# Prediktiv analys

FÖRELÄSNING 7

# Dagens agenda

- Gradient descent
- Evaluering av klassifikationsmodeller:
  - Error metrics
  - Confusion Matrix
  - Threshold value
- Inlämningsuppgift
- Classification trees
- Rule inference

# Förra föreläsning

- Klassificering
- Olika typer av klassificering metoder,
- Vad sannolikhet är och hur det används i klassificering
- Modellen Logistic regression
- Kategoriska features: Ordinal encoding, One-hot encoding, dummy variabel

# Loss/cost function

- **Loss function** – funktion som berättar hur mycket fel vi har i en predikton
  *ex Ordinary least square, Gradient descent*
- **Cost function** – funktion som berättar hur mycket fel vi har i alla prediktioner. En samling loss funktioner
  *ex Ordinary least square, Gradient descent*
- **Error metric** – mäter fel i en prediktiv model
  *ex MSE, RMSE, MAE*

- När vi tränar en machine learning modell (.fit i scikit learn) är det en learning algoritme som optimerar modellen.
- Denna learning algoritmen brukar man säga optimerar en **loss function** eller **cost function**
- Termerna används huller om buller

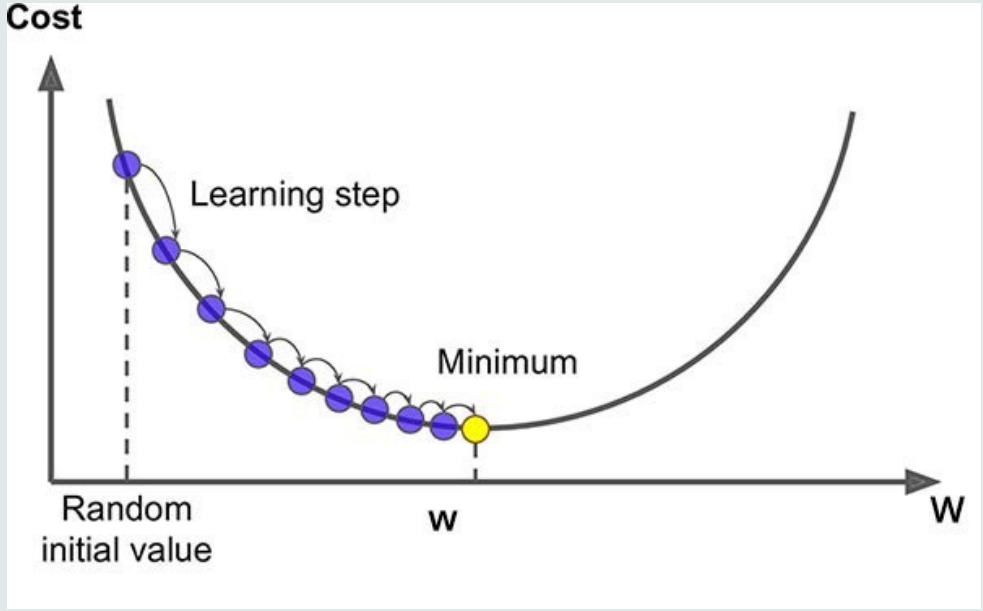
# Gradient descent

- Logistic regression använder gradient descent som learning algoritm (optimering av modellen)
- Nästan alla många machine learning modeller kan använda gradient descent
- Gradient descent är en optimerings algoritme för att hitta värden på modellen (vikterna) för att minimera felet mellan vad vi predikterar och rätt värde på $y$
- Intuition:
  - Tänk en stor skål. Denna är plotten av en cost function (fel mellan $\hat{y}$ och $y$)
  - En random position i denna skålen är felet för nuvarande värden på vikterna i modellen (koefficientera, $w$)
  - Botten av skålen är felet när vi har bästa möjliga värden på vikterna till modellen. Alltså minimum av cost function
  - Så man testar olika värden på vikterna, utvärderar värdet på cost funktionen och väljer nya värden på vikterna som har lägre fel
  - Efter tillräckligt många repetitioner är vi i botten av skålen och värden vi har på vikterna är de vi använder i vår färdiga tränade modell
  - Vi rör oss neråt i skålen med hjälp av derivation!

# Gradient descent

$$y = b + wx$$

- $b$ är bias (intercept), $w$ weight (slope), $x$ independent features, $y$ dependent target
- Gradient descent – optimerings-algoritme för att hitta minimum
- Mål: hitta $w$ och $b$ som beskriver fårhållandet mellan $y$ och $x$ korrekt
- Använder: Loss function (error metric/cost function)
- Loss function: hur mycket predikterade värden skiljer sig från faktiskt värde på $y$
- => Hitta $w$ och $b$ som minimerar loss funktion
- Detta görs genom att ta derivatan av loss funktionen med avseende på $w$ och $b$
- Sedan rör man sig i riktningen till derivatan för att ändra $w$ och $b$ tills de *konvergerar* (uppnår optimalt värde)



# Gradient descent

Steg för att utföra gradient descent med linjär regression:

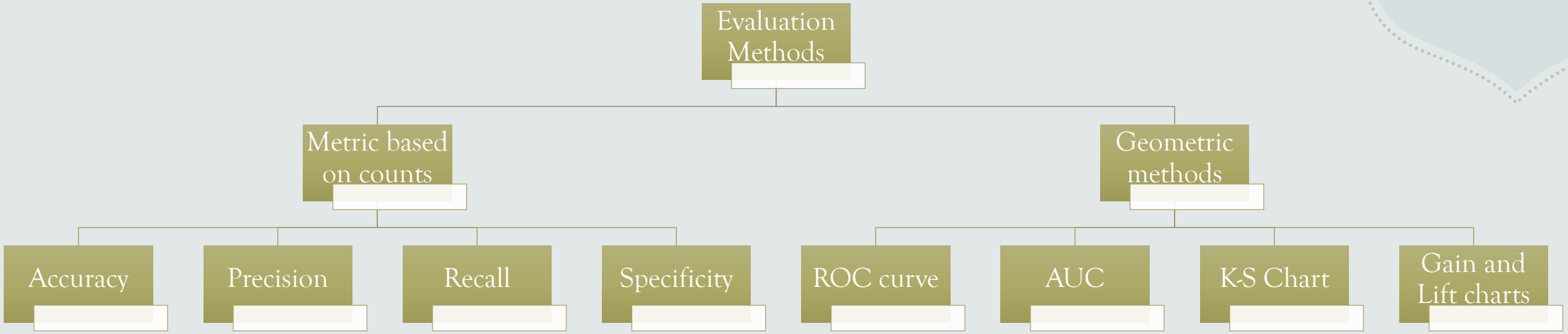
1. Initialisera vikter $w$ och bias $b$. Antigen med random tal eller med 0
2. Skapa prediktioner med dessa initiala värden på $w$ och $b$
3. Jämföra de predikterade värden på $y$ med det sanne värdet på $y$ och definiera loss funktionen med båda de predikterade och sanna värden på $y$
4. Beräkna hur loss funktionen ändrar sig med hjälp av derivatan beroende på $w$ och $b$
5. Uppdatera $w$ och $b$ så att de minimerar loss funktionen

# Model Evaluation for classification

- Evaluation methods for classification
- Confusion matrix
- Accuracy
- Precision
- Recall
- Threshold value

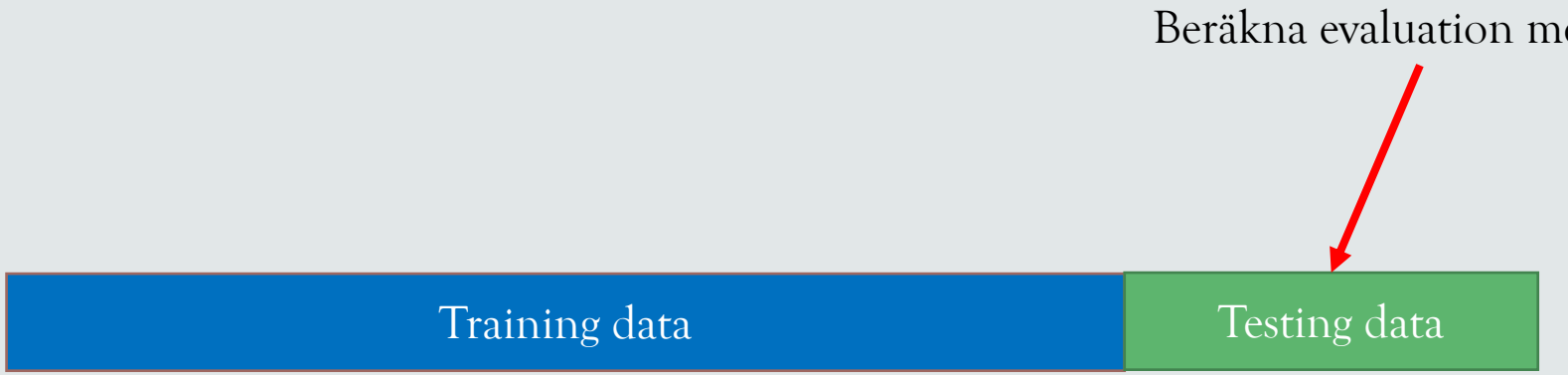
# Model Evaluation

- There are many evaluation metrics used to measure the performance of classification models. As with regression, the intuition behind all of them is to measure how close the predicted values are to the observed target values.
- There are two broad typed of evaluation methods:



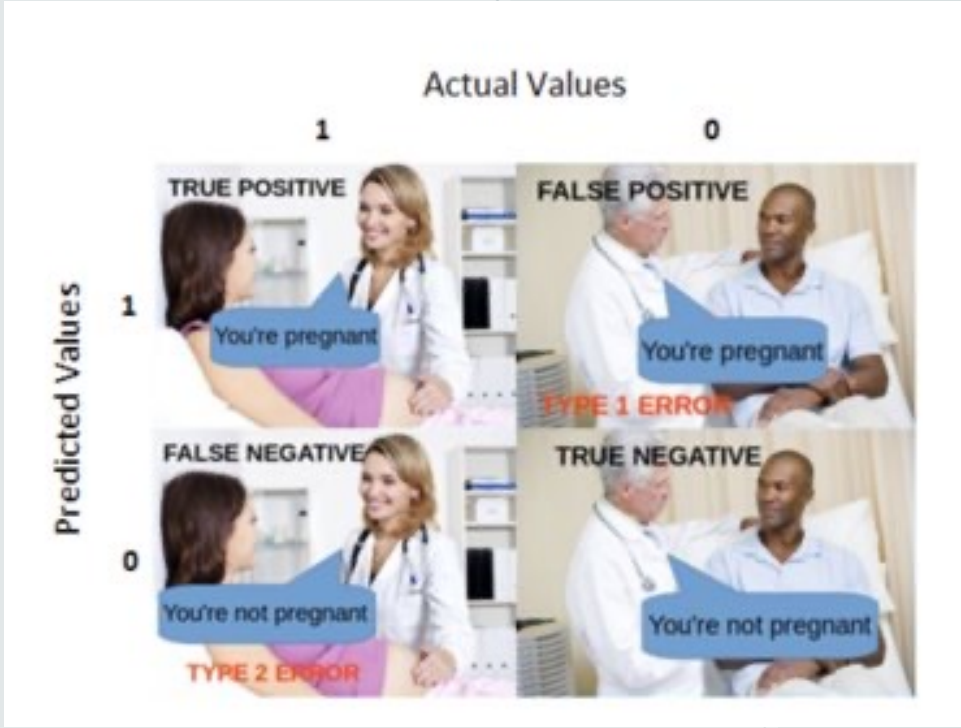
# Holdout Cross-Validation

- Som med regression använder vi cross-validation för att uppskatta hur väl modellen fungerar med osedd, ny data



# Confusion Matrix

- För binary classification
- Många av de viktigaste metricsene för att utvärdera klassificering får vi från confusion matrix
- Det är en två-dimensionell tabell med observerade och predikterade värden
- Kom ihåg att vi har valt en positivt klass (=1) och en negativ klass (=0) från början
- När vi gör en confusion matrix är det dessa positiva och negative klasserna som används i schemat för att se hur väl vi har predikterat
- True betyder att vi har predikterat rätt klass
- False betyder vi inte har predikterat rätt klass




| | | Predicted | |
|---|---|---|---|
| | | 0 | 1 |
| Observed | 0 | True Negatives ☺ (TN) | False Positives ☹ (FP) |
| | 1 | False Negatives ☹ (FN) | True Positives ☺ (TP) |

# Metrics till Confusion Matrix

- **Accuracy:** Andelen av totala antal predikterningar som är korrekt

$$Accuracy = \frac{TP + TN}{Total}$$

- **Precision:** Andel av positive prediktioner som faktisk är korrekt

$$Recall = \frac{TP}{TP + FP}$$

- **Recall:** Andel av positivt observerade värden som är korrekt predikterad

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

- **F1 Score:** kombination av precision och recall

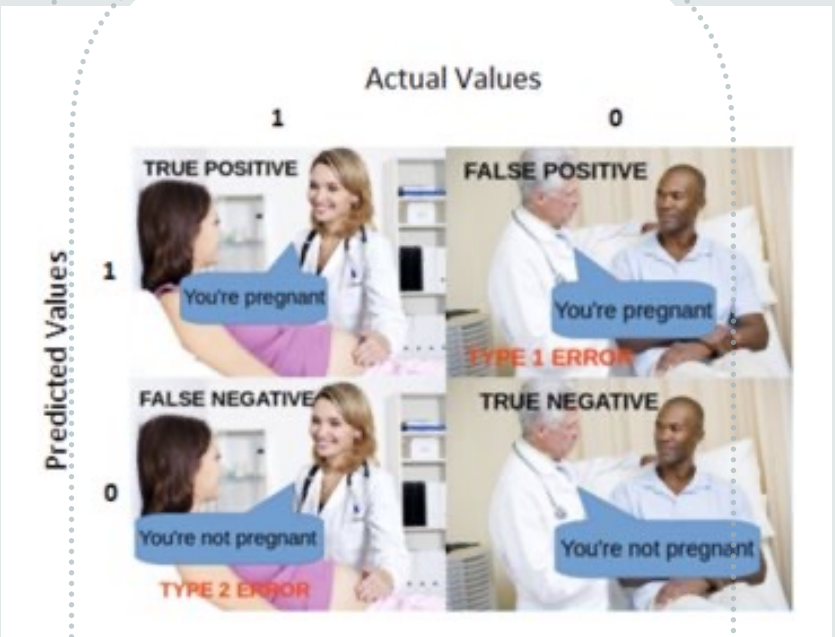
$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

| | | Predicted | |
|---|---|---|---|
| | | 0 | 1 |
| Observed | 0 | True Negatives ☺ (TN) | False Positives ☹ (FP) |
| | 1 | False Negatives ☹ (FN) | True Positives ☺ (TP) |

Alla går från 0 (sämst) till 1 (bästa) och ofta som procenter

# Vilken metric ska man använda?

- Ställ frågan:
  Vilket misstag är sämst för problemet jag försöker lösa?
  False positives eller false negatives?
- **Accuracy**: använd om båda misstag är mer eller mindre lika dåliga
- **Precision**: använd när ni vill undvika **false positives**
  *ex spam email, när false positive betyder en icke-spam mail har blivit klassad som spam och man går miste om mail*
- **Recall**: använd när du vill undvika **false negatives**
  *ex sjuka patienter, om en sjuk patient (faktisk positiv) blir predikterad till inte sjuk (false negative)*
- **F1:** balans mellan precision och recall och vi har obalanserad data



# Vilken metric ska man använda?

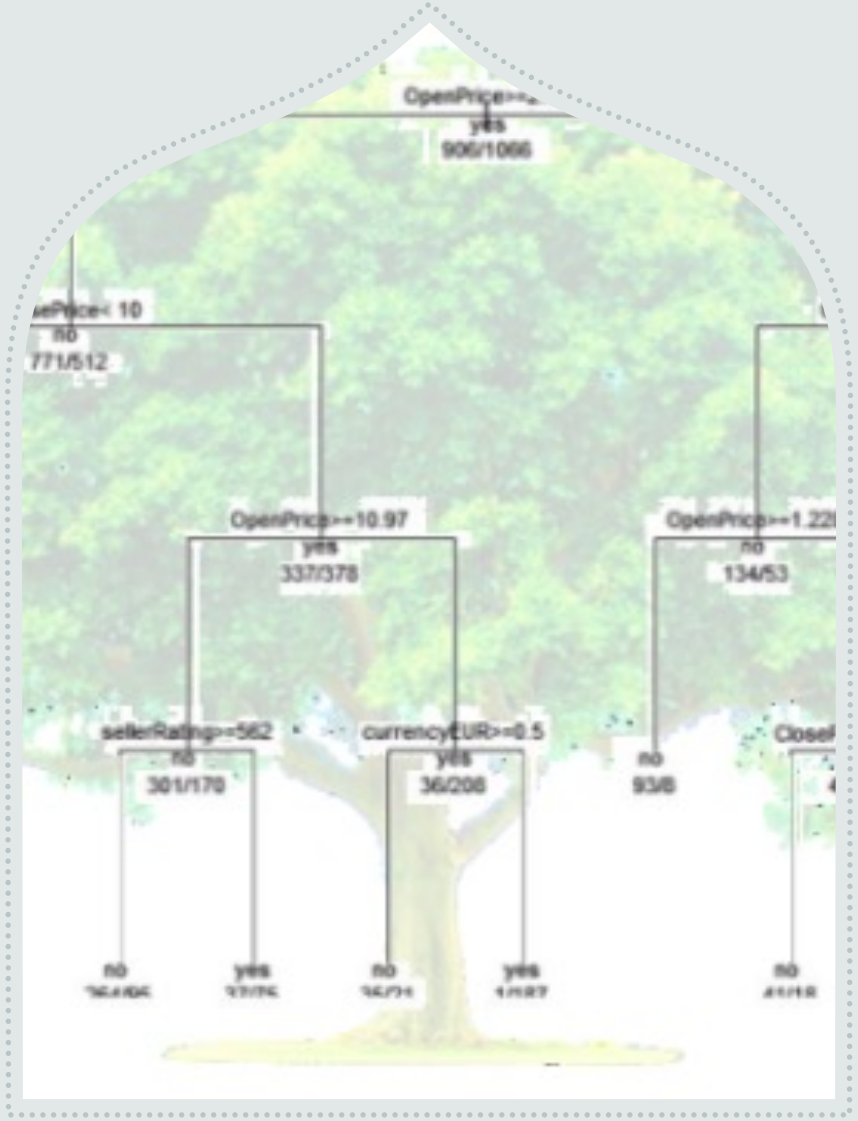
- **Exempel:** Credit card transaction klassificering, positiv klass: Bedrägeri transaktioner versus bra transaktioner
- **False positive**: bra transaktioner predikterad (klassificerad) bedrägeri
- **False negative**: bedrägeri transaktioner predikterad (klassificerad) som bra
- Vilken är sämst?
- False negative
- Metricen vi borde bry oss om är **recall**

# Threshold Value och metrics

- Modeller baserad på sannolikheter predikterar att det är den positive eller negativa klassen (1 eller 0) vid att använda beräknade sannolikhet och en **threshold value**
- Det är minsta sannolikhet som klassificerar en observation I den *positiva klassen*
- Default är 0.5 (50%)
- Den kan ändras och detta kan användas för att justera precision och recall metrics i klassificeringen

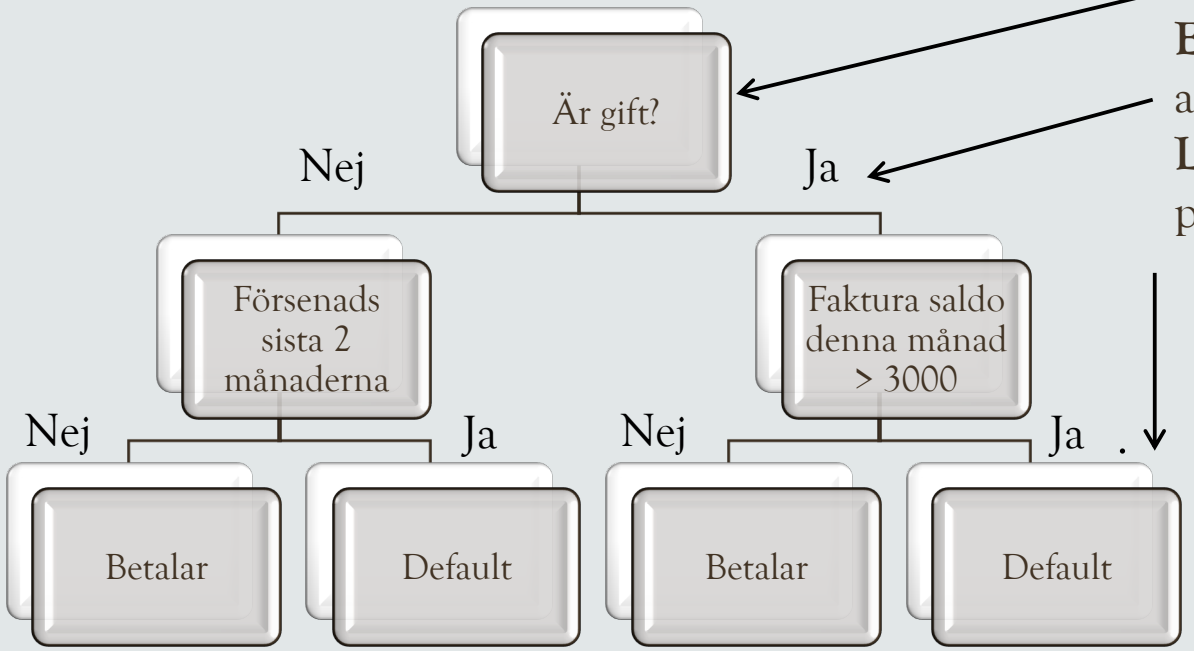
# Classification Trees

- Målet är att skapa en modell som predikterar target $y$ genom att lära sig simpla beslutsregler (if-then regler) baserat på slutsatser den drar från egenskaperna $x$
- Fördelar
  - Enkla att förstå och att visualisera
  - Kräver lite data förberedning
- Nackdelar
  - De tenderar att overfitta träningsdatan
  - De funkar inte bra när det är en obalans mellan positiva och negativa klasser t.ex. 90% negativa klasser och 10% positiva klasser



# Classification Trees

Prediktera kredit default nästa månad:




Består av:

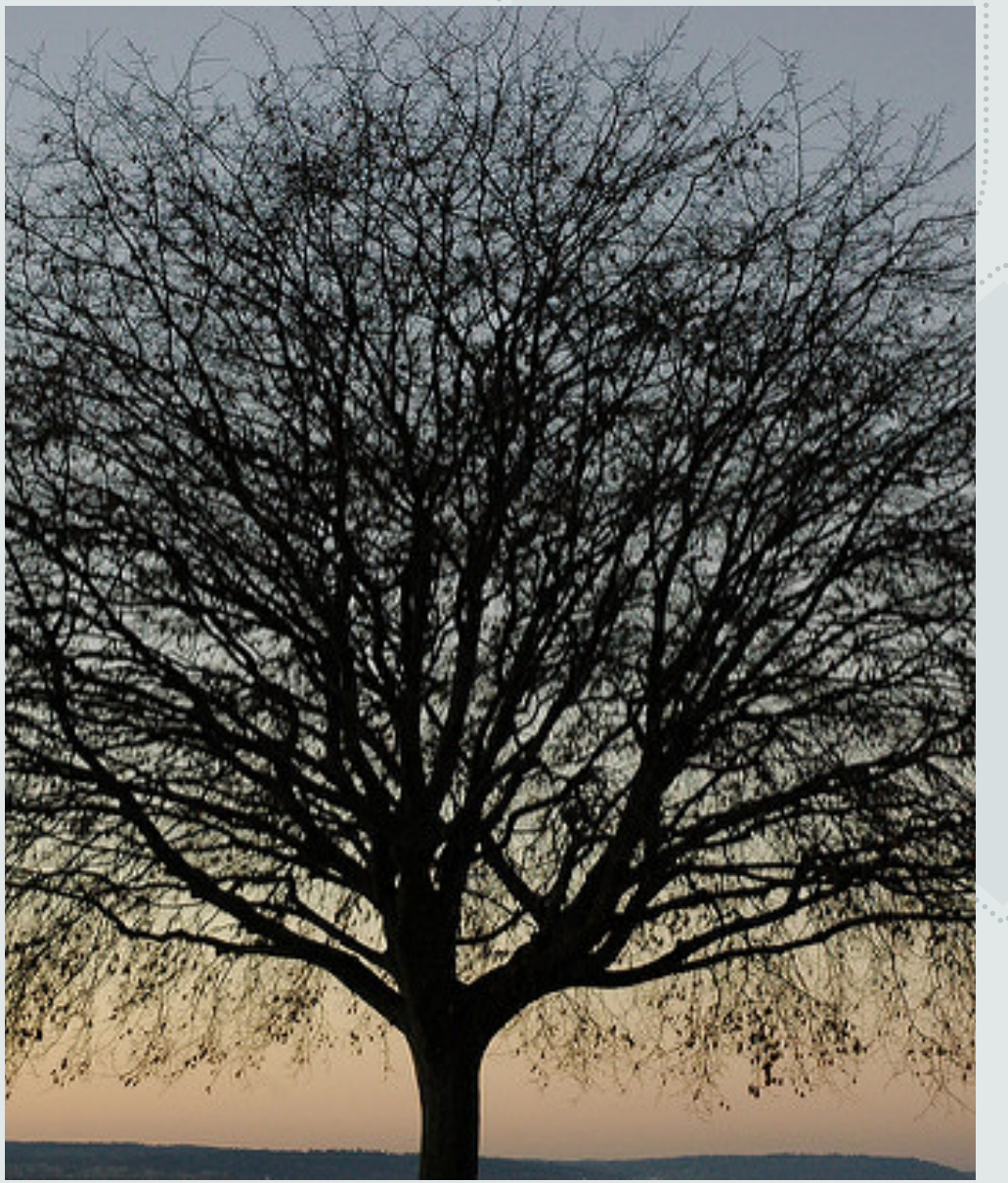
**Nodes** – testar värdet till en variabel

**Edges** – Resultatet av testen och anslutning till nästa node eller leaf

**Leaf** – ”terminal” slutliga noder som predikterar klassen

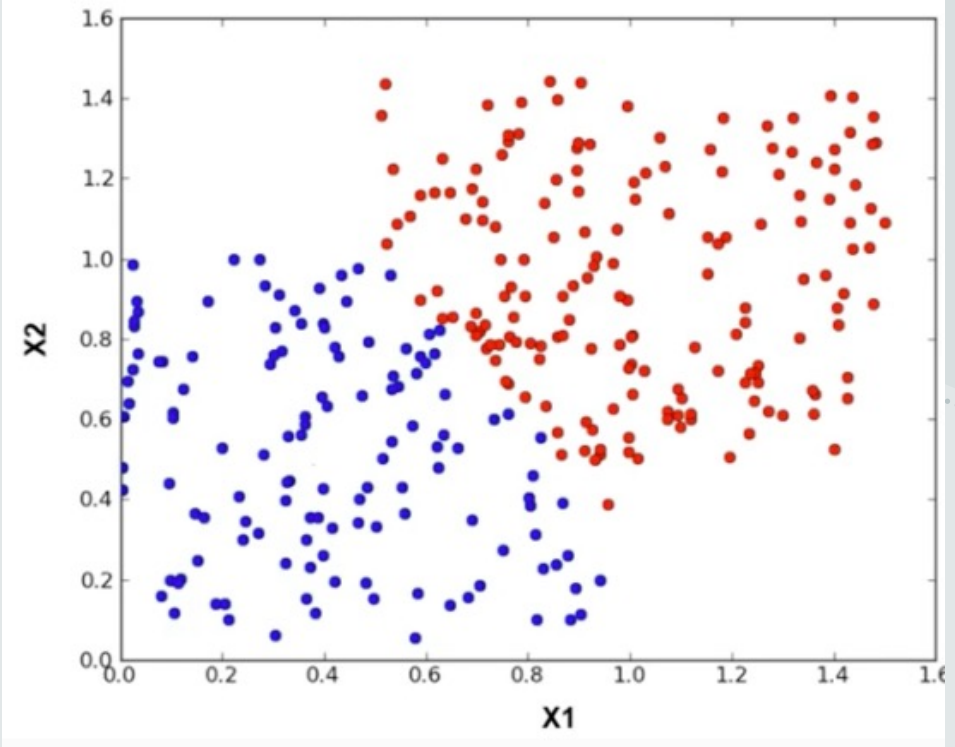
# Classification tree

- Byggd upp av *rule inference*. Iterativ process som delar upp datan och sedan delar upp den vidare på var och en av grenarna.
- Använder en beslutsalgoritm. Vid trädroten splittras datan efter den featuren som ger störst vinst i information (reducerar osäkerhet mest när man klassificerar)
- Genom en iterativ process fortsätter datan splittras
- Sätter en gräns för hur djupt trädet kan bli för att undvika overfitting



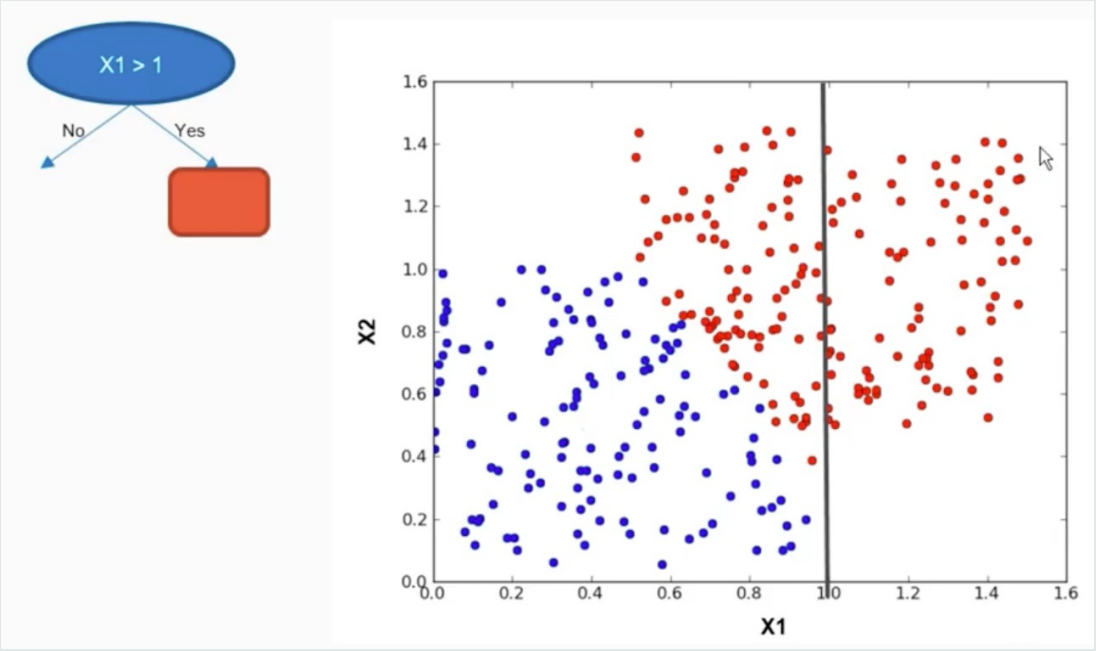
# Rule inference

- I detta exemplet är det två egenskaper (features) $x_1$ & $x_2$
- Två kategorier: Blå och röd
- Generella regeln är att algoritmen försöker dela in egenskaperna i lådor/delar sådana att delarna är så ”rena” som möjligt
- Med ”ren” menar vi att varje del innehåller nästan enbart observationer från samma kategori

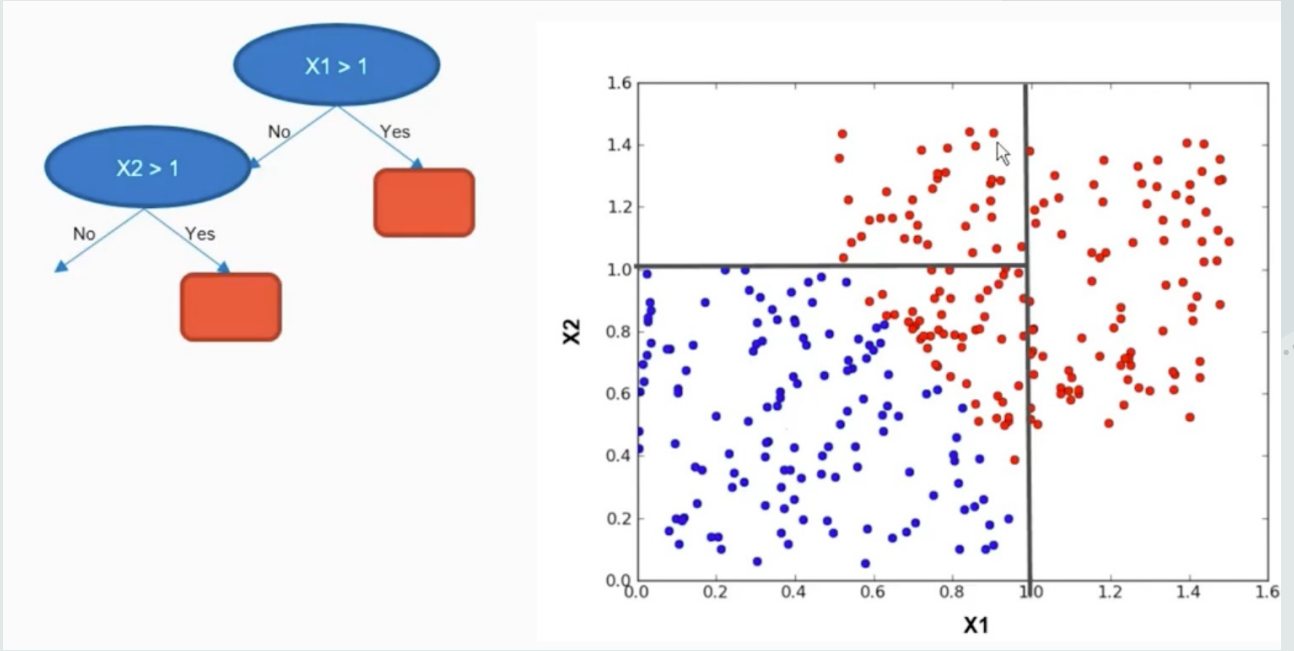


# Rule inference

Regel 1

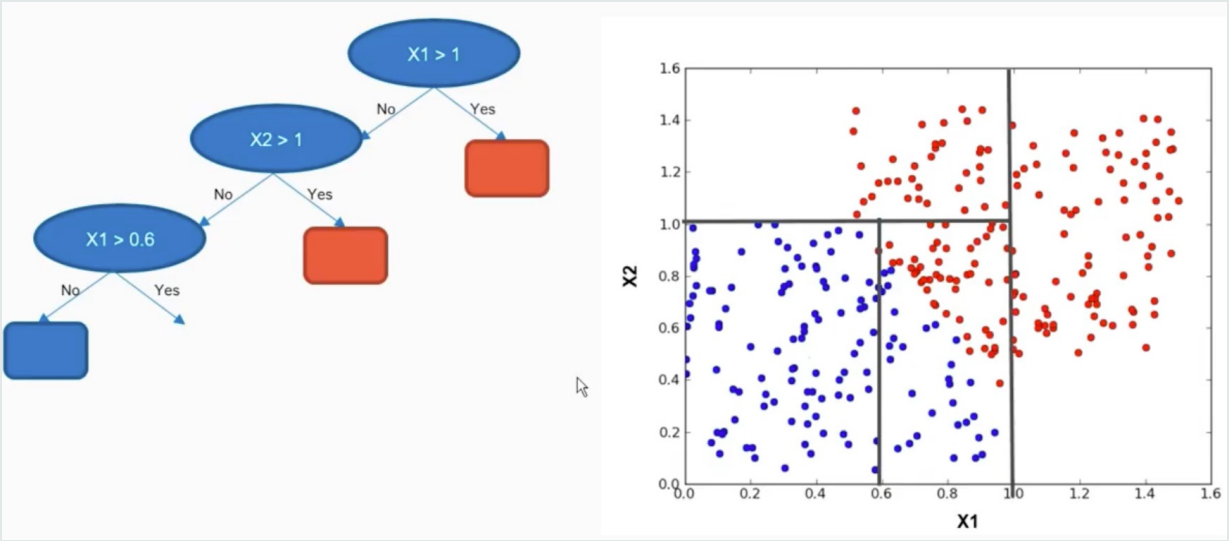



Regel 2

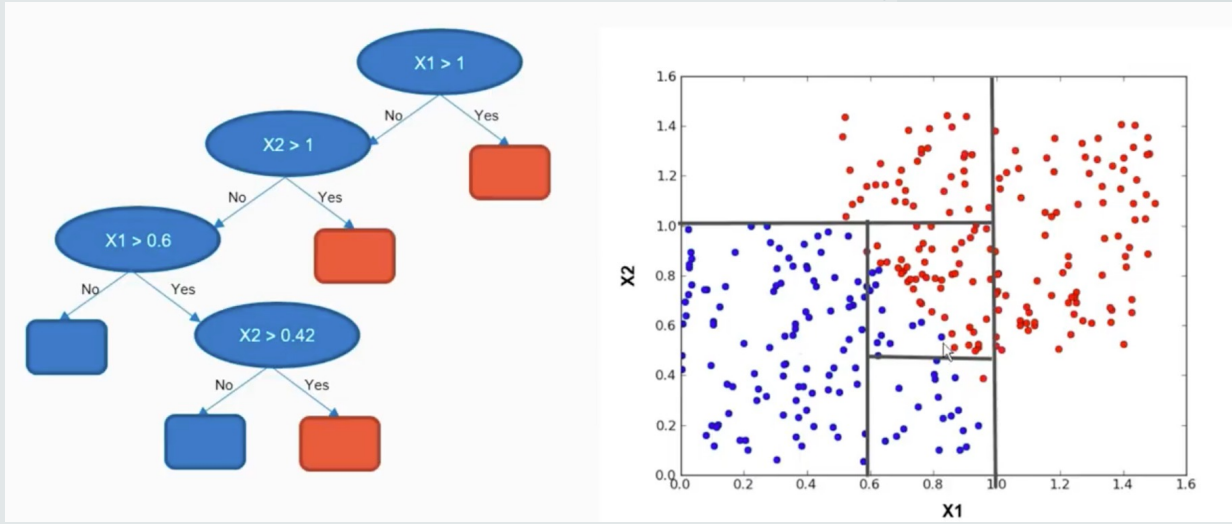


# Rule inference

Regel 3




Regel 4



# DecisionTreeClassifier Estimator in scikit-learn

- Importera DecisionTreeClassifier från träd metoden
- Importera parameter:
  - max_features: The number of features to concider when loking for the best split
  - max_depth: The maximum depth of the tree, normal  4-12) (risk for overfit if to deep
  - min_sample_split: The minimum number of samples required to split an internal node (can be used to counteract overfitting)
  - min_sample_leaf: The minimum number of samples required to be at a leaf node (can be used to counteract overfitting)

# Predicting Credit Card Default

- Default (positiv klass) är att man inte betalar krediträkningen
- Varje rad är en kund
- Hands on genomgång Python

# Vad har vi gjort idag?

# Nästa lektion