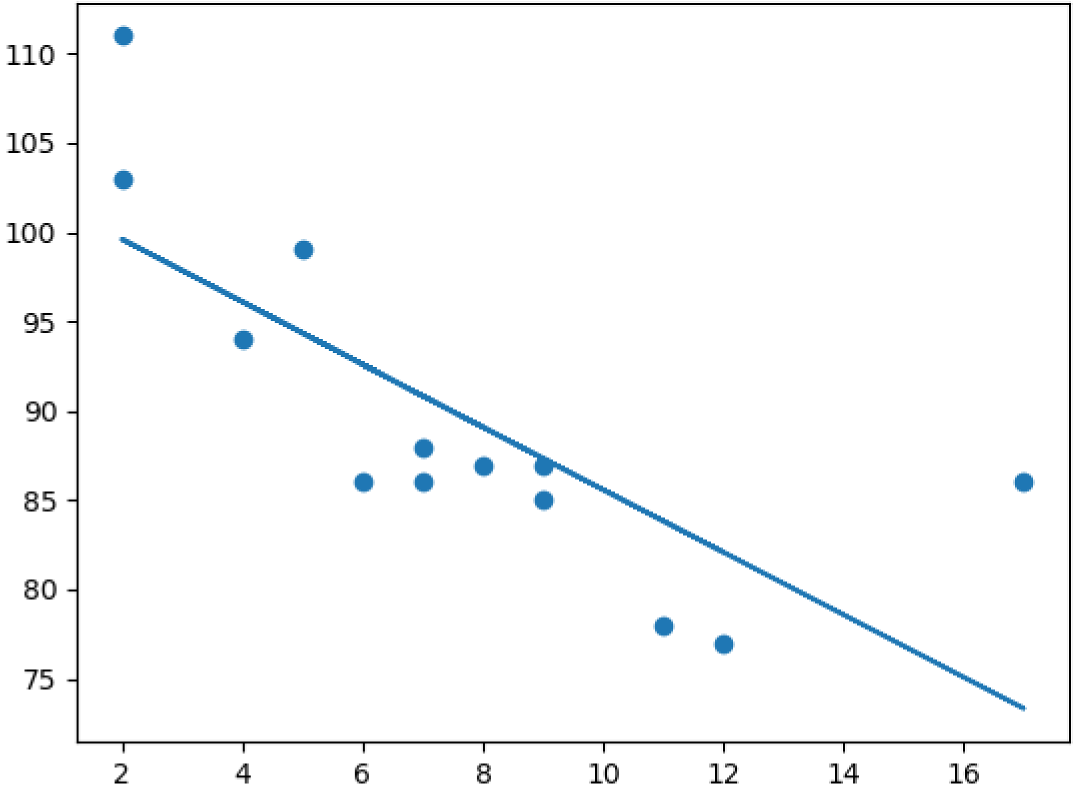
Linear Regression

- To infer a linear trend, we will use scipy library.

```
import matplotlib.pyplot as plt
from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std_err = stats.linregress(x, y)

def myfunc(x):
  return slope * x + intercept

mymodel = list(map(myfunc, x))

plt.scatter(x, y)
plt.plot(x, mymodel)
plt.show()
```

Linear Regression

- The result would be:



So, What Does The Code Means?

- Here, the function parameter has been inferred from:

```
slope, intercept, r, p, std_err = stats.linregress(x, y)
```

- Using the existing x and y.
- Let’s see

```
def myfunc(x):
  return slope * x + intercept
```

- That is, the line function form.

So, What Does The Code Means?

- To draw the line on the scatter plot, we need to infer the data of each point. Could be done by:

```
mymodel = list(map(myfunc, x))
```

- And finally, we draw a line using:

```
plt.plot(x, mymodel)
```

- That’s all folk!

Does The Function Really Good?

- Here the function: `slope * x + intercept`
- Really explains/describes the data well?

- The indicator for “fitting” is so-called r, correlation.

Based on This, Let’s Predict Future

- Let us try to predict the speed of a (N) years old car.

```
def myfunc(x):
  return slope * x + intercept
```

- Put in x to the number you want to predict, e.g., 10, 5, ⋯

Lab Exercise #1

- Here is the data x: “exercise hours/week” and y : body fat (%).

x = [12, 4, 2, 18, 15, 12, 0, 9, 3, 6, 10, 8, 9, 4, 1, 5, 21, 20, 7, 16]
y = [25, 33, 26, 16, 14, 26, 38, 21, 33, 22, 25, 27, 29, 32, 32, 20, 10, 13, 25, 14]

- Draw the scatter plot to see the relation between x and y

Lab Exercise #1

- Find a regression y = ax + b, where a = slope, b = intercept.
- Explain the relation between two variables.
- Find r to decide whether the fitting is good or not.

Now, Move on To Clustering

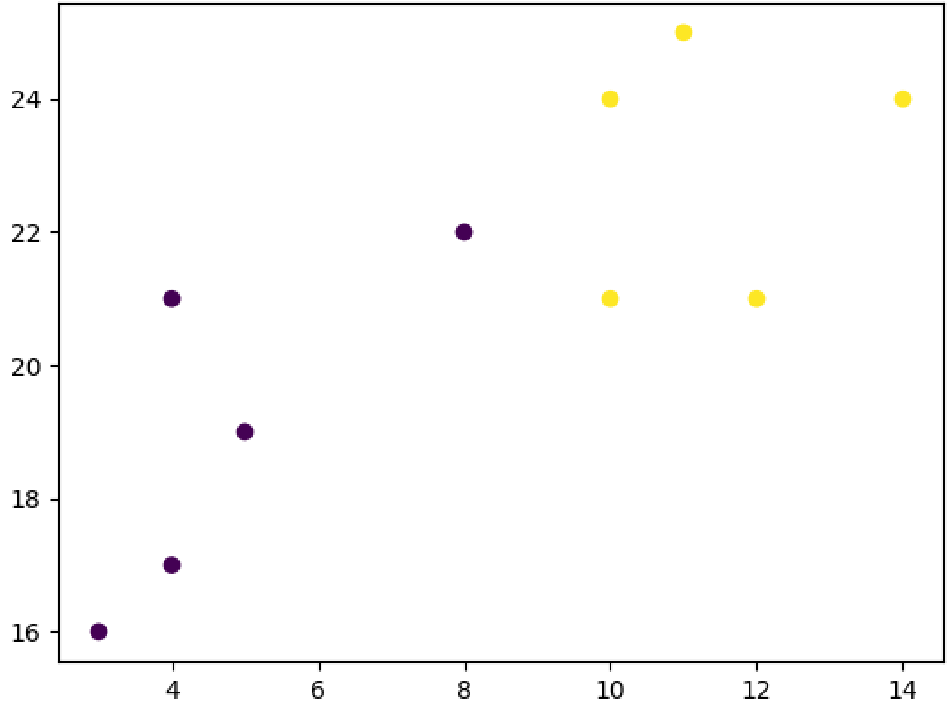
- KNN (K-nearest Neighbors)
  - Classifies a new input, based on the adjacent (prior) points.
  - It infers missing values.
  - Requires prior knowledges (pre-trained points)

  - Infers new input (missing value) class by using K nearest points.

Starting from Prior Data

- Let’s define (x, y) data with two-classes.

```
import matplotlib.pyplot as plt

x = [4, 5, 10, 4, 3, 11, 14 , 8, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
classes = [0, 0, 1, 0, 0, 1, 1, 0, 1, 1]

plt.scatter(x, y, c=classes)
plt.show()
```



Applying K = 1 KNN

- Putting in all prior knowledges into KNN:

```
from sklearn.neighbors import KNeighborsClassifier

data = list(zip(x, y))
knn = KNeighborsClassifier(n_neighbors=1)

knn.fit(data, classes)
```

Using This,

- Predict a new point (8, 21)

```
new_x = 8
new_y = 21
new_point = [(new_x, new_y)]

prediction = knn.predict(new_point)

plt.scatter(x + [new_x], y + [new_y], c=classes + [prediction[0]])
plt.text(x=new_x-1.7, y=new_y-0.7, s=f"new point, class: {prediction[0]}")
plt.show()
```

Using This,

- Repeat this with K = 5 KNN

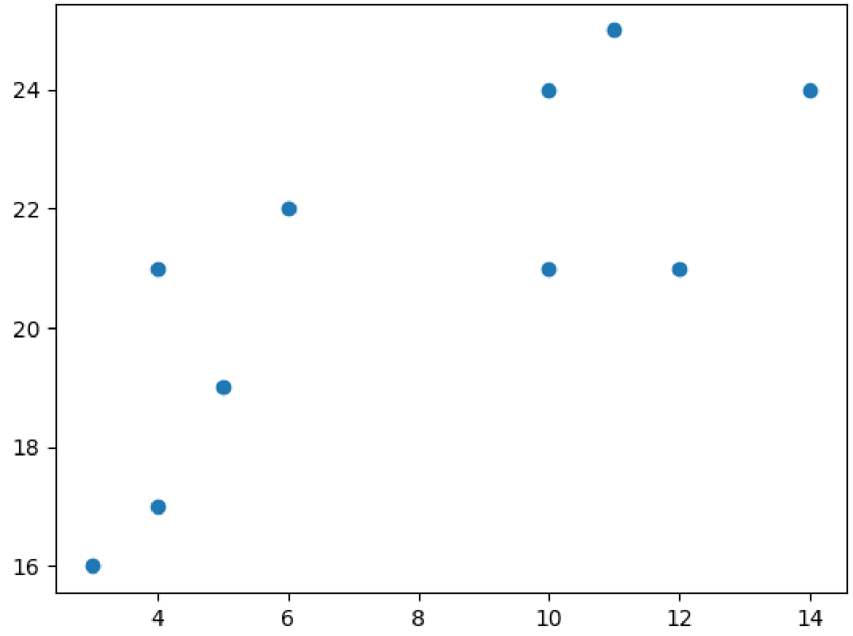
```
from sklearn.neighbors import KNeighborsClassifier

data = list(zip(x, y))
knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(data, classes)
```

- And repeat the above page.

Now, Let’s Play With Multiple Classes

- Hierarchical Clustering
  - Clusters based on measuring the dissimilarities between data.
- Useful for “group” the data
- Often referred as “Dendrogram.”
- Starting from this data:

```
import numpy as np
import matplotlib.pyplot as plt

x = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]

plt.scatter(x, y)
plt.show()
```

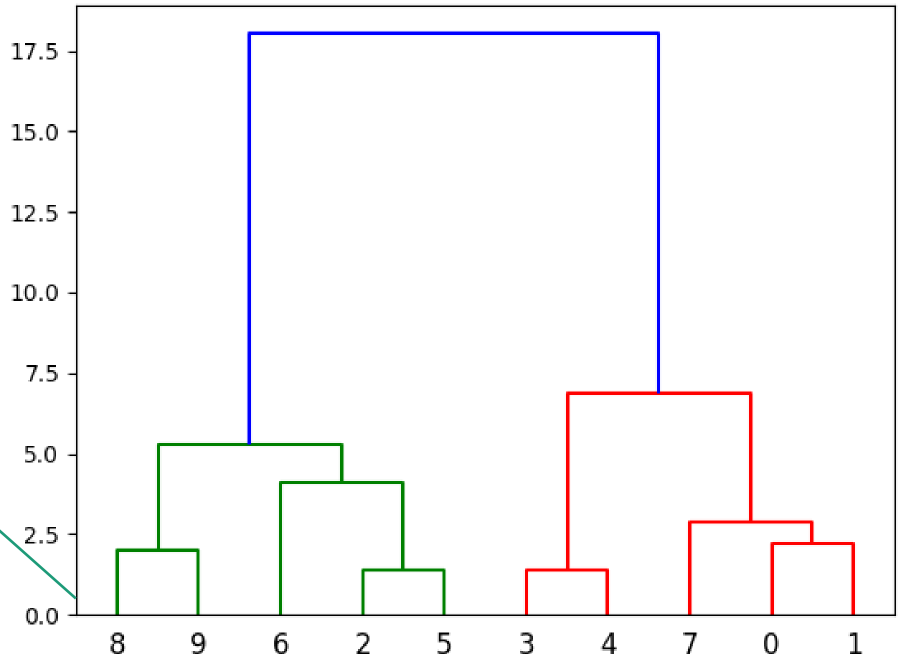


Now, Let’s Play With Multiple Classes

- Use a dendrogram, explains which data differs how much.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage

x = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]

data = list(zip(x, y))

linkage_data = linkage(data, method='ward’,
metric='euclidean')
dendrogram(linkage_data)

plt.show()
```

Here, watch the [nth-index]



Lab Exercises #2 - KNN

- We will use Pizza-simp.csv
- There are five different pizza's nutrition data. Also, we will us e ONLY two data columns and one class variable today.
- Brand = A – E, p = protein, f = fat

- Draw the scatter plot first.

Lab Exercises #2 - KNN

- Let’s assume that we found a new pizza, with p = 18.0, f = 25.0

- Use KNN algorithm, with K = 3, predict the new pizza’s brand (A-E).

- How about if we use K = 5?

Lab Exercise #3 - Hierarchical Clustering

- Calculate the mean values of protein and fat (p and f) for each pizza brands A-E.
- Using these five points, draw a dendrogram of five pizza brands.

- Which brand(s) are having similar characteristics in terms of nutrition?