Machine Learning (ML)

Chapter 6:

Classification

(Logistic Regression, Bayes' theorem, KNN and Evaluate the performance)

Saeed Saeedvand, Ph.D.

Outline

In this Chapter:

- ✓ Classification Problems
- ✓ Bayes' theorem
- ✓ Bayes optimal classifier
- ✓ Logistic Regression (trained by GD)
- ✓ Sigmoid function
- ✓ Maximum Likelihood Estimation (MLE)
- ✓ KNN (k-nearest neighbors)
- ✓ Evaluate the performance of a classification
  - ➢ Confusion Matrix
  - ➢ Accuracy
  - ➢ Precision
  - ➢ Recall
  - ➢ F1-Score
  - ➢ Area Under the ROC Curve (AUC-ROC)

Aim of this chapter:

- ✓ Understand important machine learning classification algorithms and learn how to evaluate the classification results.

Classification (Reminder)

Classification Problems

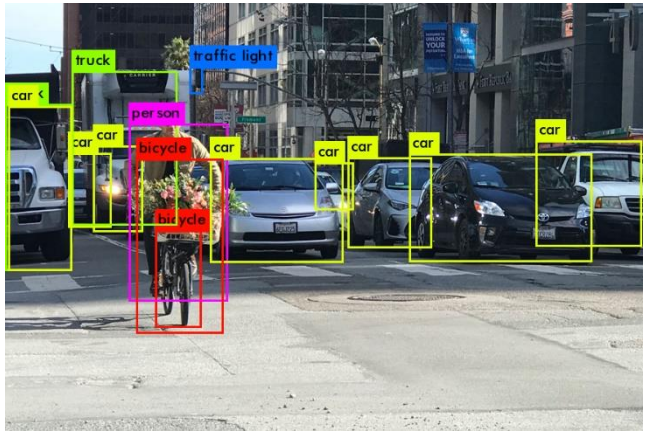
- ✓Type of ML problem in which the goal is to predict the class or category of a **given input** based on a set of **labeled training data**.
- ✓Learn a **mapping function** from input features to class labels.
- ✓For instance, in **image classification**, an image is inputted, and the classes could be different types of objects or animals, such as cats, dogs, or birds.

Classification (Reminder)

Classification Problems

There are different popular algorithms for classification task in supervised learning:

- ✓ Naive Bayes
- ✓ **Logistic Regression**
- ✓ **k-Nearest Neighbors (K-NN)**
- ✓ Decision Trees
- ✓ Random Forest
- ✓ Support Vector Machines (SVM)
- ✓ Neural Networks
- ✓ Gradient Boosting



Classification

What is the Conditional class probabilities?

✓ Probability of a specific class given an input value x.

$$P_k(x) = Pr(Y = k|X = x), where\ k = 1, 2, .., K$$

Probability of belonging input x to class k

class of variable

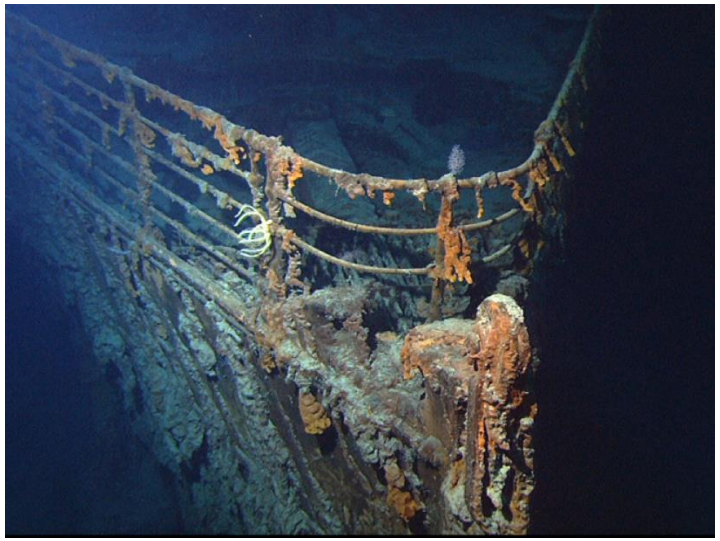
✓ These probabilities can be used to **classify new, unlabeled observations.**

✓ We need to trained ML models, which has learned to **map input features** to **class of labels** based on labeled training data.

Classification

Bayes' theorem

- ✓Bayes' theorem is widely used in ML and data science.
- ✓The most important concept in probability theory to model and reason uncertainty.
- ✓ In **1998,** Tommy Thomson, et.al used it to uncover a ship that sunk in century (worth 50,000,000$).
- ✓ By incorporating multiple sources of information into a probabilistic model, and prioritize their search efforts.



Classification

Bayes' theorem

- ✓Describes the relationship between conditional probabilities.
- ✓Probability of a output y given some observed evidence x:

$$P(y|x)$$

Bayes tells us how to update our belief for new inputs

Classification

Spam detection as an example

- ✓ If we have **categorical data naturally** we call it qualitative.
- ✓ In Spam detection the objective is to **classify emails** as either "spam" or "ham"
  - We can write C = (spam, ham)
- ✓ The task is to **construct a classifier C(X)** that can assign a class label from C to a new unlabeled observation X.
- ✓ To do this we have **different algorithms** in Statistical Learning and Machine Learning.

Classification

Spam detection as an example

✓ As start it's crucial to get the extract different features for predictors:

$$X = (X_1, X_2, X_3, \dots, X_p)$$

Binary feature

1. **Word frequency:** Indicating whether a word appears in the email or not.
2. **HTML tags: E**ach tag of interest appears in the email or not.
3. **Sender information:** Email comes from a known spammer or legitimate sender.
4. **Message length:** Binary feature indicating whether the message is longer than a certain threshold.
5. **Image analysis:** Binary feature indicating whether the email contains images with embedded text.
6. **Time/date sent:** Email was sent at **certain times of day** or on **certain days of the week**.
7. **Content analysis:** Whether the email contains certain phrases or language patterns that are commonly used in spam emails.

Classification

Bayes' theorem

- ✓ Using Bayes' theorem, we can update our belief or probability of an email as (spam or ham)
- ✓ This is based on based on the observed evidence or email content.

$$P(y|x) = P(y) \times P(x|y) / P(x)$$

$$P(y|x) = \frac{P(x|y) * P(y)}{P(x)}$$

$$P(y|x) = \frac{P(x|y) * P(y)}{P(y)P(x|y) + P(\neg y)P(x|\neg y)}$$

Random variable:

- In spam detection class label (spam or not spam)

Random variable:

- Input features (e.g., email contents)

Bayes' theorem

✓ Using Bayes' theorem, formally includes different parts as follows.

Prior probability **Marginal probability**

$$P(y|x) = P(y) \times P(x|y) / P(x)$$

Posterior probability **Likelihood**

Classification

Bayes' theorem

Posterior probability

$$P(y|x) = P(y) \times P(x|y) / P(x)$$

Posterior probability:

- Probability of being in one of class based on its features.
- Probability of the class label (y) given the observed features (x).

$$P(\text{spam}|x=\begin{pmatrix}0\\0\\0\\1\end{pmatrix}) = 0.2 \qquad P(\text{ham}|x=\begin{pmatrix}0\\0\\0\\1\end{pmatrix}) = 0.8$$

Classification

Bayes' theorem

Prior probability **Marginal probability**

$$P(y|x) = P(y) \times P(x|y) / P(x)$$

Posterior probability **Likelihood**

```
import numpy as np

# Existing Email Dataset
X = np.array([[1, 0, 1, 0], [0, 0, 0, 1], [1, 0, 0, 1], [1, 1, 0, 1], [1, 1, 0, 1], [1, 1, 0, 1],
[0, 1, 1, 0], [1, 1, 1, 1]])
Y = np.array([1, 0, 0, 0, 1, 0, 1, 1])  # Lables 1 for spam, 0 for ham

# Incoming New email
x = np.array([1, 1, 0, 1])
```

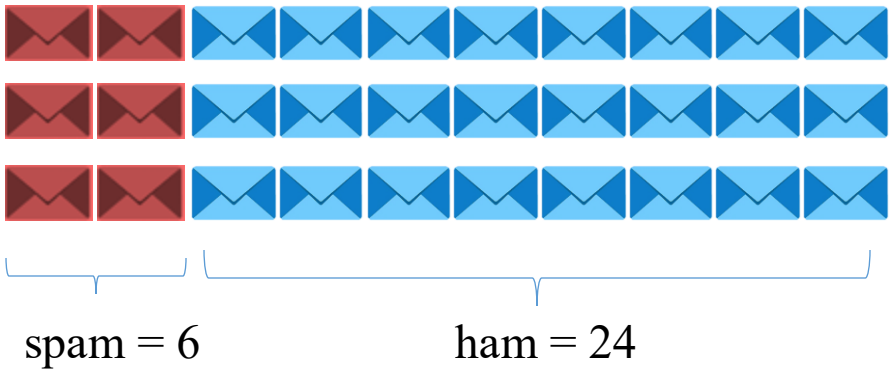
Classification

Bayes' theorem

New Data: New Email can be spam or ham

If new email is spam should we change our previous belief completely that other incoming emails are spam too?

No we need to update our belief
(Bayes' theorem)



Classification

Bayes' theorem

Prior probability

$$P(y|x) = P(y) \times P(x|y) / P(x)$$

Prior probability:

✓ Represents initial belief or the probability of a class label y (before observing the input instance x)

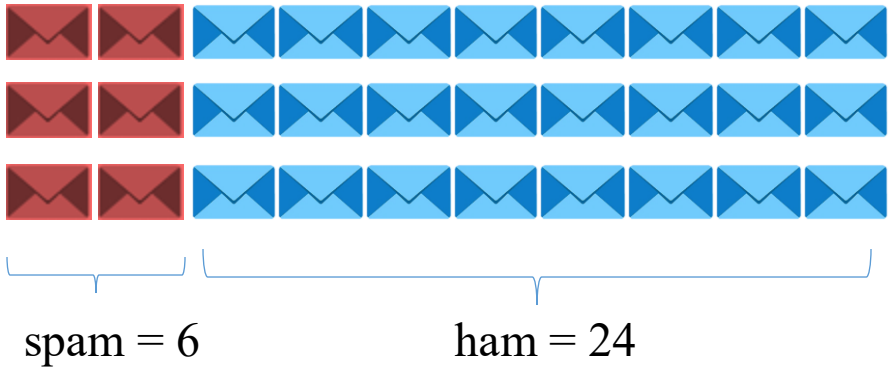
✓ Reflects any **prior knowledge or assumptions** we may have about the **dataset** **(called domain expertise)** or external knowledge .

if we don't know any of them?

- In the absence of any external information, a reasonable assumption is to assume that probabilities are equally likely to occur.

Classification

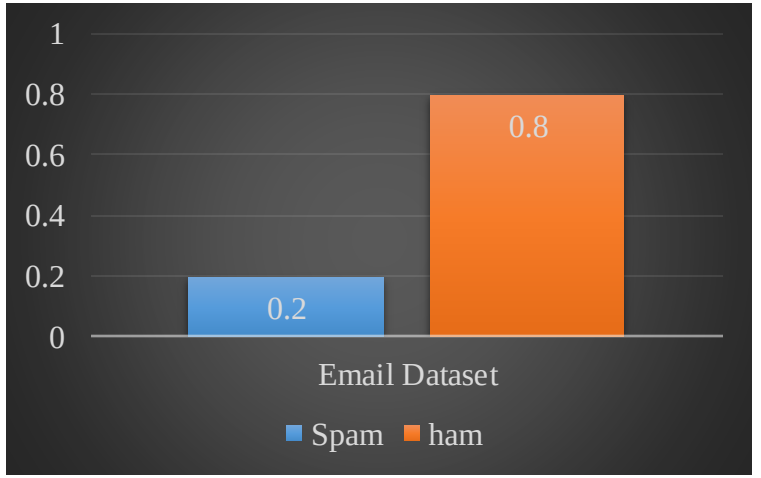
Bayes' theorem



Prior probability:

$$P(y) = P(\text{ham}) = \frac{24}{(6+24)} = 0.8$$

$$P(y) = P(\text{spam}) = \frac{6}{(6+24)} = 0.2$$



Classification

Bayes' theorem

Prior probability

$$P(y|x) = P(y) \times P(x|y) / P(x)$$

```
# Prior probabilities (we call domain expertise in absence of external knowledge)
P_spam = sum(Y) / len(Y)
P_ham = 1 - P_spam
```

Classification

Bayes' theorem

Likelihood

$$P(y|x) = P(y) \times P(x|y) / P(x)$$

Likelihood:

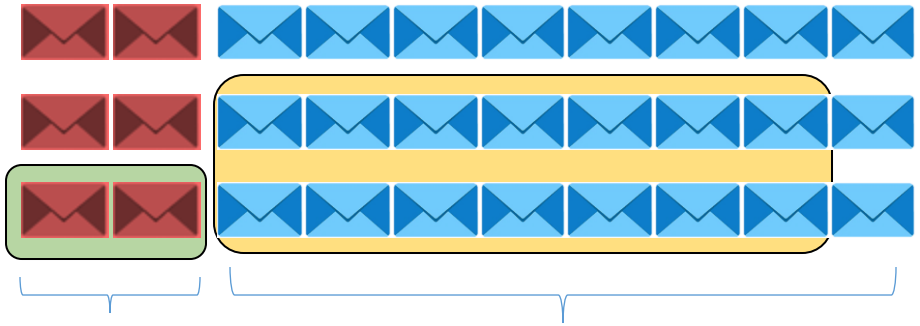
- ✓ The probability of observing every feature combination of input instance x given the class label y in dataset.
- ✓ Likelihood is often modeled as a probability distribution over the input features, such as a Gaussian distribution.
- ✓ Also can be called Density function of measurement.

Bayes' theorem

New Data: New Email can be spam or ham

Likelihood:

- ✓ How many of each class are **fitting** with common features of incoming data?



$$P(x|spam) = \frac{2}{6} = 0.33 \qquad P(x|ham) = \frac{14}{24} = 0.58$$

Classification

Bayes' theorem

Likelihood

$$P(y|x) = P(y) \times P(x|y) / P(x)$$

```
# Estimate conditional probabilities and Likelihood (features new email)
P_x_ham = np.mean(X[Y == 0], axis=0)
P_x_spam = np.mean(X[Y == 1], axis=0)
P_x_ham_x = np.prod(P_x_ham[x == 1]) * np.prod(1 - P_x_ham[x == 0])
P_x_spam_x = np.prod(P_x_spam[x == 1]) * np.prod(1 - P_x_spam[x == 0])
```

Naive Bayes assumption: (Include each feature independently, extension of previous slide)

P(x|spam) = P(x_1|spam) * P(x_2|spam) * ... * P(x_p|spam)

P(x|ham) = P(x_1| ham) * P(x_2| ham) * ... * P(x_p| ham)

Note: assuming that the input variables are conditionally independent so that we multiply not sum

Classification

Bayes' theorem

Marginal probability

$$P(y|x) = P(y) \times P(x|y) / P(x)$$

Marginal probability:

✓ Represents the probability of observing the evidence in the **overall population** (for all class labels).

How to write overall population:

$$P(y)P(x|y) + P(\neg y)P(x|\neg y)$$

$$=$$

$$P(\text{ham})P(x|\text{ham}) + P(\textit{spam})P(x|\textit{spam})$$

i.e., the email content for $, or … in both spam and ham

Add all the joint probabilities of observing the features x and each possible class label y:

Classification

Bayes' theorem

Marginal probability

$$P(y|x) = P(y) \times P(x|y) / P(x)$$

As shown in previous slide:

$$P(x) = P(\text{ham})P(x|\text{ham}) + P(\textit{spam})P(x|\textit{spam})$$

```
# Marginal probability (for the observing the features)
P_x = P_x_spam_x * P_spam + P_x_ham_x * P_ham
```

Classification

Bayes' theorem

$$P(y|x) = P(y) \times P(x|y) / P(x)$$

Posterior probability

```
# Posterior probabilities (Bayes' theorem)
P_spam_x = P_x_spam_x * P_spam / P_x
P_ham_x = P_x_ham_x * P_ham / P_x

print('Posterior probability for ham:', P_ham_x)
print('Posterior probability for spam:', P_spam_x)
```

Classification

Bayes optimal classifier

✓Find the **maximum probability using** Bayes' theorem per each class and assign to that corresponding class.

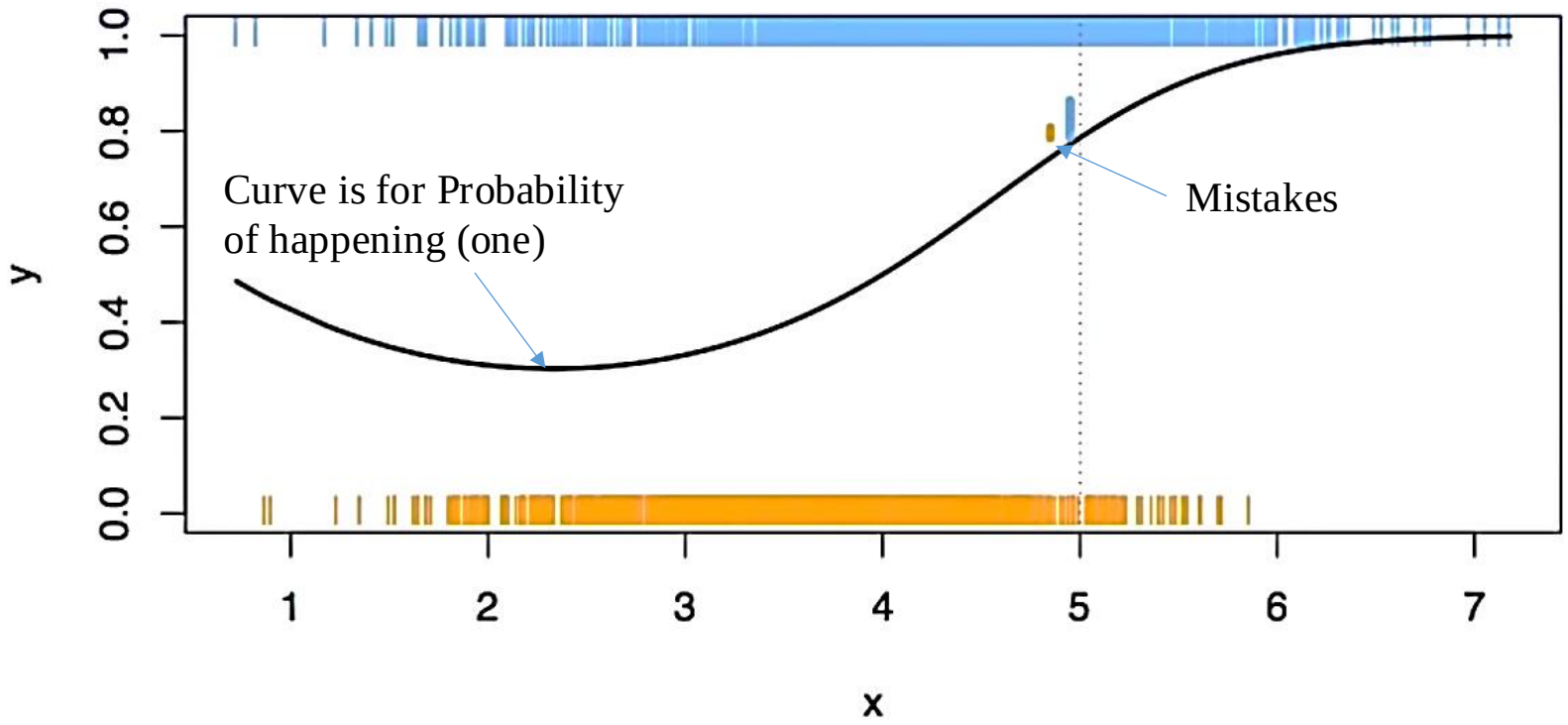
$$y^* = \underset{j}{\operatorname{argmax}}(P(y_1|x), P(y_2|x))$$

```
# Bayes optimal classifier (labeling)
if P_spam_x > P_ham_x:
    print('New Email: Spam')
else:
    print('New Email: Ham')
```

Classification

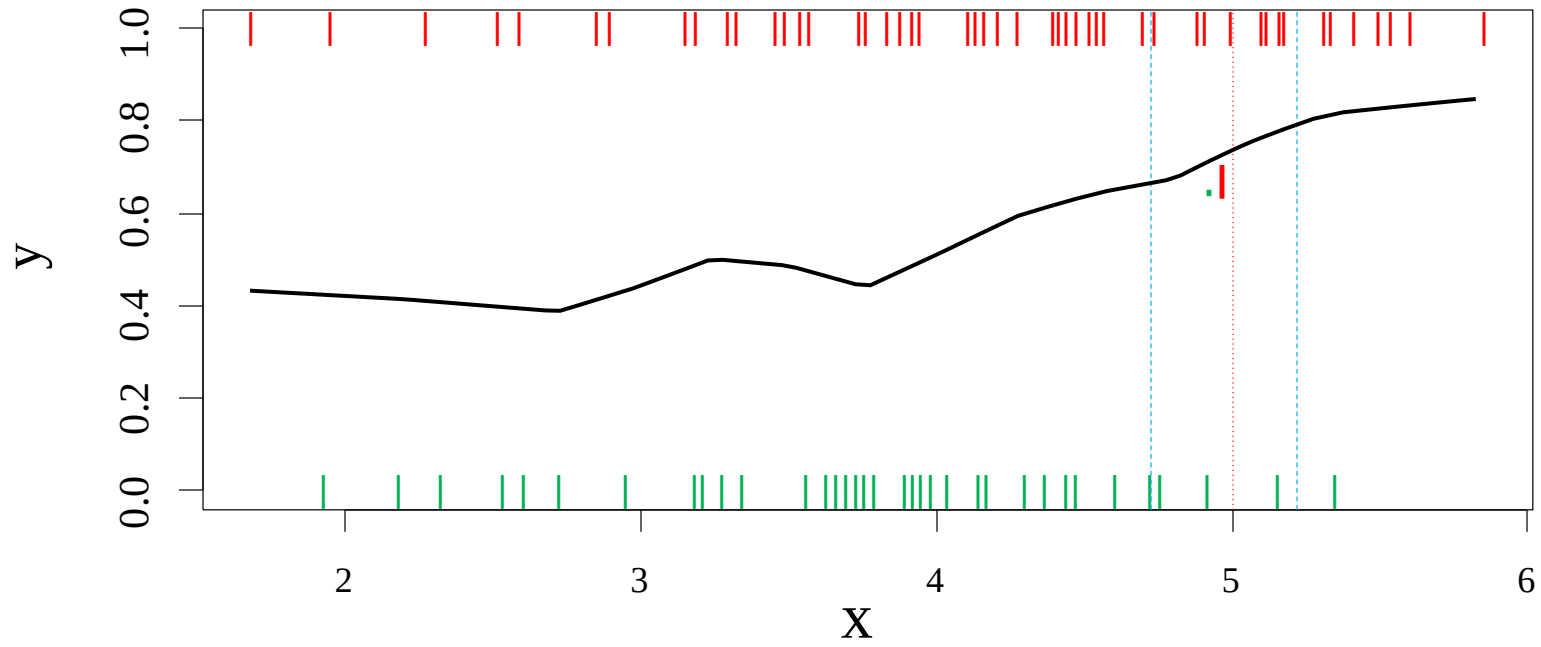
Spam detection

Bayes optimal classifier



Classification

- ✓ In the case that there is not enough data (100 points).
- ✓ Cannot calculate conditional probability for missing input x.
- ✓ So we can use the idea of **nearest neighbor** to generate model again.



Classification

Logistic Regression:

- ✓ A popular machine learning approach.
- ✓ A supervised learning algorithm for binary classification problems.
- ✓ We want to predict the probability that an input example belongs to one of **two classes**.
- ✓ A simple and efficient algorithm that we can train different **large datasets** with **many features** with it.
- ✓ It is based on the concepts of linear regression.
- ✓ Logistic regression can be trained using GD, SGD, … optimizations.

Classification

Logistic Regression:

- ✓ Works by modeling the relationship between the **input features** and the binary output variable.
- ✓ It uses a logistic function (**sigmoid function**) to do this.
- ✓ Same as linear regression it can work with continues data like price and discrete data.

For example:

Input: age, smoking amount, gender, overweigh
Class: Diseased, not diseased

Linear regression (Reminder)

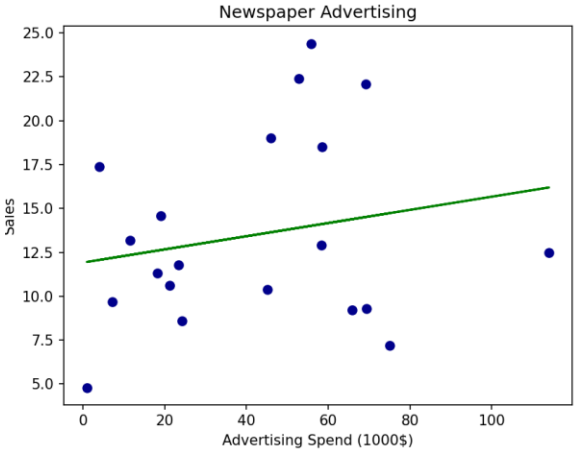
Why cannot use Linear regression directly for classification?

- ✓ Linear regression model was:

$$\hat{y}_i = \hat{\theta}_0 + \hat{\theta}_1 x_i + \varepsilon$$

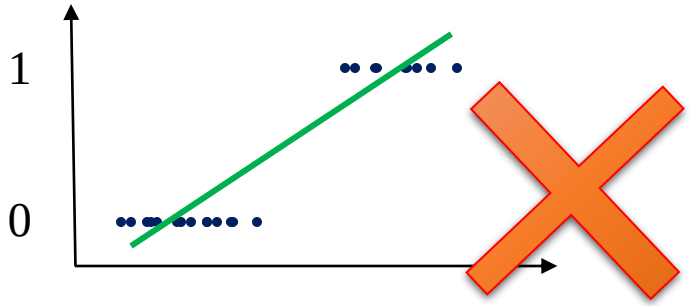
slope

Intercept



- ✓ In classification problems we have class of true and false:

In **logistic regression** we want to find the probability of occurrence in range of value of 0 and 1 only, not like previous.



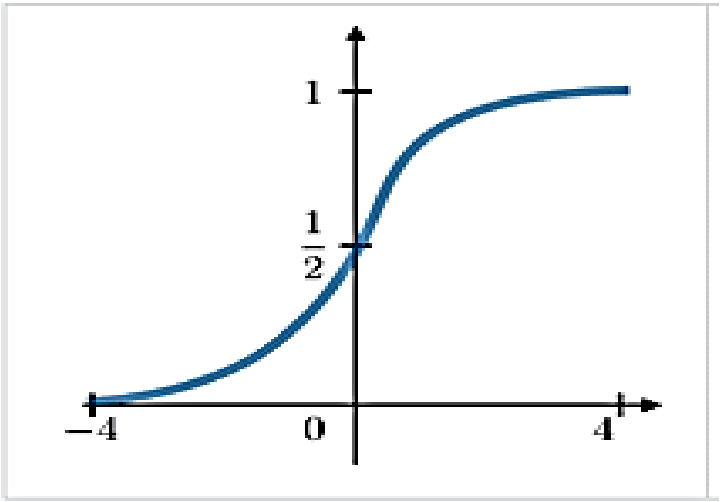
Logistic Regression

What is the Sigmoid function or Logistic function

- ✓ Logistic function, also known as the sigmoid function.
- ✓ It is a mathematical function that maps any real-valued input to a value between 0 and 1 (**exactly what we want**).
- ✓ **Commonly used** in logistic regression and other binary classification like deep learning models.
- ✓ It models the relationship between the **input** features and the **output** variable.

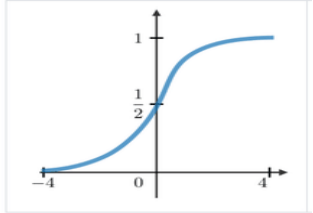
$$h(x) = \frac{1}{1 + e^{-x}}$$

real-valued input



Logistic Regression

How to write Sigmoid function for Logistic Regression



$$h(x) = \frac{1}{1 + e^{-x}}$$

- ✓ **Probability** of the dependent variable to be 1 by given input can be written (called likelihood):

$$p_i(\hat{y} = 1 | x_1, x_2, \dots, x_i) = \frac{1}{1 + e^{-(\hat{\theta}_0 + \hat{\theta}_1 x_1 + \cdots + \hat{\theta}_i x_i + \varepsilon)}}$$

Independent variables

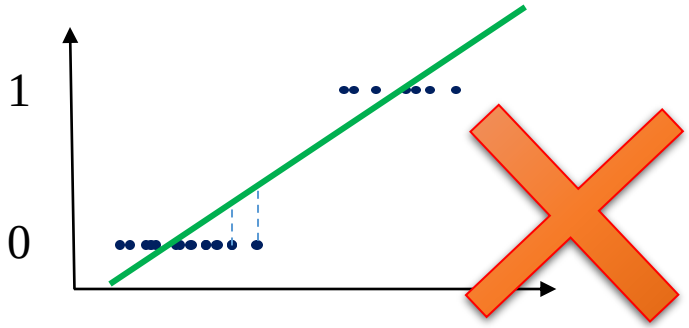
Now we find the coefficients by GD or SGD

What is the objective function then?

Logistic Regression

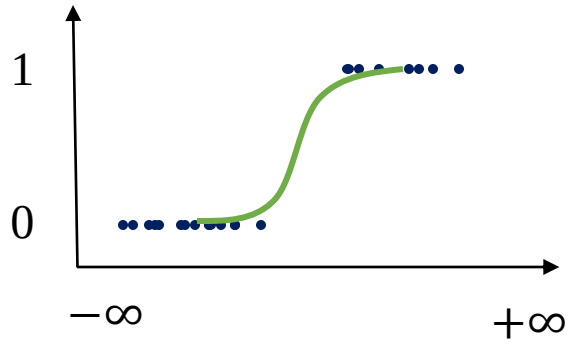
Loss Function

✓ In logistic regression, we cannot use MSE or R^2 to write cost function for the.



✓ **We Use:**

- Maximum likelihood
  - ➢ With negative log-likelihood function or cross-entropy loss function.



Logistic Regression - Loss Function

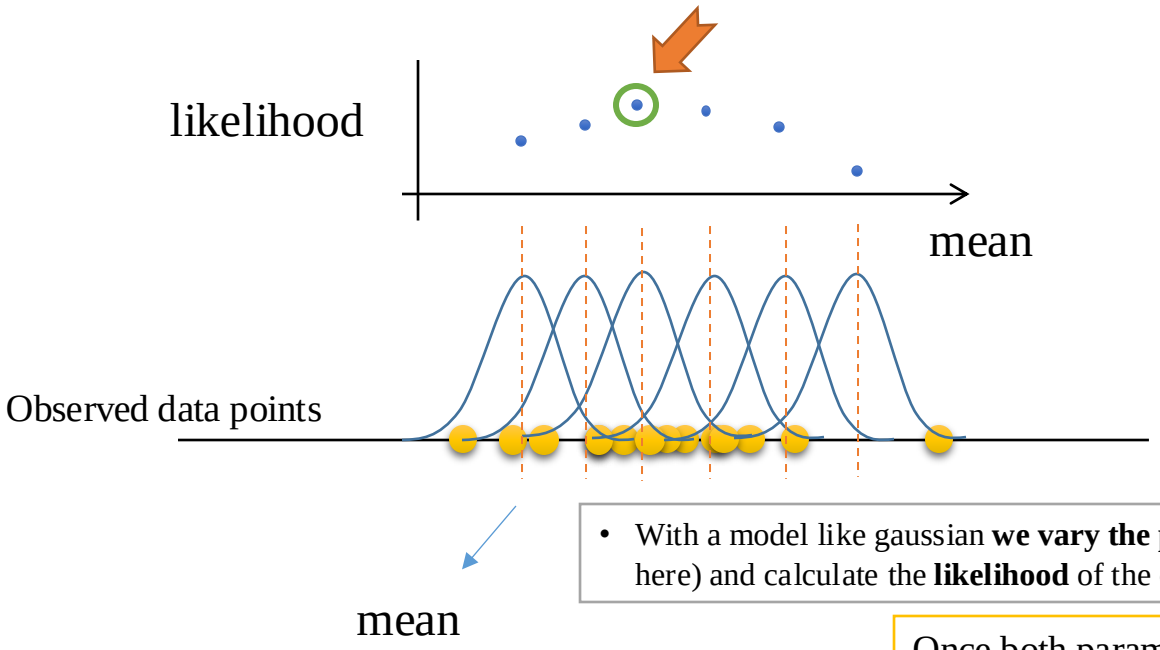
Maximum Likelihood Estimation (MLE)

- ✓ MLE method is commonly used to estimate the parameters of Machine Learning algorithms like:
  - Logistic regression, NN, NLP, Signal Processing, Computer Vision, ….
- ✓ The concept of MLE is different than finding the MSE which was showing us the model's error to use it to find coefficients.
- ✓ MLE can be directly used to be applied on Logistic Regression without SGD but costly!
  - Works by finding best fit of a distribution (for example normal distribution) on the data.

Logistic Regression - Loss Function

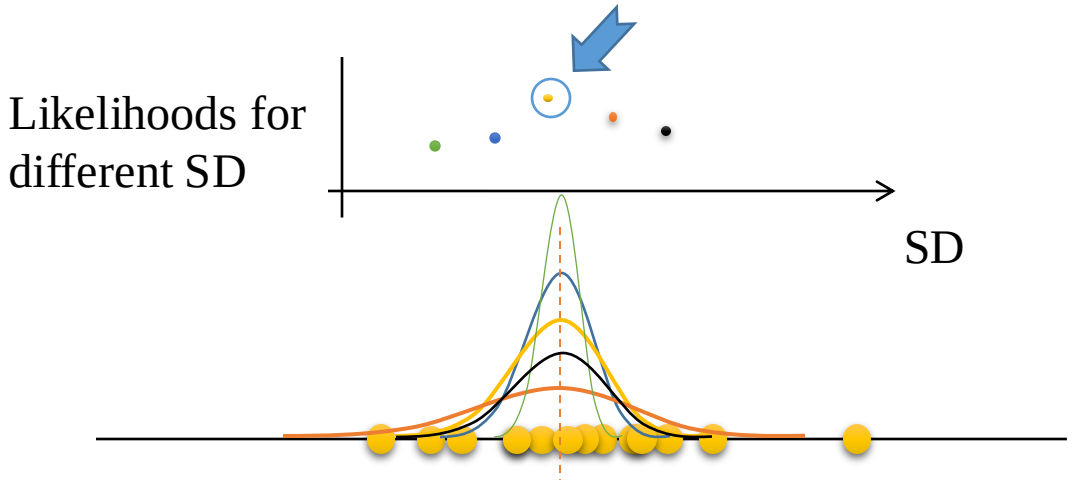
Maximum Likelihood Estimation (MLE)

1 Finding Maximum Likelihood:



- With a model like gaussian **we vary the parameters** (mean here) and calculate the **likelihood** of the observed data.

2 Finding standard deviation:



- Same but **vary the parameters** (SD here) this time.

Once both parameters are optimized:

✓ We have the **final Gaussian model** that best explains the observed data.

Logistic Regression - Loss Function

Maximum Likelihood Estimation (MLE)

- ✓ Maximizing the likelihood function is a difficult optimization problem.
- ✓ Instead, the optimization process in logistic regression often involves minimizing the negative log-likelihood function.
- ✓ Which is mathematically equivalent to maximizing the likelihood function.
- ✓ In fact we minimize the negative log-likelihood but approach **called** the Maximum Likelihood Estimates (MLE) of the **coefficients also same as** cross-entropy.

Logistic Regression with GD

How to apply the GD?

Step 1

Initialize the parameters randomly:

- Choose initial values for $\hat{\theta}_0$ and $\hat{\theta}_1$ (For example, $\hat{\theta}_0 = 0$ and $\hat{\theta}_1 = 0$,…) in sigmoid function or logistic function:

$$p_i(\hat{y} = 1|x_1, x_2, \dots, x_i) = \frac{1}{1 + e^{-(\hat{\theta}_0 + \hat{\theta}_1 x_1 + \cdots + \hat{\theta}_i x_i + \varepsilon)}}$$

Logistic Regression with GD

Calculate the Cost function:

Step 2

✓ We can use Gradient Descent (GD) with minimizing Negative log-likelihood to estimate the parameters of a logistic regression model.

Negative log-likelihood

✓ We Calculate the value of the cost function (Negative log-likelihood) given the current values of the coefficients $\hat{\theta}$:

$$J(\hat{\theta}) = -\sum [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

Known as cross-entropy

We transform the product of probabilities into a sum of logarithms
It makes the optimization process computationally easier.

Negative because of minimizing log-likelihood mathematically equivalent to maximizing the likelihood function

Logistic Regression with GD

Step 3

Calculate the gradient:

✓ Now we can use Gradient of cost function (negative log-likelihood function):

$$\nabla J(\hat{\theta}) = \frac{\partial J(\hat{\theta})}{\partial(\hat{\theta})}$$

$$\frac{\partial J(\hat{\theta})}{\partial(\hat{\theta}_0)} = \sum_{i=1}^{n} [p_i - \hat{y}_i]$$

$$\frac{\partial J(\hat{\theta})}{\partial(\hat{\theta}_1)} = \sum_{i=1}^{n} [p_i - \hat{y}_i]\, x_1$$

…

$$\frac{\partial J(\hat{\theta})}{\partial(\hat{\theta}_i)} = \sum_{i=1}^{n} [p_i - \hat{y}_i]\, x_i$$

Predicted class of the model for i-th data point usually [0 or 1]

The predicted probability of $\hat{y}_i = 1$

$$p_i(\hat{y} = 1 | x_1, x_2, \dots, x_i) = \frac{1}{1 + e^{-(\hat{\theta}_0 + \hat{\theta}_1 x_1 + \cdots + \hat{\theta}_i x_i + \varepsilon)}}$$

Parameters of the logistic regression model

feature for observation i

Logistic Regression with GD

Step 4

Update the coefficient parameters:

✓ We Update the values of $\hat{\theta}_0$ and $\hat{\theta}_1$, ... using the gradients and a learning rate α:

New $\hat{\theta}_0$ and $\hat{\theta}_1$,…

$$\hat{\theta}_0 = \hat{\theta}_0 - \alpha \frac{\partial J(\hat{\theta})}{\partial(\hat{\theta}_0)}$$

$$\hat{\theta}_1 = \hat{\theta}_1 - \alpha \frac{\partial J(\hat{\theta})}{\partial(\hat{\theta}_1)}$$

…

$$\hat{\theta}_i = \hat{\theta}_i - \alpha \frac{\partial J(\hat{\theta})}{\partial(\hat{\theta}_i)}$$

Logistic Regression with GD

Step 5

Repeat

✓Run steps 2 to 4 until the cost function gets a minimum or a stopping criterion is met (changes are very small).

Now also we can use SGD, or other approaches instead of GD to optimize logistic Regression

✓logistic regression models can achieve good accuracy for image classification tasks especially in binary class (e.g. cat and dog)

Assignment

Implement Logistic Regression for a classification problem (optionally can be image classification), and report your findings. Use a small or mid-size Kaggle dataset: https://www.kaggle.com/datasets

Logistic Regression Difficulties

Challenge

- ✓ **Logistic regression** can work reasonably well even in the presence of moderate non-linearities.
- ✓ In situations when the nonlinearity is very high, some other ML models may capture non-linear relationships better.
  - Decision trees, Random forests, Neural Networks, or Support Vector Machines (SVMs)

KNN (k-nearest neighbors)

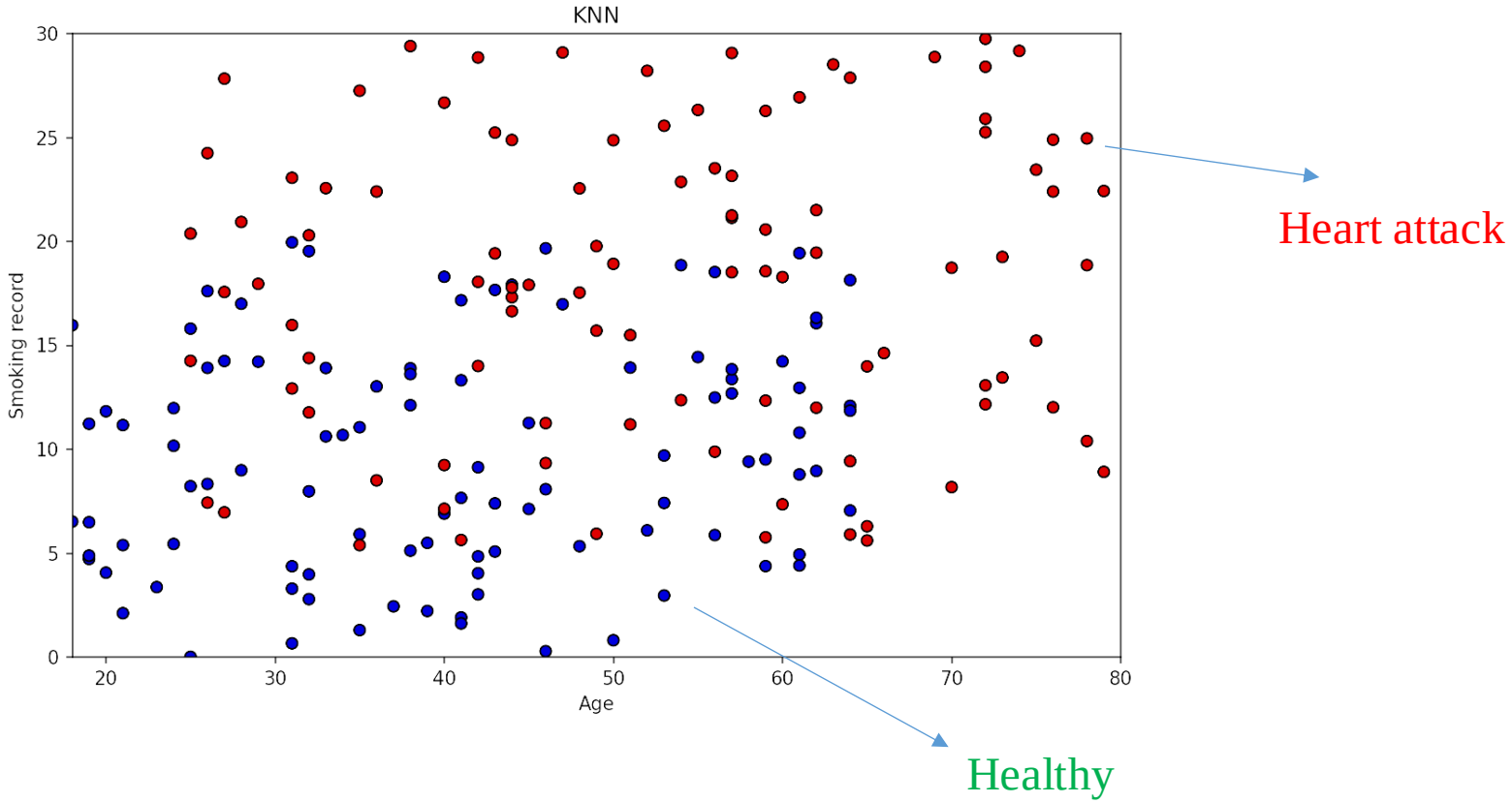
Definition

- ✓ A machine learning algorithm for classification, that can also be used for regression tasks.
- ✓ KNN is non-parametric classification
- ✓ KNN does not build a model during the training phase, and instead it stores the entire training dataset.
- ✓ First we need a labeled dataset as for all supervised methods.
- ✓ In regression tasks for KNN assigns a numerical value based on the average of the k-nearest neighbors' target values (class).

KNN (k-nearest neighbors)

Example

✓ Data for smoking, age and heart attack records (supervised).



KNN (k-nearest neighbors)

Step 1 **Choose the number of neighbors (k) and a distance metric:**

parameter: K depends on dataset and should be adjusted

Metric: Euclidian distance:

$$d(p_1, p_2) = \sqrt{(p_{1\_x1} - p_{2\_x1})^2 + (p_{1\_x2} - p_{2\_x2})^2 + \cdots + (p_{1\_xn} - p_{2\_xn})^2}$$

- **Or we can use Manhattan distance metric:**

$$d(p_1, p_2) = |p_{1\_x1} - p_{2_{x1}}| + \cdots + |p_{1_{xn}} - p_{2_{xn}}|$$

- **Or Minkowski distance metric:** generalized version of the Euclidean and Manhattan distances:

positive real number

$$d(p_1, p_2) = \left(\sum |p_{1\_x1} - p_{2\_x1}|^p\right)^{1/p}$$

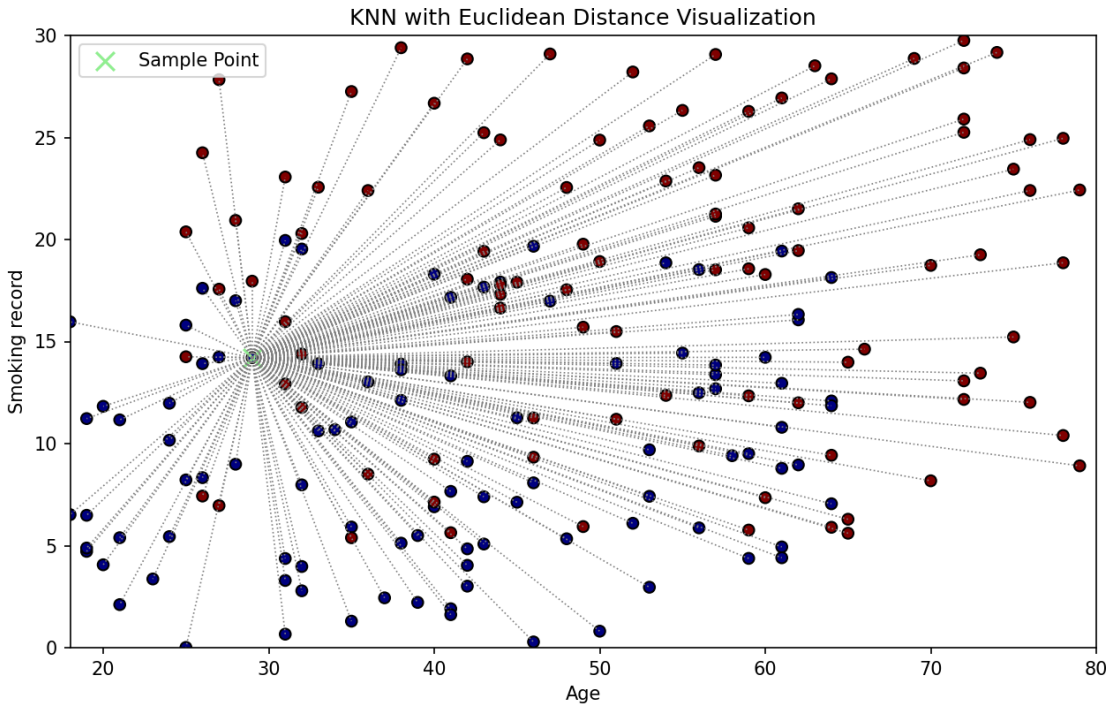
p=1, it becomes the Manhattan distance

p=2, it becomes the Euclidean distance

KNN (k-nearest neighbors)

Step 2

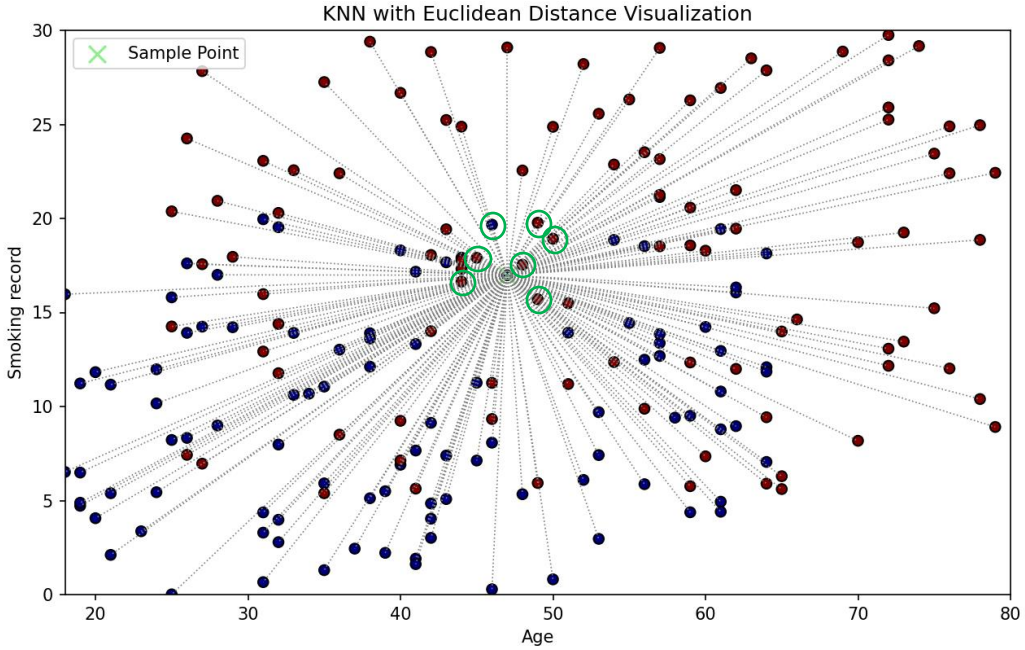
✓ Calculate the distance between all the existing labeled dataset and each new data to be classified. (here k=7)

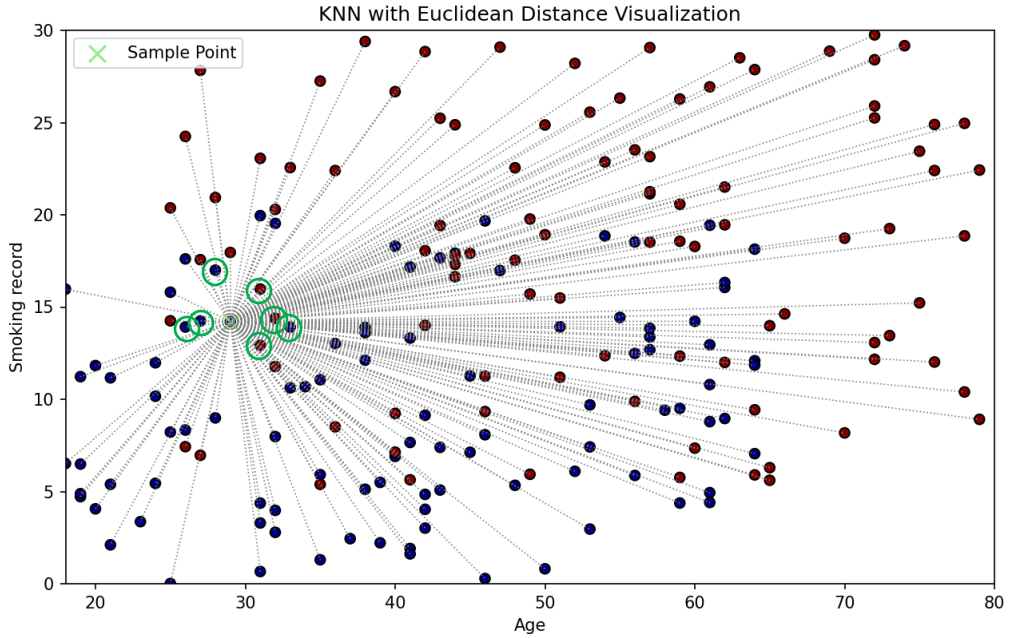


KNN (k-nearest neighbors)

Step 3

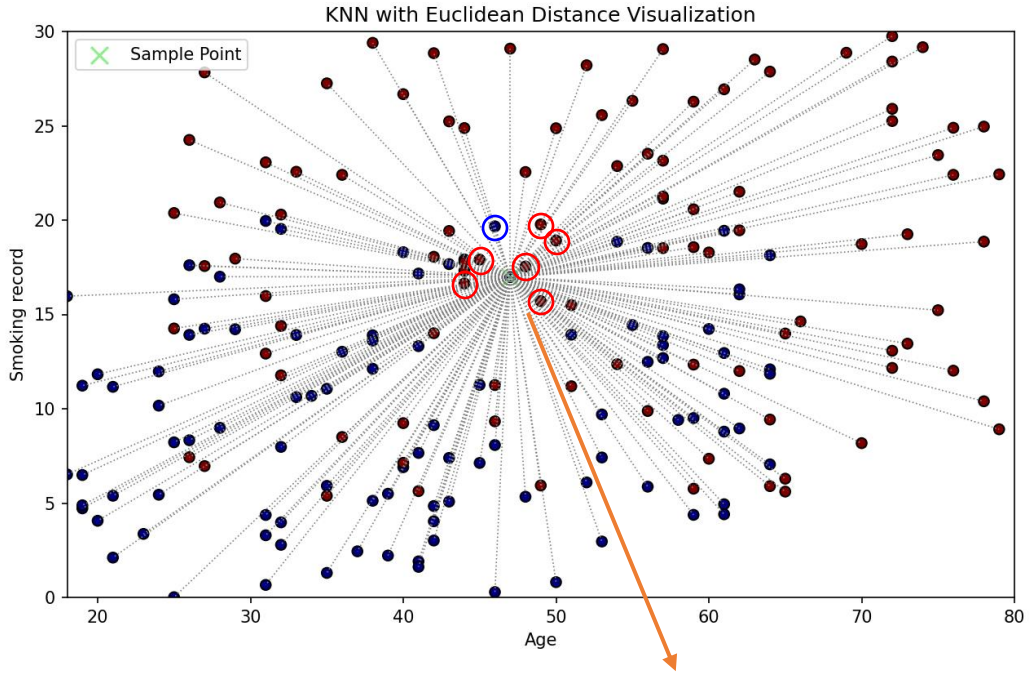
✓ Select the k-nearest instances (i.e., the instances with the smallest distances) to the query instance.

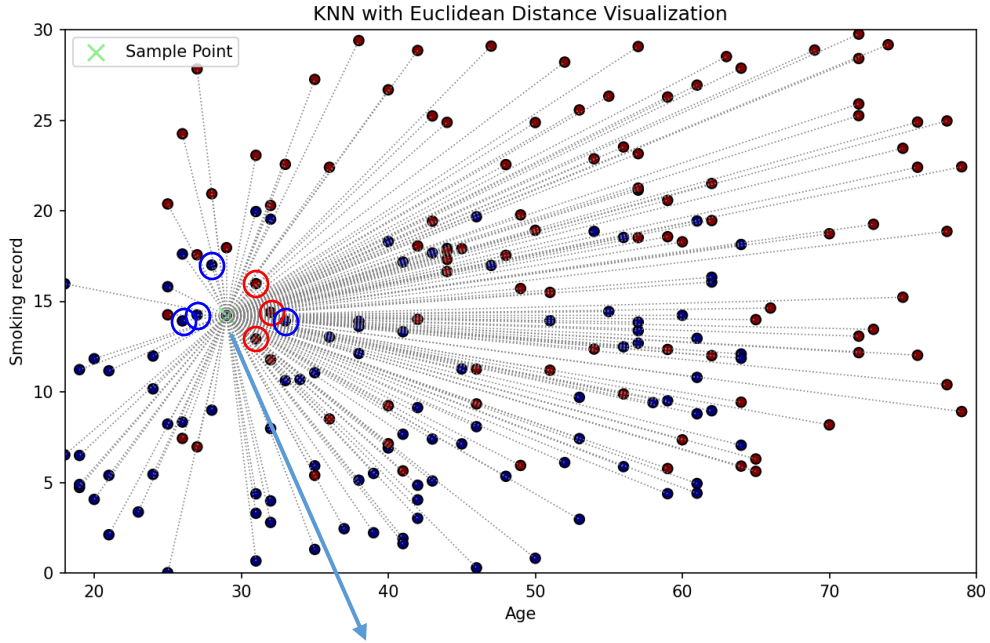




KNN (k-nearest neighbors)

Step 4 ✓ Assign the sample to majority class among the k-nearest neighbors.



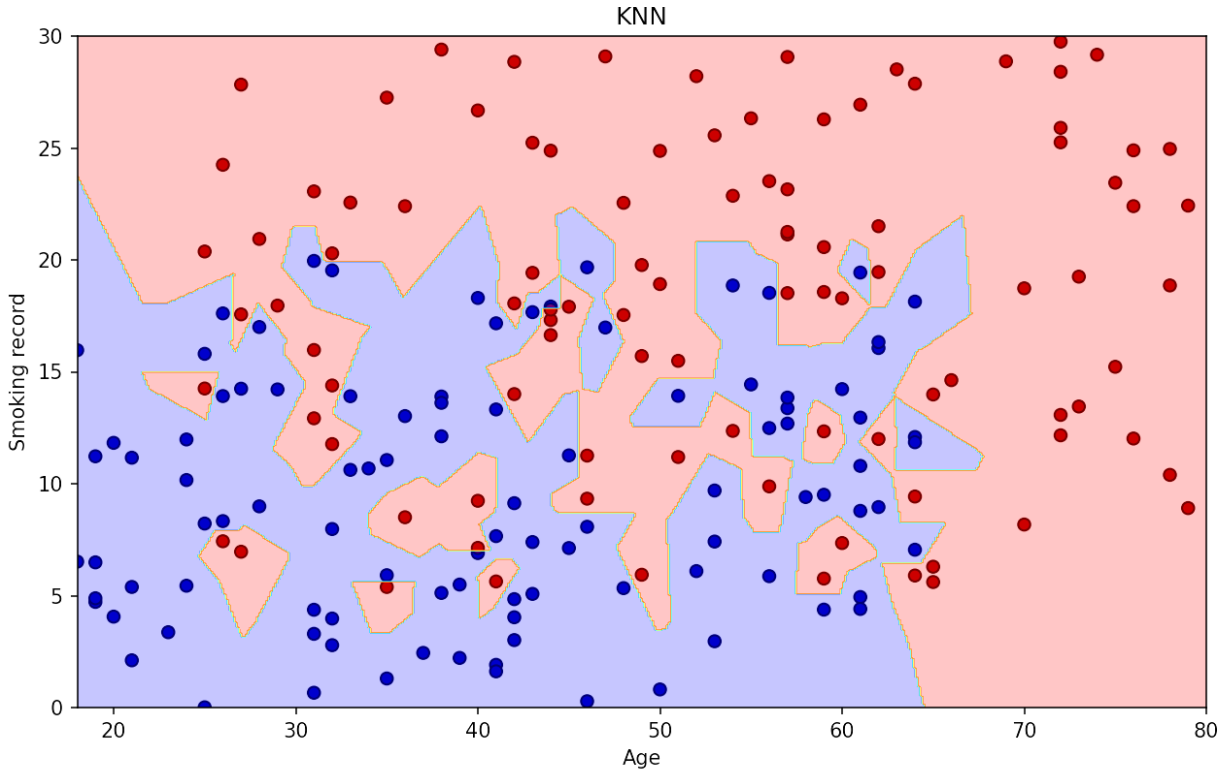


Note: In the case of regression tasks, the **average** of the k-nearest neighbors' target values can be returned.

KNN (k-nearest neighbors)

KNN, K=1

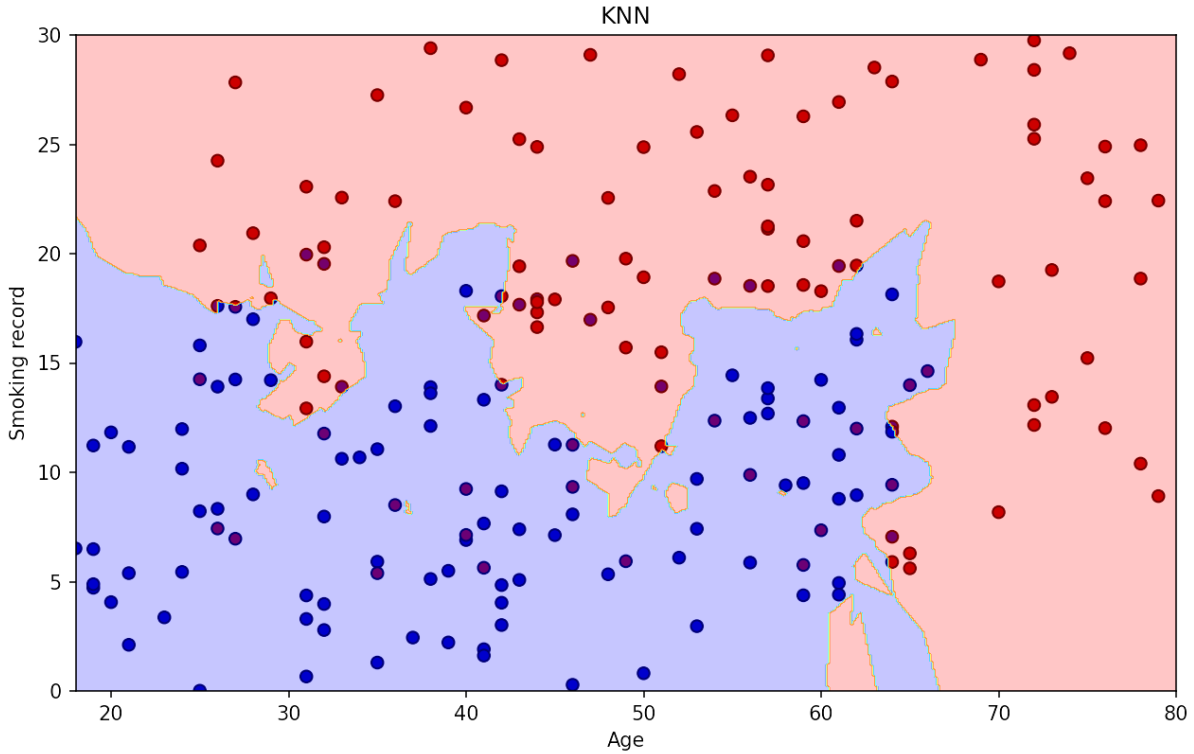
✓ Decision boundary for same example with K=1.



KNN (k-nearest neighbors)

KNN, K=7

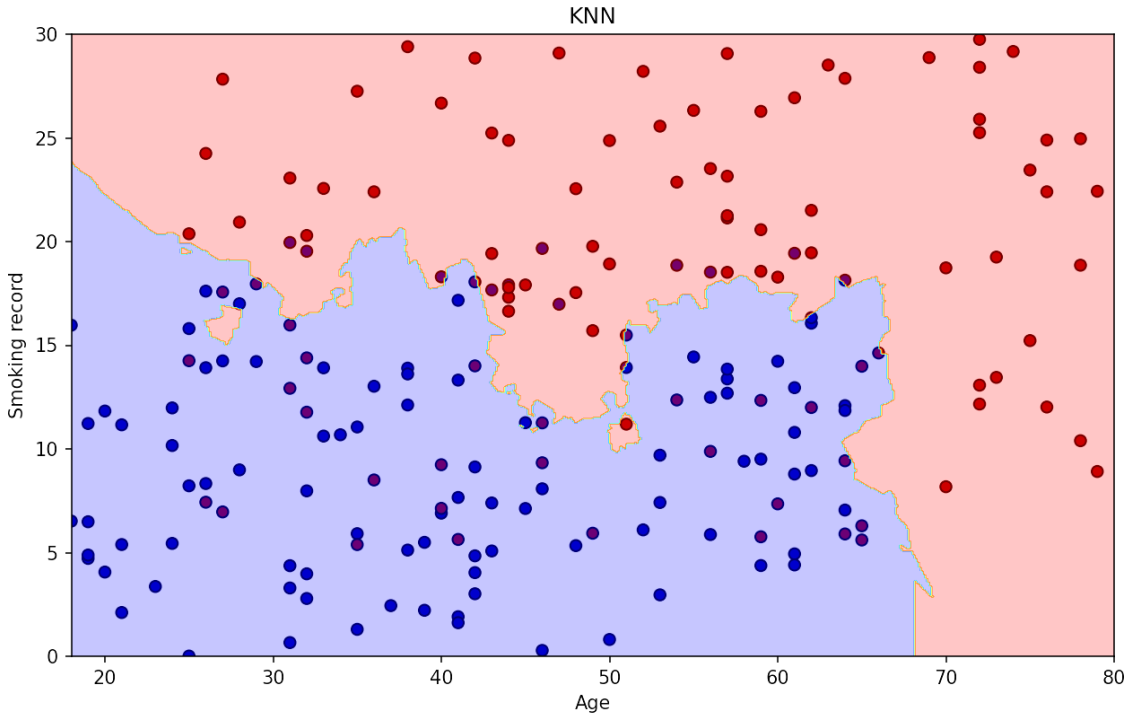
✓ If we change number of neighborhoods to 7.



KNN (k-nearest neighbors)

KNN, K=15

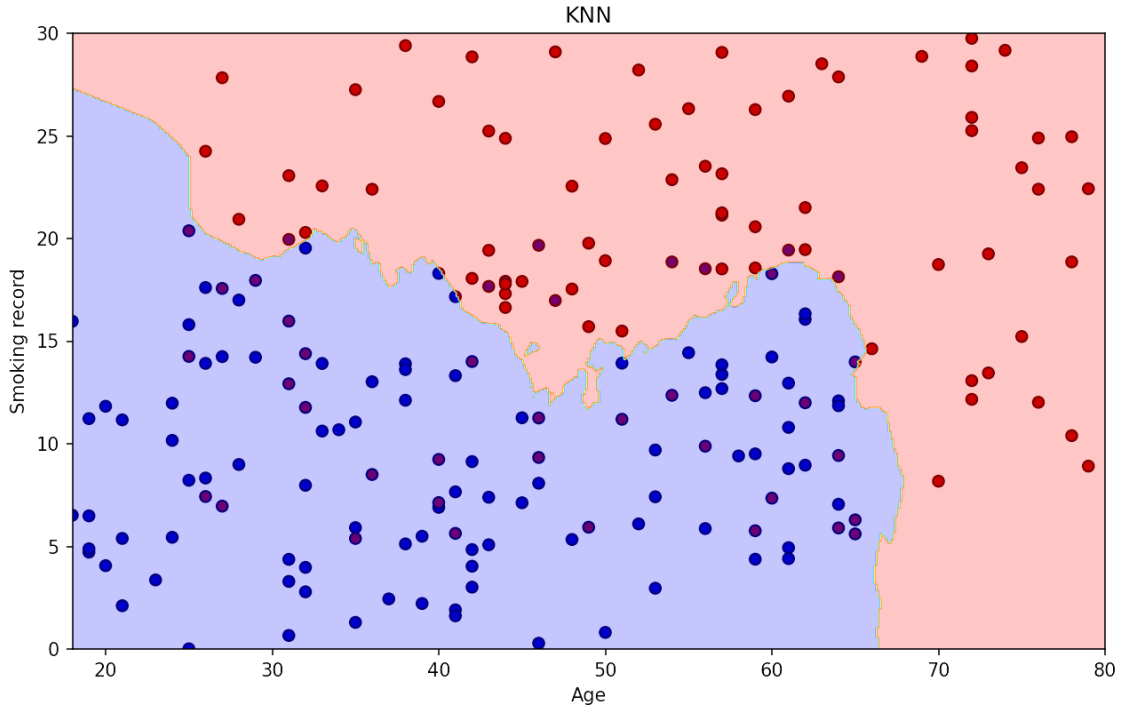
✓ If we change number of neighborhoods to 15.



KNN (k-nearest neighbors)

KNN, K=30

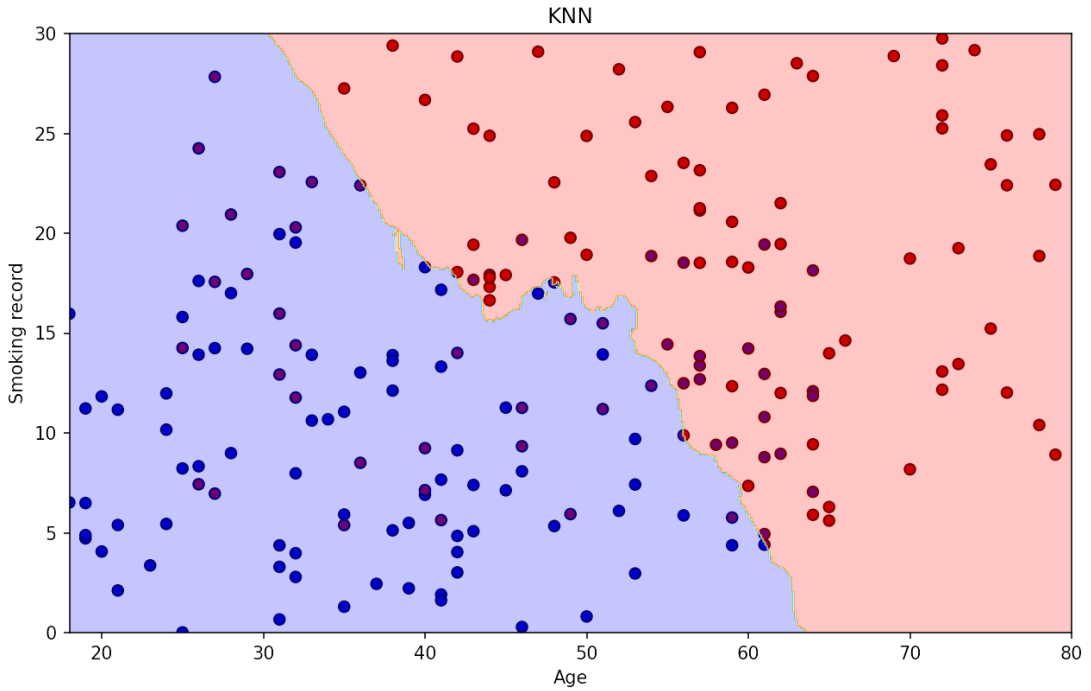
✓ If we change number of neighborhoods to 30.



KNN (k-nearest neighbors)

KNN, K=100

✓ Data for smoking, age and hurt attack records (supervised).



KNN (k-nearest neighbors)

How to find Optimal K?

- ✓ Cross-validation approaches can be used to evaluate the performance of KNN algorithm (e.g. k-fold cross-validation).
- ✓ We can use Euclidean distance in cross-validation for KNN.
- ✓ To **find the optimal value of the K** in KNN we can find the average accuracy across all k iterations then chose the best.

If we use library it is very easy to implement KNN: (in ML class we want learn its implementation)

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=1, metric='euclidean')
classifier.fit(X, y)
```

Evaluate the performance of a classification

Evaluation approaches for classification

- ✓ Confusion Matrix
- ✓ Accuracy
- ✓ Precision
- ✓ Recall
- ✓ F1-Score
- ✓ Area Under the ROC Curve (AUC-ROC)

Evaluate the performance of a classification

Confusion Matrix

- ✓ The most common evaluation metric to evaluate the performance of **a classification algorithm**.
- ✓ A **table used** to evaluate the performance of a classification algorithm.
- ✓ We Summarizes the number of predictions made by the classifier (next slide)

Evaluate the performance of a classification

Confusion Matrix (disease example)

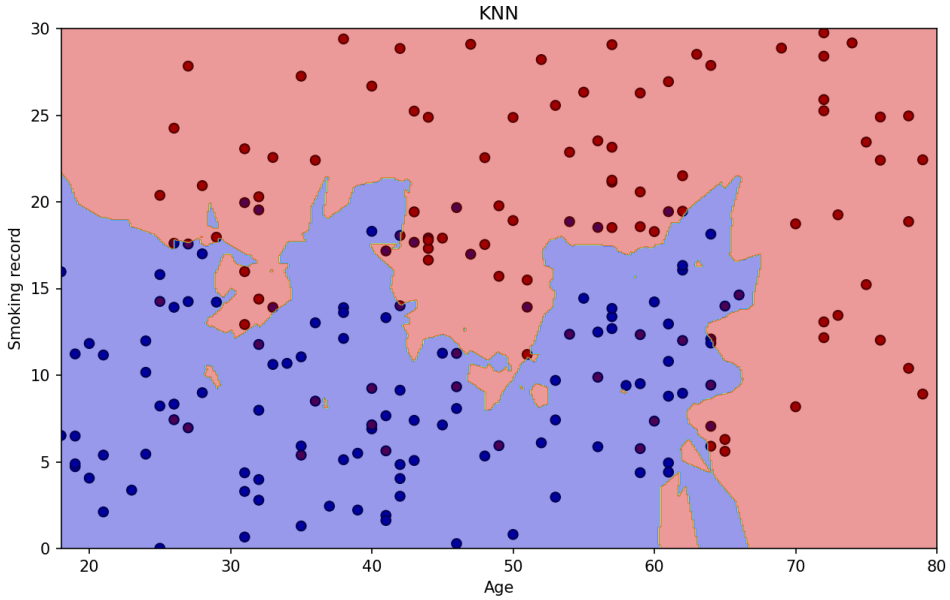
- **True Positive (TP):** The model correctly predicts the positive class.
  - If the model predicts that a patient has a certain disease and the patient actually has the disease.
- **False Positive (FP):** The model incorrectly predicts the positive class.
  - If the model predicts that a patient has a certain disease but the patient actually does not have the disease
- **True Negative (TN):** The model correctly predicts the negative class.
  - If the model predicts that a patient does not have a certain disease and the patient actually does not have the disease.
- **False Negative (FN):** The model incorrectly predicts the negative class.
  - If the model predicts that a patient does not have a certain disease but the patient actually has the disease.

Evaluate the performance of a classification

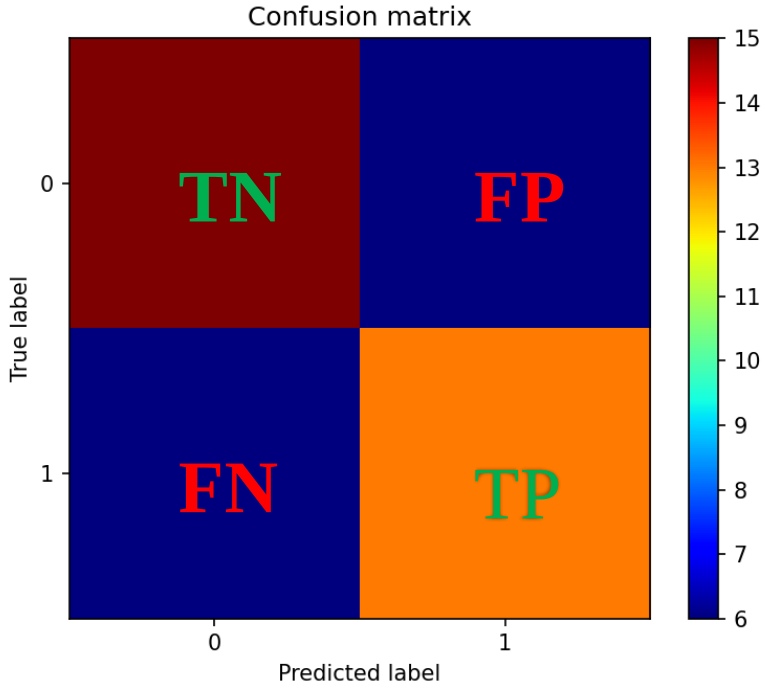
Confusion Matrix

✓ Example:

K = 7



K = 7



Note: this confusion matrix is performed on all the same training sets, not only the test data sample.

Evaluate the performance of a classification

Evaluate Classification

Accuracy

- ✓ It measures the overall proportion of correct predictions.

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

Precision:

- ✓ Measures the proportion of positive predictions that are correct.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Evaluate the performance of a classification

Evaluate Classification

Recall (sensitivity or true positive rate):

✓ It measures the proportion of actual positives that are correctly predicted.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

F1-Score:

✓ It combines precision and recall into a single metric.

$$\textbf{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Evaluate the performance of a classification

Evaluate Classification

Area Under the ROC Curve (AUC-ROC)

- ✓ The Receiver Operating Characteristic (ROC) we plot a **curve as** True Positive Rate (TPR) against the False Positive Rate (FPR) for different threshold values.
- ✓ The AUC-ROC value ranges between 0 and 1, where a value of 1 indicates perfect performance and a value of 0.5 indicates random performance.
- ✓ If the AUC-ROC is less than 0.5, the classifier is performing worse than a random classifier! (we made the classification task harder).

$$TPR = \frac{TP}{TP + FN}, \qquad FPR = \frac{FP}{FP + TN}$$

same as Precision

Ratio of true positives to the total number of positives

Ratio of false positives to the total number of negatives

Evaluate the performance of a classification

Area Under the ROC Curve (AUC-ROC)

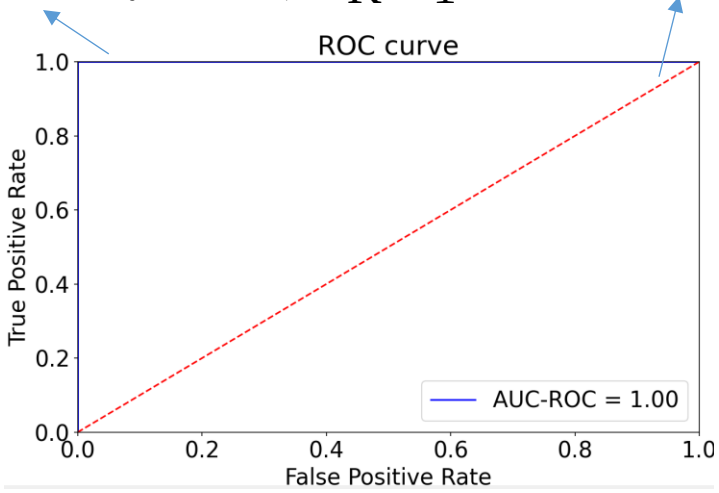
✓ We iterate threshold between 0 to1 to check if the predicted probability is high enough to classify a sample as positive class (greater than or equal to e.g.: 0.1, 0.3, 0.7, 0.9 at each step).

ROC curve (blue)

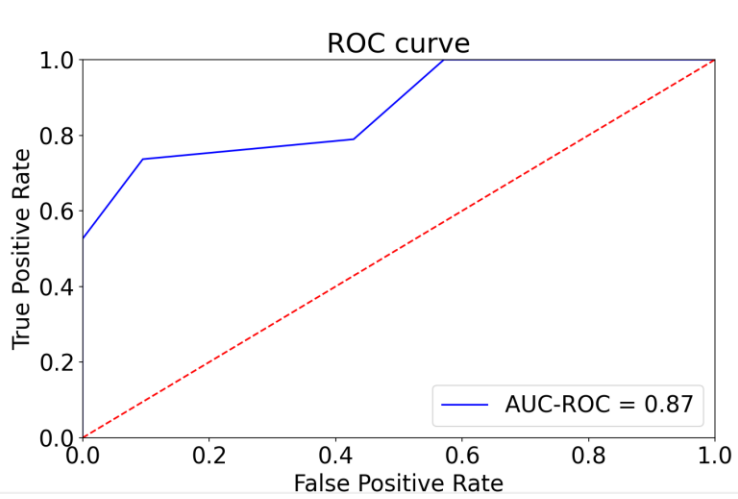
Reference line showing the performance of a random classifier

(A sample for TPR only in this slide)

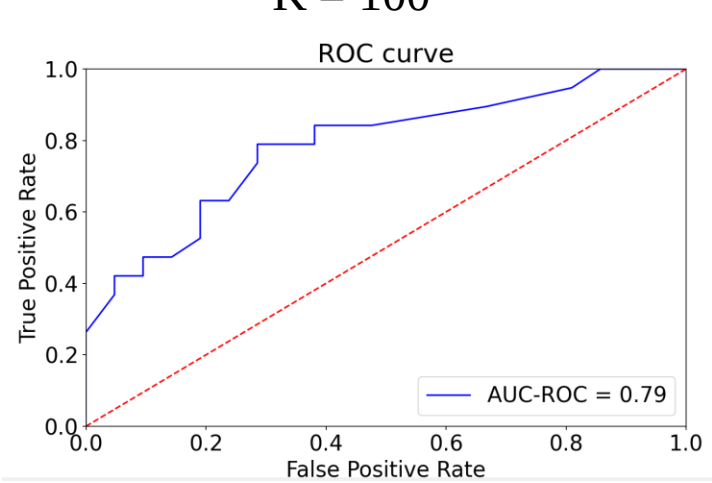
K = 1



K = 7



K = 100



ROC curve and the AUC-ROC score: a higher AUC-ROC score and a curve that is closer to the top-left corner of the plot indicate better classifier performance.

Usually 200 iterations for threshold.

Evaluate the performance of a classification

Evaluate Classification

$$TPR = \frac{TP}{TP + FN}, \quad FPR = \frac{FP}{FP + TN}$$

Area Under the ROC Curve (AUC-ROC)

AUC-ROC Score: now we can use all those thresholds together to calculate AUC-ROC score:

$$\begin{aligned}\textbf{AUC-ROC} = (&FPR[0] \times TPR[1] - FPR[1] \times TPR[0] + \\ &FPR[1] \times TPR[2] - FPR[2] \times TPR[1] + \\ &\ldots + \\ &FPR[n-2] \times TPR[n-1] - FPR[n-1] \times TPR[n-2] + \\ &FPR[n-1] \times TPR[n]) / 2\end{aligned}$$

Known as Trapezoidal rule:

- ✓ A numerical integration method used to approximate the definite integral of a function.

We can use library:

```
from sklearn.metrics import roc_auc_score, roc_curve
```

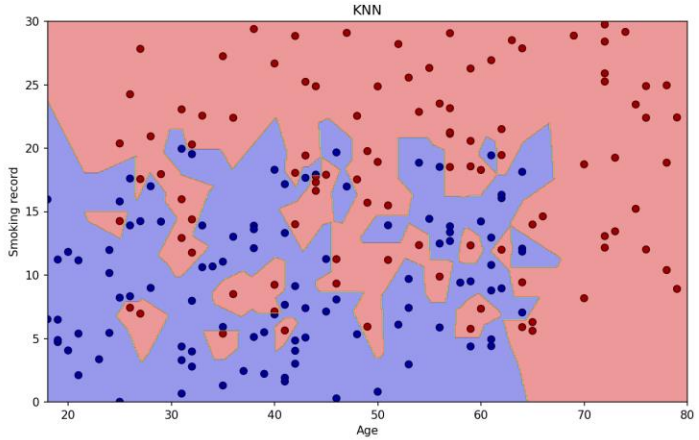
AUC-ROC Score: Higher value is better and it tells you **how well your model separates the positive and negative classes**

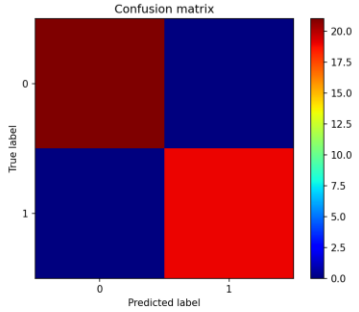
Evaluate the performance of a classification

Evaluate Classification

Important: We need to perform this for test data.

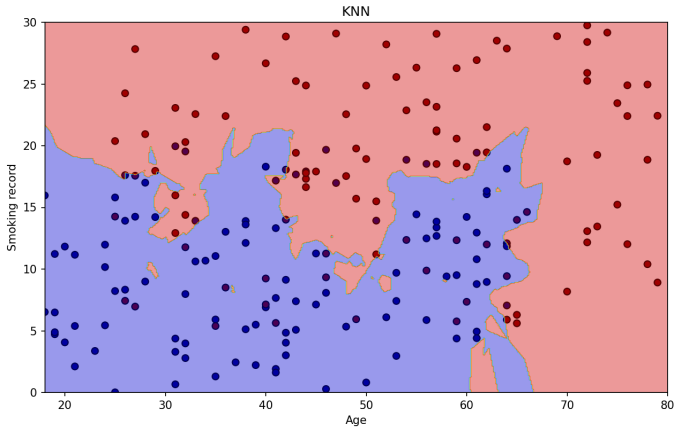
K = 1

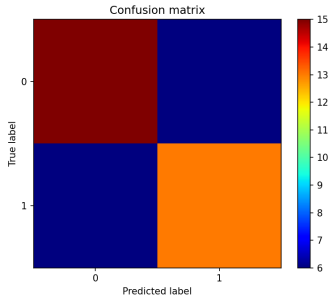




Accuracy: 1.00
Precision: 1.00
Recall: 1.00
F1 score: 1.00
AUC-ROC score: 1.00

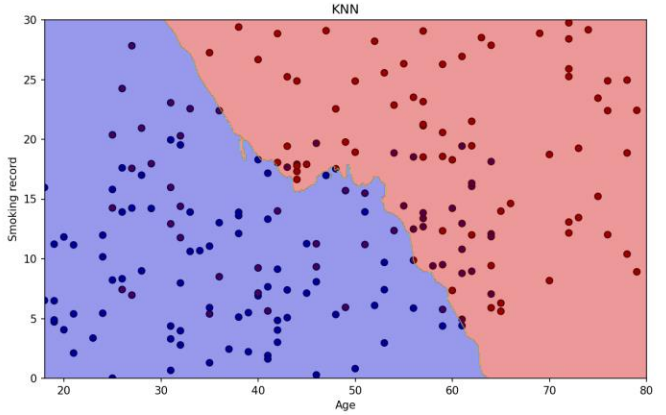
K = 7

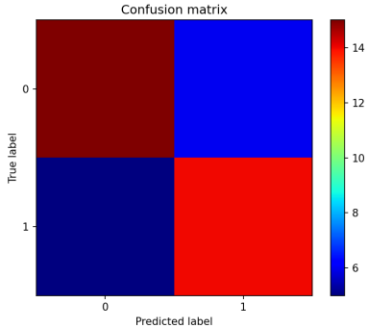




Accuracy: 0.82
Precision: 0.88
Recall: 0.74
F1 score: 0.80
AUC-ROC score: 0.87

K = 100





Accuracy: 0.72
Precision: 0.70
Recall: 0.74
F1 score: 0.72
AUC-ROC score: 0.79

Summery

- ✓ Bayes' theorem
- ✓ Bayes optimal classifier
- ✓ We understood Logistic Regression and how to train it with GD.
- ✓ We understood Maximum Likelihood Estimation (MLE), which was based on the negative log-likelihood function Known as cross-entropy.
- ✓ We learned the KNN and saw how to evaluate the performance of a classification by:
  - ➢ Confusion Matrix
  - ➢ Accuracy
  - ➢ Precision
  - ➢ Recall
  - ➢ F1-Score
  - ➢ AUC-ROC

Practice and Assignment

Implement KNN and evaluate metrics: A) Choose your desired problem and use Kaggle or Create simple data with 10-100 samples of desired problem, B) evaluate all the performances by metrics we discussed in the class.
Note: do not use library