LECTURE 17

Classification

Building models of classification in sklearn

Data Science, Spring 2024 @ Knowledge Stream

Sana Jabbar

Outline

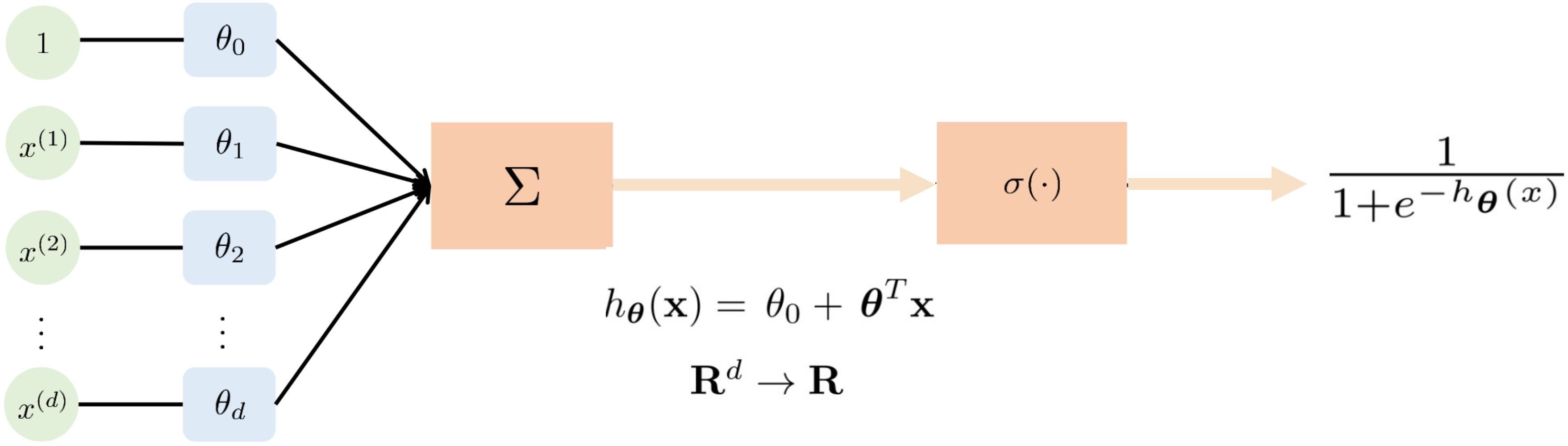
Lecture 17

- Introduction to Classification
- Types of Classification
- Classification Algorithms
- Performance Metrics
- Overfitting and Underfitting
- Conclusion

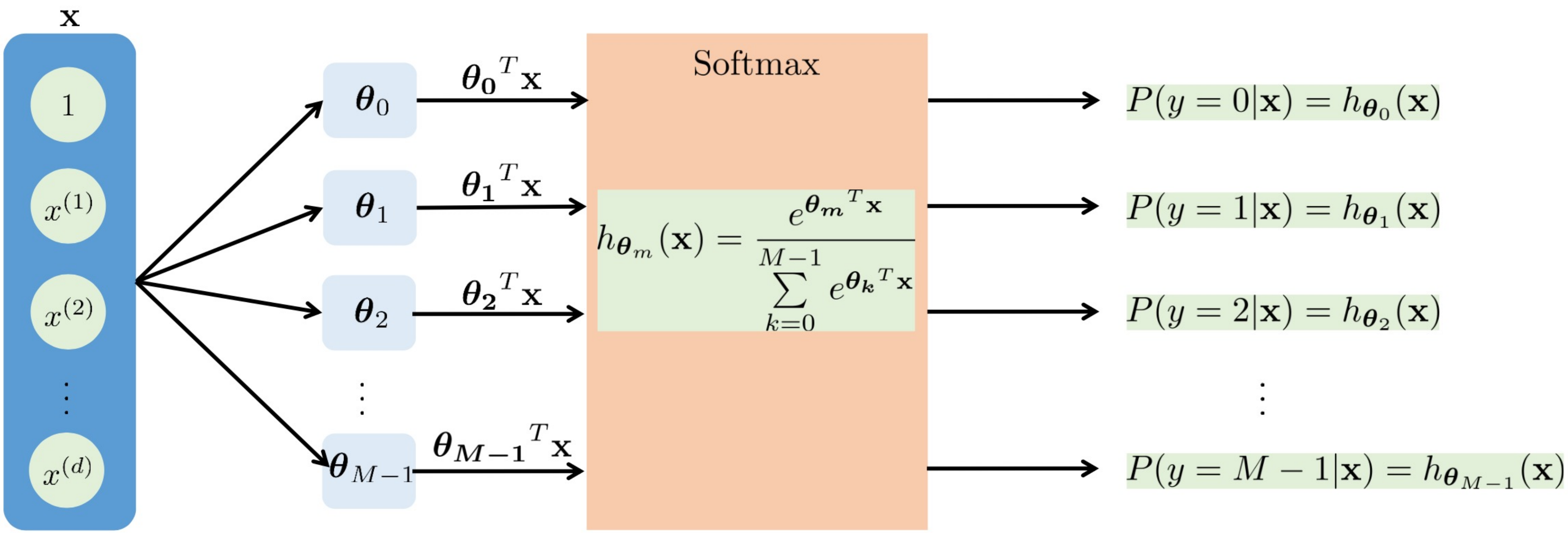
Classification

- Classification is defined as the process of recognition and grouping of objects
- Classification refers to a predictive modeling problem where a class label is predicted for a given example of input data
- For Classification, the training dataset must be sufficiently representative of the problem and have many examples of each class label.
    1. Logistic Regression
    2. Decision Trees
    3. Naive Bayes
    4. K-Nearest Neighbours
    5. Support Vector Machine
    6. Random Forest

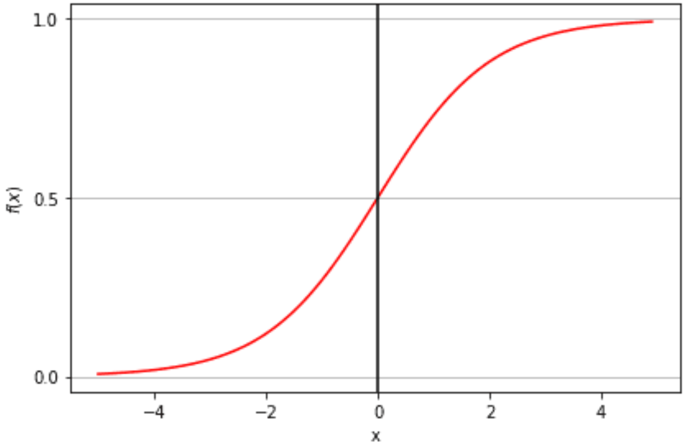
Logistic Regression



Logistic Regression

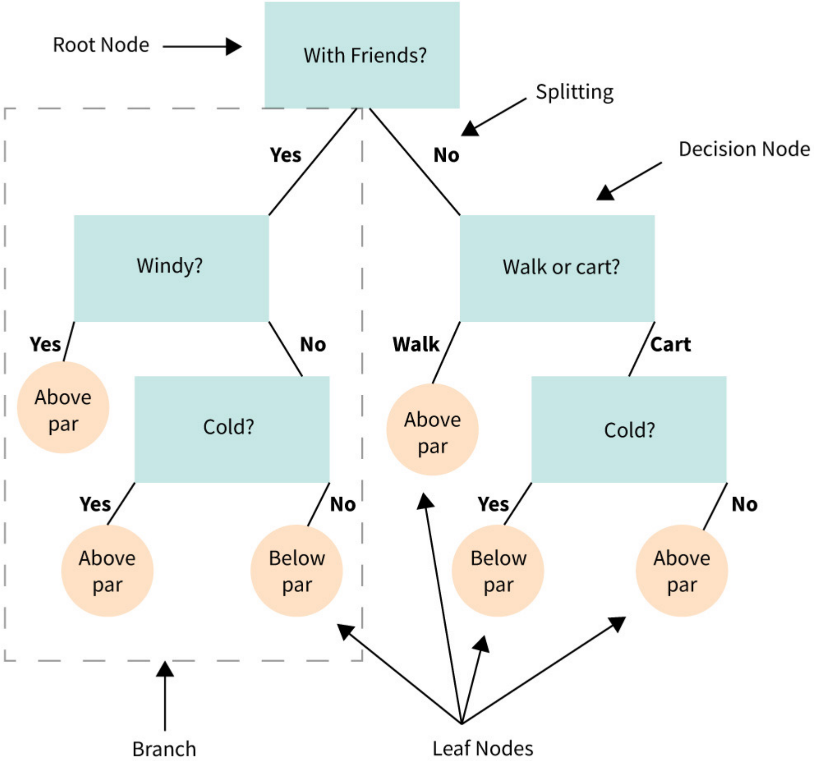


```
X = df['data']
Y = df['target']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=16)

from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(random_state=16)
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)

from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
```



Decision Trees (DTs)

- A hierarchical, tree structure, consists of a root node, branches, internal nodes and leaf nodes
- Algorithms for binary classification
    1. Logistic Regression
    2. Decision Trees
    3. Naive Bayes
    4. k-Nearest Neighbours
    5. Support Vector Machine
    6. Random Forest

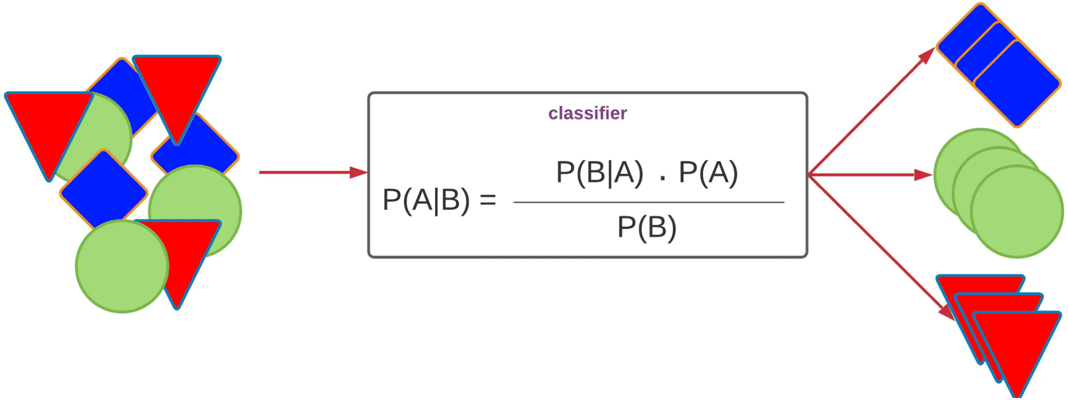


Decision Tree

```
X = df['data']
Y = df['target']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=16)

from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier ()
dt.fit(X_train, y_train)
y_pred = dt.predict(X_test)

from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
```

Naive Bayes

- The Naïve Bayes classifier is used for classification tasks, like text classification.
- Algorithms for binary classification
    1. Logistic Regression
    2. Decision Trees
    3. Naive Bayes
    4. k-Nearest Neighbours
    5. Support Vector Machine
    6. Random Forest



Naive Bayes

- The Naive Bayes classifier is a family of probabilistic machine learning models based on Bayes' theorem.
- The "naive" assumption is of feature independence.
- It's commonly used for text classification tasks, particularly for document categorization and spam email filtering.
- Estimate $P(y|\boldsymbol{X})$ from the data using the Bayes Theorem

$$P(y \mid \mathbf{x}) = \underset{y \in \mathcal{Y}}{\text{maximize}} \quad \frac{P(\mathbf{x} \mid y)\, P(y)}{P(\mathbf{x})}$$

Naive Bayes

Given Outlook, Temperature, Humidity, and Wind Information, we want to carry out a prediction for Play: Yes or No.

Mathematically

$$P(\text{Play} = \text{Yes} \mid \text{Outlook},\ \text{Temp.},\ \text{Humidity},\ \text{Wind})$$

$$P(\text{Play} = \text{No} \mid \text{Outlook},\ \text{Temp.},\ \text{Humidity},\ \text{Wind})$$

Predict for sunny outlook, humidity high, cool temp, and weak wind.

Day	Outlook	Temp.	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Naive Bayes

$$P(\text{Play} = \text{Yes} \mid \text{Outlook} = \text{Sunny},\ \text{Temp} = \text{Cool},\ \text{Humidity} = \text{High},\ \text{Wind} = \text{Weak})$$

$$= \frac{P(\text{Outlook} = \text{Sunny},\ \text{Temp} = \text{Cool},\ \text{Humidity} = \text{High},\ \text{Wind} = \text{Strong} \mid \text{Play} = \text{Yes})\, P(\text{Play} = \text{Yes})}{P(\text{Outlook} = \text{Sunny},\ \text{Temp} = \text{Cool},\ \text{Humidity} = \text{High},\ \text{Wind} = \text{Strong})}$$

- Feature are mutually independent given the label!

$$P(\text{Outlook} = \text{Sunny},\ \text{Temp} = \text{Cool},\ \text{Humidity} = \text{High},\ \text{Wind} = \text{Strong} \mid \text{Play} = \text{Yes})$$

$$= P(\text{Outlook} = \text{Sunny} \mid \text{Play} = \text{Yes})\ P(\text{Temp} = \text{Cool} \mid \text{Play} = \text{Yes})$$
$$P(\text{Humidity} = \text{High} \mid \text{Play} = \text{Yes})\, P(\text{Wind} = \text{Strong} \mid \text{Play} = \text{Yes})$$

Example:

$$P(\text{Outlook} = \text{Sunny} \mid \text{Play} = \text{Yes}) = \frac{2}{9}$$

$$P(\text{Temp} = \text{Cool} \mid \text{Play} = \text{Yes}) = \frac{3}{9}$$

$$P(\text{Humidity} = \text{High} \mid \text{Play} = \text{Yes}) = \frac{3}{9}$$

$$P(\text{Wind} = \text{Strong} \mid \text{Play} = \text{Yes}) = \frac{3}{9}$$

$$P(\text{Play} = \text{Yes}) = \frac{9}{14}$$

$$P(\text{Outlook} = \text{Sunny} \mid \text{Play} = \text{No}) = \frac{3}{5}$$

$$P(\text{Temp} = \text{Cool} \mid \text{Play} = \text{No}) = \frac{1}{5}$$

$$P(\text{Humidity} = \text{High} \mid \text{Play} = \text{No}) = \frac{4}{5}$$

$$P(\text{Wind} = \text{Strong} \mid \text{Play} = \text{No}) = \frac{3}{5}$$

$$P(\text{Play} = \text{No}) = \frac{5}{14}$$

Day	Outlook	Temp.	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Naive Bayes

$$P(\text{Outlook} = \text{Sunny} \mid \text{Play} = \text{Yes})\, P(\text{Temp} = \text{Cool} \mid \text{Play} = \text{Yes})\, P(\text{Humidity} = \text{High} \mid \text{Play} = \text{Yes})\, P(\text{Wind} = \text{Strong} \mid \text{Play} = \text{Yes})$$

$$\times P(\text{Play} = \text{Yes}) \qquad = \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} \qquad = 0.0053$$

$$P(\text{Outlook} = \text{Sunny} \mid \text{Play} = \text{No})\, P(\text{Temp} = \text{Cool} \mid \text{Play} = \text{No})\, P(\text{Humidity} = \text{High} \mid \text{Play} = \text{No})\, P(\text{Wind} = \text{Strong} \mid \text{Play} = \text{No})$$

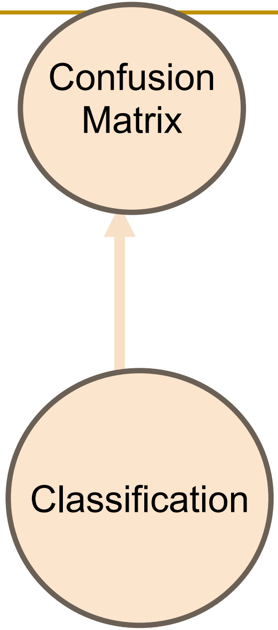
$$\times P(\text{Play} = \text{No}) \qquad = \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} \qquad = 0.0206$$

$$P(\text{Play} = \text{Yes} \mid \text{Outlook} = \text{Sunny},\ \text{Temp} = \text{Cool},\ \text{Humidity} = \text{High},\ \text{Wind} = \text{Strong}) = \frac{0.0053}{0.0053 + 0.0206} = 0.2046$$

$$P(\text{Play} = \text{No} \mid \text{Outlook} = \text{Sunny},\ \text{Temp} = \text{Cool},\ \text{Humidity} = \text{High},\ \text{Wind} = \text{Strong}) = \frac{0.0206}{0.0053 + 0.0206} = 0.7954$$

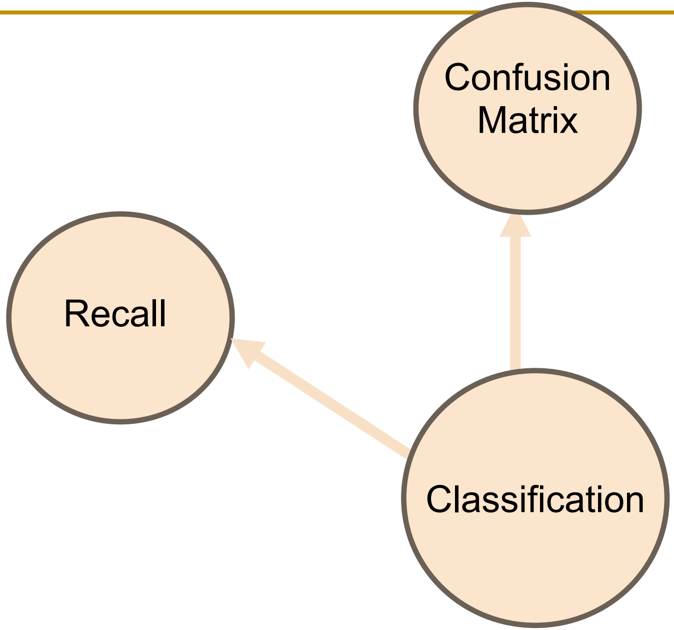
Play = No is more likely!

Performance Matrices



	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

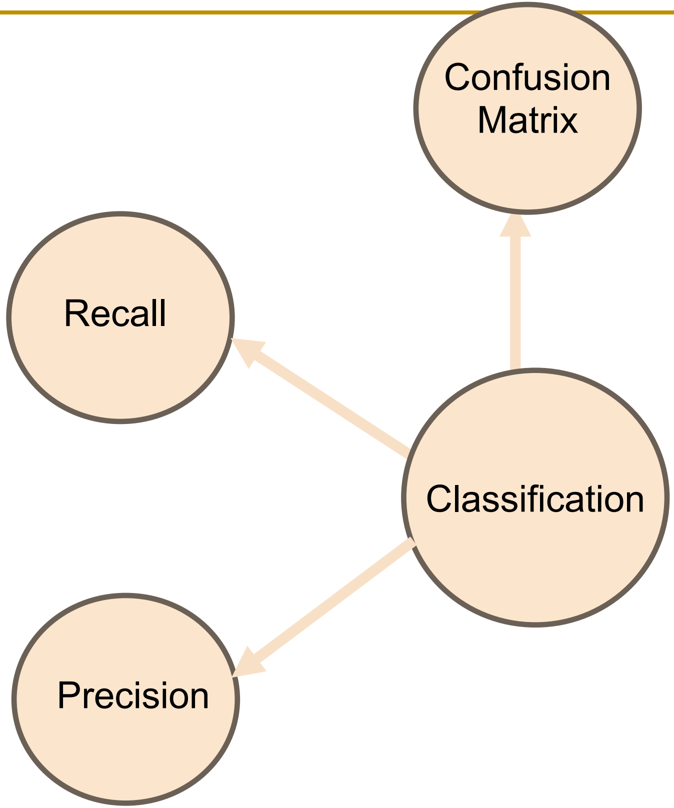
Performance Matrices



	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

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

Performance Matrices

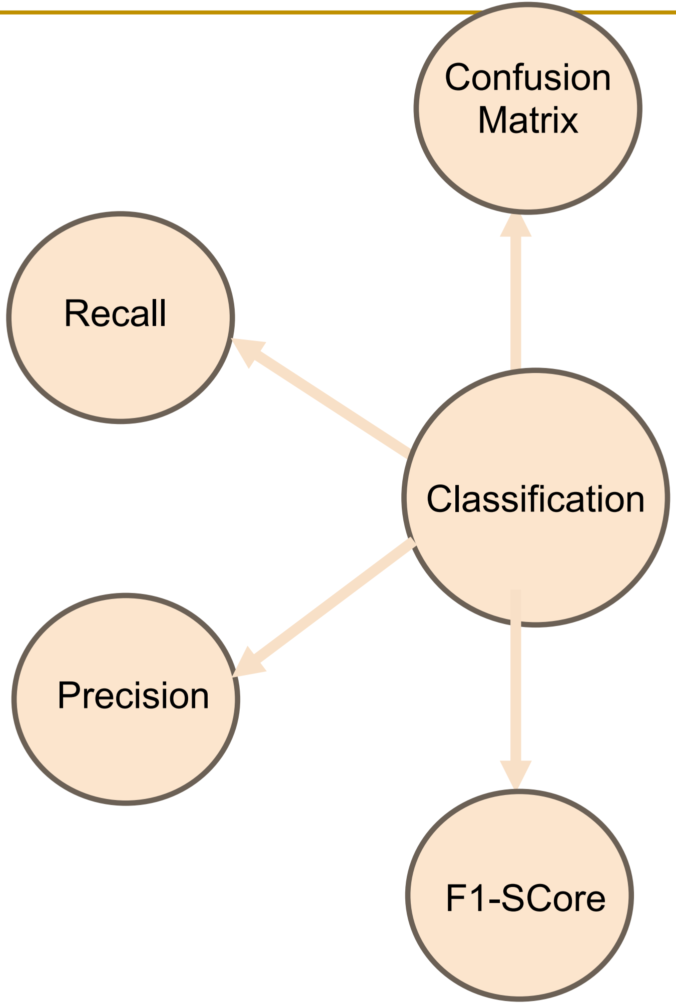


	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

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

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

Performance Matrices



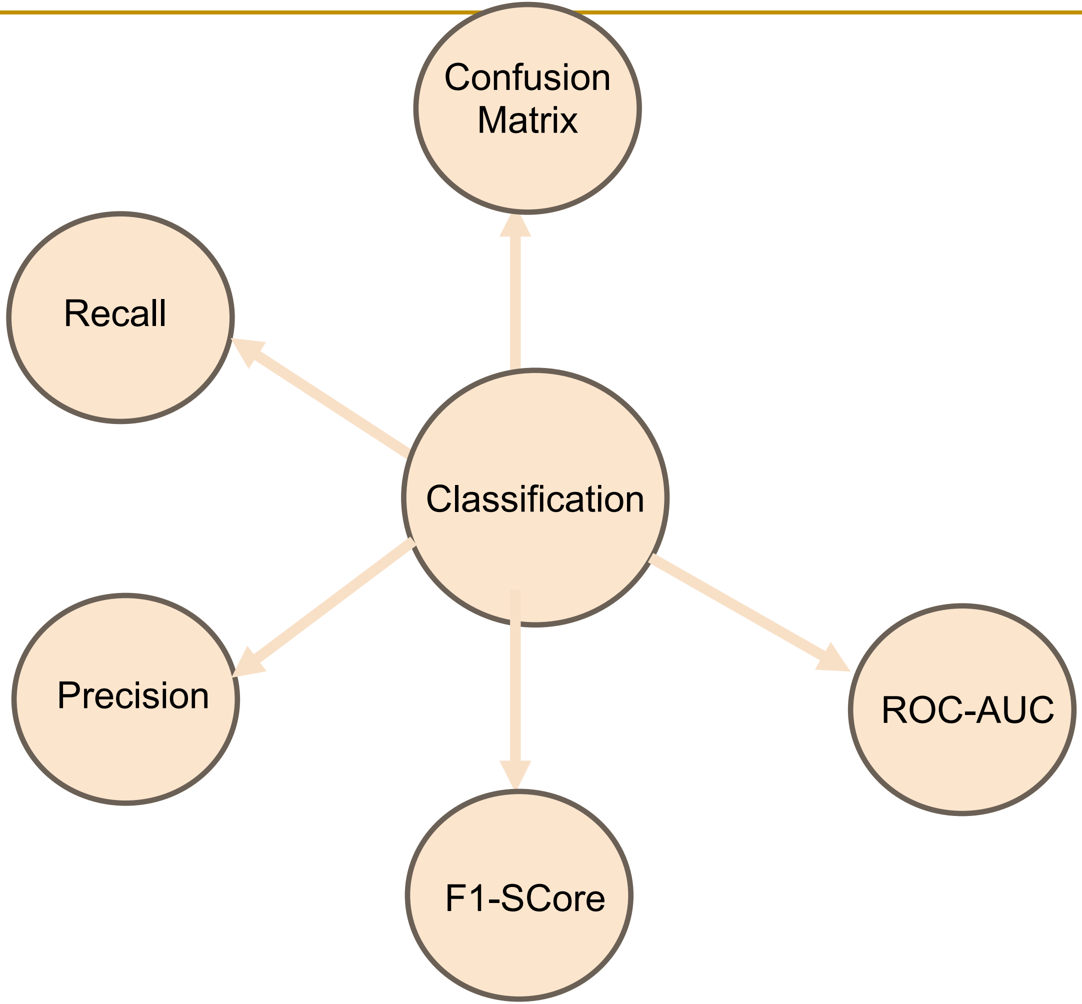
	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

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

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

$$\text{F1 Score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

Performance Matrices



	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

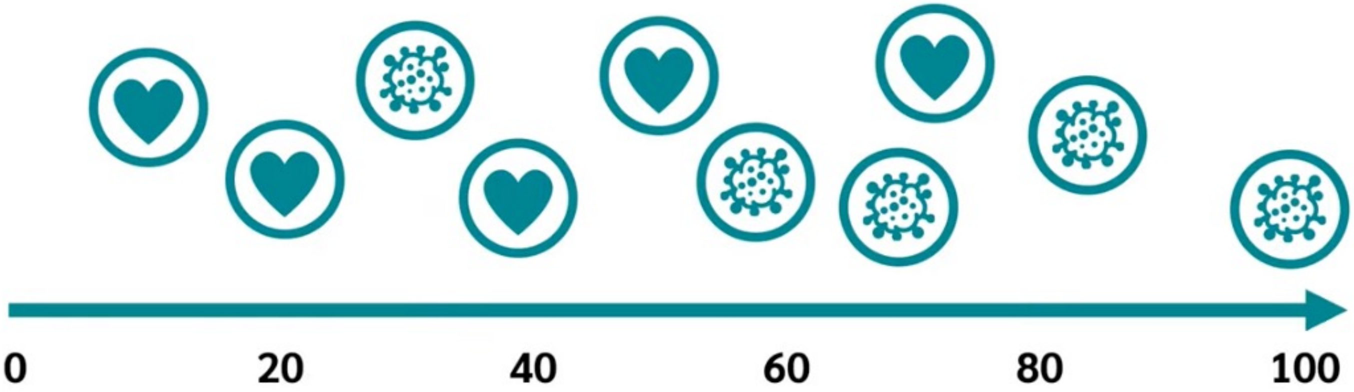
ROC curve is a graphical representation of the performance of the binary classifier

$$TPR = \frac{TP}{FN + TP}$$

$$FPR = \frac{FP}{TN + FP}$$

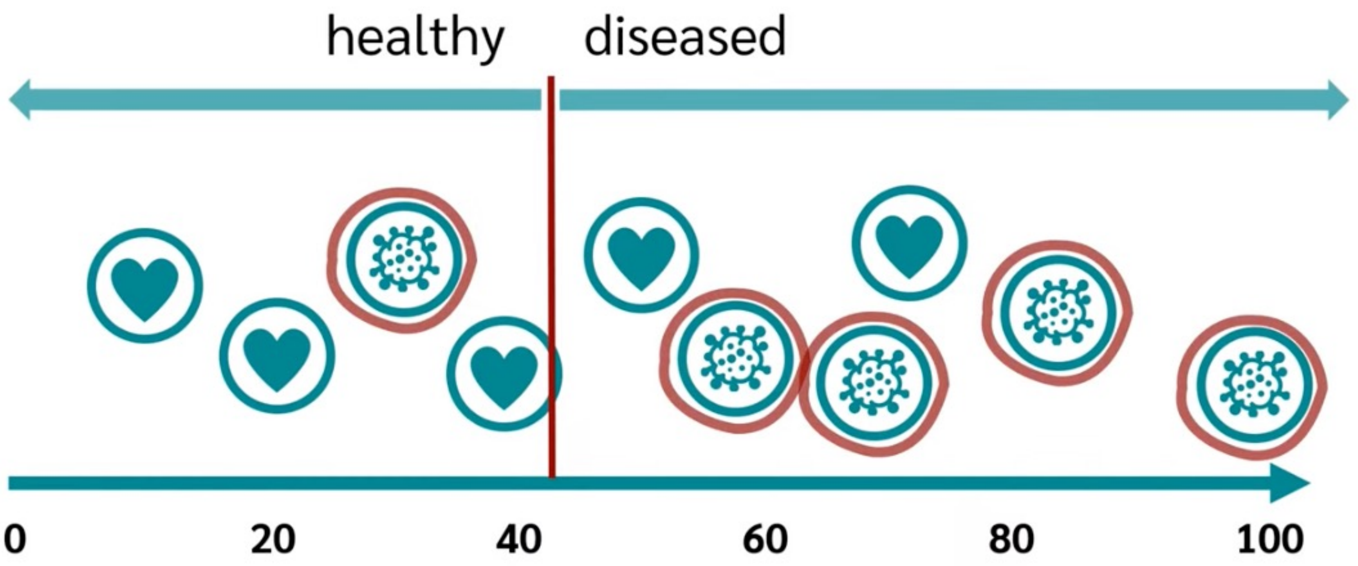
ROC Curve

- We get the data of 10 people about how high blood level is and whether there is disease or not.



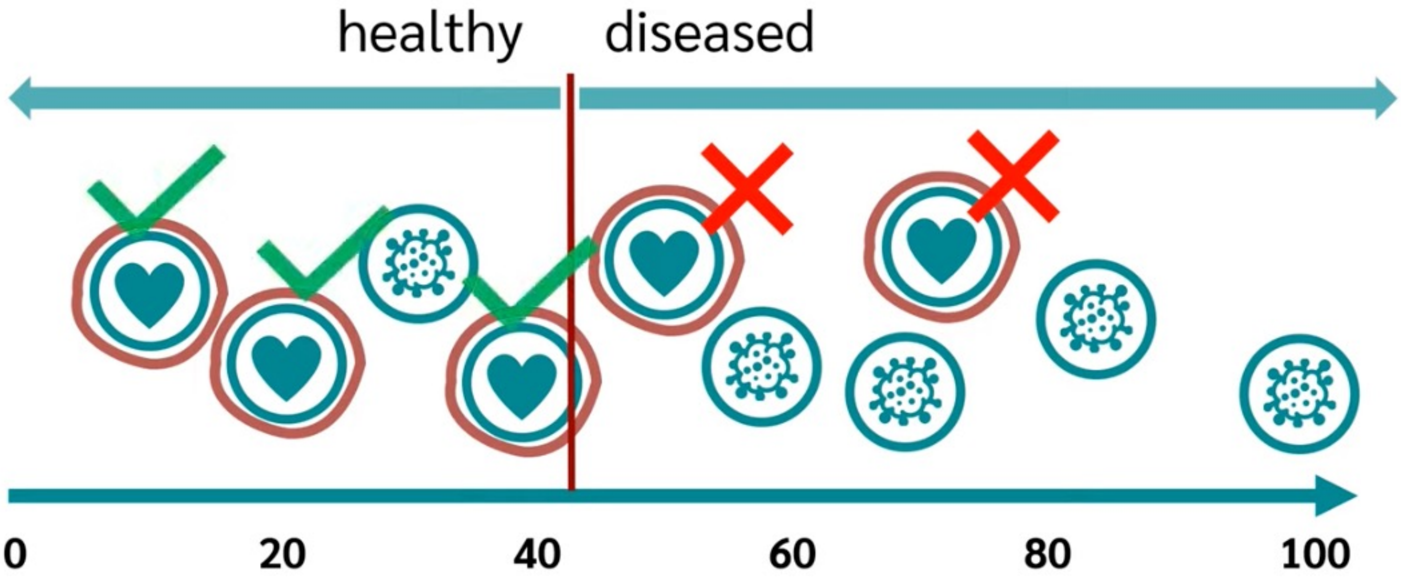
ROC Curve

- 5 individuals have the disease and 4 are classified correctly and 1 is misclassified.
- The true positive rate (TPR) is 0.8



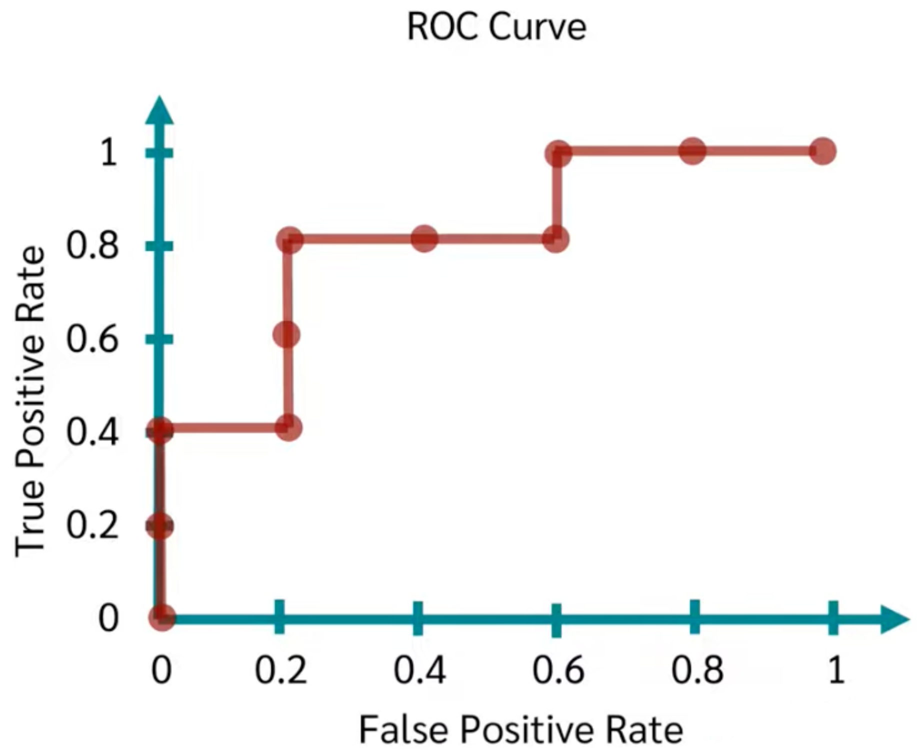
ROC Curve

- 5 individuals are healthy and 3 are classified correctly and 2 are misclassified.
- The false positive rate (FPR) is 0.4



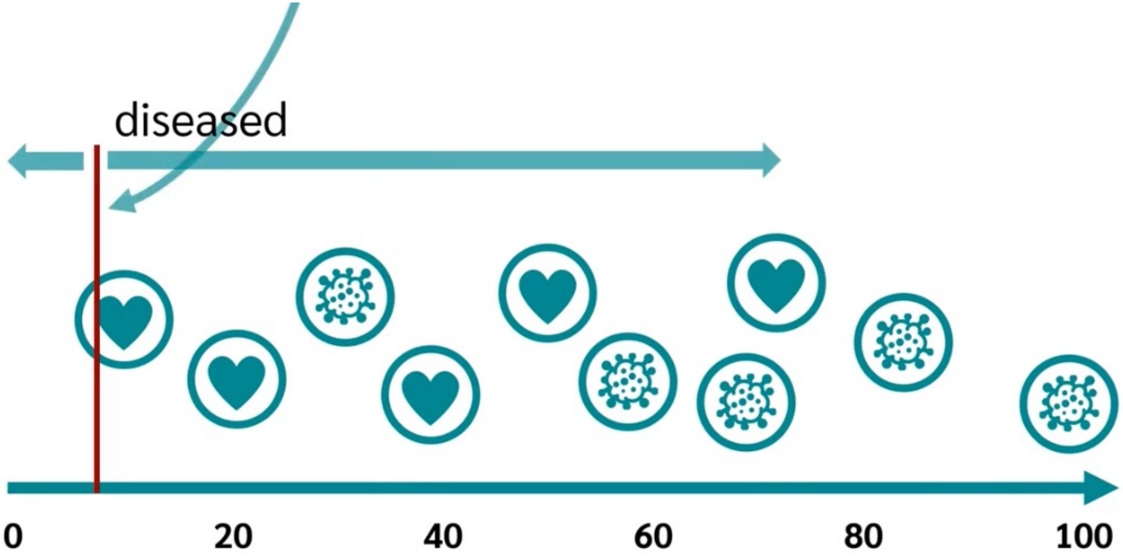
ROC Curve

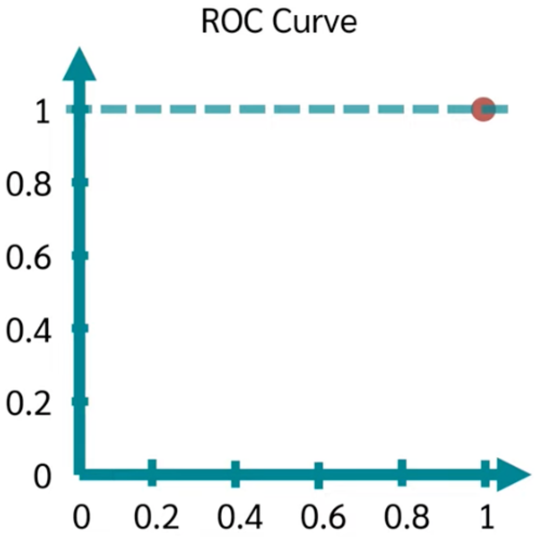
- We calculate the value of TPR and FPR for different value of thresholds.



ROC Curve

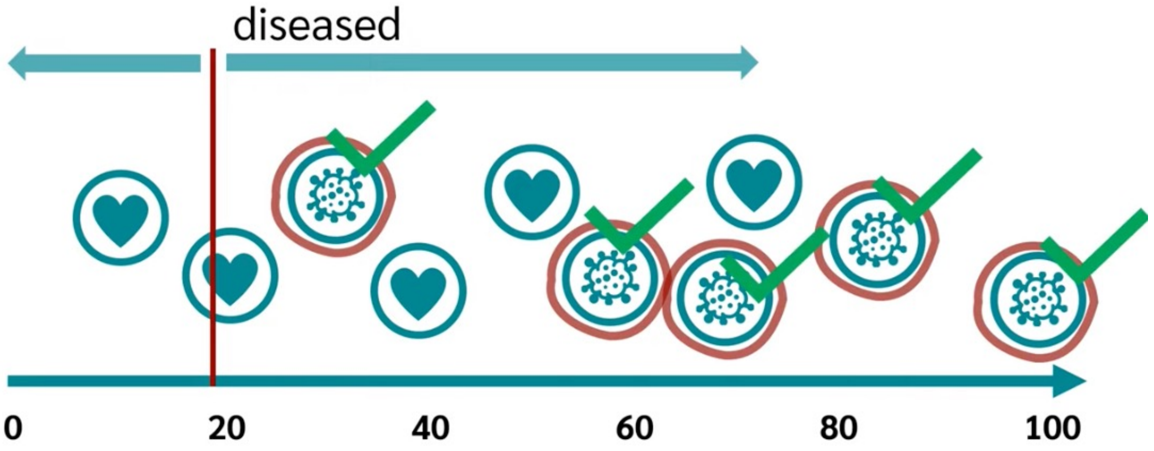
- The true positive rate (TPR) is 1
- The false positive rate (FPR) is 1

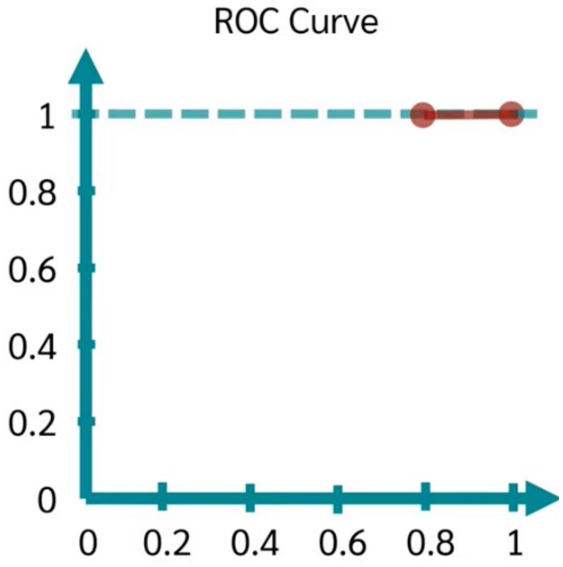




ROC Curve

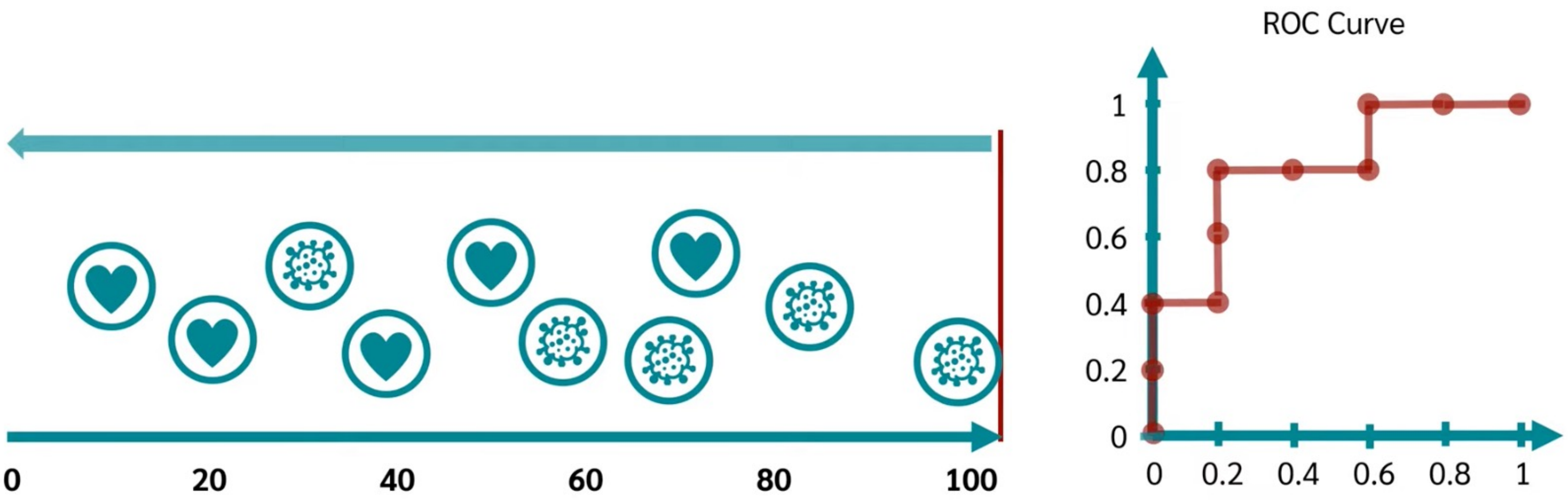
- The true positive rate (TPR) is 1
- The false positive rate (FPR) is 0.8





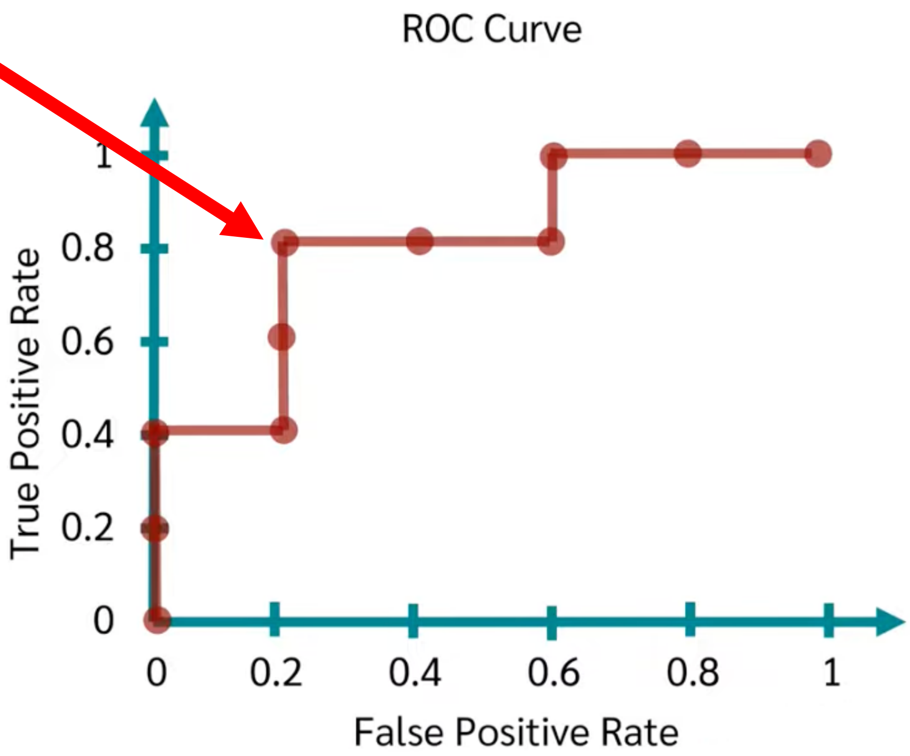
ROC Curve

- The true positive rate (TPR) is 0
- The false positive rate (FPR) is 0



ROC Curve

- For this point (0.2, 0.8)
- 80% of individuals are correctly classified as with diseases.
- 20% of indiviuals are misclassified as with diseases.



ROC Curve

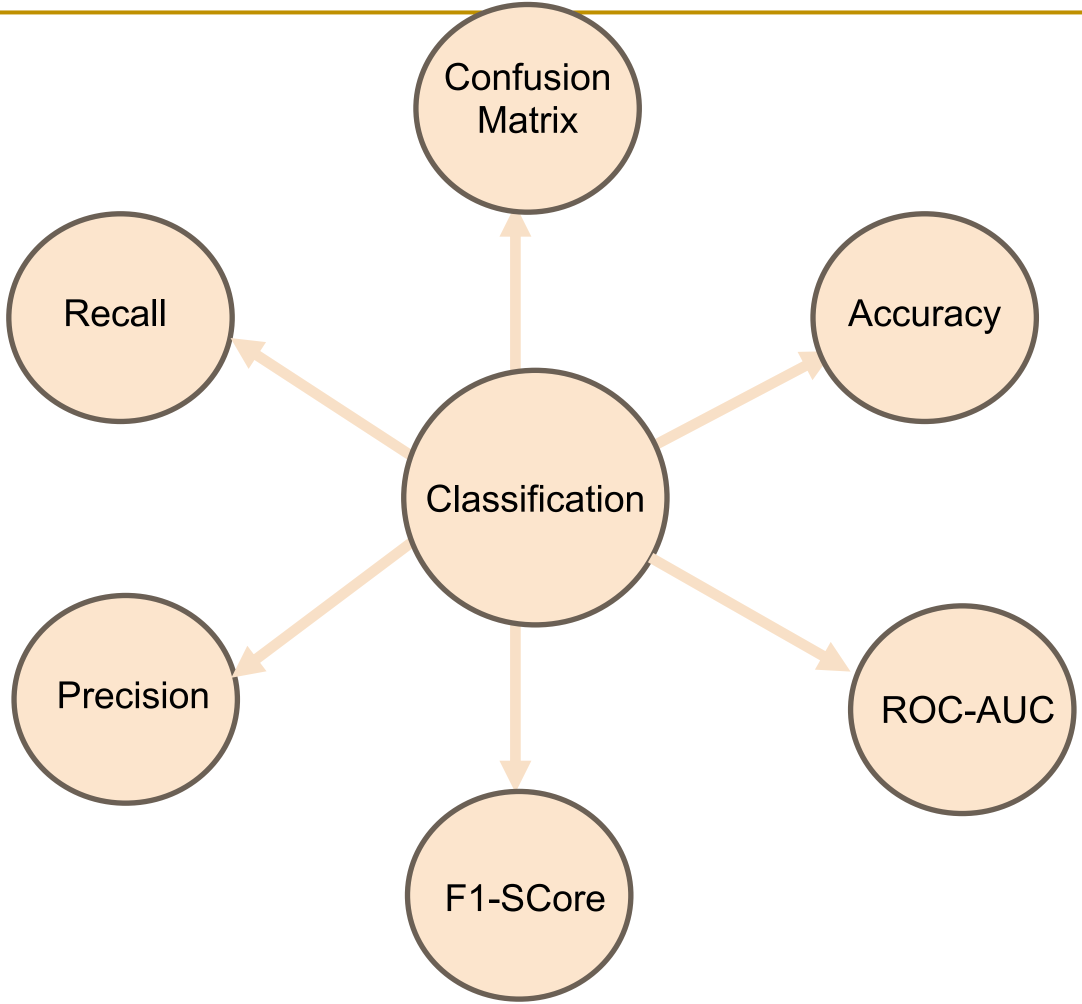
- from sklearn.metrics import roc_curve

- from sklearn.metrics import roc_auc_score

- fpr, tpr, thresholds = roc_curve(y_true, y_scores)

- auc = roc_auc_score(y_true, y_scores)

Performance Matrices

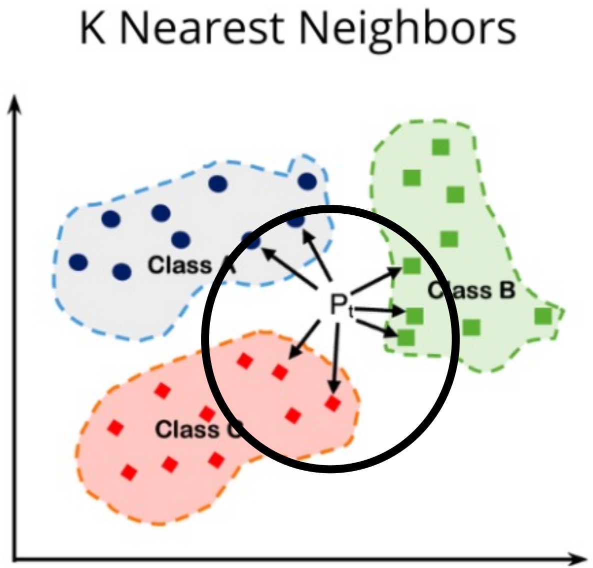


	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

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

K-Nearest Neighbor

- K nearest neighbours is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure.
- Algorithms for classification
  1. Logistic Regression
  2. Decision Trees
  3. Naive Bayes
  4. k-Nearest Neighbours
  5. Support Vector Machine
  6. Random Forest



KNN Algorithm

- The k-nearest neighbours algorithm stores all the available data
- Classifies a new data point based on the similarity measure (e.g., distance functions).
- The data point is classified by a majority vote of its neighbours, with the data point being assigned to the class most common amongst its K nearest neighbours measured by a distance function.

KNN Algorithm

- Loading the training and the test data.
- Choose the nearest data points (the value of K). K can be any integer.
- Do the following, for each test data point
  - Use Euclidean distance $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$ or Manhattan distance $\sum_{i=1}^{k}|x_i - y_i|$

    to calculate the distance between test data and each row of training.
  - Sort the data set in ascending order based on the distance value.
  - From the sorted array, choose the top K rows.
  - Based on the most appearing class of these rows, it will assign a class to the test point.
  - End

Companies Using KNN

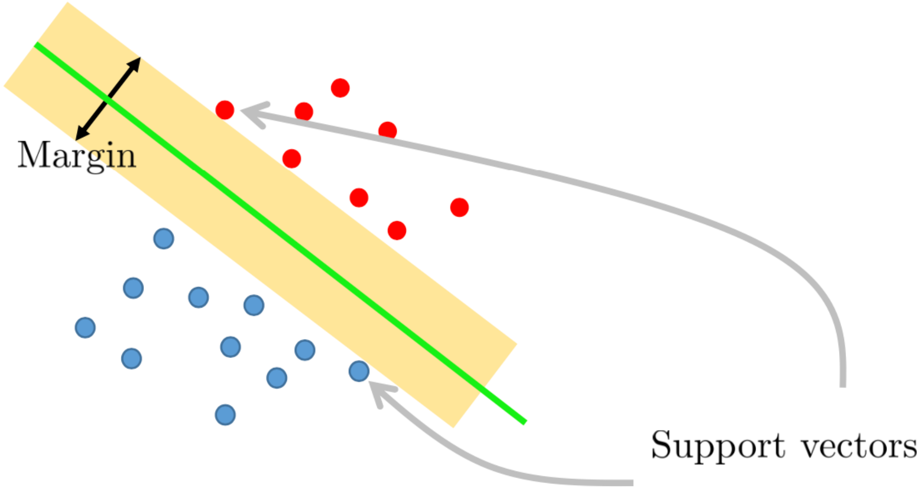
- Companies like Amazon or Netflix use KNN when recommending books to buy or movies to watch.
- How do these companies make recommendations?

Well, these companies gather data on the books you have read or movies you have watched on their website and apply KNN.

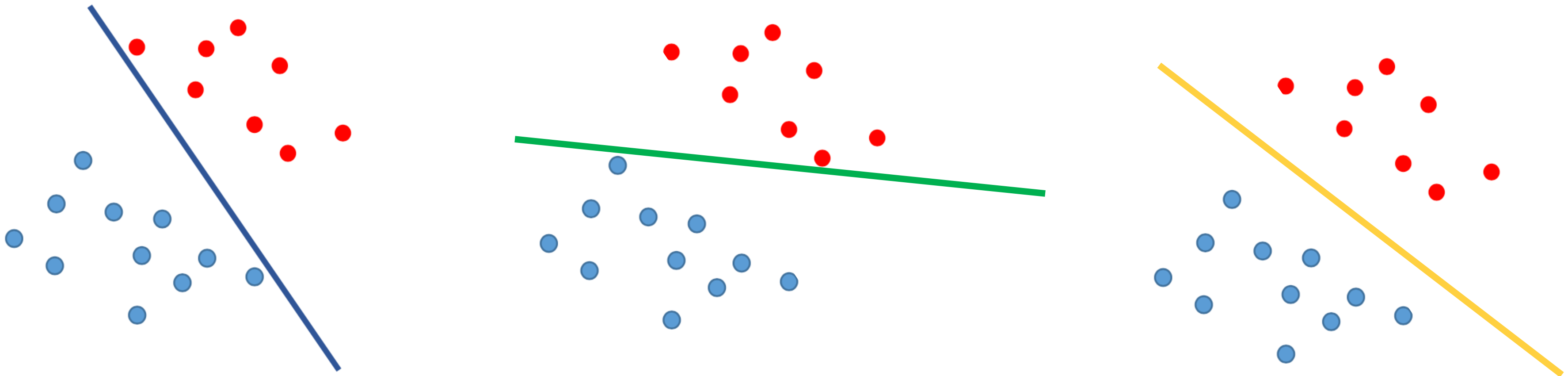
The companies will input your available customer data and compare that to other customers who have purchased similar books or have watched similar movies.

https://www.mygreatlearning.com/blog/knn-algorithm-introduction/

- **SVM** is used to classify data by finding the optimal decision boundary that maximally separates different classes
- Algorithms for binary classification
    1. Logistic Regression
    2. Decision Trees
    3. Naive Bayes
    4. k-Nearest Neighbours
    5. Support Vector Machine
    6. Random Forest



- We have linear separable classes.
- We can have multiple hyperplanes separating these classed.

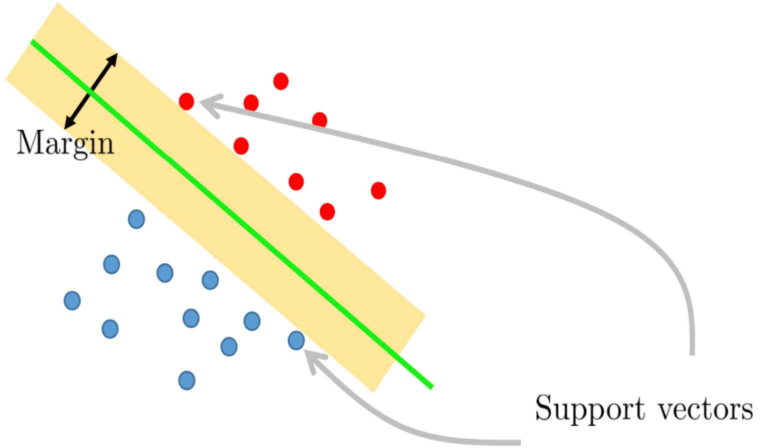


Q: Which one is the best decision boundary?

A: Maximum Margin Classifier (e.g., Support Vector Machine)

Support Vector Machine (SVM)

- Idea: Choose a fat separator
- The best boundary is the one that maximizes the margin or distance between the boundary and the “difficult points” close to the decision boundary.
- Margin against the data points are called support vectors.
- **Hard margin idea:** Find a maximum margin classifier with no errors on the training data.
- **Soft margin idea:** Find the maximum margin classifier while minimizing the number of training errors.



Support Vector Machine (SVM)

- from sklearn.svm import SVC

```
svm_classifier = SVC(kernel='linear', C=1.0)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
```

You can choose different kernels (e.g., 'linear', 'poly', 'rbf') by specifying the kernel parameter. The C parameter controls the trade-off between maximizing the margin and minimizing the classification error.