



UNIVERSITAS
INDONESIA
Veritas, Probatum, Justitia

KECERDASAN BUATAN

5 | CLASSIFICATION

Dr. Prima Dewi Purnamasari
Program Studi Teknik Komputer FTUI

Machine Learning - Classification

In Machine Learning, classification is a **supervised** learning approach, which can be thought of as a means of **categorizing** or "classifying" some unknown items into a discrete set of "classes."

Classification attempts to learn the relationship between a set of **feature** variables and a **target** variable of interest.

The target attribute in classification is a **categorical variable with discrete values**.

How does classification work?

Classification determines the class label for an unlabeled test case.

| age | ed | employ | address | income | debtinc | creddebt | othdebt | default |
|-----|----|--------|---------|--------|---------|----------|---------|---------|
| 41 | 3 | 17 | 12 | 176 | 9.3 | 11.359 | 5.009 | 1 |
| 27 | 1 | 10 | 6 | 31 | 17.3 | 1.362 | 4.001 | 0 |
| 40 | 1 | 15 | 14 | 55 | 5.5 | 0.856 | 2.169 | 0 |
| 41 | 1 | 15 | 14 | 120 | 2.9 | 2.659 | 0.821 | 0 |
| 24 | 2 | 2 | 0 | 28 | 17.3 | 1.787 | 3.057 | 1 |
| 41 | 2 | 5 | 5 | 25 | 10.2 | 0.393 | 2.157 | 0 |
| 39 | 1 | 20 | 9 | 67 | 30.6 | 3.834 | 16.668 | 0 |
| 43 | 1 | 12 | 11 | 38 | 3.6 | 0.129 | 1.239 | 0 |
| 24 | 1 | 3 | 4 | 19 | 24.4 | 1.358 | 3.278 | 1 |
| 36 | 1 | 0 | 13 | 25 | 19.7 | 2.778 | 2.147 | 0 |

} Categorical Variable

| age | ed | employ | address | income | debtinc | creddebt | othdebt | default |
|-----|----|--------|---------|--------|---------|----------|---------|---------|
| 37 | 2 | 16 | 10 | 130 | 9.3 | 10.23 | 3.21 | |

| age | ed | employ | address | Income | debtinc | creddebt | othdebt | default |
|-----|----|--------|---------|--------|---------|----------|---------|---------|
| 41 | 3 | 17 | 12 | 176 | 9.3 | 11.359 | 5.009 | 1 |
| 27 | 1 | 10 | 6 | 31 | 17.3 | 1.362 | 4.001 | 0 |
| 40 | 1 | 15 | 14 | 55 | 5.5 | 0.856 | 2.169 | 0 |
| 41 | 1 | 15 | 14 | 120 | 2.9 | 2.659 | 0.821 | 0 |
| 24 | 2 | 2 | 0 | 28 | 17.3 | 1.787 | 3.057 | 1 |
| 41 | 2 | 5 | 5 | 25 | 10.2 | 0.393 | 2.157 | 0 |
| 39 | 1 | 20 | 9 | 67 | 30.6 | 3.834 | 16.668 | 0 |
| 43 | 1 | 12 | 11 | 38 | 3.6 | 0.129 | 1.239 | 0 |
| 24 | 1 | 3 | 4 | 19 | 24.4 | 1.358 | 3.278 | 1 |
| 36 | | | | | 15.7 | 2.778 | 2.147 | 0 |

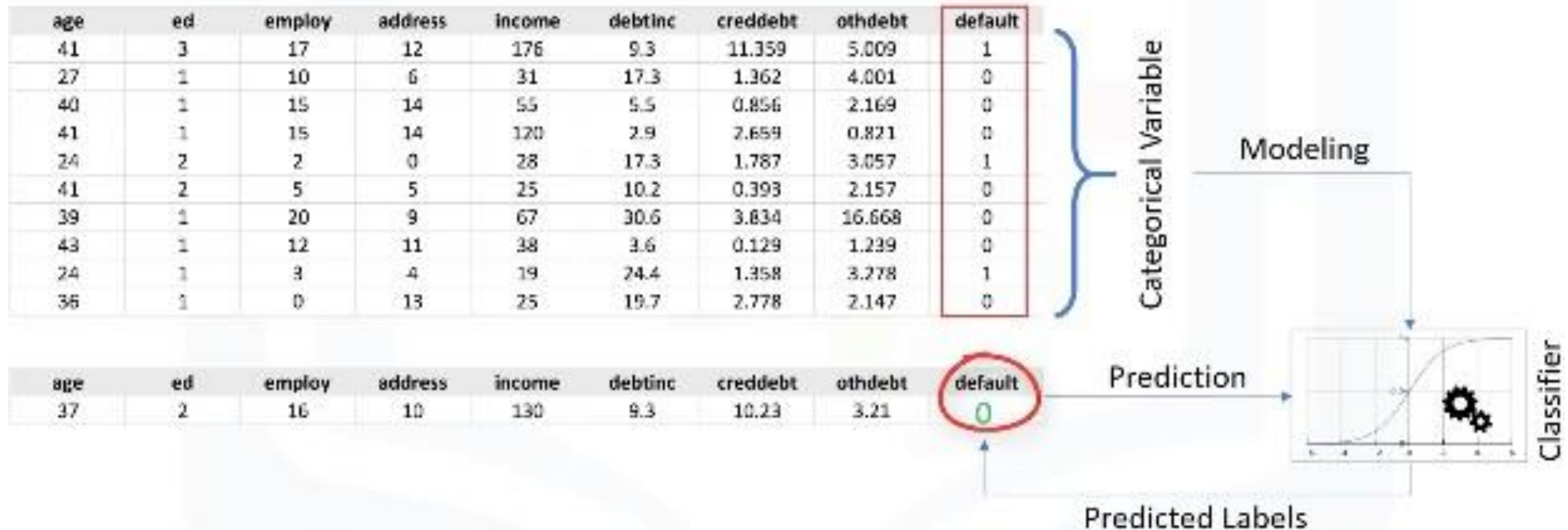
Machine Learning - Classification

- Let's explain this with an example. A good sample of classification is the loan default prediction. Suppose a bank is concerned about the potential for loans not to be repaid.
- If previous loan default data can be used to predict which customers are likely to have problems repaying loans, these "bad risk" customers can either have their loan application declined or offered alternative products.

Machine Learning - Classification

The goal of a loan default predictor is to use existing loan default data, which is information about the customers (such as age, income, education, etc.),

To build a classifier, pass a new customer or potential future defaulter to the model, and then label it (i.e. the data points) as "Defaulter" or "Not Defaulter", or for example, 0 or 1.



Machine Learning

- Classification

This is how a classifier predicts an unlabeled test case.

Please notice that this specific example was about a binary classifier with two values.

We can also build classifier models for both binary classification and multi-class classification.

For example, imagine that you collected data about a set of patients, all of whom suffered from the same illness.

| Age | Sex | BP | Cholesterol | Na | K | Drug |
|-----|-----|--------|-------------|-------|-------|-------|
| 23 | F | HIGH | HIGH | 0.793 | 0.031 | drugY |
| 47 | M | LOW | HIGH | 0.739 | 0.056 | drugC |
| 47 | M | LOW | HIGH | 0.697 | 0.069 | drugC |
| 28 | F | NORMAL | HIGH | 0.564 | 0.072 | drugX |
| 61 | F | LOW | HIGH | 0.559 | 0.031 | drugY |
| 22 | F | NORMAL | HIGH | 0.677 | 0.079 | drugX |
| 49 | F | NORMAL | HIGH | 0.79 | 0.049 | drugY |
| 41 | M | LOW | HIGH | 0.767 | 0.069 | drugC |
| 60 | M | NORMAL | HIGH | 0.777 | 0.051 | drugY |
| 43 | M | LOW | NORMAL | 0.526 | 0.027 | drugY |

Machine Learning - Classification

During their course of treatment, each patient responded to one of three medications.

You can use this labeled dataset, with a classification algorithm, to build a classification model.

Then you can use it to find out which drug might be appropriate for a future patient with the same illness.



Classification Use Case

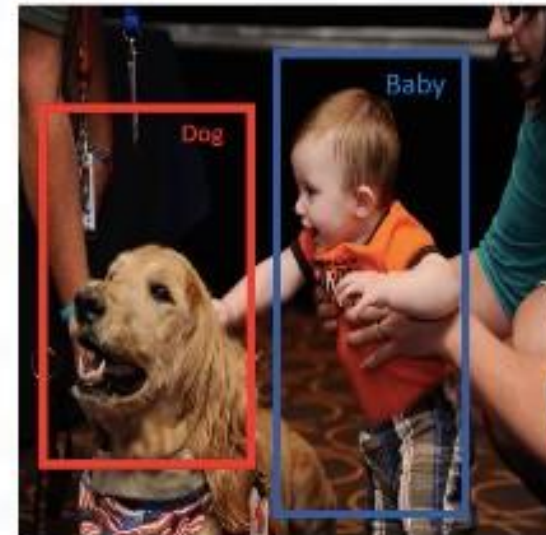
As you can see, it is a sample of multi-class classification. Classification has different business use cases as well, e.g.:

- To predict the category to which a customer belongs;
- For Churn detection, where we predict whether a customer switches to another provider or brand;
- Or to predict whether or not a customer responds to a particular advertising campaign.

| | tenure | age | address | income | ed | employ | equip | callcard | wireless | churn |
|---|--------|------|---------|--------|-----|--------|-------|----------|----------|-------|
| 0 | 11.0 | 33.0 | 7.0 | 136.0 | 5.0 | 5.0 | 0.0 | 1.0 | 1.0 | Yes |
| 1 | 33.0 | 33.0 | 12.0 | 33.0 | 2.0 | 0.0 | 0.0 | 0.0 | 0.0 | Yes |
| 2 | 23.0 | 30.0 | 9.0 | 30.0 | 1.0 | 2.0 | 0.0 | 0.0 | 0.0 | No |
| 3 | 38.0 | 35.0 | 5.0 | 76.0 | 2.0 | 10.0 | 1.0 | 1.0 | 1.0 | No |
| 4 | 7.0 | 35.0 | 14.0 | 80.0 | 2.0 | 15.0 | 0.0 | 1.0 | 0.0 | ? |

Classification Use Case

- Data classification has several applications in a wide variety of industries.
- Essentially, many problems can be expressed as associations between feature and target variables, especially when labeled data is available.
- This provides a broad range of applicability for classification.
- For example, classification can be used for email filtering, speech recognition, handwriting recognition, bio-metric identification, document classification, and much more.



Popular Classification Algorithms

Decision Trees

Naïve Bayes

K-nearest
neighbor

Logistic
regression

Support Vector
Machines

Neural
Networks

K-Nearest Neighbor (KNN)

K-Nearest Neighbors

Imagine that a telecommunications provider has segmented its customer base by service usage patterns, categorizing the customers into four groups.

| X: Independent variable | | | | | | | | | | | Y: Dependent variable |
|-------------------------|--------|-----|---------|---------|--------|----|--------|--------|--------|--------|-----------------------|
| | region | age | marital | address | income | ed | employ | retire | gender | reside | custcat |
| 0 | 2 | 44 | 1 | 8 | 64 | 4 | 5 | 0 | 0 | 2 | 1 |
| 1 | 3 | 33 | 1 | 7 | 135 | 5 | 5 | 0 | 0 | 6 | 4 |
| 2 | 3 | 52 | 1 | 24 | 118 | 1 | 29 | 0 | 1 | 2 | 3 |
| 3 | 2 | 33 | 0 | 12 | 33 | 2 | 0 | 0 | 1 | 1 | 1 |
| 4 | 2 | 30 | 1 | 8 | 30 | 1 | 2 | 0 | 0 | 4 | 3 |
| 5 | 2 | 38 | 0 | 17 | 78 | 2 | 18 | 0 | 1 | 1 | 3 |
| 6 | 3 | 22 | 1 | 2 | 19 | 2 | 4 | 0 | 1 | 5 | 2 |
| 7 | 2 | 35 | 0 | 5 | 78 | 2 | 10 | 0 | 0 | 3 | 4 |
| 8 | 3 | 50 | 1 | 7 | 168 | 4 | 31 | 0 | 0 | 5 | ? |

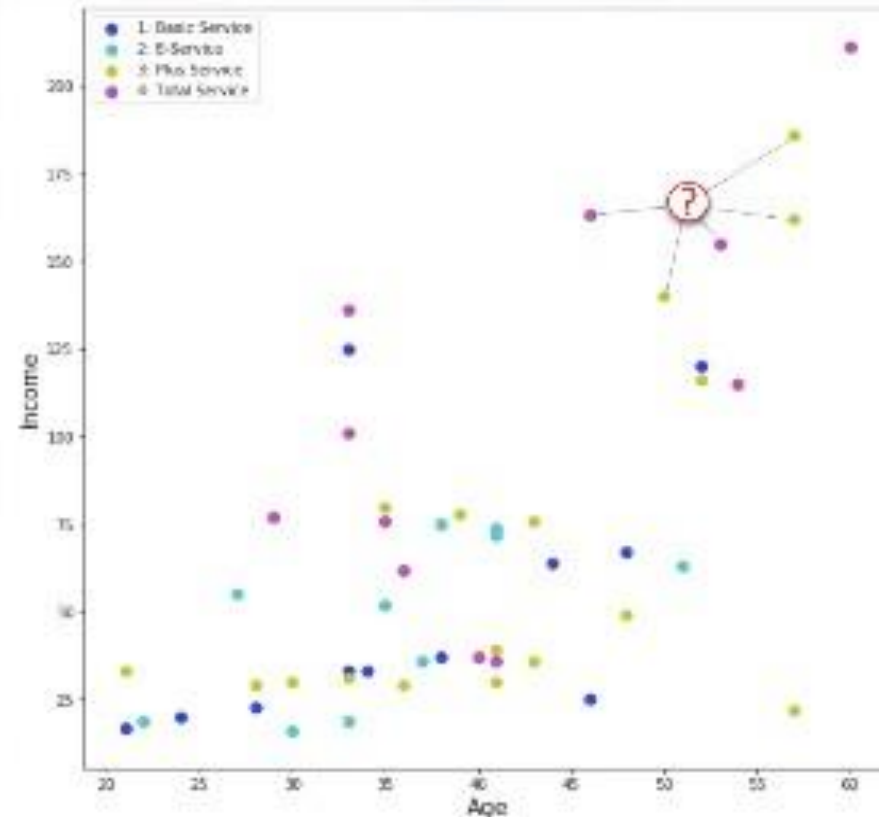
| Value | Label |
|-------|---------------|
| 1 | Basic Service |
| 2 | E-Service |
| 3 | Plus Service |
| 4 | Total Service |

Our objective is to build a classifier, for example using the rows 0 to 7, to predict the class of row 8. We will use a specific type of classification called K-nearest neighbor.



K-Nearest Neighbors

- A method for **classifying** cases based on their similarity to other cases
- Cases that are near each other are said to be “**neighbors**”
- Based on **similar cases with same class labels** are near each other



Let's use only two fields as predictors - specifically, Age and Income, and then plot the customers based on their group membership.

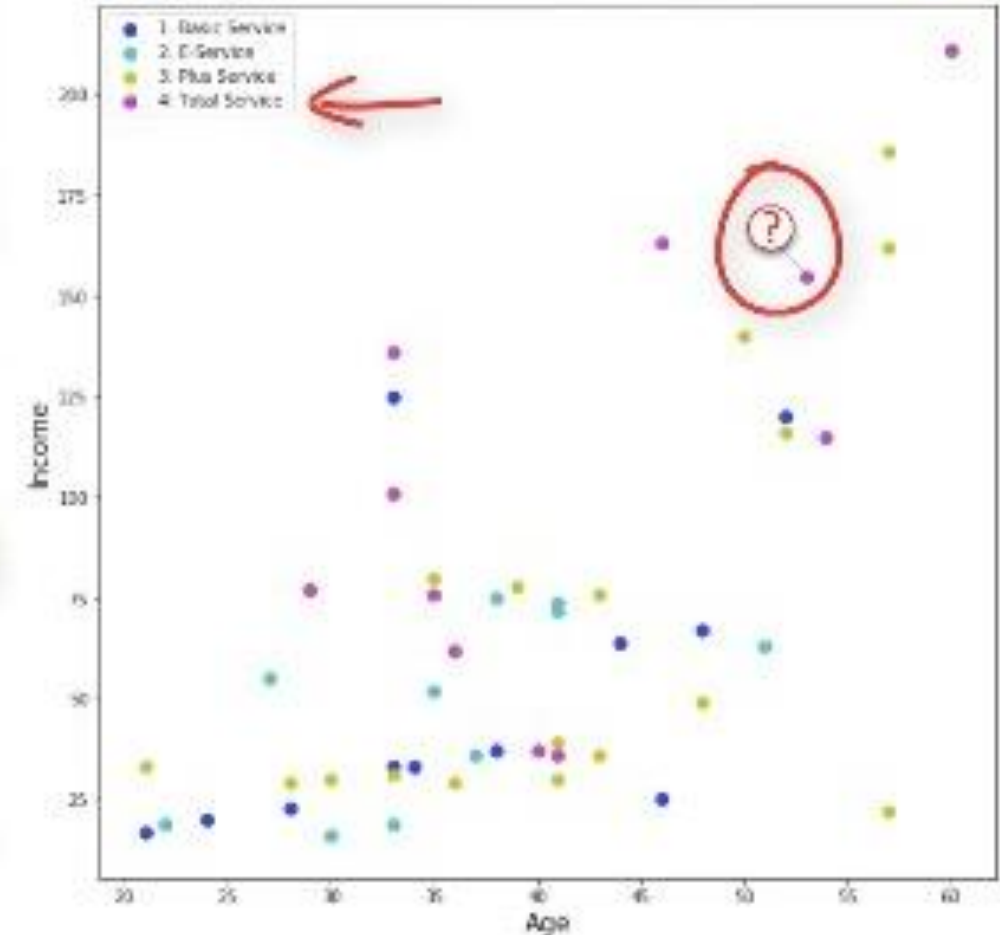
| | region | age | marital | address | income | ed | employ | retire | gender | reside | custcat |
|---|--------|-----|---------|---------|--------|----|--------|--------|--------|--------|---------|
| 0 | 2 | 44 | 1 | 8 | 64 | 4 | 5 | 0 | 0 | 2 | 1 |
| 1 | 3 | 33 | 1 | 7 | 135 | 5 | 5 | 0 | 0 | 6 | 4 |
| 2 | 3 | 52 | 1 | 24 | 118 | 1 | 29 | 0 | 1 | 2 | 3 |
| 3 | 2 | 32 | 0 | 12 | 33 | 2 | 0 | 0 | 1 | 1 | 1 |
| 4 | 2 | 30 | 1 | 8 | 30 | 1 | 2 | 0 | 0 | 4 | 3 |
| 5 | 2 | 38 | 0 | 17 | 78 | 2 | 18 | 0 | 1 | 1 | 3 |
| 6 | 3 | 22 | 1 | 2 | 19 | 2 | 4 | 0 | 1 | 6 | 2 |
| 7 | 2 | 35 | 0 | 5 | 78 | 2 | 10 | 0 | 0 | 3 | 4 |
| 8 | 3 | 50 | 1 | 2 | 168 | 4 | 31 | 0 | 0 | 5 | ? |

Now, let's say that we have a new customer, for example, record number 8 with a known age and income. How can we find the class of this customer?

Can we find one of the closest cases and assign the same class label to our new customer?

Can we also say that the class of our new customer is most probably group 4 (i.e. total service) because its nearest neighbor is also of class 4?

| | region | age | marital | address | income | ed | employ | retire | gender | reside | custcat |
|---|--------|-----|---------|---------|--------|----|--------|--------|--------|--------|---------|
| 0 | 2 | 44 | 1 | 8 | 64 | 4 | 5 | 0 | 0 | 2 | 1 |
| 1 | 3 | 33 | 1 | 7 | 136 | 5 | 5 | 0 | 0 | 6 | 4 |
| 2 | 3 | 52 | 1 | 24 | 118 | 1 | 29 | 0 | 1 | 2 | 3 |
| 3 | 2 | 33 | 0 | 12 | 33 | 2 | 0 | 0 | 1 | 1 | 1 |
| 4 | 2 | 30 | 1 | 8 | 30 | 1 | 2 | 0 | 0 | 4 | 3 |
| 5 | 2 | 38 | 0 | 17 | 78 | 2 | 18 | 0 | 1 | 1 | 3 |
| 6 | 3 | 22 | 1 | 2 | 19 | 2 | 4 | 0 | 1 | 6 | 2 |
| 7 | 2 | 35 | 0 | 5 | 78 | 2 | 10 | 0 | 0 | 3 | 4 |
| 8 | 3 | 50 | 1 | 7 | 168 | 4 | 31 | 0 | 0 | 5 | ? |

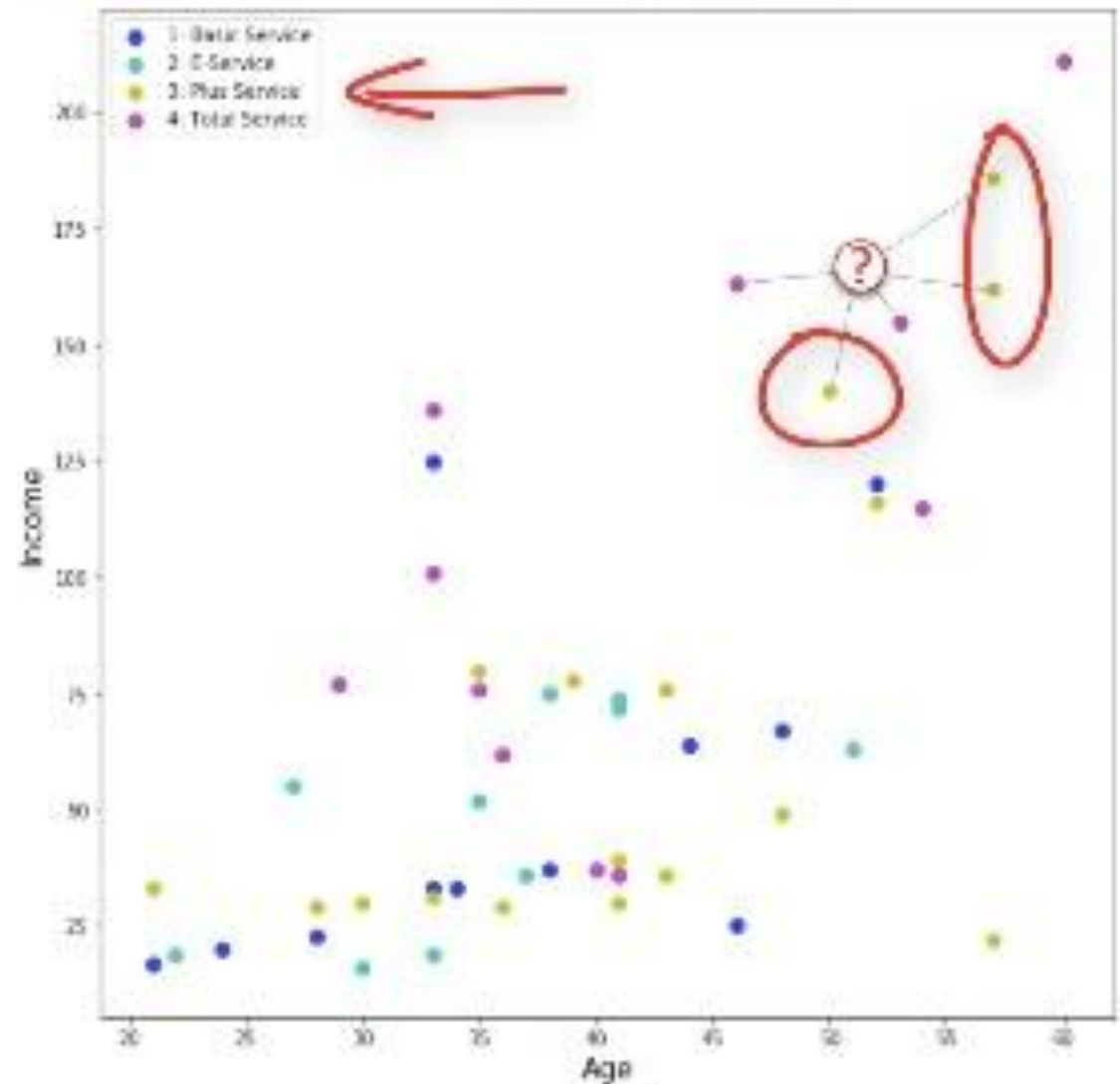


Yes, we can. In fact, it is the first-nearest neighbor.

It might be a poor judgment, especially if the first nearest neighbor is a very specific case, or an outlier.

What if we chose the five nearest neighbors, and did a majority vote among them to define the class of our new customer?

- 3 yellow and 2 magenta
- yellow wins → class = 3



KNN Algorithm

In a classification problem, the k-nearest neighbors algorithm works as follows:

1. Pick a value for K.
2. Calculate the distance from the new case (holdout from each of the cases in the dataset).
3. Search for the K observations in the training data that are 'nearest' to the measurements of the unknown data point.
4. predict the response of the unknown data point using the most popular response value from the K nearest neighbors.

Key questions

- How to select the correct K
- How to compute the similarity between cases



Customer 1

Age

54



Customer 2

Age

50

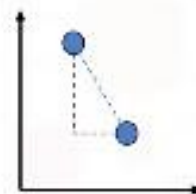
$$\text{Dis}(x_1, x_2) = \sqrt{\sum_{i=0}^n (x_{1i} - x_{2i})^2}$$

Calculating distance → e.g. Euclidean

- What about if we have more than one feature, for example Age and Income?
- If we have income and age for each customer, we can still use the same formula, but this time, we're using it in a 2-dimensional space.



| Customer 1 | |
|------------|--------|
| Age | Income |
| 54 | 190 |



| Customer 2 | |
|------------|--------|
| Age | Income |
| 50 | 200 |

$$\begin{aligned}\text{Dis}(x_1, x_2) &= \sqrt{\sum_{i=1}^n (x_{1i} - x_{2i})^2} \\ &= \sqrt{(54 - 50)^2 + (190 - 200)^2} = 10.77\end{aligned}$$

3 dimensional



Customer 1

| Age | Income | Education |
|-----|--------|-----------|
| 54 | 190 | 3 |



Customer 2

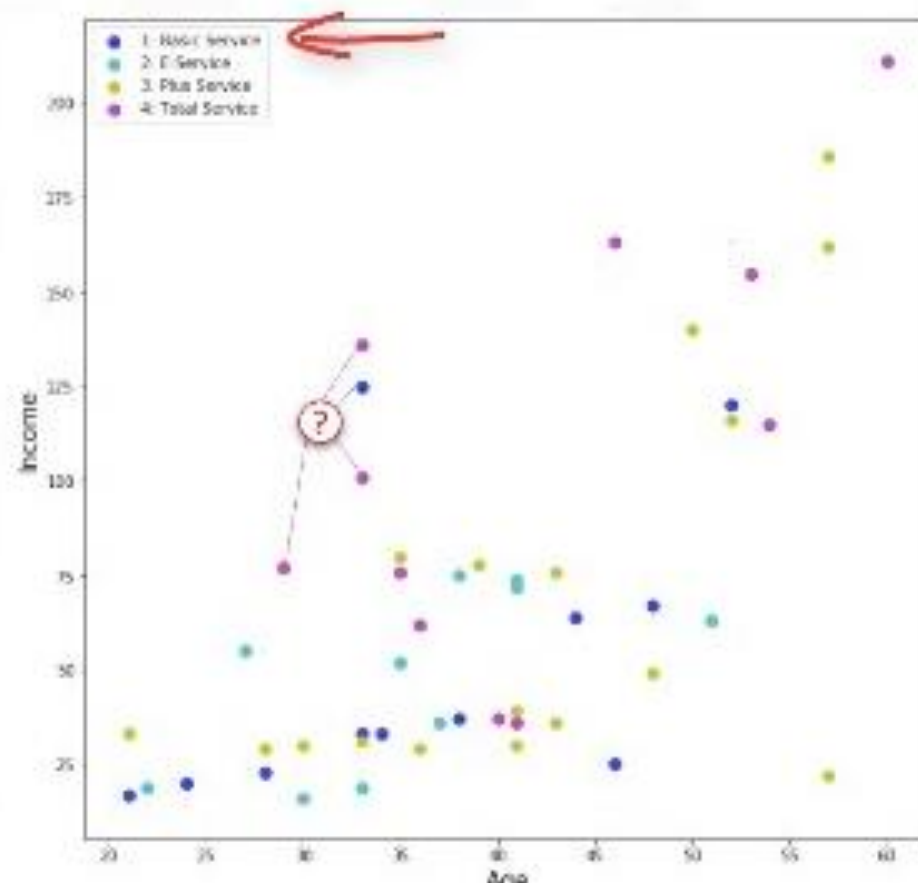
| Age | Income | Education |
|-----|--------|-----------|
| 50 | 200 | 8 |

$$\begin{aligned}\text{Dis}(x_1, x_2) &= \sqrt{\sum_{i=0}^n (x_{1i} - x_{2i})^2} \\ &= \sqrt{(54 - 50)^2 + (190 - 200)^2 + (3 - 8)^2} = 11.87\end{aligned}$$

Choose the right K

What happens if we choose a very low value of K, let's say, $k=1$? The first nearest point would be Blue, which is class 1.

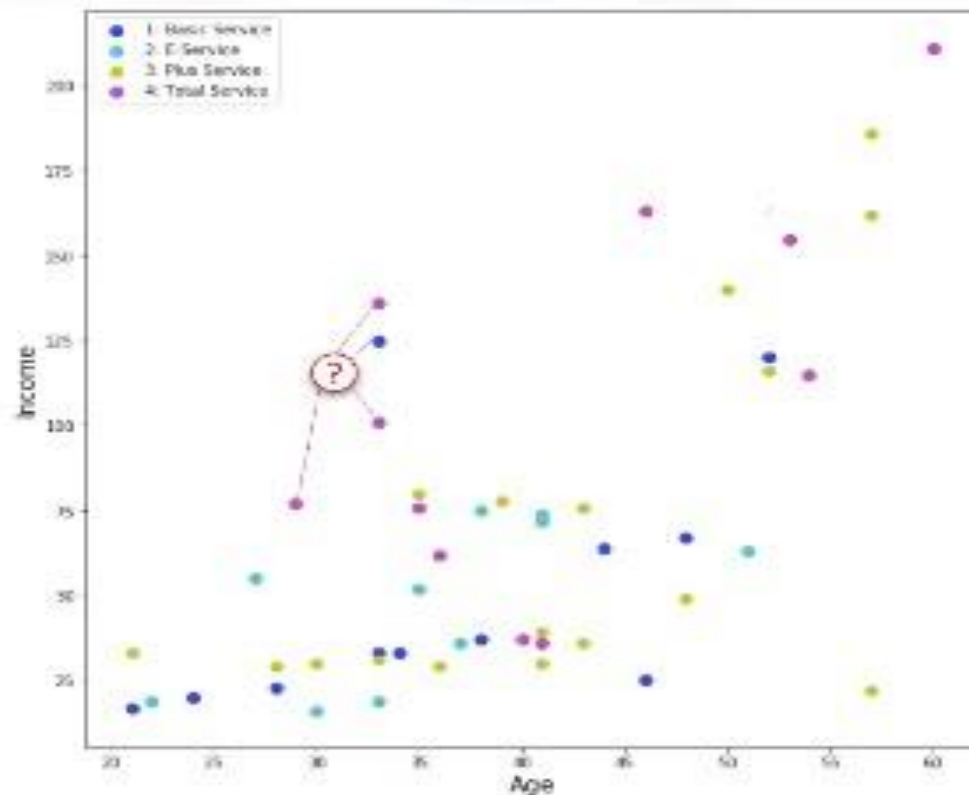
- This would be a bad prediction, since more of the points around it are Magenta, or class 4.
- It means the prediction process is **not generalized enough** to be used for out-of-sample cases.
- Out-of-sample data is data that is outside of the dataset used to train the model.
- In other words, it cannot be trusted to be used for prediction of unknown samples.
- It's important to remember that **over-fitting** is bad, as we want a general model that works for any data, not just the data used for training.



Now, on the opposite side of the spectrum, if we choose a very high value of K , such as $K=20$, then the model becomes **overly generalized** → **underfitting**

So, how we can find the best value for K ?

- $K = 1$ class 1
- $K = 20$?



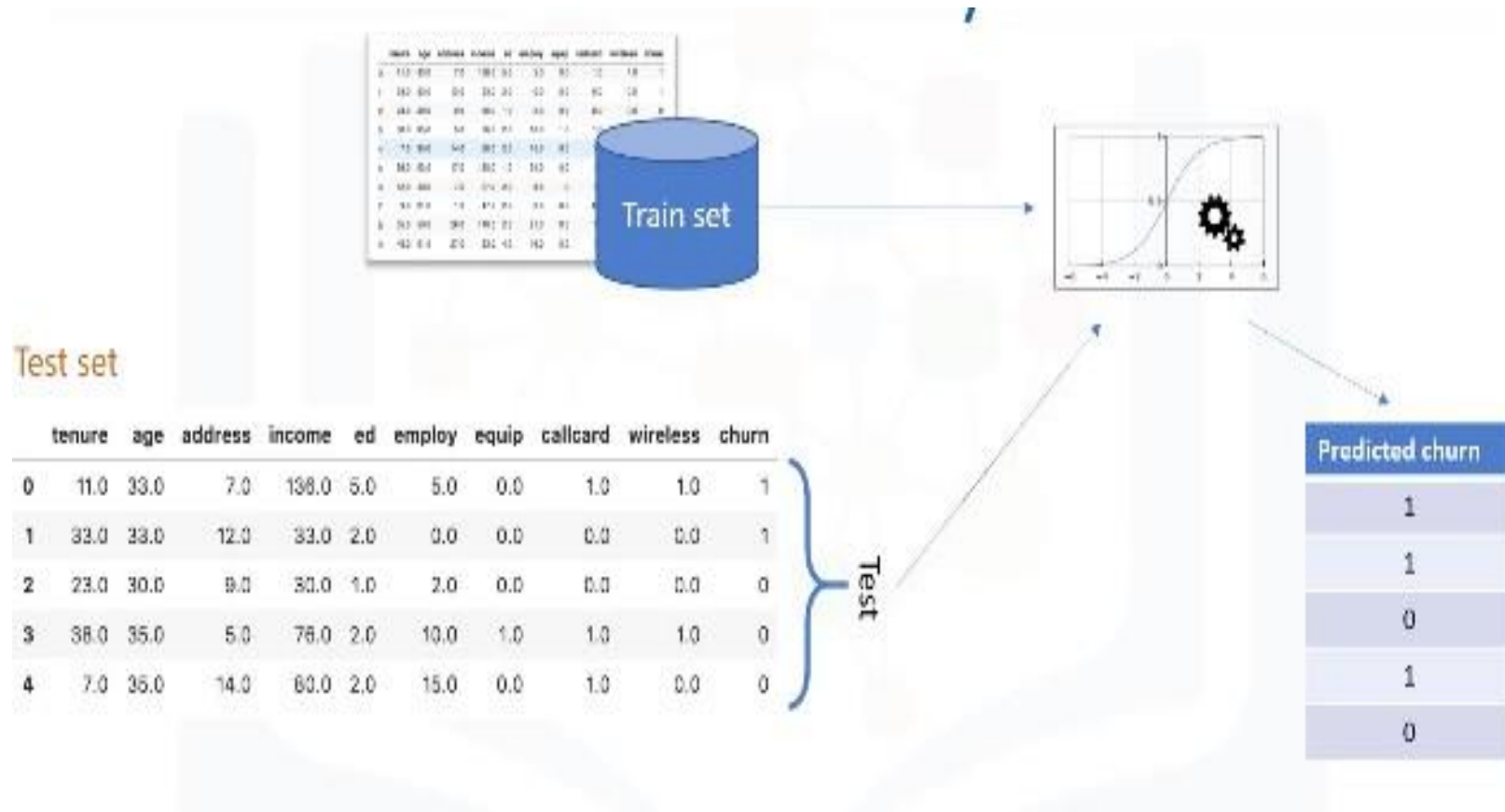
Solution

- The general solution is to reserve a part of your data for testing the accuracy of the model.
- Once you've done so, choose $k = 1$, and then use the training part for modeling, and calculate the accuracy of prediction using all samples in your test set.
- Repeat this process, increasing the k , and see which k is best for your model.

Model Evaluation

We have trained the model, and now we want to calculate its accuracy using the test set. We pass the test set to our model, and we find the predicted labels.

Now the question is, **“How accurate is this model?”**



Evaluation Metrics

- Evaluation metrics explain the performance of a model.
- Basically, we compare the actual values in the test set with the values predicted by the model, to calculate the accuracy of the model.
- Evaluation metrics provide a key role in the development of a model, as they provide insight to areas that might require improvement.
- There are different model evaluation metrics but we just talk about three of them here, specifically: Jaccard index, F1-score, and Log Loss.

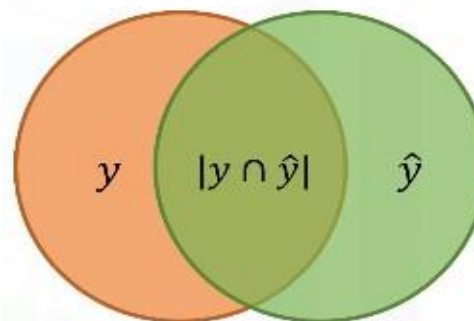
Jaccard Index

- Let's first look at one of the simplest accuracy measurements, the Jaccard index – also known as the Jaccard similarity coefficient.
- Let's say y shows the true labels of the churn dataset. And \hat{y} shows the predicted values by our classifier.

Jaccard index

y : Actual labels

\hat{y} : Predicted labels



Jaccard Index

- Jaccard = size of the intersection divided by the size of the union of two label sets.
- For example, for a test set of size 10, with 8 correct predictions, or 8 intersections, the accuracy by the Jaccard index would be 0.66.
- If the entire set of predicted labels for a sample strictly matches with the true set of labels, then the subset accuracy is 1.0; otherwise it is 0.0.

y : Actual labels

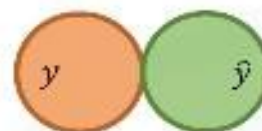
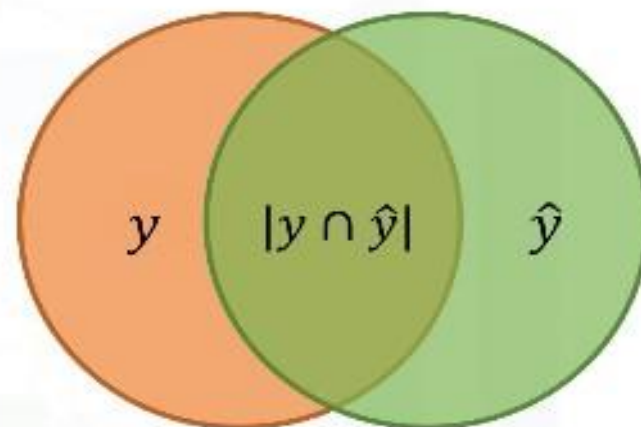
\hat{y} : Predicted labels

$$J(y, \hat{y}) = \frac{|y \cap \hat{y}|}{|y \cup \hat{y}|} = \frac{|y \cap \hat{y}|}{|y| + |\hat{y}| - |y \cap \hat{y}|}$$

y : [0, 0, 0, 0, 0, 1, 1, 1, 1, 1]

\hat{y} : [1, 1, 0, 0, 0, 1, 1, 1, 1, 1]

$$J(y, \hat{y}) = \frac{8}{10+10-8} = 0.66$$



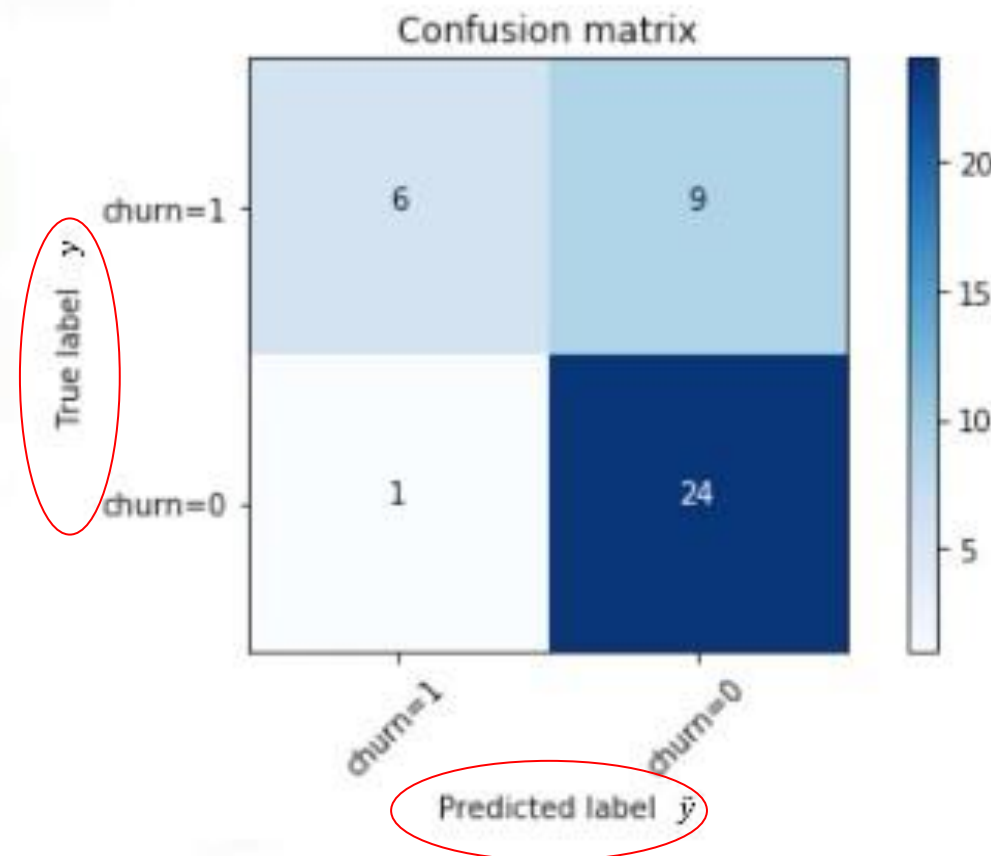
$$J(y, \hat{y}) = 0.0$$



$$J(y, \hat{y}) = 1.0$$

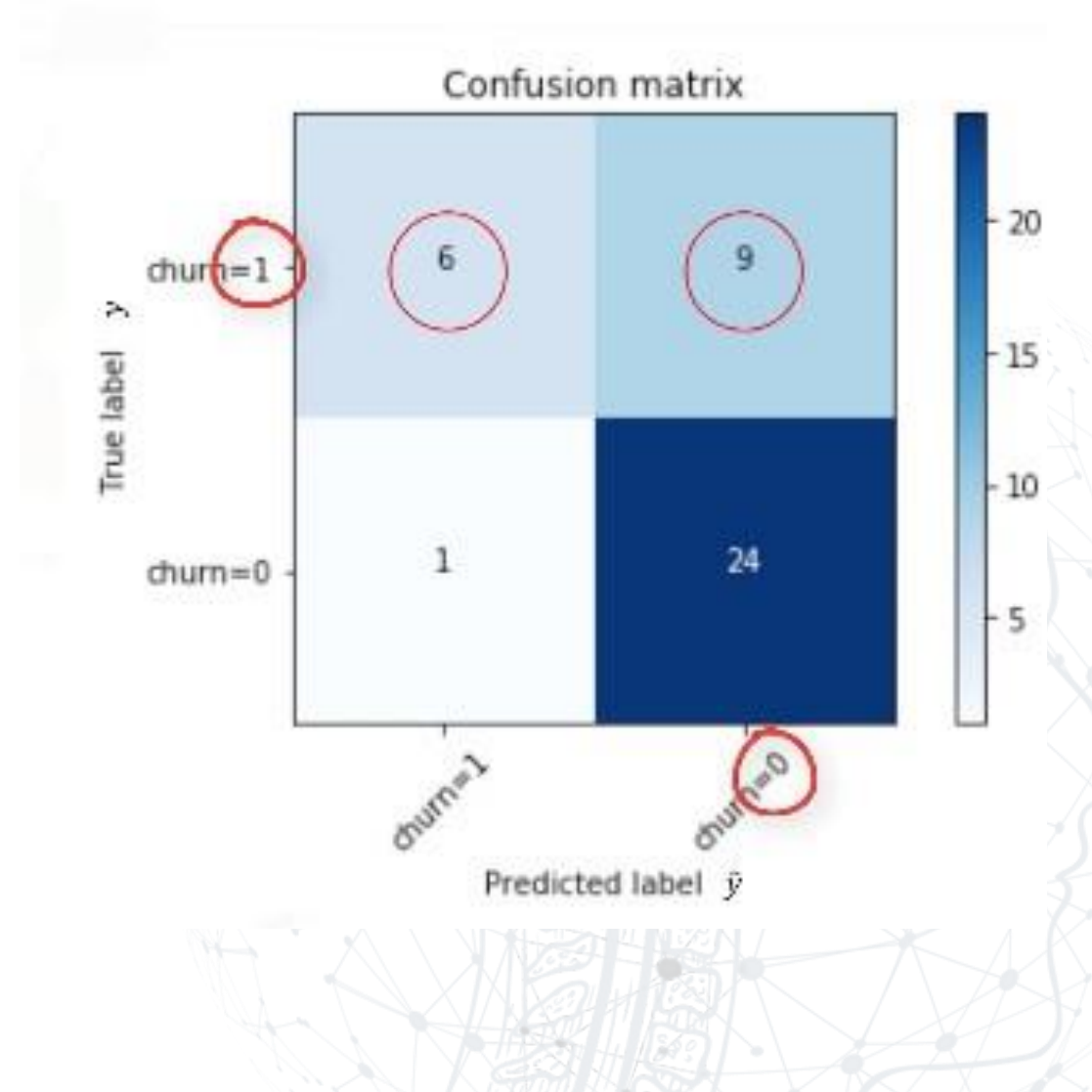
Confusion Matrix

- This matrix shows the corrected and wrong predictions, in comparison with the actual labels.
- Each confusion matrix row shows the Actual/True labels in the test set and the columns show the predicted labels by classifier.
- For example, let's assume that our test set has only 40 rows.



Confusion Matrix

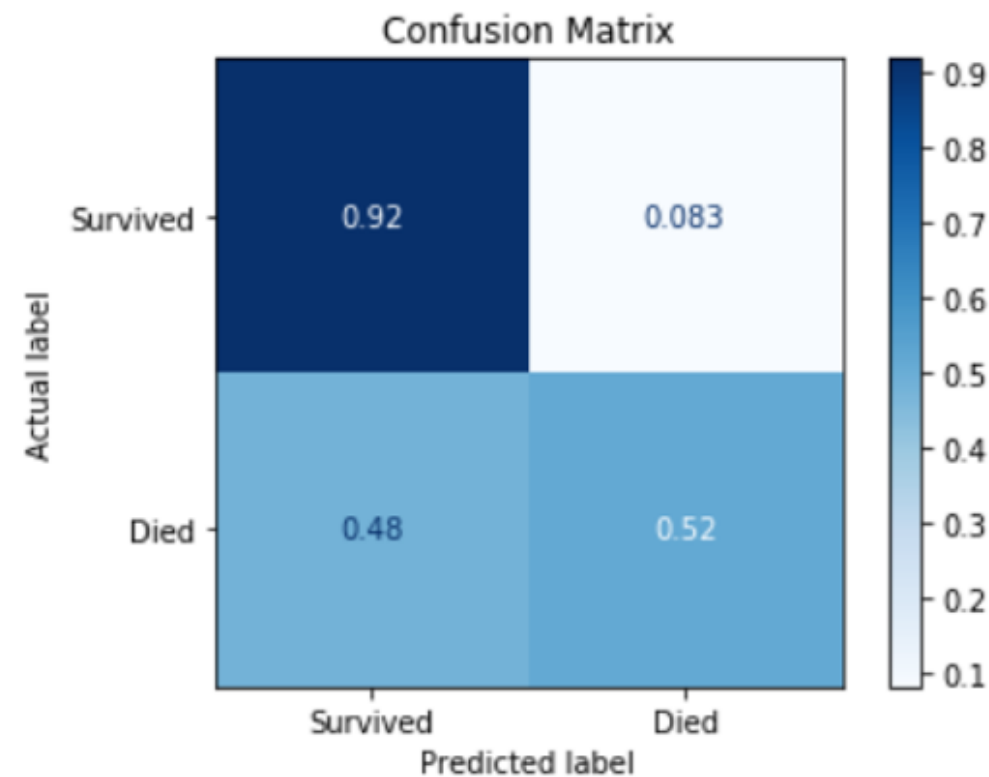
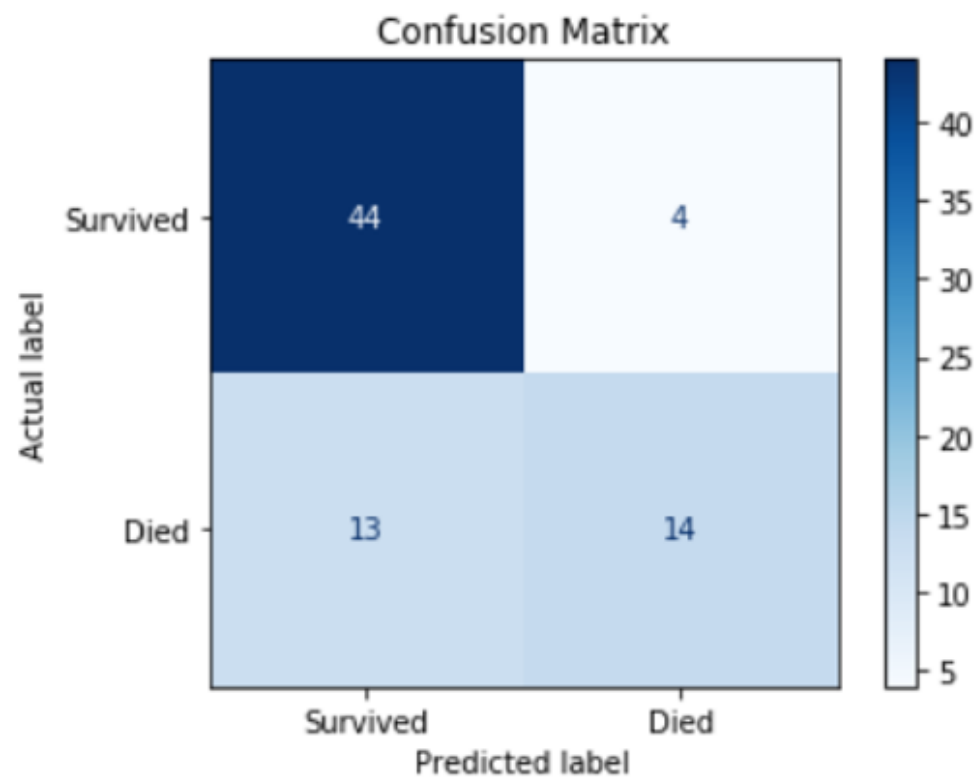
- The first row: actual churn value = 1 → 15 people
 - correctly predicted 6 of them as 1,
 - and 9 of them as 0 (wrong).
- Second row: actual churn value = 0 → 24 people
 - correctly predicted 24 of them as 0,
 - and 1 of them as 1 (wrong) .
- So, it has done a good job in predicting the customers with a churn value of 0, but not very good in predicting value of 1





UNIVERSITAS
ESIA

Confusion Matrix: Not Normalized/Normalized











- A good thing about the confusion matrix is that it shows the model's ability to correctly predict or separate the classes.
- In the specific case of a binary classifier, such as this example, we can interpret these numbers as the count of **true positives, false positives, true negatives, and false negatives**.
- Based on the count of each section, we can calculate the precision and recall of each label.

TP, TN, FP, FN

| ACTUAL VALUES | PREDICTED VALUES | |
|---------------|---------------------|---------------------|
| | Positive | Negative |
| | Positive | Negative |
| Positive | True Positive (TP) | False Negative (FN) |
| Negative | False Positive (FP) | True Negative (TN) |

| PREDICTED VALUES | ACTUAL VALUES | |
|------------------|---------------------|---------------------|
| | Positive | Negative |
| | Positive | Negative |
| Positive | True Positive (TP) | False Positive (FP) |
| Negative | False Negative (FN) | True Negative (TN) |

| ACTUAL VALUES | PREDICTED VALUES | |
|---------------|---|--|
| | Apple | Strawberry |
| | Apple | Strawberry |
| Apple |  |  |
| Strawberry |  |  |

| ACTUAL VALUES | PREDICTED VALUES | |
|---------------|---|---|
| | Pregnant | Not Pregnant |
| | Pregnant | Not Pregnant |
| Pregnant |  |  |
| Not Pregnant |  |  |

Precision – Recall – F1 Score

- **Precision** is a measure of the accuracy, provided that a class label has been predicted.

$$\text{Precision} = \text{True Positive} / (\text{True Positive} + \text{False Positive})$$

- **Recall** is the true positive rate.

$$\text{Recall} = \text{True Positive} / (\text{True Positive} + \text{False Negative}).$$

- **F1 score** is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (which represents perfect precision and recall) and its worst at 0.

- $$\text{F1-score} = 2 \times (\text{prc} \times \text{rec}) / (\text{prc} + \text{rec})$$



Precision – Recall – F1 Score

PREDICTED VALUES

ACTUAL VALUES

| | Positive | Negative | |
|----------|---|------------------------------------|--|
| Positive | True Positive (TP) | False Positive (FP) | Precision= $\frac{TP}{TP+FP}$ |
| Negative | False Negative (FN) | True Negative (TN) | NPV= $\frac{TN}{TN+FN}$ |
| | Recall/Sensitivity= $\frac{TP}{TP+FN}$ | Specificity= $\frac{TN}{TN+FP}$ | ACCURACY= $\frac{TP+TN}{TP+TN+FN+FP}$ |

ACTUAL VALUES

PREDICTED VALUES

| | Positive | Negative | |
|----------|----------------------------------|----------------------------|---|
| Positive | True Positive (TP) | False Negative (FN) | Recall/Sensitivity= $\frac{TP}{TP+FN}$ |
| Negative | False Positive (FP) | True Negative (TN) | Specificity= $\frac{TN}{TN+FP}$ |
| | Precision= $\frac{TP}{TP+FP}$ | NPV= $\frac{TN}{TN+FN}$ | ACCURACY= $\frac{TP+TN}{TP+TN+FN+FP}$ |



UNIVERSITAS
INDONESIA
Veritas, Probatio, Justitia

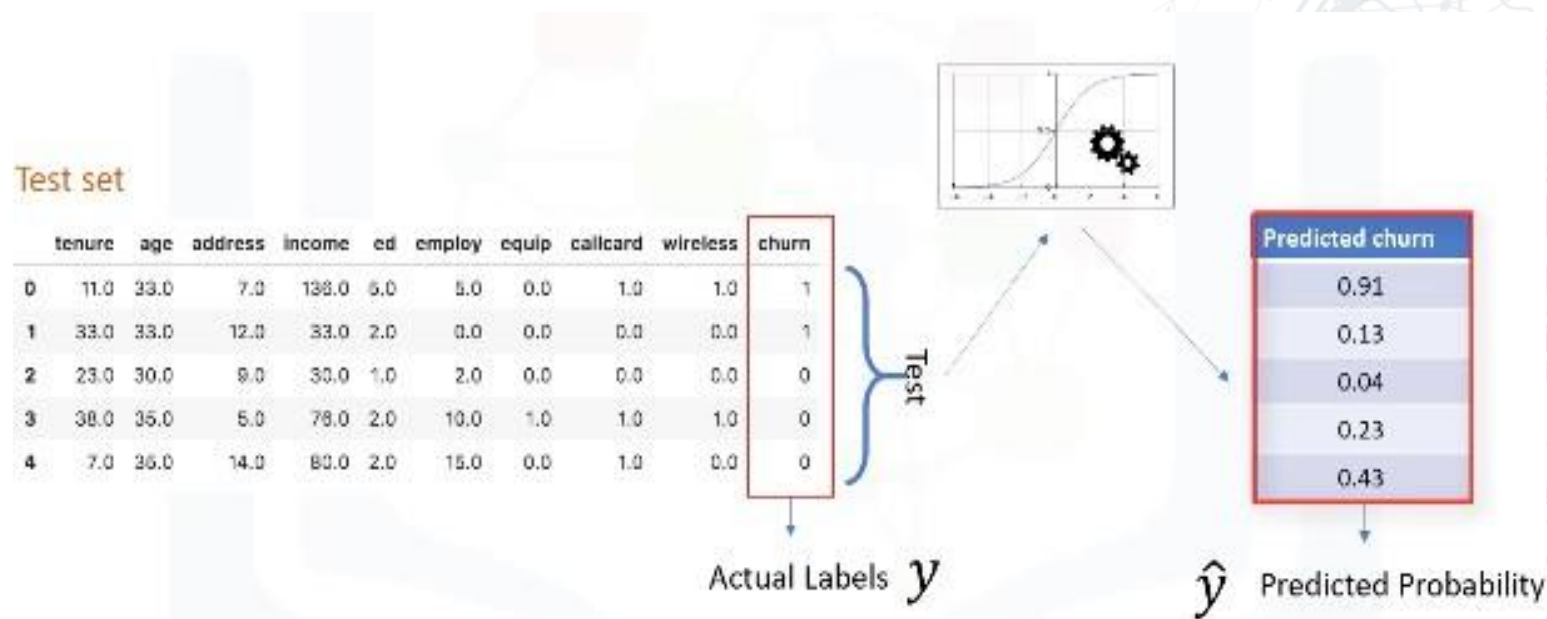
Accuracy

- It is a good way to show that a classifier has a good value for both recall and precision. It is defined using the F1-score equation.
- For example, the F1-score for class 0 (i.e. churn=0), is 0.83, and the F1-score for class 1 (i.e. churn=1), is 0.55.
- And finally, we can tell the average accuracy for this classifier is the average of the F1-score for both labels, which is 0.72 in our case.
- Please notice that both Jaccard and F1-score can be used for multi-class classifiers as well.

| | precision | recall | f1-score |
|----------------|-----------|--------|----------|
| Churn = 0 | 0.73 | 0.96 | 0.83 |
| Churn = 1 | 0.86 | 0.40 | 0.55 |
| Avg Accuracy = | | | 0.72 |

Log-loss

- Sometimes, the output of a classifier is **the probability of a class label**, instead of the label.
- For example, in logistic regression, the output can be the probability of customer churn, i.e., yes (or equals to 1). This probability is a value between 0 and 1.



Evaluation Matrix

- Logarithmic loss (also known as Log loss) measures the performance of a classifier where the predicted output is a probability value between 0 and 1.
- So, for example, predicting a probability of 0.13 when the actual label is 1, would be bad and would result in a high log loss.

Test set

| | tenure | age | address | income | ed | employ | equip | callcard | wireless | churn |
|---|--------|------|---------|--------|-----|--------|-------|----------|----------|-------|
| 0 | 11.0 | 33.0 | 7.0 | 138.0 | 6.0 | 5.0 | 0.0 | 1.0 | 1.0 | 1 |
| 1 | 33.0 | 33.0 | 12.0 | 33.0 | 2.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1 |
| 2 | 23.0 | 30.0 | 9.0 | 50.0 | 1.0 | 2.0 | 0.0 | 0.0 | 0.0 | 0 |
| 3 | 38.0 | 35.0 | 5.0 | 78.0 | 2.0 | 10.0 | 1.0 | 1.0 | 1.0 | 0 |
| 4 | 7.0 | 26.0 | 14.0 | 80.0 | 2.0 | 15.0 | 0.0 | 1.0 | 0.0 | 0 |

Predicted churn

| |
|------|
| 0.91 |
| 0.13 |
| 0.04 |
| 0.23 |
| 0.43 |

Evaluation Matrix

- We can calculate the log loss for each row using the log loss equation, which measures how far each prediction is, from the actual label.

$$(y \times \log(\hat{y}) + (1 - y) \times \log(1 - \hat{y}))$$

- Then, we calculate the average log loss across all rows of the test set.
- It is obvious that more ideal classifiers have progressively smaller values of log loss. So, the classifier with lower log loss has better accuracy.

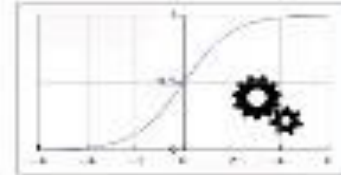
Log loss

Performance of a classifier where the predicted output is a probability value between 0 and 1.

Test set

| | tenure | age | address | income | ed | employ | equip | callcard | wireless | churn |
|---|--------|------|---------|--------|-----|--------|-------|----------|----------|-------|
| 0 | 11.0 | 23.0 | 7.0 | 138.0 | 6.0 | 5.0 | 0.0 | 1.0 | 1.0 | 1 |
| 1 | 33.0 | 33.0 | 12.0 | 33.0 | 2.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1 |
| 2 | 23.0 | 30.0 | 9.0 | 30.0 | 1.0 | 2.0 | 0.0 | 0.0 | 0.0 | 0 |
| 3 | 38.0 | 35.0 | 6.0 | 78.0 | 2.0 | 10.0 | 1.0 | 1.0 | 1.0 | 0 |
| 4 | 7.0 | 26.0 | 14.0 | 80.0 | 2.0 | 15.0 | 0.0 | 1.0 | 0.0 | 0 |

Actual Labels y



| Predicted churn | LogLoss |
|-----------------|---------|
| 0.91 | 0.11 |
| 0.13 | 2.04 |
| 0.04 | 0.04 |
| 0.23 | 0.26 |
| 0.43 | 0.56 |

$LogLoss = 0.60$

\hat{y} Predicted Probability

LogLoss: 0.00 ... 0.35 ... 0.60 ... 1.00

Higher Accuracy

$$LogLoss = -\frac{1}{n} \sum (y \times \log(\hat{y}) + (1 - y) \times \log(1 - \hat{y}))$$

Decision Tree

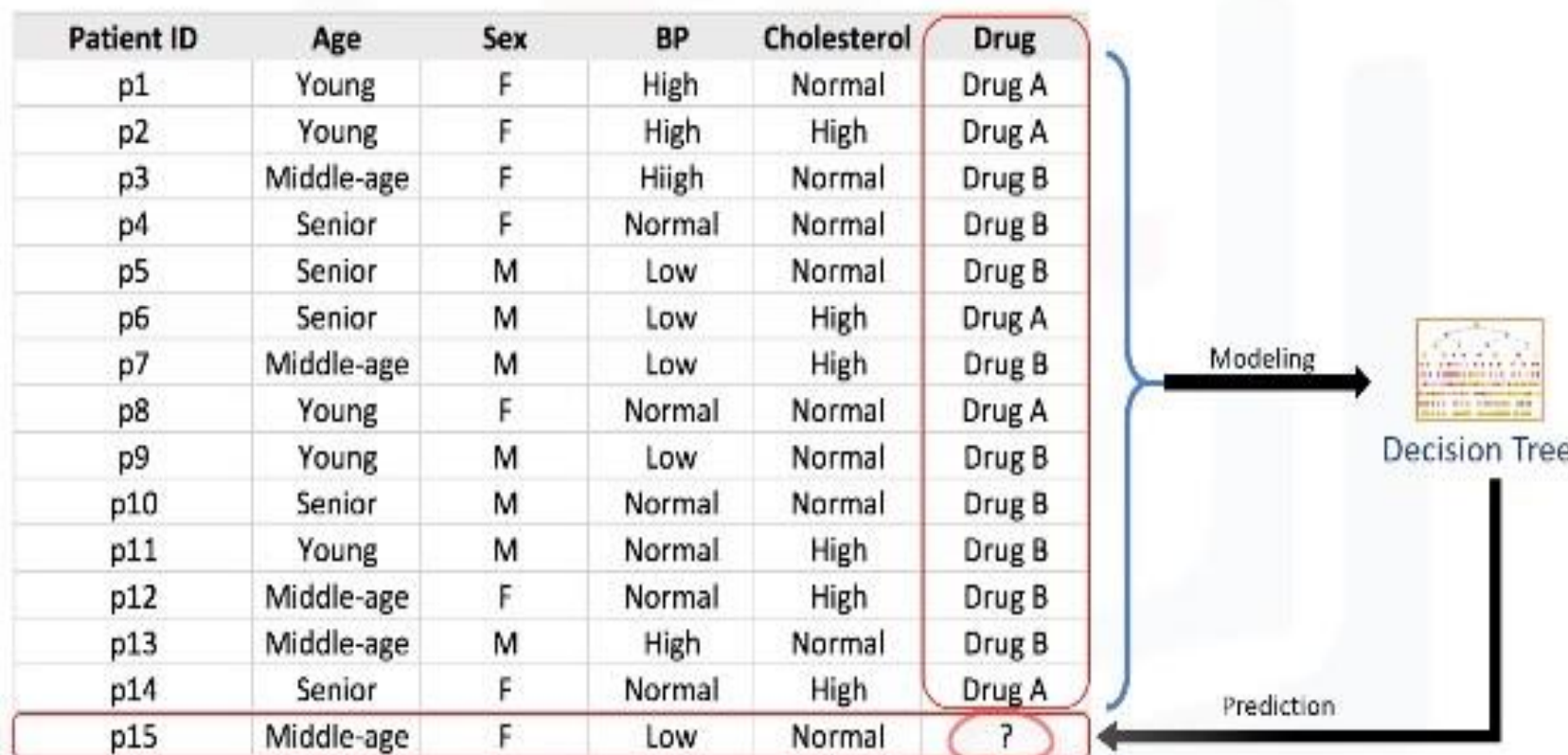
Introduction to Decision Trees

- Imagine that you're a medical researcher compiling data for a study.
- You've already collected data about a set of patients, all of whom suffered from the same illness.

| Patient ID | Age | Sex | BP | Cholesterol | Drug |
|------------|------------|-----|--------|-------------|--------|
| p1 | Young | F | High | Normal | Drug A |
| p2 | Young | F | High | High | Drug A |
| p3 | Middle-age | F | High | Normal | Drug B |
| p4 | Senior | F | Normal | Normal | Drug B |
| p5 | Senior | M | Low | Normal | Drug B |
| p6 | Senior | M | Low | High | Drug A |
| p7 | Middle-age | M | Low | High | Drug B |
| p8 | Young | F | Normal | Normal | Drug A |
| p9 | Young | M | Low | Normal | Drug B |
| p10 | Senior | M | Normal | Normal | Drug B |
| p11 | Young | M | Normal | High | Drug B |
| p12 | Middle-age | F | Normal | High | Drug B |
| p13 | Middle-age | M | High | Normal | Drug B |
| p14 | Senior | F | Normal | High | Drug A |
| p15 | Middle-age | F | Low | Normal | ? |

Decision Trees

- The feature sets of this dataset are Age, Gender, Blood Pressure, and Cholesterol of our group of patients, and the target is the drug that each patient responded to.
- It is a sample of binary classifiers, and you can use the training part of the dataset to build a decision tree, and then, use it to predict the class of an unknown patient ... in essence, to come up with a decision on which drug to prescribe to a new patient.



How to Build Decision Trees

Decision trees are built by splitting the training set into distinct nodes, where one node contains all of, or most of, one category of the data.

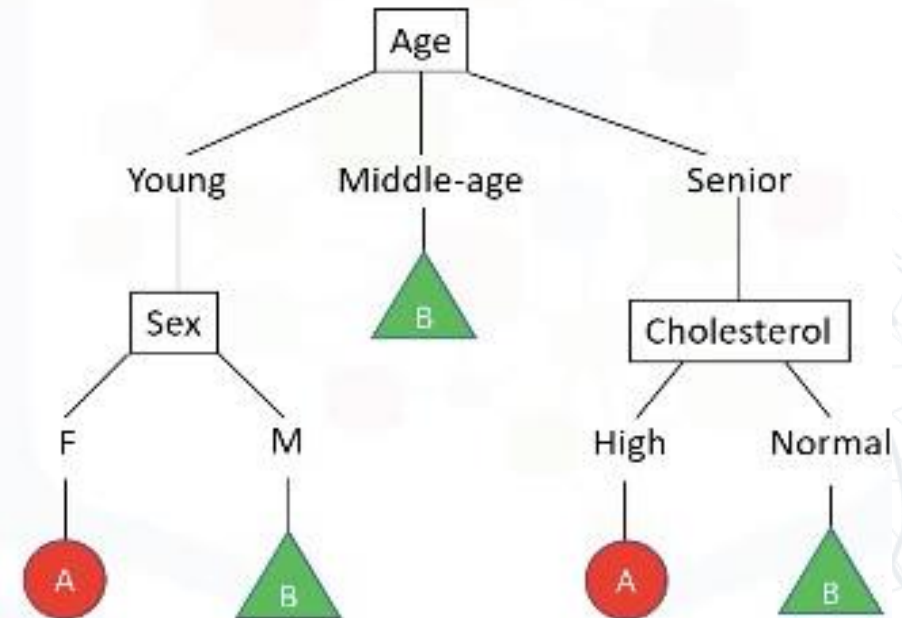
If we look at the diagram here, we can see that it's a patient classifier.

So, as mentioned, we want to prescribe a drug to a new patient, but the decision to choose drug A or B, will be influenced by the patient's situation.

We start with the Age, which can be Young, Middle-aged, or Senior.

If the patient is Middle-aged, then we'll definitely go for Drug B.

On the other hand, if he is a Young or a Senior patient, we'll need more details to help us determine which drug to prescribe.

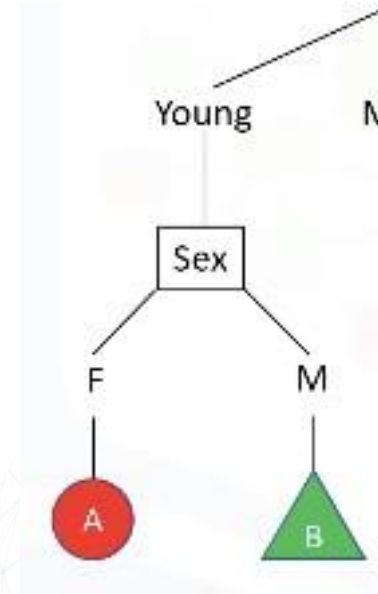




How to Build Decision Trees

The additional decision variables can be things such as Cholesterol levels, Gender or Blood Pressure.

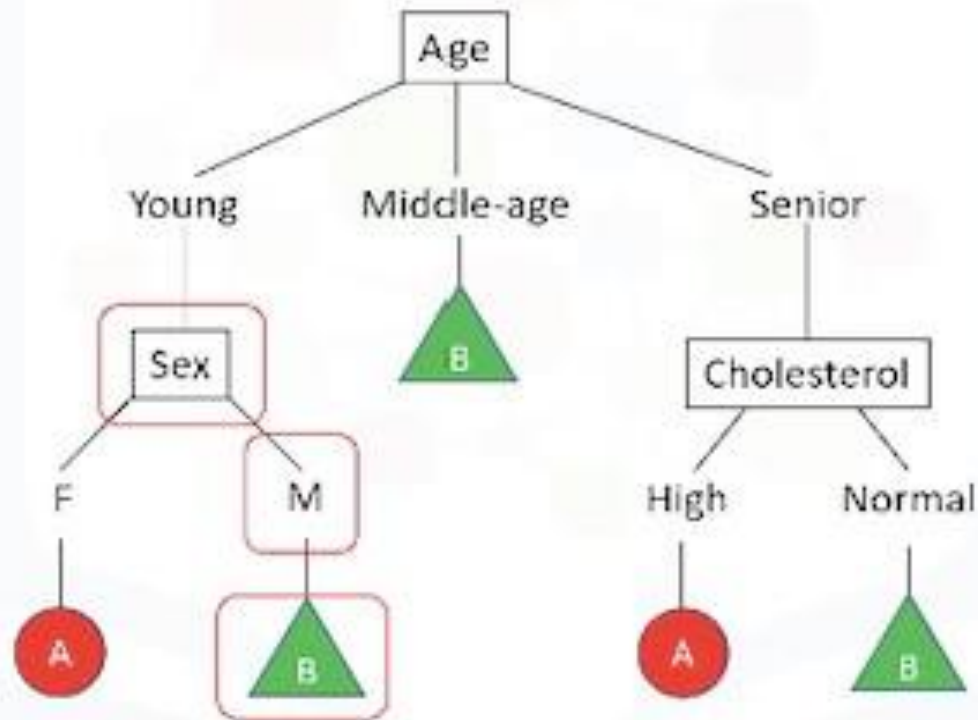
For example, if the patient is Female, then we will recommend Drug A, but if the patient is Male, then we'll go for Drug B. As you can see, decision trees are about testing an attribute and branching the cases, based on the result of the test.



How to Build Decision Trees

Each internal node corresponds to a test. And each branch corresponds to a result of the test. And each leaf node assigns a patient to a class.

Now the question is how can we build such a decision tree?

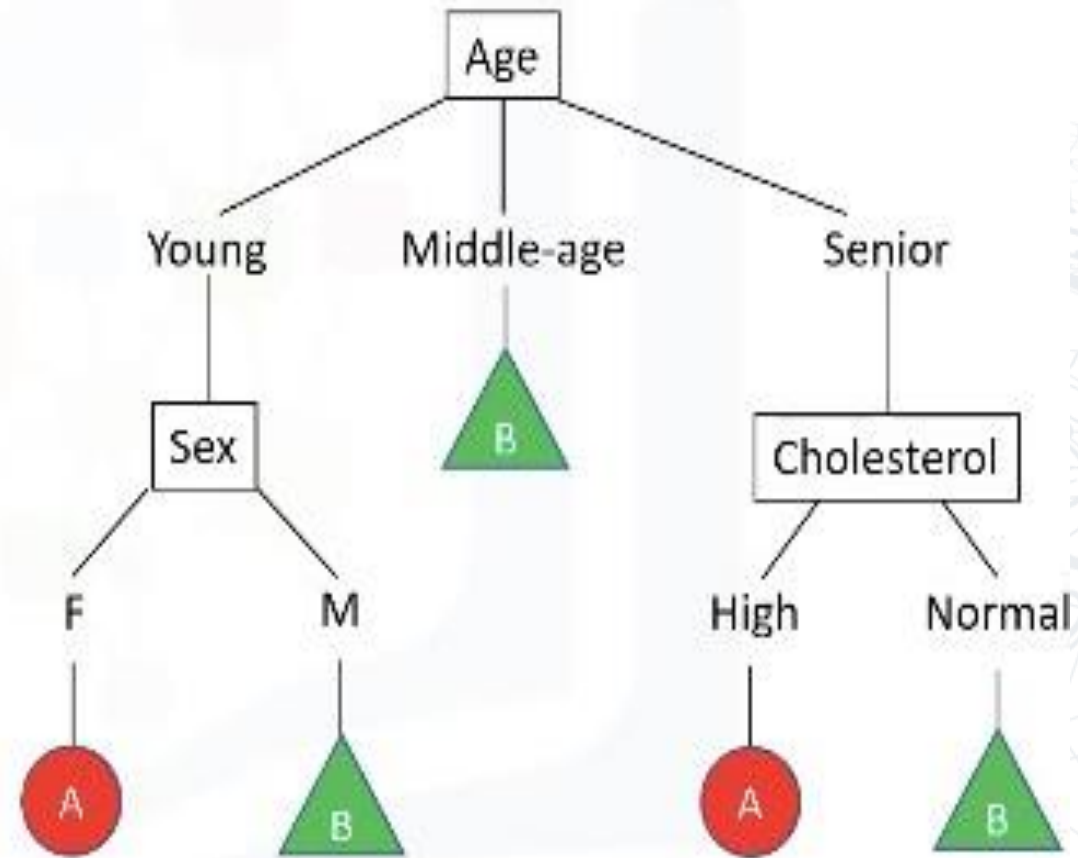


- Each **internal node** corresponds to a test
- Each **branch** corresponds to a result of the test
- Each **leaf node** assigns a classification



Decision Tree Algorithm

1. Choose an attribute from your dataset.
2. Calculate the significance of attribute in splitting of data.
3. Split data based on the value of the best attribute.
4. Go to step 1.



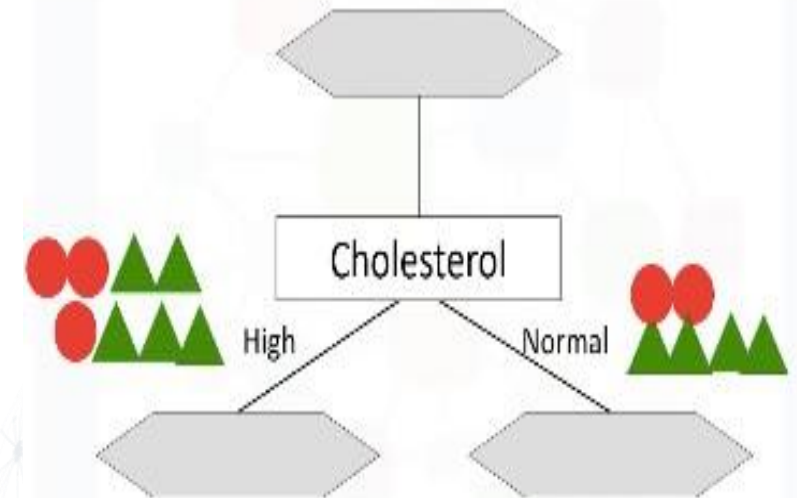
Building Decision Trees

- Consider the drug dataset again.
- The question is, “How do we build the decision tree based on that dataset?”

| Patient ID | Age | Sex | BP | Cholesterol | Drug |
|------------|------------|-----|--------|-------------|--------|
| p1 | Young | F | High | Normal | Drug A |
| p2 | Young | F | High | High | Drug A |
| p3 | Middle-age | F | High | Normal | Drug B |
| p4 | Senior | F | Normal | Normal | Drug B |
| p5 | Senior | M | Low | Normal | Drug B |
| p6 | Senior | M | Low | High | Drug A |
| p7 | Middle-age | M | Low | High | Drug B |
| p8 | Young | F | Normal | Normal | Drug A |
| p9 | Young | M | Low | Normal | Drug B |
| p10 | Senior | M | Normal | Normal | Drug B |
| p11 | Young | M | Normal | High | Drug B |
| p12 | Middle-age | F | Normal | High | Drug B |
| p13 | Middle-age | M | High | Normal | Drug B |
| p14 | Senior | F | Normal | High | Drug A |
| p15 | Middle-age | F | Low | Normal | ? |

Building Decision Trees

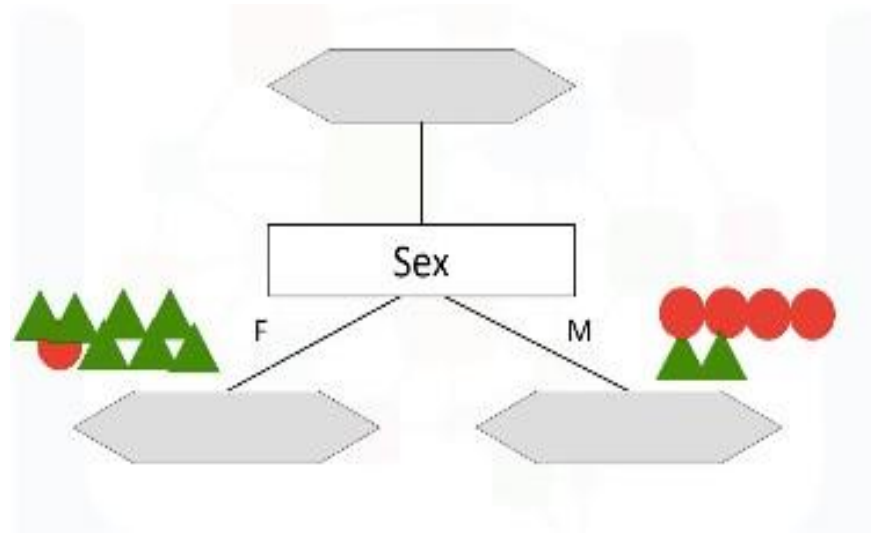
- Decision trees are built using recursive partitioning to classify the data.
- The algorithm chooses the most predictive feature to split the data on.
 - “which attribute is the best, or more predictive, to split data based on the feature.”
- Let's say we pick “Cholesterol” as the first attribute to split data. It will split our data into 2 branches.
- As you can see, if the patient high “Cholesterol,” we cannot say with high confidence that Drug B might be suitable for him.
- Also, if the Patient's “Cholesterol” is normal, we still don't have sufficient evidence or information to determine if either Drug A or Drug B is, in fact, suitable.



Thus, Cholesterol is not a good attribute now

Building Decision Trees

- So, let's try another attribute. This time, we pick the “sex” attribute of patients.
- It will split our data into 2 branches, Male and Female.
- As you can see, if the patient is Female, we can say Drug B might be suitable for her with high certainty.

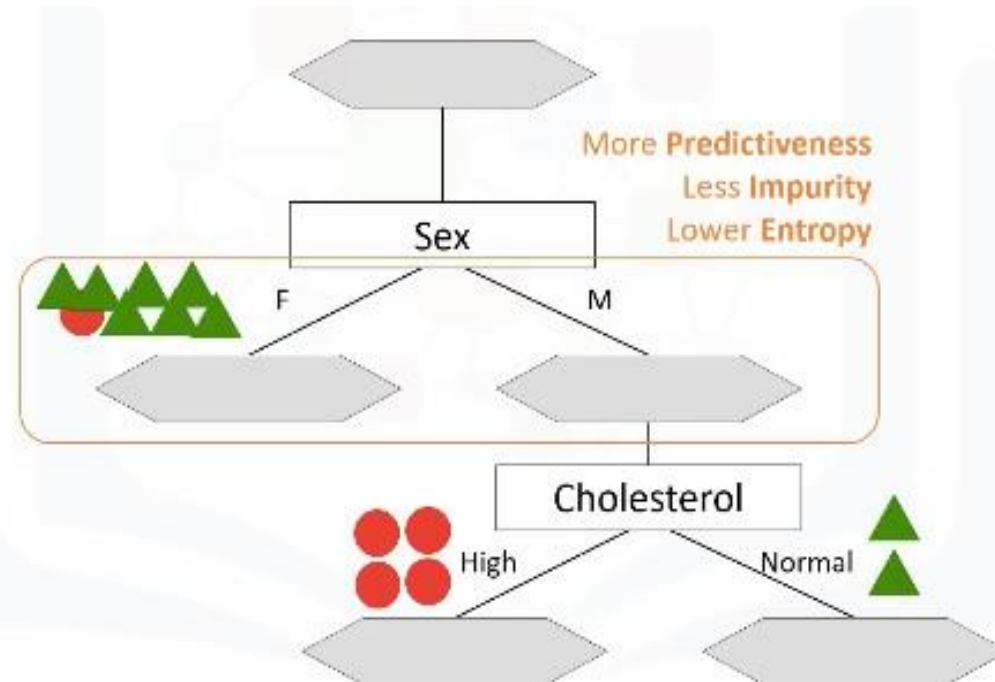


Building Decision Trees

- But, if the patient is Male, we don't have sufficient evidence or information to determine if Drug A or Drug B is suitable.
- However, it is still a better choice in comparison with the "Cholesterol" attribute, because the result in the nodes are more pure.
- It means, nodes that are either mostly Drug A or Drug B.
- So, we can say the "Sex" attribute is more significant than "Cholesterol," or in other words, it's more predictive than the other attributes.
- Indeed, "predictiveness" is based on decrease in "impurity" of nodes.
- We're looking for the best feature to decrease the "impurity" of patients in the leaves, after splitting them up based on that feature.
- So, the "Sex" feature is a good candidate in the following case, because it almost found the pure patients. Let's go one step further.
- For the Male patient branch, we again test other attributes to split the subtree.

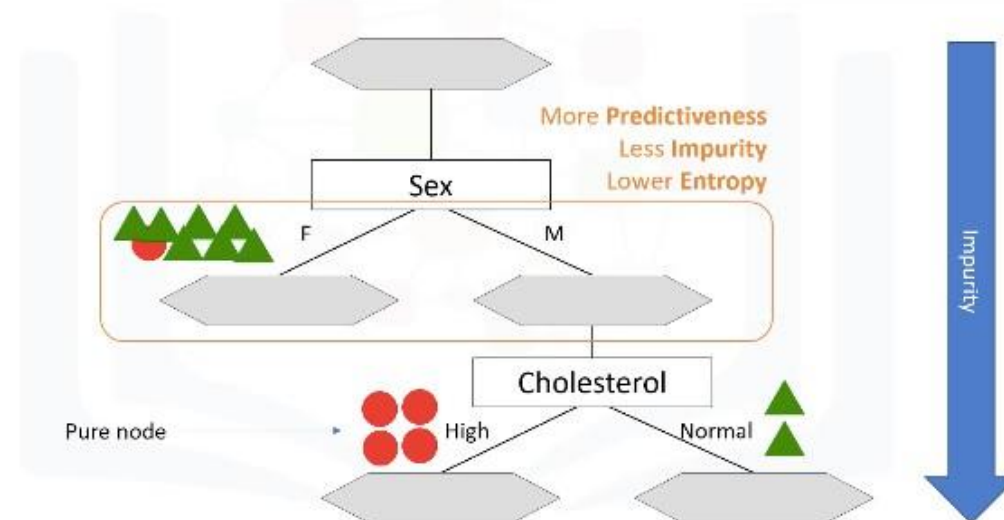
Building Decision Trees

- We test “Cholesterol” again here.
- As you can see, it results in even more pure leaves. So, we can easily make a decision here.



Building Decision Trees

- For example, if a patient is “Male”, and his “Cholesterol” is “High”, we can certainly prescribe Drug A, but if it is “Normal”, we can prescribe Drug B with high confidence.
- As you might notice, the choice of attribute to split data is very important, and it is all about
- A node in the tree is considered pure if all nodes fall into a specific category
- In fact, the method uses recursive records into segments by minimizing the “Impurity” of nodes is calculated by “Entropy” of data in the node.



Decision Trees - Entropy

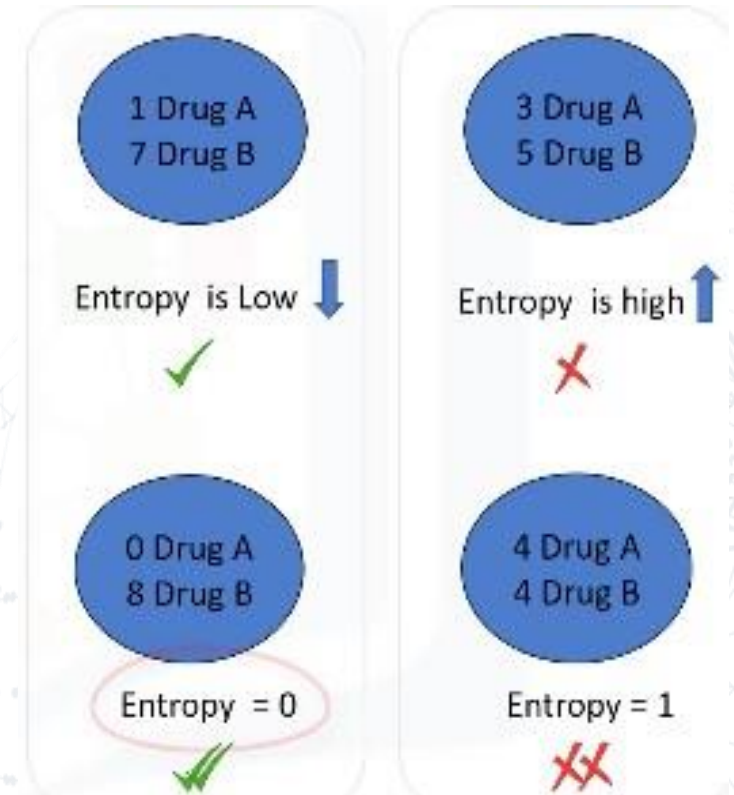
- So, what is “Entropy”?
- Entropy is the amount of information disorder, or the amount of randomness in the data.
- The entropy in the node depends on how much random data is in that node and is calculated for each node. In decision trees, we're looking for trees that have the smallest entropy in their nodes.





Decision Trees - Entropy

- The entropy is used to calculate the homogeneity of the samples in that node.
- If the samples are completely homogeneous the entropy is zero and if the samples are equally divided, it has an entropy of one.
- This means, if all the data in a node are either Drug A or Drug B, then the entropy is zero, but if half of the data are Drug A and other half are B, then the entropy is one.



Decision Trees - Entropy

- You can easily calculate the entropy of a node using the frequency table of the attribute through the Entropy formula, where P is for the proportion or ratio of a category, such as Drug A or B.
- Please remember, though, that you don't have to calculate these, as it's easily calculated by the libraries or packages that you use.

$$\text{Entropy} = - p(A)\log(p(A)) - p(B)\log(p(B))$$

- As an example, let's calculate the entropy of the dataset before splitting it. We have 9 occurrences of Drug B and 5 of Drug A.
- You can embed these numbers into the Entropy formula to calculate the impurity of the target attribute before splitting it. In this case, it is 0.94.



| Patient ID | Age | Sex | BP | Cholesterol | Drug |
|------------|------------|-----|--------|-------------|--------|
| p1 | Young | F | High | Normal | Drug A |
| p2 | Young | F | High | High | Drug A |
| p3 | Middle-age | F | High | Normal | Drug B |
| p4 | Senior | F | Normal | Normal | Drug B |
| p5 | Senior | M | Low | Normal | Drug B |
| p6 | Senior | M | Low | High | Drug A |
| p7 | Middle-age | M | Low | High | Drug B |
| p8 | Young | F | Normal | Normal | Drug A |
| p9 | Young | M | Low | Normal | Drug B |
| p10 | Senior | M | Normal | Normal | Drug B |
| p11 | Young | M | Normal | High | Drug B |
| p12 | Middle-age | F | Normal | High | Drug B |
| p13 | Middle-age | M | High | Normal | Drug B |
| p14 | Senior | F | Normal | High | Drug A |

S: [9 B, 5 A]

$$E = -p(B)\log(p(B)) - p(A)\log(p(A))$$

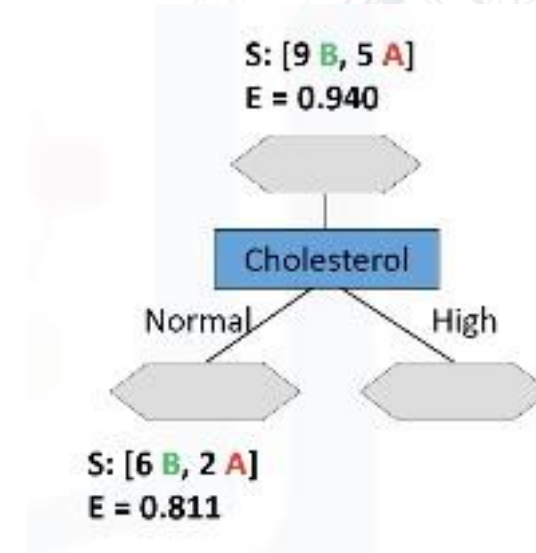
$$E = - (9/14)\log(9/14) - (5/14)\log(5/14)$$

$$E = 0.940$$

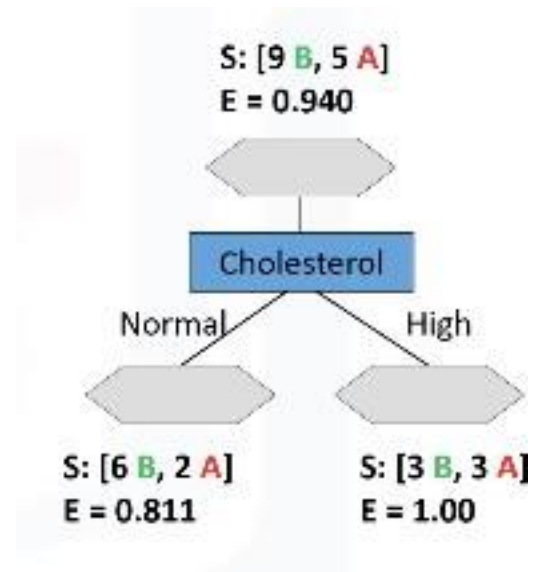
Decision Trees - Entropy

- So, what is entropy after splitting?
- Now we can test different attributes to find the one with the most “predictiveness,” which results in two more pure branches.
- Let’s first select the “Cholesterol” of the patient and see how the data gets split, based on its values.
- For example, when it is “normal,” we have 6 for Drug B, and 2 for Drug A.
- We can calculate the Entropy of this node based on the distribution of drug A and B, which is 0.8 in this case.

| Patient ID | Age | Sex | BP | Cholesterol | Drug |
|------------|------------|-----|--------|-------------|--------|
| p1 | Young | F | High | Normal | Drug A |
| p2 | Young | F | High | High | Drug A |
| p3 | Middle-age | F | High | Normal | Drug B |
| p4 | Senior | F | Normal | Normal | Drug B |
| p5 | Senior | M | Low | Normal | Drug B |
| p6 | Senior | M | Low | High | Drug A |
| p7 | Middle-age | M | Low | High | Drug B |
| p8 | Young | F | Normal | Normal | Drug A |
| p9 | Young | M | Low | Normal | Drug B |
| p10 | Senior | M | Normal | Normal | Drug B |
| p11 | Young | M | Normal | High | Drug B |
| p12 | Middle-age | F | Normal | High | Drug B |
| p13 | Middle-age | M | High | Normal | Drug B |
| p14 | Senior | F | Normal | High | Drug A |



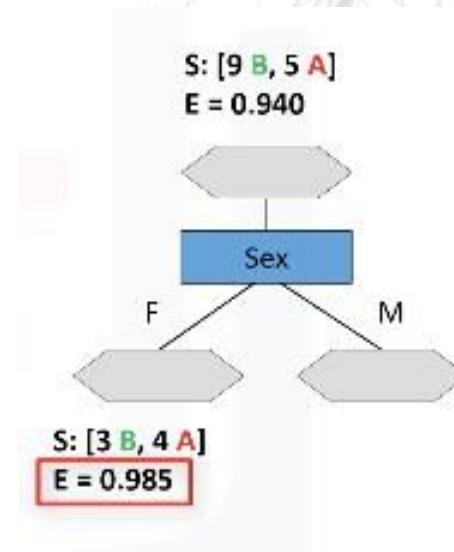
But, when Cholesterol is “High,” the data is split into 3 for drug B and 3 for drug A.
Calculating its entropy, we can see it would be 1.0.



Decision Trees - Entropy

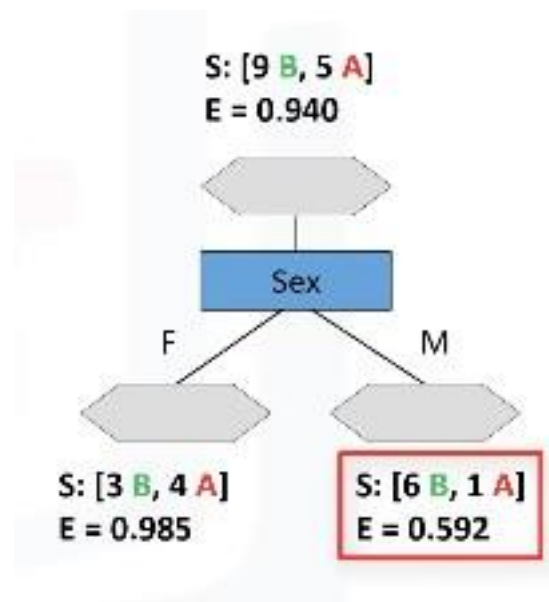
- We should go through all the attributes and calculate the “Entropy” after the split, and then chose the best attribute.
- Let’s try another field.
- Let’s choose the Sex attribute for the next check.
- As you can see, when we use the Sex attribute to split the data, when its value is “Female,” we have 3 patients that responded to Drug B, and 4 patients that responded to Drug A.
- The entropy for this node is 0.98 which is not very promising

| Patient ID | Age | Sex | BP | Cholesterol | Drug |
|------------|------------|-----|--------|-------------|--------|
| p1 | Young | F | High | Normal | Drug A |
| p2 | Young | F | High | High | Drug A |
| p3 | Middle-age | F | High | Normal | Drug B |
| p4 | Senior | F | Normal | Normal | Drug B |
| p5 | Senior | M | Low | Normal | Drug B |
| p6 | Senior | M | Low | High | Drug A |
| p7 | Middle-age | M | Low | High | Drug B |
| p8 | Young | F | Normal | Normal | Drug A |
| p9 | Young | M | Low | Normal | Drug B |
| p10 | Senior | M | Normal | Normal | Drug B |
| p11 | Young | M | Normal | High | Drug B |
| p12 | Middle-age | F | Normal | High | Drug B |
| p13 | Middle-age | M | High | Normal | Drug B |
| p14 | Senior | F | Normal | High | Drug A |



Decision Trees - Entropy

- However, on other side of the branch, when the value of the Sex attribute is Male, the result is more pure with 6 for Drug B and only 1 for Drug A.
- The entropy for this group is 0.59.

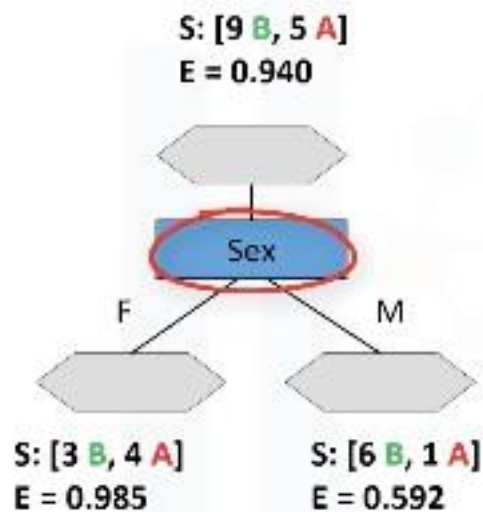


Now, the question is, between the Cholesterol and Sex attributes, which one is a better choice?

Which one is better as the first attribute to divide the dataset into 2 branches?

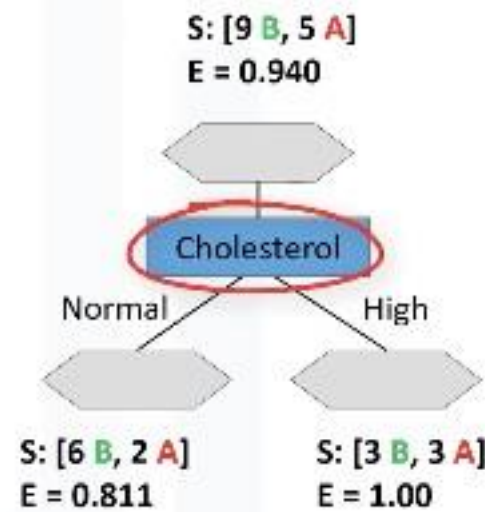
Or, in other words, which attribute results in more pure nodes for our drugs?

Or, in which tree, do we have less entropy after splitting rather than before splitting?



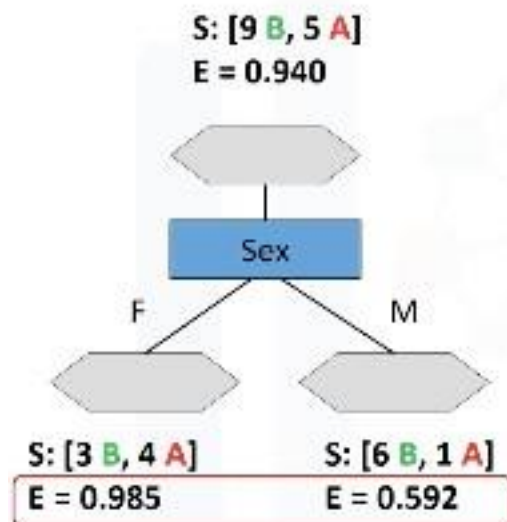
| Patient ID | Age | Sex | BP | Cholesterol | Drug |
|------------|------------|-----|--------|-------------|--------|
| p1 | Young | F | High | Normal | Drug A |
| p2 | Young | F | High | High | Drug A |
| p3 | Middle age | F | High | Normal | Drug B |
| p4 | Senior | F | Normal | Normal | Drug B |
| p5 | Senior | M | Low | Normal | Drug B |
| p6 | Senior | M | Low | High | Drug A |
| p7 | Middle age | M | Low | High | Drug B |
| p8 | Young | F | Normal | Normal | Drug A |
| p9 | Young | M | Low | Normal | Drug B |
| p10 | Senior | M | Normal | Normal | Drug B |
| p11 | Young | M | Normal | High | Drug B |
| p12 | Middle age | F | Normal | High | Drug B |
| p13 | Middle age | M | High | Normal | Drug B |
| p14 | Senior | F | Normal | High | Drug A |

?



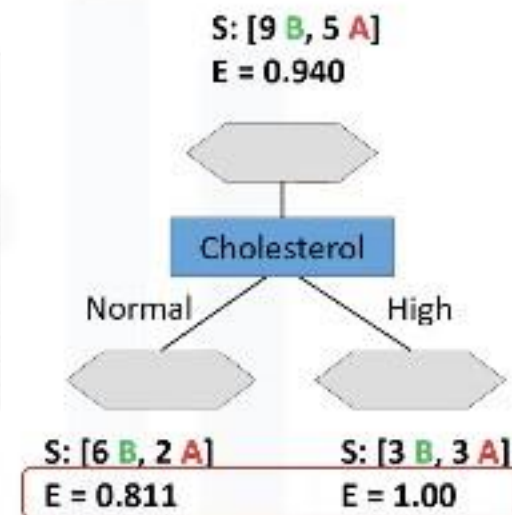
Decision Trees - Entropy

- The "Sex" attribute with entropy of 0.98 and 0.59, or the "Cholesterol" attribute with entropy of 0.81 and 1.0 in its branches?
- The answer is, "The tree with the higher information gain after splitting."



| Patient ID | Age | Sex | BP | Cholesterol | Drug |
|------------|------------|-----|--------|-------------|--------|
| p1 | Young | F | High | Normal | Drug A |
| p2 | Young | F | High | High | Drug A |
| p3 | Middle-age | F | High | Normal | Drug B |
| p4 | Senior | F | Normal | Normal | Drug B |
| p5 | Senior | M | Low | Normal | Drug B |
| p6 | Senior | M | Low | High | Drug A |
| p7 | Middle-age | M | Low | High | Drug B |
| p8 | Young | F | Normal | Normal | Drug A |
| p9 | Young | M | Low | Normal | Drug B |
| p10 | Senior | M | Normal | Normal | Drug B |
| p11 | Young | M | Normal | High | Drug B |
| p12 | Middle-age | F | Normal | High | Drug B |
| p13 | Middle-age | M | High | Normal | Drug B |
| p14 | Senior | F | Normal | High | Drug A |

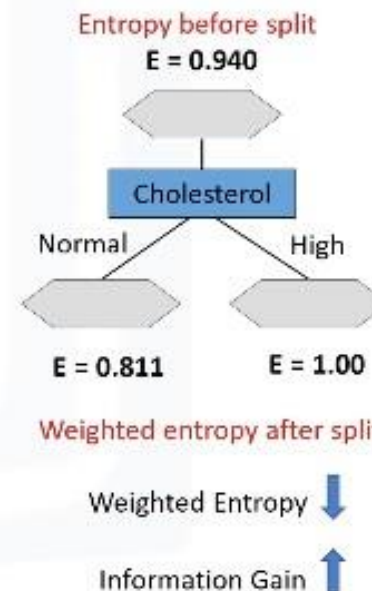
?



Information gain

- Information gain is the information that can increase the level of certainty after splitting.
- It is the entropy of a tree before the split minus the weighted entropy after the split by an attribute.

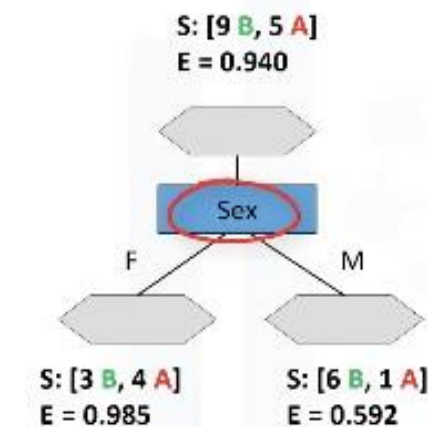
$$\text{Information Gain} = (\text{Entropy before split}) - (\text{weighted entropy after split})$$





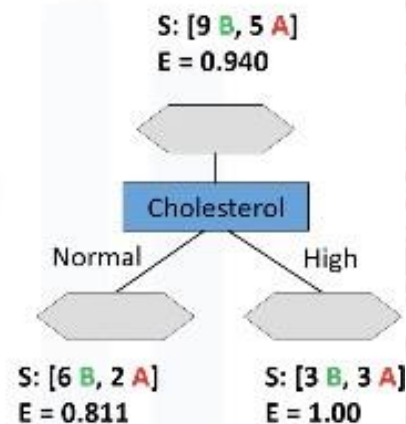
Decision Trees

- We can think of information gain and entropy as opposites.
- As entropy, or the amount of randomness, decreases, the information gain, or amount of certainty, increases, and vice-versa.
- So, constructing a decision tree is all about finding attributes that return the highest information gain.
- Let's see how "information gain" is calculated for the Sex attribute.
- As mentioned, the information gain is the entropy of the tree before the split, minus the weighted



| Patient ID | Age | Sex | BP | Cholesterol | Drug |
|------------|------------|-----|--------|-------------|--------|
| p1 | Young | F | High | Normal | Drug A |
| p2 | Young | F | High | High | Drug A |
| p3 | Middle age | F | High | Normal | Drug B |
| p4 | Senior | F | Normal | Normal | Drug B |
| p5 | Senior | M | Low | Normal | Drug B |
| p6 | Senior | M | Low | High | Drug A |
| p7 | Middle age | M | Low | High | Drug B |
| p8 | Young | F | Normal | Normal | Drug A |
| p9 | Young | M | Low | Normal | Drug B |
| p10 | Senior | M | Normal | Normal | Drug B |
| p11 | Young | M | Normal | High | Drug B |
| p12 | Middle age | F | Normal | High | Drug B |
| p13 | Middle age | M | High | Normal | Drug B |
| p14 | Senior | F | Normal | High | Drug A |

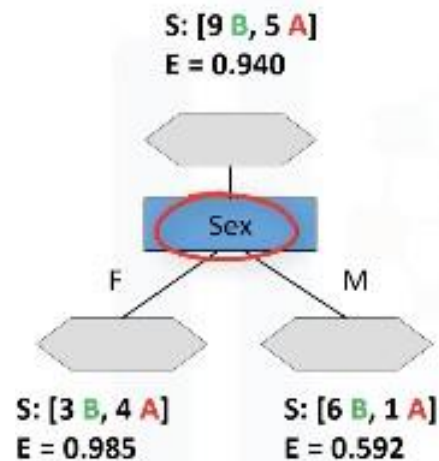
?



Gain (s, Sex)

Decision Trees

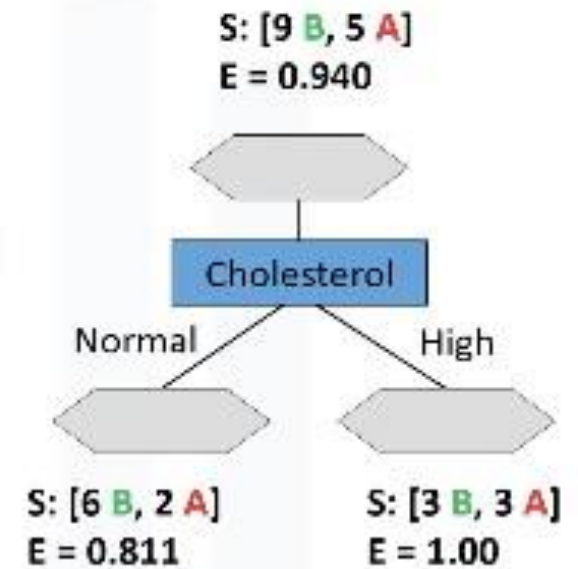
- The portion of Female patients is 7 out of 14, and its entropy is 0.985.
- Also, the portion of men is 7 out of 14, and the entropy of the Male node is 0.592.
- The result is the weighted entropy after the split. So, the information gain of the tree if we use the “Sex” attribute to split the dataset is 0.151.



$$\begin{aligned} \text{Gain}(s, \text{Sex}) &= 0.940 - [(7/14)0.985 + (7/14)0.592] \\ &= 0.151 \end{aligned}$$

Decision Trees

- As you can see, we will consider the entropy over the distribution of samples falling under each leaf node, and we'll take a weighted average of that entropy – weighted by the proportion of samples falling under that leaf.
- We can calculate the information gain of the tree if we use “Cholesterol” as well. It is 0.48.

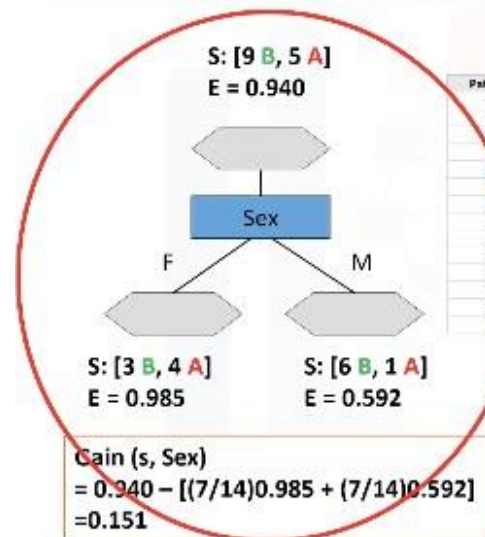


$$\begin{aligned}\text{Gain}(s, \text{Cholesterol}) &= 0.940 - [(8/14) \cdot 0.811 + (6/14) \cdot 1.0] \\ &= 0.048\end{aligned}$$

Decision Trees

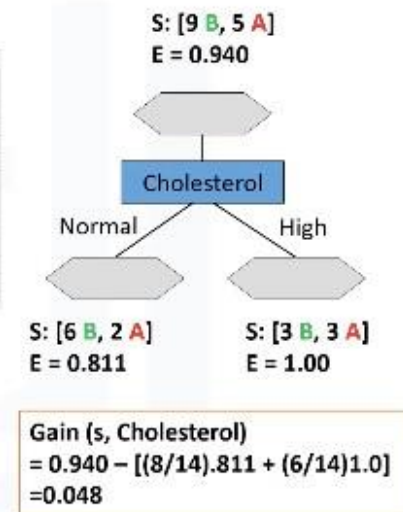
- Now, the question is, “Which attribute is more suitable?”
- Well, as mentioned, the tree with the higher information gain after splitting. This means the “Sex” attribute.

Which attribute is the best?



| Patient ID | Age | Sex | BP | Cholesterol | Drug |
|------------|------------|-----|--------|-------------|--------|
| p1 | Young | F | High | Normal | Drug A |
| p2 | Young | F | High | High | Drug A |
| p3 | Middle age | F | High | Normal | Drug B |
| p4 | Senior | F | Normal | Normal | Drug B |
| p5 | Senior | M | Low | Normal | Drug B |
| p6 | Senior | M | Low | High | Drug A |
| p7 | Middle age | M | Low | High | Drug B |
| p8 | Young | F | Normal | Normal | Drug A |
| p9 | Young | M | Low | Normal | Drug B |
| p10 | Senior | M | Normal | Normal | Drug B |
| p11 | Young | M | Normal | High | Drug B |
| p12 | Middle age | F | Normal | High | Drug B |
| p13 | Middle age | M | High | Normal | Drug B |
| p14 | Senior | F | Normal | High | Drug A |

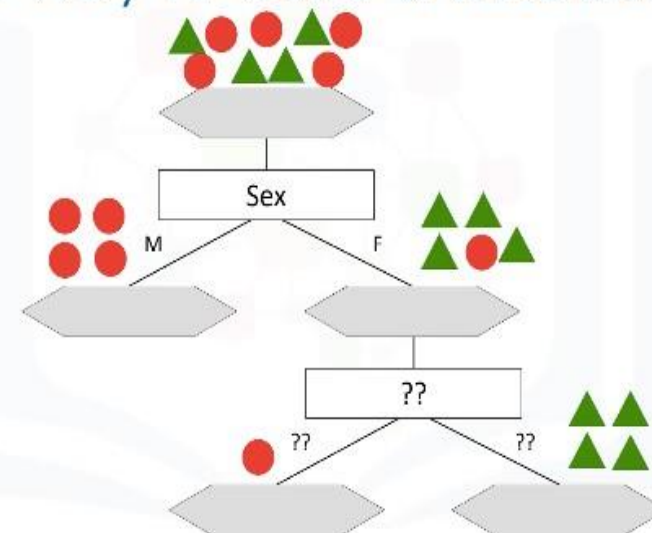
?



Decision Trees

- So, we select the “Sex” attribute as the first splitter.
- Now, what is the next attribute after branching by the “Sex” attribute?
- Well, as you can guess, we should repeat the process for each branch, and test each of the other attributes to continue to reach the most pure leaves.
- This is the way that you build a decision tree!

Correct way to build a decision tree



Other impurity index: Gini impurity index

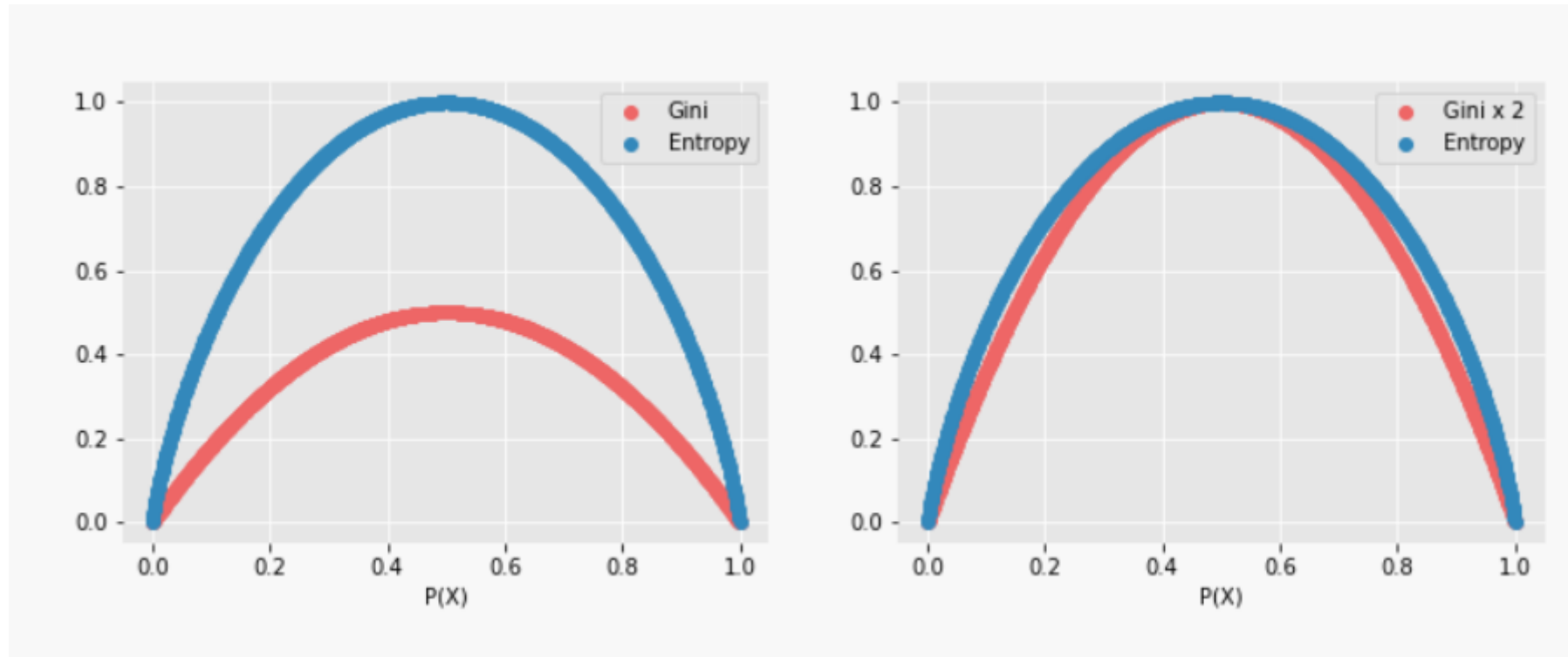
$$GiniIndex = 1 - \sum_j p_j^2$$

- The gini impurity measures the frequency at which any element of the dataset will be mislabelled when it is randomly labeled.
- The minimum value of the Gini Index is 0. This happens when the node is pure, this means that all the contained elements in the node are of one unique class. Therefore, this node will not be split again. Thus, the optimum split is chosen by the features with less Gini Index. Moreover, it gets the maximum value when the probability of the two classes are the same.

$$Gini_{min} = 1 - (1^2) = 0$$

$$Gini_{max} = 1 - (0.5^2 + 0.5^2) = 0.5$$

Gini vs Entropy



- The gini criterion is much faster because it is less computationally expensive.
- On the other hand, the obtained results using the entropy criterion are slightly better

Logistic Regression

Logistic Regression

- Logistic regression is analogous to linear regression but tries to predict a categorical or discrete target field instead of a numeric one.
- In linear regression, we might try to predict a continuous value of variables, such as the price of a house, blood pressure of patient, or fuel consumption of a car.
- But, in logistic regression, we predict a variable which is **binary**, such as, Yes/No, TRUE/FALSE, successful or Not successful, pregnant/Not pregnant, and so on, all of which can all be coded as 0 or 1.
- In logistic regression, **dependent variables should be continuous**; if categorical, they should be dummy or indicator-coded.
- This means we have to transform them to some continuous value.
- Please note that logistic regression can be used for both binary classification and multiclass classification, but for simplicity, we'll focus on binary classification.

When to use Logistic Regression

- Here are four situations in which Logistic regression is a good candidate:
- First, when the target field in your data is categorical, or specifically, is binary, such as 0/1, yes/no, churn or no churn, positive/negative, and so on.
- Second, when you need the probability of your prediction, for example, if you want to know what the probability is, of a customer buying a product.
 - **Logistic regression returns a probability score between 0 and 1 for a given sample of data.** In fact, logistic regressing predicts the probability of that sample, and we map the cases to a discrete class based on that probability.

- Third, if your data is linearly separable. The decision boundary of logistic regression is a line or a plane or a hyper-plane.
 - A classifier will classify all the points on one side of the decision boundary as belonging to one class and all those on the other side as belonging to the other class.
 - For example, if we have just two features (and are not applying any polynomial processing), we can obtain an inequality like
$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 > 0$$
 - which is a half-plane, easily plottable.
 - Please note that in using logistic regression, we can also achieve a complex decision boundary using polynomial processing as well. You'll get more insight from decision boundaries when you understand how logistic regression works.

- Fourth, you need to understand the impact of a feature.
 - You can select the best features based on the statistical significance of the logistic regression model coefficients or parameters.
 - That is, after finding the optimum parameters, a feature x with the weight θ_1 close to 0, has a smaller effect on the prediction, than features with large absolute values of θ_1 .
 - Indeed, it allows us to understand the impact an independent variable has on the dependent variable while controlling other independent variables.

| | X | | | | | | | | | y |
|---|----------|------|---------|--------|-----|--------|-------|----------|----------|----------|
| | tenure | age | address | income | ed | employ | equip | callcard | wireless | churn |
| 0 | 11.0 | 33.0 | 7.0 | 136.0 | 5.0 | 5.0 | 0.0 | 1.0 | 1.0 | Yes |
| 1 | 33.0 | 33.0 | 12.0 | 33.0 | 2.0 | 0.0 | 0.0 | 0.0 | 0.0 | Yes |
| 2 | 23.0 | 30.0 | 9.0 | 30.0 | 1.0 | 2.0 | 0.0 | 0.0 | 0.0 | No |
| 3 | 38.0 | 35.0 | 5.0 | 76.0 | 2.0 | 10.0 | 1.0 | 1.0 | 1.0 | No |

Example

- We define the independent variables as X , and dependent variable as Y .
- Notice that, for the sake of simplicity, we can code the target or dependent values to 0 or 1.
- The goal of logistic regression is to build a model to predict the class of each sample (which in this case is a customer) as well as the probability of each sample belonging to a class.
- Given that, let's start to formalize the problem.
- X is our dataset, in the space of real numbers of m by n , that is, of m dimensions or features and n records.



| | tenure | age | address | income | ed | employ | equip | callcard | wireless | churn |
|---|--------|------|---------|--------|-----|--------|-------|----------|----------|-------|
| 0 | 11.0 | 33.0 | 7.0 | 136.0 | 5.0 | 5.0 | 0.0 | 1.0 | 1.0 | 1.0 |
| 1 | 33.0 | 33.0 | 12.0 | 33.0 | 2.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| 2 | 23.0 | 30.0 | 9.0 | 30.0 | 1.0 | 2.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 3 | 38.0 | 35.0 | 5.0 | 76.0 | 2.0 | 10.0 | 1.0 | 1.0 | 1.0 | 0.0 |

$$X \in \mathbb{R}^{m \times n}$$
$$y \in \{0,1\}$$

And y is the class that we want to predict, which can be either zero or one.

Ideally, a logistic regression model, so called \hat{y} (y-hat), can predict that the class of a customer is 1, given its features x .

$$\hat{y} = P(y=1|x)$$

It can also be shown quite easily, that the probability of a customer being in class 0 can be calculated as 1 minus the probability that the class of the customer is 1.

$$\hat{y} = P(y=1|x)$$

$$P(y=0|x) = 1 - P(y=1|x)$$

Logistic Regression Vs Linear Regression

- We go over linear regression and see why it cannot be used properly for some binary classification problems. We will also look at the Sigmoid function, which is the main part of logistic regression.
- Let's look at the telecommunication dataset again.

| X | | | | | | | | | | y |
|---|--------|------|---------|--------|-----|--------|-------|----------|----------|-------|
| | tenure | age | address | income | ed | employ | equip | callcard | wireless | churn |
| 0 | 11.0 | 33.0 | 7.0 | 136.0 | 5.0 | 5.0 | 0.0 | 1.0 | 1.0 | Yes |
| 1 | 33.0 | 33.0 | 12.0 | 33.0 | 2.0 | 0.0 | 0.0 | 0.0 | 0.0 | Yes |
| 2 | 23.0 | 30.0 | 9.0 | 30.0 | 1.0 | 2.0 | 0.0 | 0.0 | 0.0 | No |
| 3 | 38.0 | 35.0 | 5.0 | 76.0 | 2.0 | 10.0 | 1.0 | 1.0 | 1.0 | No |
| 4 | 7.0 | 35.0 | 14.0 | 80.0 | 2.0 | 15.0 | 0.0 | 1.0 | 0.0 | No |

$$\hat{y} = P(y=1|x)$$

Logistic Regression Vs Linear Regression

- The goal of logistic regression is to build a model to predict the class of each customer, and also the probability of each sample belonging to a class.
- Ideally, we want to build a model, \hat{y} , that can estimate that the class of a customer is 1, given its features, x .
- I want to emphasize that y is the “labels vector,” also called “actual values” that we would like to predict, and \hat{y} is the vector of the predicted values by our model.
- Mapping the class labels to integer numbers, can we use linear regression to solve this problem?

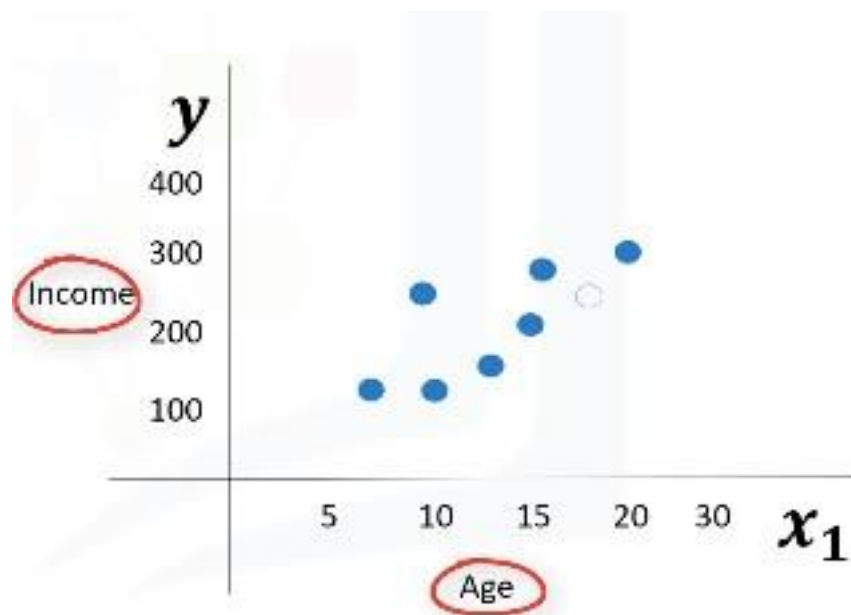
Logistic Regression Vs Linear Regression

- First, let's recall how linear regression works to better understand logistic regression.
- Forget about the churn prediction for a minute, and assume our goal is to predict the income of customers in the dataset.
- This means that instead of predicting churn, which is a categorical value, let's predict income, which is a continuous value.
- So, how can we do this?
- Let's select an independent variable, such as customer age, and predict a dependent variable, such as income.

| | tenure | age | address | income | ed | employ | equip | callicard | wireless | churn |
|---|--------|------|---------|--------|-----|--------|-------|-----------|----------|-------|
| 0 | 11.0 | 33.0 | 7.0 | 126.0 | 5.0 | 5.0 | 0.0 | 1.0 | 1.0 | 1 |
| 1 | 33.0 | 33.0 | 12.0 | 33.0 | 2.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1 |
| 2 | 23.0 | 30.0 | 9.0 | 30.0 | 1.0 | 2.0 | 0.0 | 0.0 | 0.0 | 0 |
| 3 | 29.0 | 35.0 | 5.0 | 76.0 | 2.0 | 10.0 | 1.0 | 1.0 | 1.0 | 0 |
| 4 | 7.0 | 35.0 | 14.0 | 80.0 | 2.0 | 15.0 | 0.0 | 1.0 | 0.0 | 0 |

Logistic Regression Vs Linear Regression

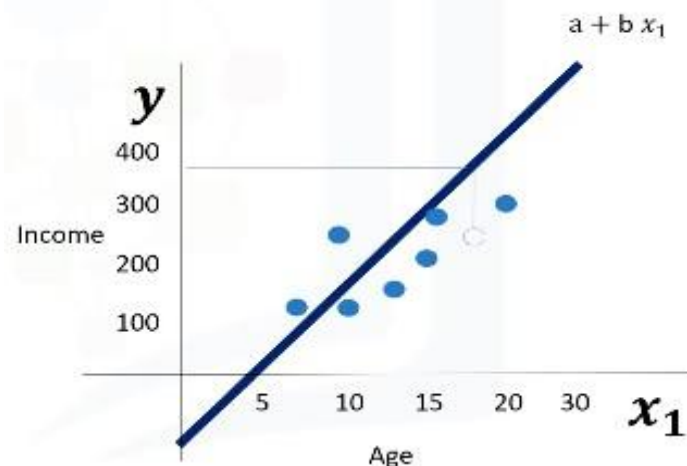
- Of course we can have more features, but for the sake of simplicity, let's just take one feature here.
- We can plot it, and show age as an independent variable, and income as the target value we would like to predict.



Logistic Regression Vs Linear Regression

With linear regression, you can fit a line or polynomial through the data.

We can find this line through the training of our model, or calculating it mathematically based on the sample sets.



We'll say this is a straight line through the sample set. This line has an equation shown as $a + b x_1$.

Now, use this line to predict the continuous value y , that is, use this line to predict the income of an unknown customer based on his or her age. And it is done.

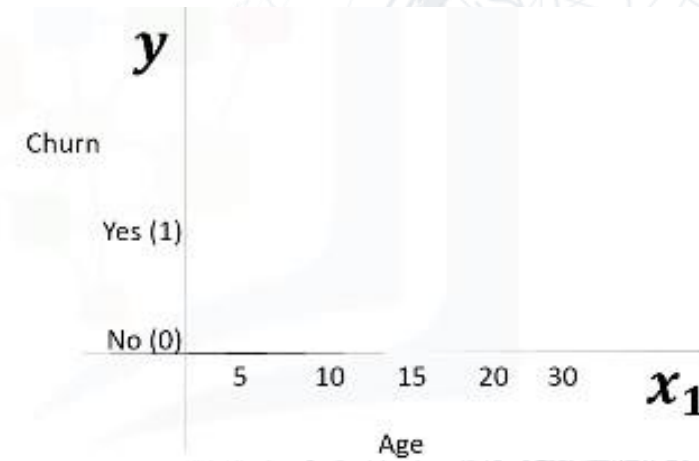
Logistic Regression Vs Linear Regression

What if we want to predict churn? Can we use the same technique to predict a categorical field such as churn? OK, let's see.

Say we're given data on customer churn and our goal this time is to predict the churn of customers based on their age.

We have a feature, age denoted as x_1 , and a categorical feature, churn, with two classes: churn is yes and churn is no.

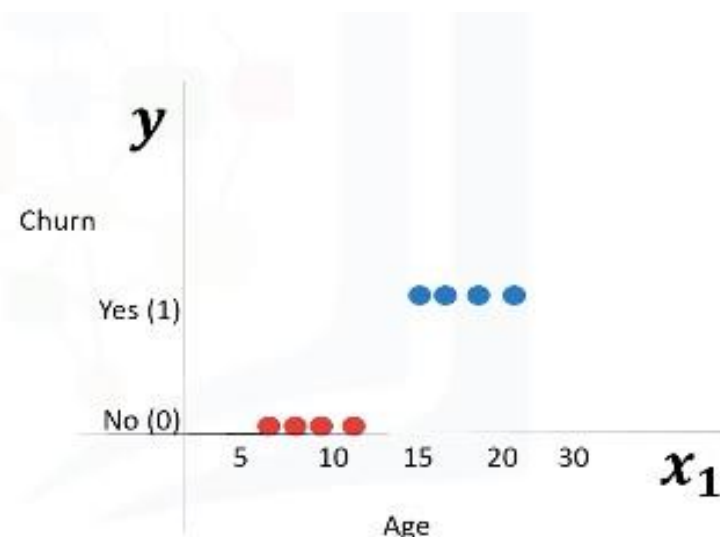
| | tenure | age | address | income | ed | employ | equip | callicard | wireless | churn |
|---|--------|------|---------|--------|-----|--------|-------|-----------|----------|-------|
| 0 | 11.0 | 33.0 | 7.0 | 136.0 | 5.0 | 5.0 | 0.0 | 1.0 | 1.0 | 1 |
| 1 | 33.0 | 33.0 | 12.0 | 33.0 | 2.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1 |
| 2 | 23.0 | 30.0 | 8.0 | 30.0 | 1.0 | 2.0 | 0.0 | 0.0 | 0.0 | 0 |
| 3 | 39.0 | 35.0 | 5.0 | 76.0 | 2.0 | 10.0 | 1.0 | 1.0 | 1.0 | 0 |
| 4 | 7.0 | 35.0 | 14.0 | 80.0 | 2.0 | 15.0 | 0.0 | 1.0 | 0.0 | 0 |



Logistic Regression Vs Linear Regression

As mentioned, we can map yes and no to integer values, 0 and 1.

How can we model it now? Well, graphically, we could represent our data with a scatter plot. But this time we have only 2 values for the y axis.



In this plot, class zero is denoted in red and class one is denoted in blue.

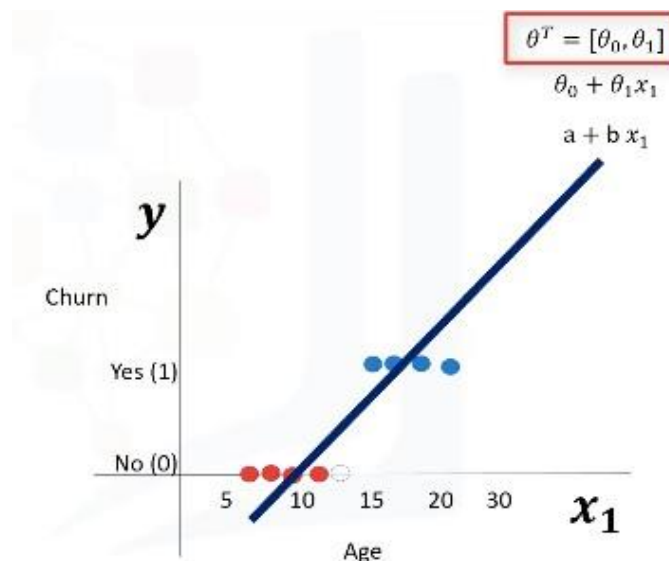
Our goal here is to make a model based on existing data, to predict if a new customer is red or blue.

Let's do the same technique that we used for linear regression here to see if we can solve the problem for a categorical attribute, such as churn.

Logistic Regression Vs Linear Regression

With linear regression, you again can fit a polynomial through the data, which is shown traditionally as $a + b \cdot x$.

This polynomial can also be shown traditionally as $\theta_0 + \theta_1 x_1$. This line has 2 parameters, which are shown with vector θ , where the values of the vector are θ_0 and θ_1 .



Logistic Regression Vs Linear Regression

We can also show the equation of this line formally as $\theta^T X$.

And generally, we can show the equation for a multi-dimensional space as $\theta^T X$, where θ is the parameters of the line in 2-dimensional space, or parameters of a plane in 3-dimensional space, and so on.

As θ is a vector of parameters, and is supposed to be multiplied by X , it is shown conventionally as $X^T \theta$. θ is also called the “weight vector” or “confidences of the equation” – with both of these terms used interchangeably.

$$\theta^T X = \theta_0 + \theta_1 x_1$$

$$\theta^T X = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots$$

$$\theta^T = [\theta_0, \theta_1, \theta_2, \dots]$$

Logistic Regression Vs Linear Regression

And X is the feature set, which represents a customer.

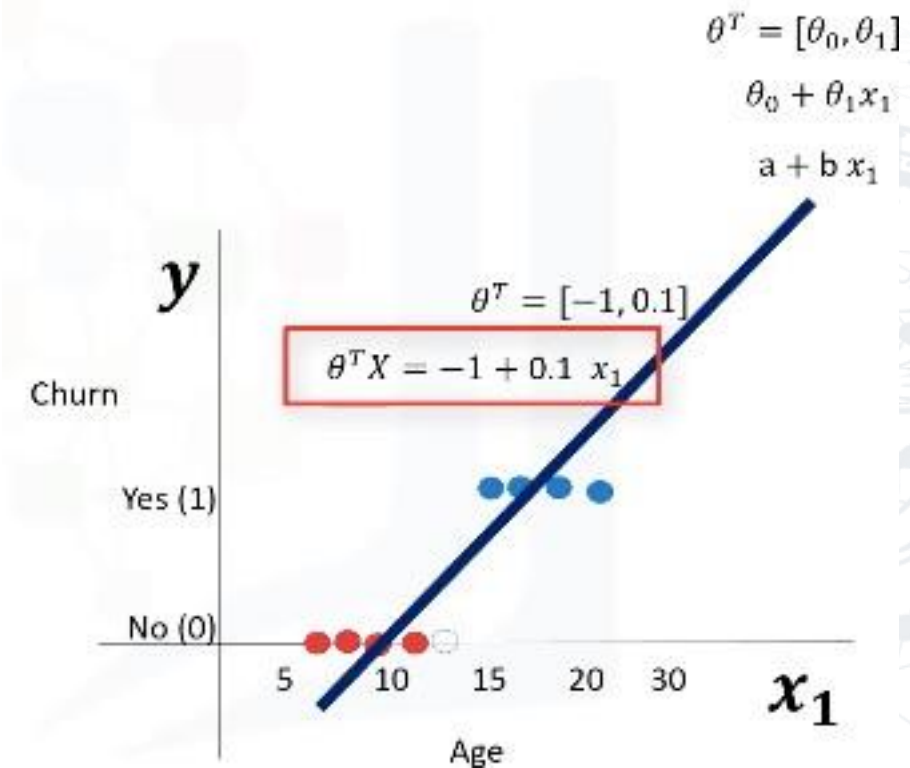
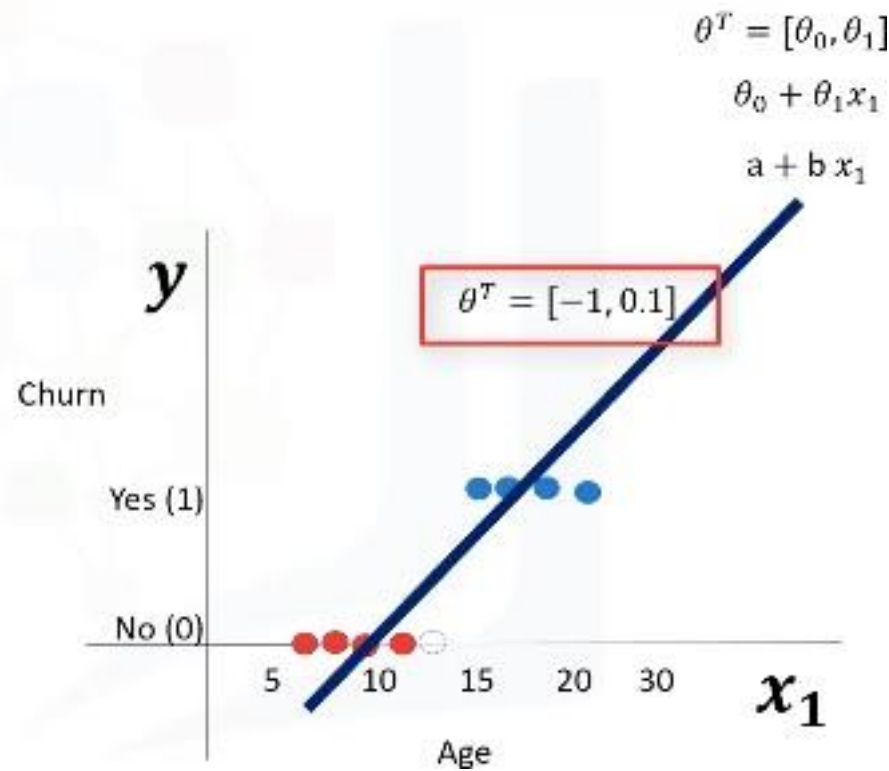
Anyway, given a dataset, all the feature sets X , θ parameters, can be calculated through an optimization algorithm or mathematically, which results in the equation of the fitting line.

$$X = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \dots \end{bmatrix}$$



Logistic Regression Vs Linear Regression

For example, the parameters of this line are -1 and 0.1 and the equation for the line is $-1 + 0.1 x_1$.



Logistic Regression Vs Linear Regression

- Now, we can use this regression line to predict the churn of the new customer.
- For example, for our customer, or let's say a data point with x value of age=13, we can plug the value into the line formula, and the y value is calculated and returns a number

$$\theta^T X = \theta_0 + \theta_1 x_1$$

$$p_1 = [13] \rightarrow \begin{aligned} \theta^T X &= -1 + 0.1 \cdot x_1 \\ &= -1 + 0.1 \times 13 \\ &= 0.3 \end{aligned}$$

Logistic Regression Vs Linear Regression

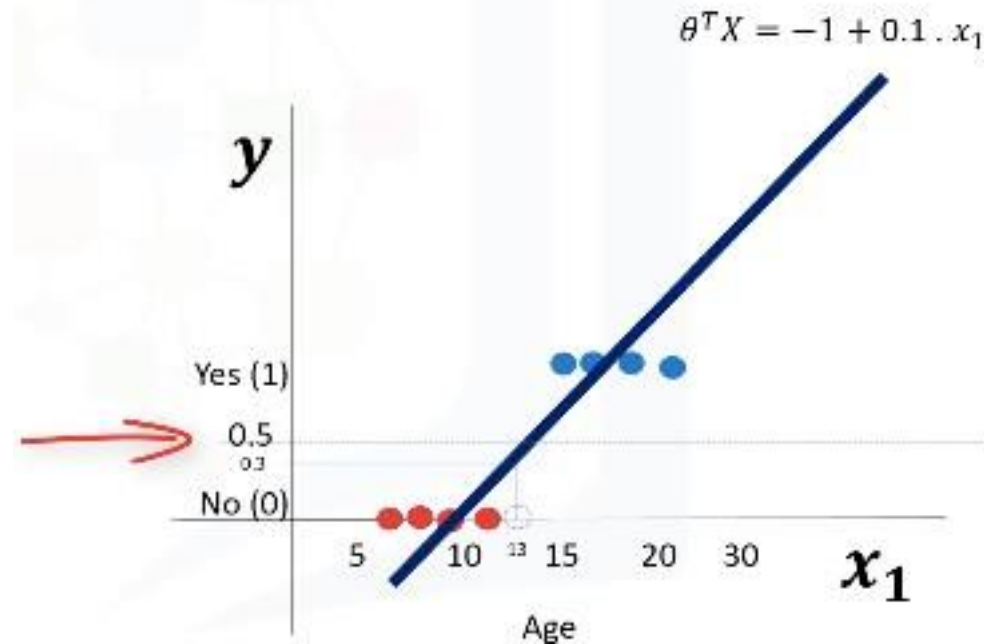
For instance, for p_1 point we have: $\theta^T X = -1 + 0.1 \times x_1 = -1 + 0.1 \times 13 = 0.3$ We can show it on our graph.

Now we can define a threshold here, for example at 0.5, to define the class.

$$\theta^T X = \theta_0 + \theta_1 x_1$$

$$p_1 = [13] \rightarrow$$

$$\begin{aligned} \theta^T X &= -1 + 0.1 \cdot x_1 \\ &= -1 + 0.1 \times 13 \\ &= 0.3 \end{aligned}$$



Logistic Regression Vs Linear Regression

So, we write a rule here for our model, \hat{y} , which allows us to separate class 0 from class 1.

If the value of $\theta^T X$ is less than 0.5, then the class is 0, otherwise, if the value of $\theta^T X$ is more than 0.5, then the class is 1.

And because our customer's y value is less than the threshold, we can say it belongs to class 0, based on our model.

$$\theta^T X = \theta_0 + \theta_1 x_1$$

$$p_1 = [13] \rightarrow \begin{aligned} \theta^T X &= -1 + 0.1 \cdot x_1 \\ &= -1 + 0.1 \times 13 \\ &= 0.3 \end{aligned}$$

$$\hat{y} = \begin{cases} 0 & \text{if } \theta^T X < 0.5 \\ 1 & \text{if } \theta^T X \geq 0.5 \end{cases}$$

$$\begin{aligned} \theta^T X &= 0.3 \\ \theta^T X &< 0.5 & \rightarrow & \text{Class 0} \end{aligned}$$

Logistic Regression Vs Linear Regression

- But there is one problem here; what is the probability that this customer belongs to class 0? As you can see, it's not the best model to solve this problem.
- Also, there are some other issues, which verify that linear regression is not the proper method for classification problems.

Logistic Regression Vs Linear Regression

So, as mentioned, if we use the regression line to calculate the class of a point, it always returns a number, such as 3, or -2, and so on.

$$\theta^T X = \theta_0 + \theta_1 x_1 + \dots$$

Then, we should use a threshold, for example, 0.5, to assign that point to either class of 0 or 1.

This threshold works as a step function that outputs 0 or 1, regardless of how big or small, positive or negative, the input is.

Logistic Regression Vs Linear Regression

So, using the threshold, we can find the class of a record.

Notice that in the step function, no matter how big the value is, as long as it's greater than 0.5, it simply equals 1.

And vice versa, regardless of how small the value y is, the output would be zero if it is less than 0.5. In other words, there is no difference between a customer who has a value of one or 1000; the outcome would be 1.

$$\hat{y} = \begin{cases} 0 & \text{if } \theta^T X < 0.5 \\ 1 & \text{if } \theta^T X \geq 0.5 \end{cases}$$

Logistic Regression Vs Linear Regression

Instead of having this step function, wouldn't it be nice if we had a smoother line – one that would project these values between zero and one?

Indeed, the existing method does not really give us the probability of a customer belonging to a class, which is very desirable.

We need a method that can give us the probability of falling in a class as well.

So, what is the scientific solution here?

Well, if instead of using $\theta^T X$

$$\theta^T X = \theta_0 + \theta_1 x_1 + \dots$$

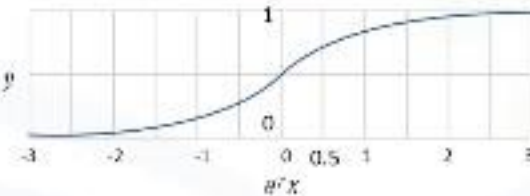
we use a specific function called sigmoid, then, sigmoid of $\theta^T X$ gives us the probability of a point belonging to a class, instead of the value of y directly.

$$\sigma(\theta^T X) = \sigma(\theta_0 + \theta_1 x_1 + \dots)$$

Logistic Regression Vs Linear Regression

Sigmoid will be explained in a second, but for now, please accept that it will do the trick. Instead of calculating the value of $\theta^T X$ directly, it returns the probability that a $\theta^T X$ is very big or very small. It always returns a value between 0 and 1 depending on how large the $\theta^T X$ actually is. Now, our model is $\sigma(\theta^T X)$, which represents the probability that the output is 1, given x .

$$\sigma(\theta^T X) = \sigma(\theta_0 + \theta_1 x_1 + \dots)$$



$$\hat{y} = \sigma(\theta^T X)$$

$$P(y=1|x)$$

Sigmoid function

Now the question is, “What is the sigmoid function?”

- Let's look in detail what sigmoid really is.
- The sigmoid function, also called the logistic function, resembles the step function and is used by the following expression in the logistic regression.
- The sigmoid function looks a bit complicated at first, but don't worry about remembering this equation. It'll make sense to you after working with it.

$$\sigma(\theta^T X) = \frac{1}{1 + e^{-\theta^T X}}$$

Sigmoid

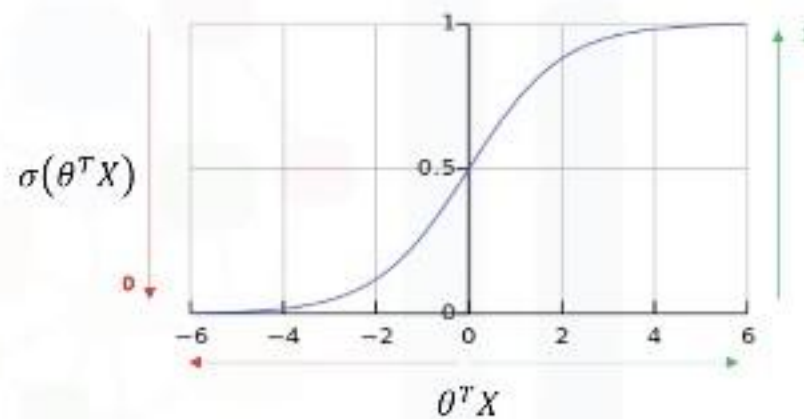
Notice that in the sigmoid equation, when $\theta^T X$ gets very big, the $e^{-(\theta^T X)}$ in the denominator of the fraction becomes almost zero, and the value of the sigmoid function gets closer to 1.

If $\theta^T X$ is very small, the sigmoid function gets closer to zero.

$$\sigma(\theta^T X) = \frac{1}{1 + e^{-\theta^T X}}$$

$$\sigma(\theta^T X) = 1$$

$$\sigma(\theta^T X) = 0$$



Sigmoid



UNIVERSITAS
INDONESIA
Veritas, Probatio, Justitia

Depicting on the sigmoid plot, when $\theta^T X$, gets bigger, the value of the sigmoid function gets closer to 1, and also,

if the $\theta^T X$ is very small, the sigmoid function gets closer to zero.

So, the sigmoid function's output is always between 0 and 1, which makes it proper to interpret the results as probabilities.

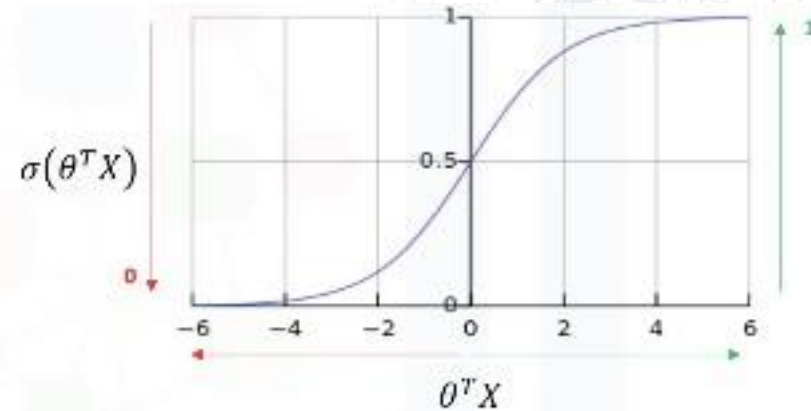
It is obvious that when the outcome of the sigmoid function gets closer to 1, the probability of $y=1$, given x , goes up, and in contrast, when the sigmoid value is closer to zero, the probability of $y=1$, given x , is very small.

$$\sigma(\theta^T X) = \frac{1}{1 + e^{-\theta^T X}}$$

$$\sigma(\theta^T X) = 1$$

$$\sigma(\theta^T X) = 0$$

$[0, 1]$



$P(y=1|x)$



$P(y=1|x)$

Sigmoid

So what is the output of our model when we use the sigmoid function?

In logistic regression, we model the probability that an input (X) belongs to the default class ($Y=1$), and we can write this formally as, $P(Y=1|X)$.

We can also write $P(y=0|X) = 1 - P(y=1|x)$.

What is the output of our model?

- $P(Y=1|X)$
- $P(y=0|X) = 1 - P(y=1|x)$

For example, the probability of a customer staying with the company can be shown as probability of churn equals 1 given a customer's income and age, which can be, for instance, 0.8.

And the probability of churn is 0, for the same customer, given a customer's income and age can be calculated as $1-0.8=0.2$.

- $P(\text{Churn}=1|\text{income,age}) = 0.8$
- $P(\text{Churn}=0|\text{income,age}) = 1 - 0.8 = 0.2$

Sigmoid

So, now, our job is to train the model to set its parameter values in such a way that our model is a good estimate of $P(y=1|x)$.

In fact, this is what a good classifier model built by logistic regression is supposed to do for us.

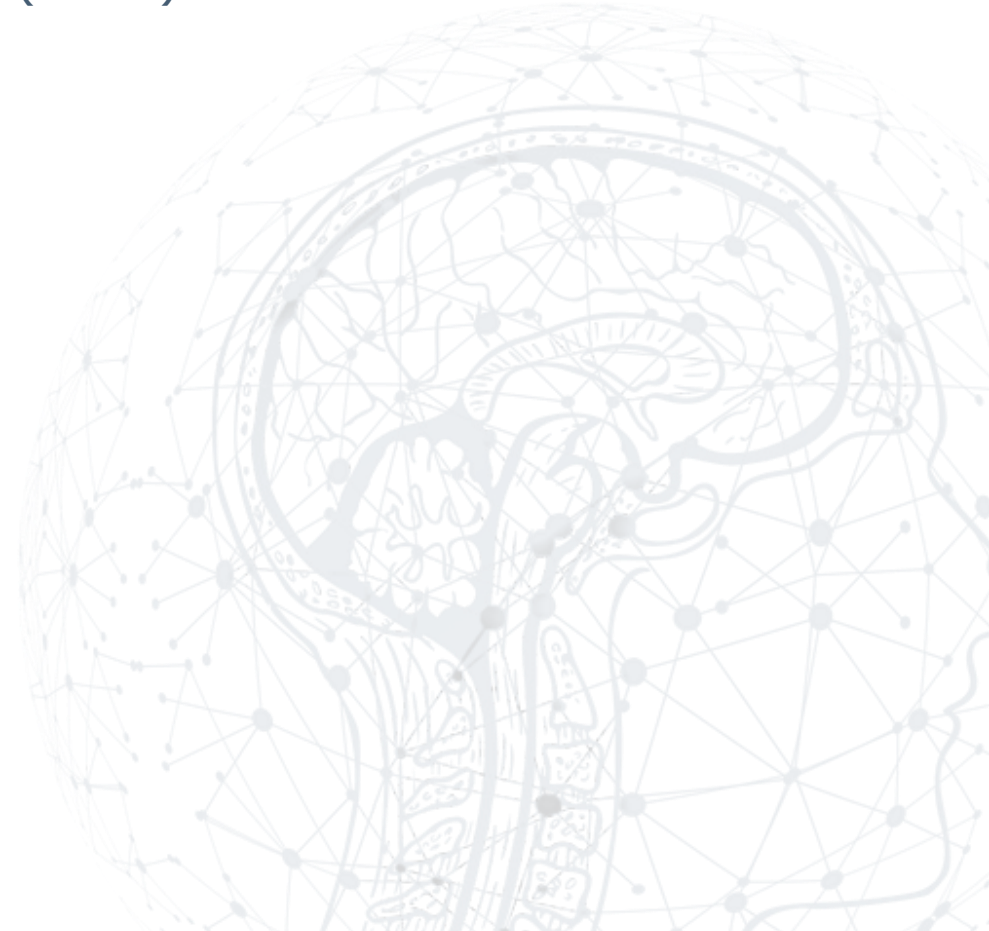
Also, it should be a good estimate of $P(y=0|x)$ that can be shown as $1-\sigma(\theta^T X)$.

$$\sigma(\theta^T X) \longrightarrow P(y=1|x)$$

$$1 - \sigma(\theta^T X) \longrightarrow P(y=0|x)$$



UNIVERSITAS
INDONESIA
Veritas, Probatio, Justitia



Sigmoid

Now, the question is: "How we can achieve this?" We can find θ through the training process, so let's see what the training process is.

Step 1. Initialize θ vector with random values, as with most machine learning algorithms, for example -1 or 2 .

$$\sigma(\theta^T X) \rightarrow P(y=1|x)$$

$$\theta = [-1, 2]$$

Step 2. Calculate the model output, which is $\sigma(\theta^T X)$, for a sample customer in your training set.

X in $\theta^T X$ is the feature vector values – for example, the age and income of the customer, for instance $[2, 5]$.

And θ is the confidence or weight that you've set in the previous step. The output of this equation is the prediction value ... in other words, the probability that the customer belongs to class 1.

Calculate $\hat{y} = \sigma(\theta^T X)$ for a customer.

$$\hat{y} = \sigma([-1, 2] \times [2, 5]) = 0.7$$

Sigmoid

Step 3. Compare the output of our model, \hat{y} , which could be a value of, let's say, 0.7, with the actual label of the customer, which is for example, 1 for churn.

Then, record the difference as our model's error for this customer, which would be $1 - 0.7$, which of course equals 0.3. This is the error for only one customer out of all the customers in the training set.

Compare the output of \hat{y} with actual output of customer, y , and record it as error.

$$\text{Error} = 1 - 0.7 = 0.3$$

Step. 4. Calculate the error for all customers as we did in the previous steps, and add up these errors.

The total error is the cost of your model, and is calculated by the model's cost function.

The cost function, by the way, basically represents how to calculate the error of the model, which is the difference between the actual and the model's predicted values.

So, the cost shows how poorly the model is estimating the customer's labels. Therefore, the lower the cost, the better the model is at estimating the customer's labels correctly. And so, what we want to do is to try to minimize this cost.

$$\text{Cost} = J(\theta)$$

Sigmoid

Step 5. But, because the initial values for θ were chosen randomly, it's very likely that the cost function is very high.

So, we change the θ in such a way to hopefully reduce the total cost.

Step 6. After changing the values of θ , we go back to step 2.

Then we start another iteration, and calculate the cost of the model again.

The training process

1. Initialize θ .
2. Calculate $\hat{y} = \sigma(\theta^T X)$ for a customer.
3. Compare the output of \hat{y} with actual output of customer, y , and record it as error.
4. Calculate the error for all customers.
5. Change the θ to reduce the cost.
6. Go back to step 2.

$$\sigma(\theta^T X) \rightarrow P(y=1|x)$$

$$\theta = [-1, 2]$$

$$\hat{y} = \sigma([-1, 2] \times [2, 5]) = 0.7$$

$$\text{Error} = 1 - 0.7 = 0.3$$

$$\text{Cost} = J(\theta)$$

$$\theta_{\text{new}}$$



UNIVERSITAS
INDONESIA
Veritas, Probatum, Institutum

Sigmoid

- And we keep doing those steps over and over, changing the values of θ each time, until the cost is low enough. So, this brings up two questions:
- first, "How can we change the values of θ so that the cost is reduced across iterations?" second, "When should we stop the iterations?"
- There are different ways to change the values of θ , but one of the most popular ways is gradient descent.
- Also, there are various ways to stop iterations, but essentially you stop training by calculating the accuracy of your model and stop it when it's satisfactory.

Support Vector Machine (SVM)

Support Vector Machines

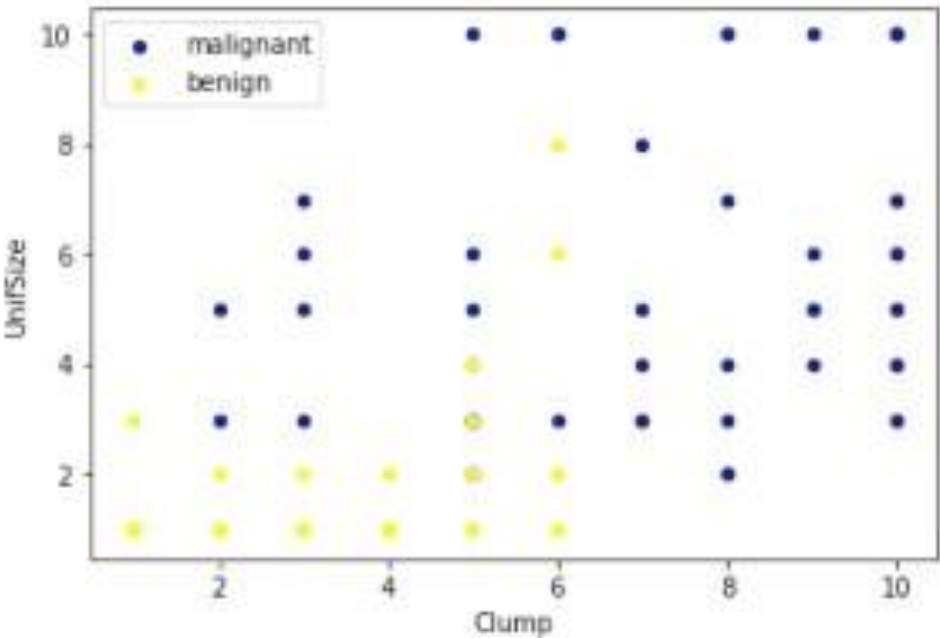
- A Support Vector Machine is a supervised algorithm that can classify cases **by finding a separator**.
- SVM works by:
 1. First, **mapping data to a high-dimensional feature space** so that data points can be categorized, even when the data are not otherwise linearly separable.
 2. Then, a **separator is estimated** for the data.
- The data should be transformed in such a way that a separator could be drawn as a hyperplane.

Imagine that you've obtained a dataset containing characteristics of thousands of human cell samples extracted from patients who were believed to be at risk of developing cancer.

Analysis of the original data showed that many of the characteristics differed significantly between **benign and malignant** samples.

For example, consider the following figure, which shows the distribution of a small set of cells, only based on their Unit Size and Clump thickness.

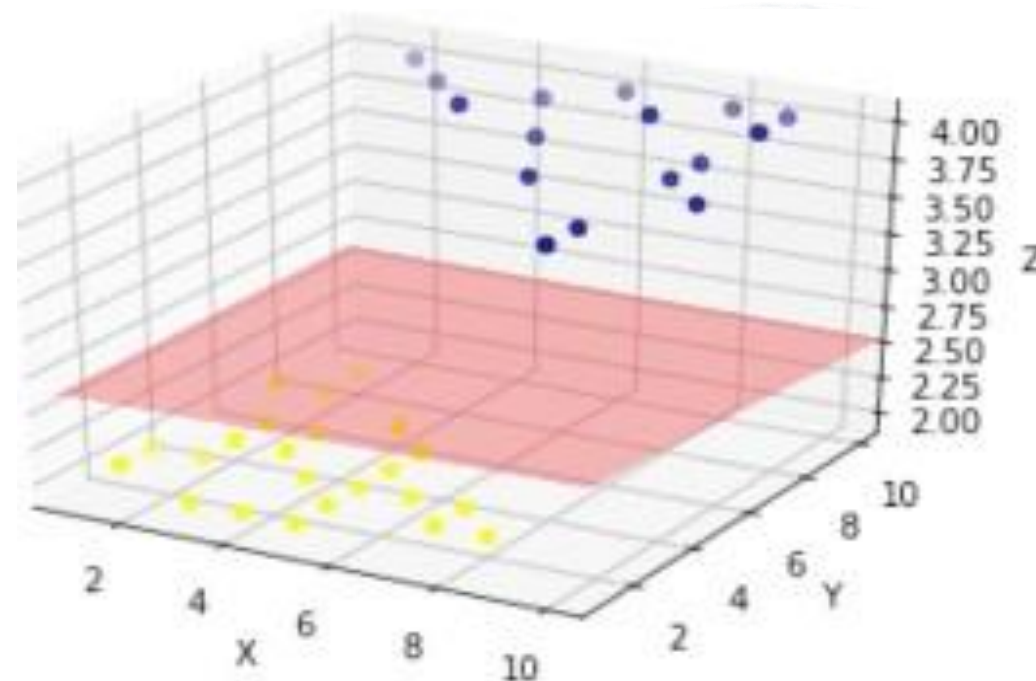
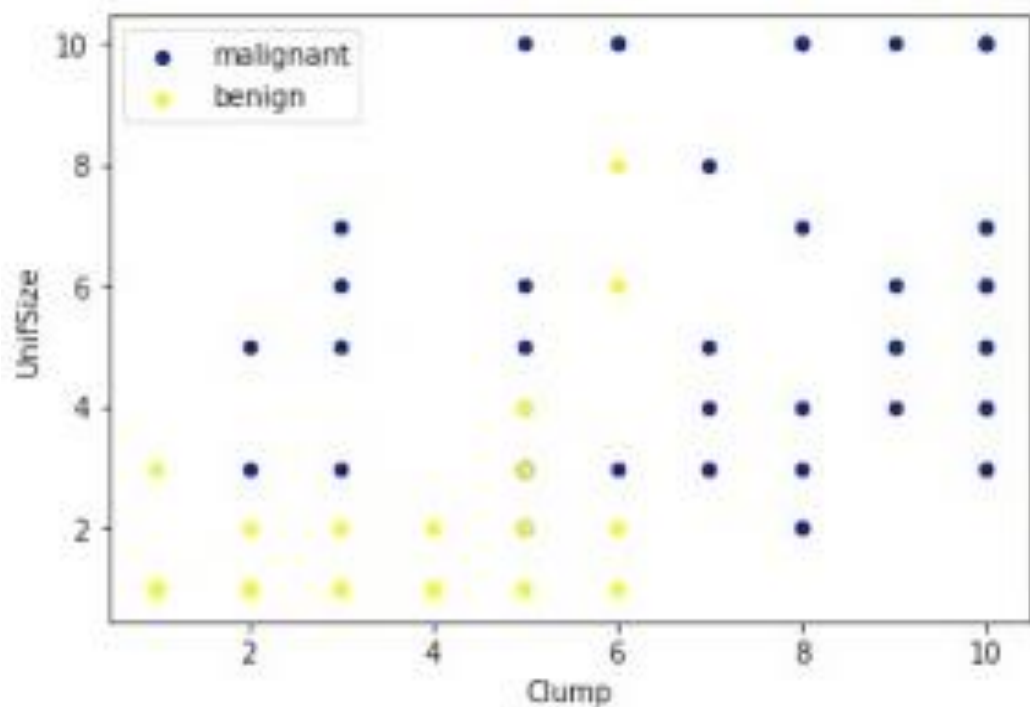
| Clump | UnifSize | UnifShape | MargAdh | SingEpiSize | BareNuc | BlandChrom | NormNucl | Mit | Class |
|-------|----------|-----------|---------|-------------|---------|------------|----------|-----|-----------|
| 5 | 1 | 1 | 1 | 2 | 1 | 3 | 1 | 1 | benign |
| 5 | 4 | 4 | 5 | 7 | 10 | 3 | 2 | 1 | benign |
| 3 | 1 | 1 | 1 | 2 | 2 | 3 | 1 | 1 | malignant |
| 6 | 8 | 8 | 1 | 3 | 4 | 3 | 7 | 1 | benign |
| 4 | 1 | 1 | 3 | 2 | 1 | 3 | 1 | 1 | benign |
| 8 | 10 | 10 | 8 | 7 | 10 | | 7 | 1 | malignant |
| 1 | 1 | 1 | 1 | 2 | 10 | 3 | 1 | 1 | benign |
| 2 | 1 | 2 | H | 2 | 1 | 3 | 1 | 1 | benign |
| 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 5 | benign |
| 4 | 2 | 1 | 1 | 2 | 1 | 2 | 1 | 1 | benign |



As you can see, the data points fall into two different categories. It represents a **linearly, non-separable**, dataset.

The two categories can be separated with a curve, but not a line.

We can **transfer this data to a higher dimensional space** ... for example, mapping it to a 3-dimensional space.

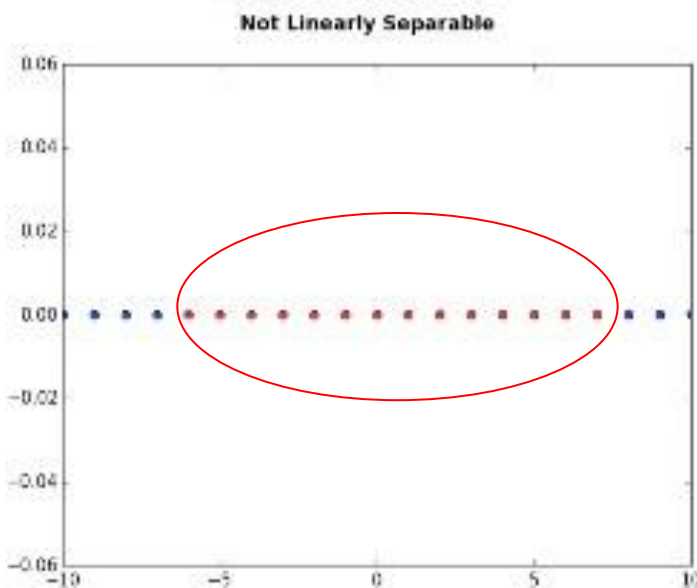


Challenging questions

- How do we transfer data in such a way that a separator could be drawn as a hyperplane?
- How can we find the best/optimized hyperplane separator after transformation?

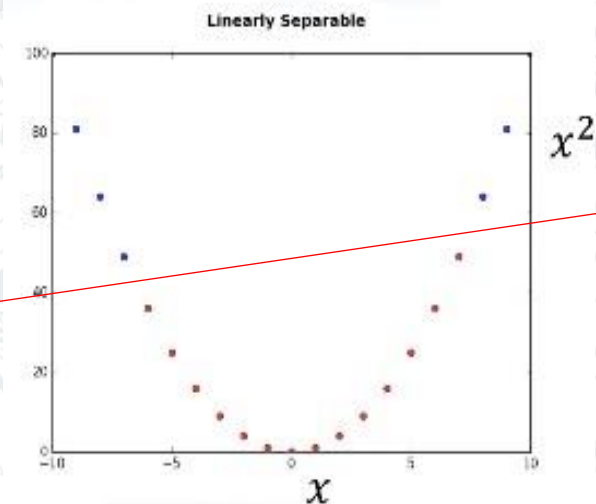
Transforming data

- Imagine that our dataset is 1-dimensional data, this means, we have only one feature x .
- Transfer it into a 2-dimensional space, e.g. mapping x into a new space using a function, with outputs x and x -squared



© IBM 2020

$$\phi(x) = [x, x^2]$$

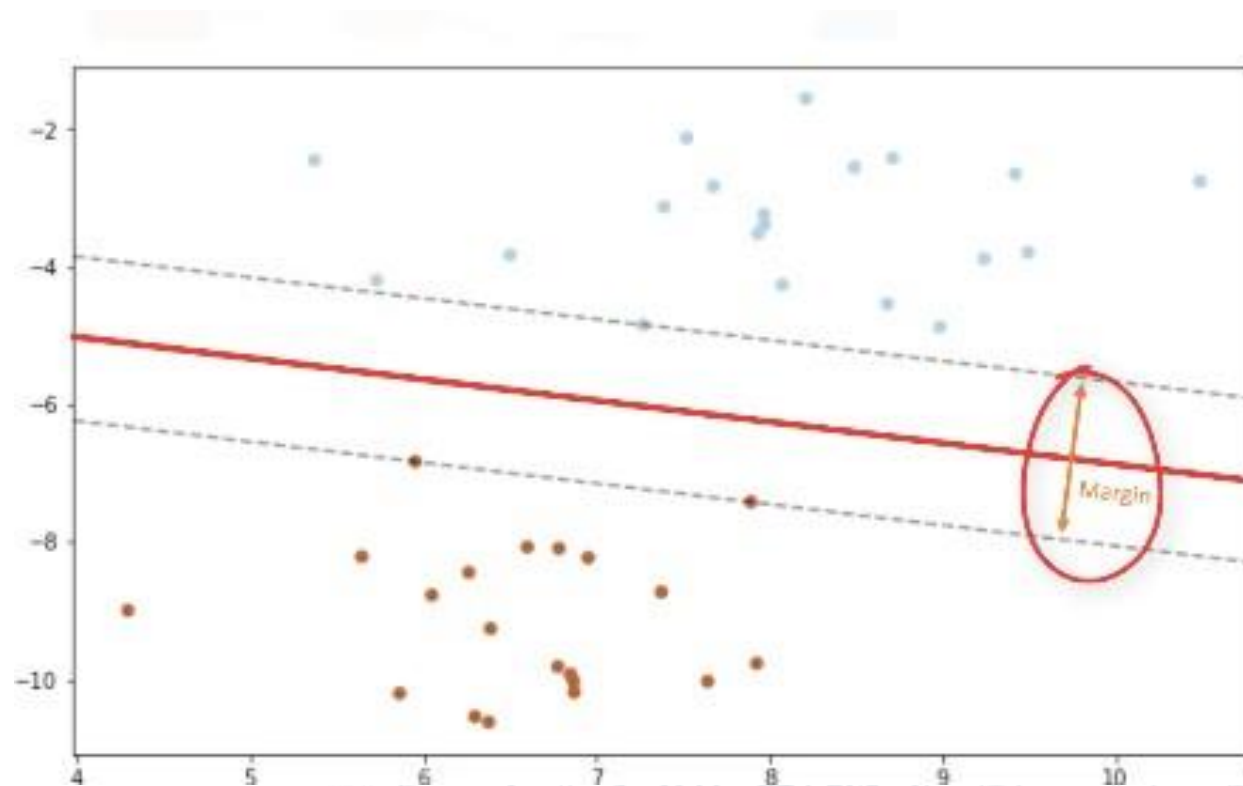


SVM Kernel

- Basically, mapping data into a higher dimensional space is called kernelling.
- The mathematical function used for the transformation is known as the kernel function, and can be of different types, such as:
 - Linear, Polynomial, Radial basis function (or RBF), and Sigmoid.
- Each of these functions has its own characteristics, its pros and cons, and its equation, but the good news is that you don't need to know them, as most of them are already implemented in libraries of data science programming languages.
- Also, as there's no easy way of knowing which function performs best with any given dataset, we usually choose different functions in turn and compare the results.

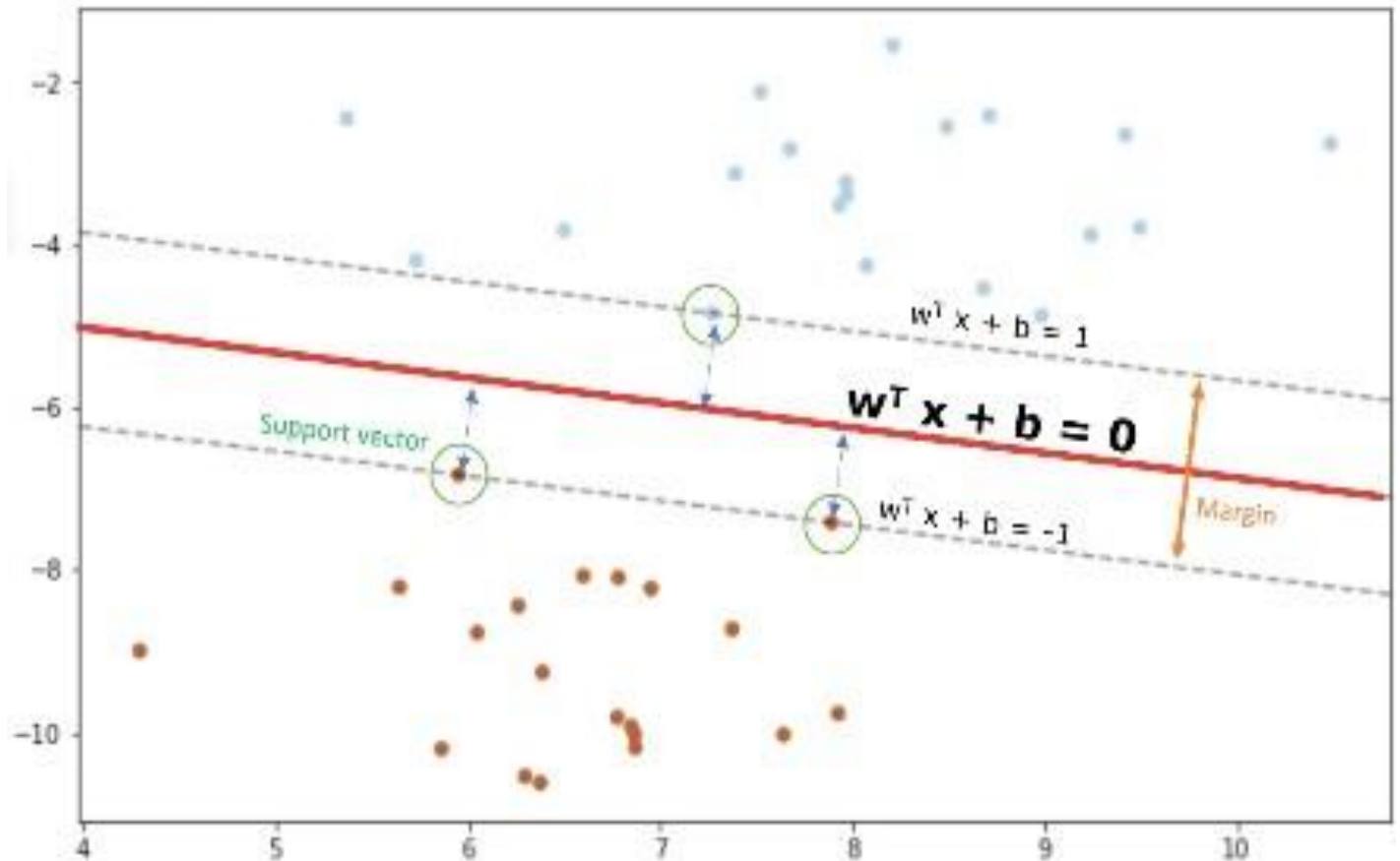
Hyperplane

- Basically, SVMs are based on the idea of finding a hyperplane that best divides a dataset into two classes, as shown here.
- One reasonable choice as the best hyperplane is **the one that represents the largest separation, or margin, between the two classes.**



So, finding the optimized hyperplane can be formalized using an equation which involves quite a bit more math,

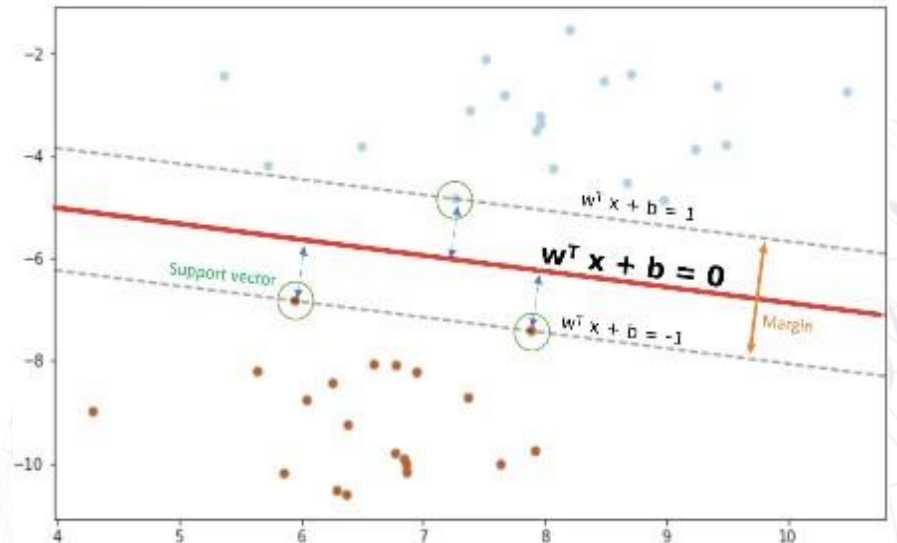
Find \mathbf{w} and b such that
 $\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$ is minimized;
and for all $\{(\mathbf{x}_i, y_i)\}: y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$





Support Vector Machines

- That said, the hyperplane is learned from training data using an optimization procedure that maximizes the margin;
- and like many other problems, this optimization problem can also be solved by gradient descent.
- Therefore, the output of the algorithm is the values 'w' and 'b' for the line. You can make classifications using this estimated line.
- It is enough to plug in input values into the line equation, then, you can calculate whether an unknown point is above or below the line.
- If the equation returns a value greater than 0, then the point belongs to the first class, which is above the line, and vice versa.



Support Vector Machines

- The two main advantages of support vector machines are that they're accurate in high dimensional spaces; and, they use a subset of training points in the decision function (called support vectors), so it's also memory efficient.
- The disadvantages of support vector machines include the fact that the algorithm is prone for over-fitting, if the number of features is much greater than the number of samples.
- Also, SVMs do not directly provide probability estimates, which are desirable in most classification problems.
- And finally, SVMs are not very efficient computationally, if your dataset is very big, such as when you have more than one thousand rows.

- **Advantages:**
 - Accurate in high-dimensional spaces
 - Memory efficient
- **Disadvantages:**
 - Prone to over-fitting
 - No probability estimation
 - Small datasets

Reference

- IBM – Cognitiveclass

