ENGR 1330 Computational Thinking with Data Science

Classification

Topic Outline

- Concept of classification
- Classifier, training set, testing set.
- Decision boundary

Objectives

- ✓ Able to train a classifier and perform prediction
- ✓ Visualize the decision boundary

Classification

- For each order Amazon receives, Amazon would like to predict: ***is this order fraudulent?***
- Doctors would like to know: ***does this patient have cancer?***
- Politicians would like to predict: ***are you going to vote for them?***

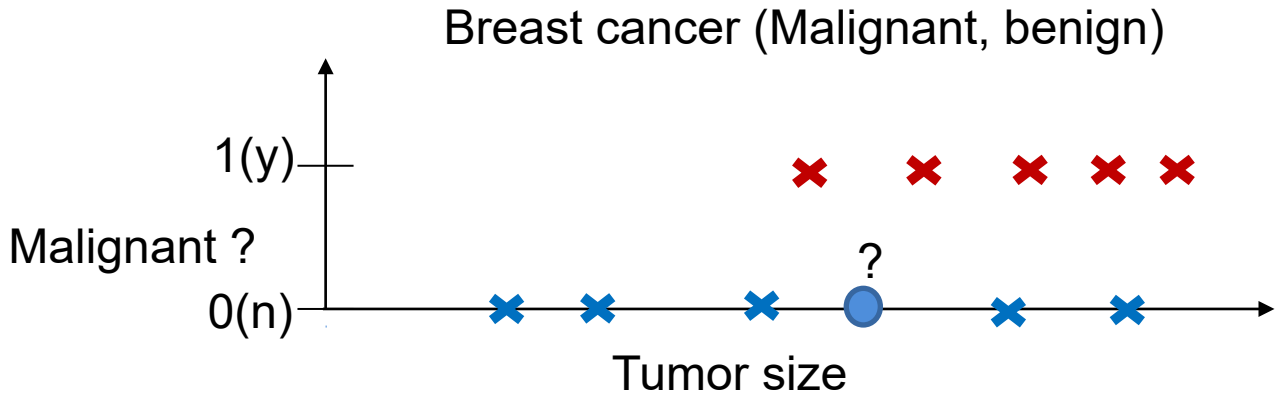
Observation: Each individual or situation where we'd like to make a prediction is called an *observation*

Each observation has multiple *attributes*, which are known (for example, the total value of the order on Amazon, or the voter's annual salary).

Each observation has a *class*, which is the answer to the question we care about (for example, fraudulent or not, or voting for you or not).

Supervised learning

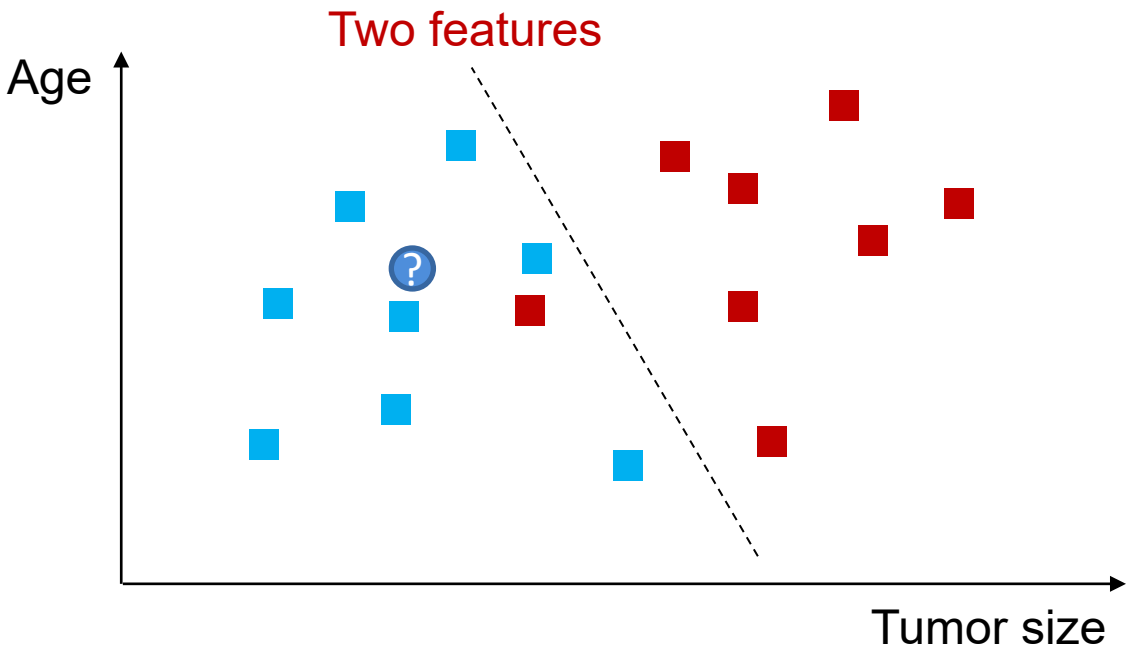
Supervised learning is the machine **learning** task of **learning** a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples [1].



Classification problem: Discrete valued output (0 or 1 or more)

[1] https://en.wikipedia.org/wiki/Supervised_learning

Supervised learning



Classification Algorithm

- ***Classifier***: Analyze the training set and come up an algorithm for predicting the class of future observations. Classifiers do not need to be perfect to be useful. In other words, you don't want the classifier to make too many errors, but it doesn't have to get the right answer every single time.
- ***Training data***: A bunch of observations, where we know the class of each observation. The collection of these pre-classified observations is also called a training set. It is used to build the classification algorithm.
- ***Testing data***: A bunch of observations, where we know the class of each observation used to test the performance of the classifier. The collection of these observation is also called a testing set.

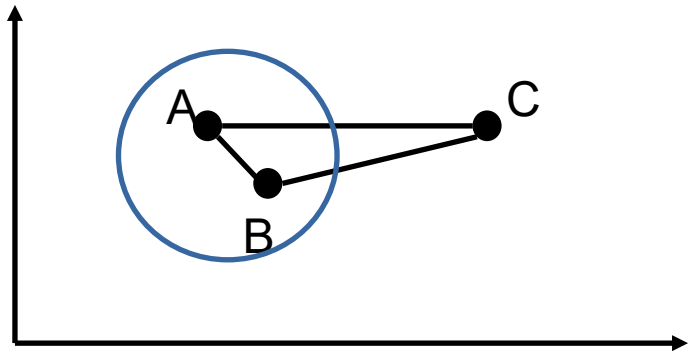
Example: Chronic kidney disease

- Given records of patients with CKD
- Predict a new patient with certain diagnosis (hemoglobin, glucose) having CKD or not?

We rely on the most similar patient that has been diagnosed to make decision on CKD

Comparing Similarity between People

- Treat each person as a point in a coordinate system of 2 dimensions.
- Similarity is the distance between two points
  - The closer of the points, the more similar they are



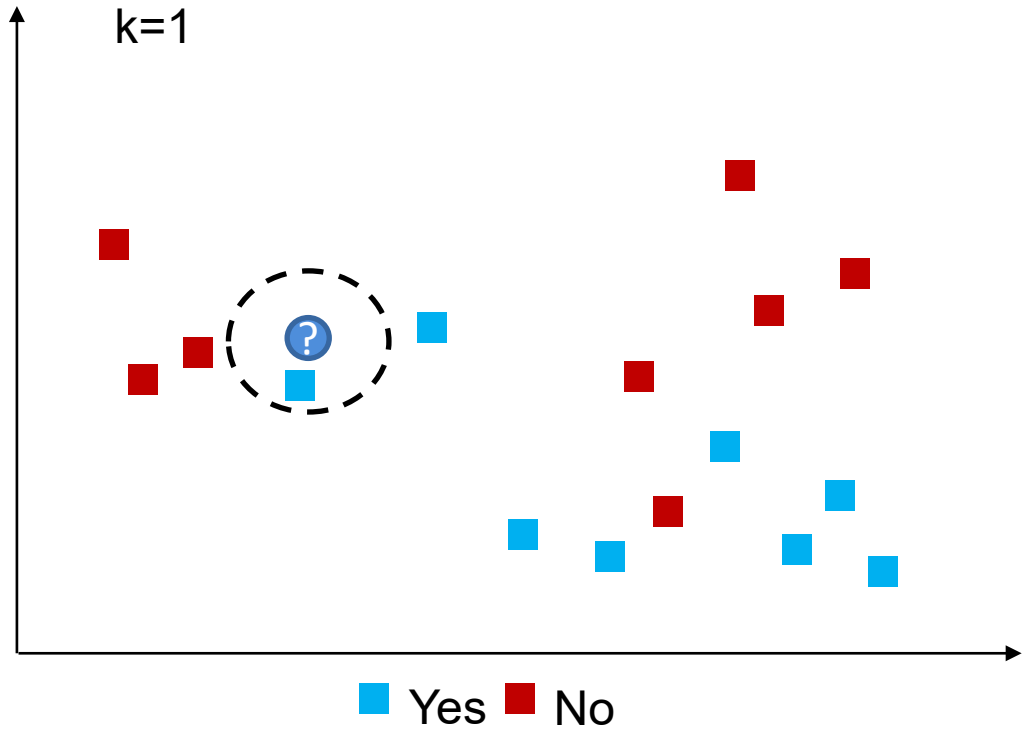
Euclid distance

$$dist = \sqrt{\sum_{k=1}^{n}(p_k - q_k)^2}$$

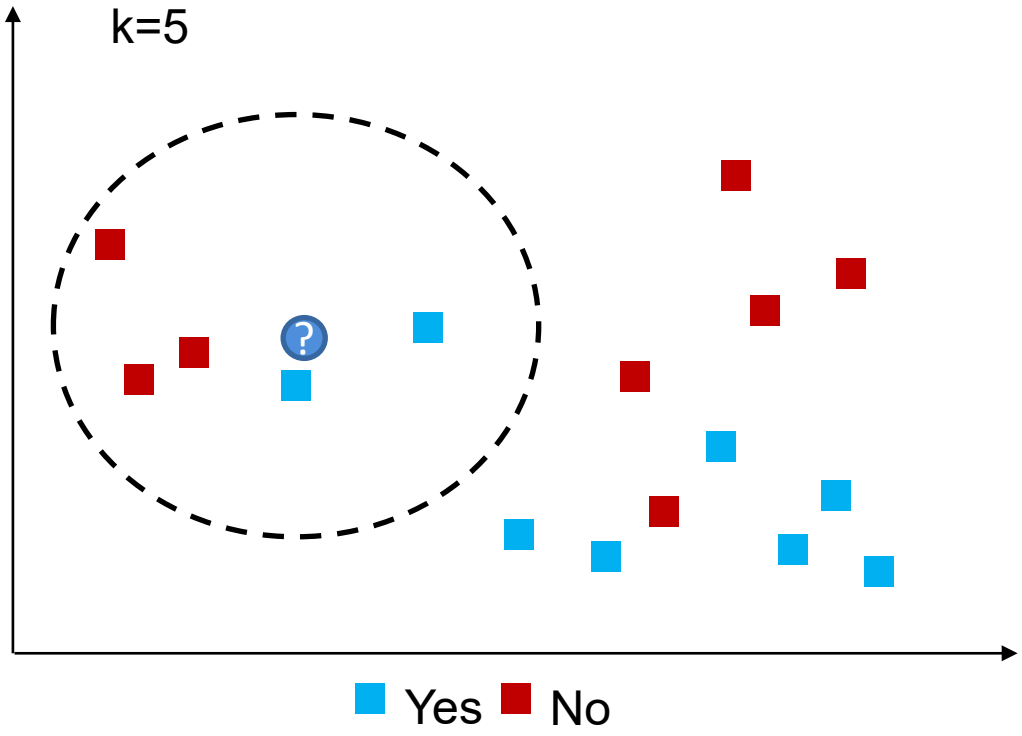
n: number of dimensions

Require standardization if the scales are different.

K Nearest Neighbors (KNN)



K Nearest Neighbors (KNN)



Nearest Neighbor Classifier

We find the nearest point in the scatterplot and check whether it is blue or gold; we predict that the new patient should receive the same diagnosis as that diagnosed patient.

Nearest neighbor

- ✓ Find the point in the training set that is nearest to the new point.
- ✓ If that nearest point is a "CKD" point, classify the new point as "CKD". If the nearest point is a "not CKD" point, classify the new point as "not CKD".

K-Nearest neighbor

- ✓ Compute distance from new point to other points in training set
- ✓ Sort them in ascending order and take k closest patients as references.
- ✓ Use the diagnosis results of k patients for counting votes and making decision.

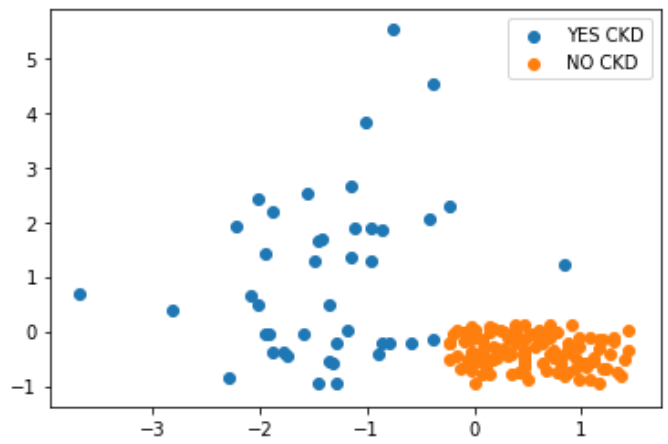
Nearest Neighbor Classifier

```
1  df = pd.read_csv("ckd.csv")
2  df
```

$$z = \frac{value - average}{SD}$$

```
df["Hemoglobin_su"] = (df["Hemoglobin"] - df["Hemoglobin"].mean()) / df["Hemoglobin"].std(ddof=0)
df["Glucose_su"] = (df["Blood Glucose Random"] - df["Blood Glucose Random"].mean()) / df["Blood Glucose Random"].std(d
df["WhiteBCC_su"] = (df["White Blood Cell Count"] - df["White Blood Cell Count"].mean()) / df["White Blood Cell Count'
df = df[["Hemoglobin_su", "Glucose_su", "WhiteBCC_su", "Class"]].copy()
df
```

```
import matplotlib.pyplot as plt

yes_ckd_df = df[df["Class"] == 1]
no_ckd_df = df[df["Class"] == 0]

plt.scatter(x=yes_ckd_df["Hemoglobin_su"], y=yes_ckd_df["Glucose_su"], label="YES CKD" )
plt.scatter(x=no_ckd_df["Hemoglobin_su"], y=no_ckd_df["Glucose_su"], label="NO CKD" )
plt.legend()

plt.show()
```

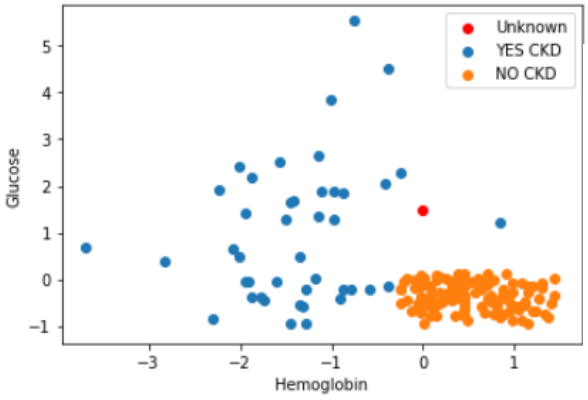


How a computer can automatically predict a new patient about CKD?

K-Nearest Neighbor Classifier

Checking new patient

```
new_patient = [0, 1.5]

yes_ckd_df = df[df["Class"] == 1]
no_ckd_df = df[df["Class"] == 0]

plt.scatter(x=new_patient[0], y=new_patient[1], color="red", label="Unknown")

plt.scatter(x=yes_ckd_df["Hemoglobin_su"], y=yes_ckd_df["Glucose_su"], label="YES CKD" )
plt.scatter(x=no_ckd_df["Hemoglobin_su"], y=no_ckd_df["Glucose_su"], label="NO CKD" )
plt.xlabel("Hemoglobin")
plt.ylabel("Glucose")
plt.legend()

plt.show()
```



Euclid distance

Distance between new_point and labeled points

```
import math as m

def euclide_distance(point1_x, point1_y, point2_x, point2_y):
    temp = (point1_x - point2_x)**2 + (point1_y - point2_y)**2
    return m.sqrt(temp)
```

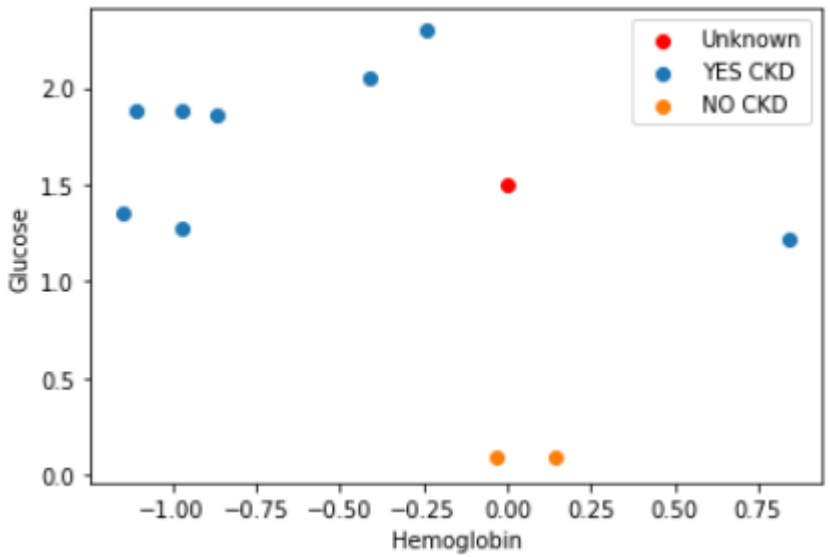
```
distances_to_new_patient = []

for index, row in df.iterrows():
    point1_x = row["Hemoglobin_su"]
    point1_y = row["Glucose_su"]

    distance = euclide_distance(point1_x, point1_y, new_point[0], new_point[1])

    distances_to_new_patient.append(distance)

df["Distance"] = distances_to_new_patient
```

K-Nearest Neighbor Classifier

```
df = df.sort_values(["Distance"], ascending=True)
closest_points = df.head(10)
closest_points
```

K nearest neighbors

	Hemoglobin_su	Glucose_su	WhiteBCC_su	Class	Distance
6	-0.413266	2.049282	0.360623	1	0.687386
13	-0.239236	2.296447	0.424788	1	0.831602
14	0.839750	1.215099	1.291014	1	0.886763
32	-0.865744	1.863908	5.750474	1	0.939118
35	-0.970162	1.276890	-0.345191	1	0.995486
34	-0.970162	1.879356	0.103963	1	1.041693
15	-1.144192	1.354130	-0.922675	1	1.153453
8	-1.109386	1.879356	-0.409356	1	1.172454
84	-0.030400	0.087407	-0.184779	0	1.412920
152	0.143630	0.087407	0.328540	0	1.419876



Plot k=10 neighbors

```
new_patient = [0, 1.5]

yes_ckd_df = closest_points[closest_points["Class"] == 1]
no_ckd_df = closest_points[closest_points["Class"] == 0]

plt.scatter(x=new_patient[0], y=new_patient[1], color="red", label="Unknown")

plt.scatter(x=yes_ckd_df["Hemoglobin_su"], y=yes_ckd_df["Glucose_su"], label="YES CKD" )
plt.scatter(x=no_ckd_df["Hemoglobin_su"], y=no_ckd_df["Glucose_su"], label="NO CKD" )
plt.xlabel("Hemoglobin")
plt.ylabel("Glucose")
plt.legend()

plt.show()
```

```
closest_points["Class"].mode().values[0]
```

```
1
```

Major votes

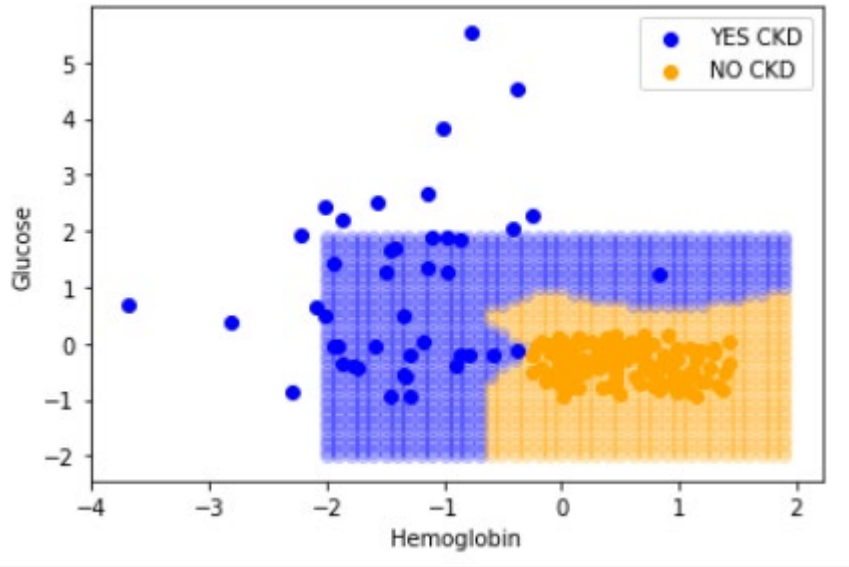
Decision Boundary

referrence_points : training data; being reference when voting
new_point : unknown class and need prediction
k_neighbors *:* number of closest points when do voting. Default=1

```
1  def classify(reference_points, new_point, k_neighbors=1):
2      distances_to_new_patient = []
3
4      for index, row in df.iterrows():
5          point1_x = row["Hemoglobin_su"]
6          point1_y = row["Glucose_su"]
7
8          distance = euclide_distance(point1_x, point1_y, new_point[0], new_point[1])
9
10         distances_to_new_patient.append(distance)
11
12     reference_points["Distance"] = distances_to_new_patient
13     reference_points = reference_points.sort_values(["Distance"], ascending=True)
14     closest_points = reference_points.head(k_neighbors)
15
16     predicted_label = closest_points["Class"].mode().values[0]
17     return predicted_label
18
```

Grouped the classification steps into a function.

Decision Boundary




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

yes_ckd_df = df[df["Class"] == 1]
no_ckd_df = df[df["Class"] == 0]

hemoglobins = np.arange(-2, 2, 0.1)
glucoses = np.arange(-2, 2, 0.1)

for h in hemoglobins:
    for g in glucoses:
        new_point = [h, g]
        predicted_val = classify(df, new_point, k_neighbors=1)
        if predicted_val == 1:
            plt.scatter(x=h, y=g, color="blue", alpha=0.2)
        else:
            plt.scatter(x=h, y=g, color="orange", alpha=0.2)

plt.scatter(x=yes_ckd_df["Hemoglobin_su"], y=yes_ckd_df["Glucose_su"], color="blue", label="YES CKD" )
plt.scatter(x=no_ckd_df["Hemoglobin_su"], y=no_ckd_df["Glucose_su"], color="orange", label="NO CKD" )
plt.xlabel("Hemoglobin")
plt.ylabel("Glucose")
plt.legend()

plt.show()
```

Decision Boundary

- Boundary between the two areas, where points on one side of the boundary will be classified 'CKD' and points on the other side will be classified 'not CKD'. This boundary is called the *decision boundary*.
- Each different classifier will have a different decision boundary; the decision boundary is just a way to visualize what criteria the classifier is using to classify points.