6.4 Nearest Neighbor classifiers

6.4.1 Inleiding

Beslissingsbomen en regel gebaseerde classifiers zijn voorbeelden van wat men **"eager learners"** noemt omdat ze ontwikkeld zijn om een model te leren die de inputs (attributen) afbeeldt op het respectievelijke klasse label van zodra de training data beschikbaar is.

Een alternatieve strategie is om het modelleerproces te vertragen totdat het nodig is om een testvoorbeeld te classificeren. Technieken die gebruik maken van deze strategie noemen we **"lazy learners"**. Een voorbeeld van een lazy learner is de **Rote classifier**, deze memoriseert de volledige training dataset en voert classificatietaken uit als de attributen van een test record precies overeen komen met één van de training records. Nadeel van deze strategie is dat sommige records nooit geclassificeerd geraken.

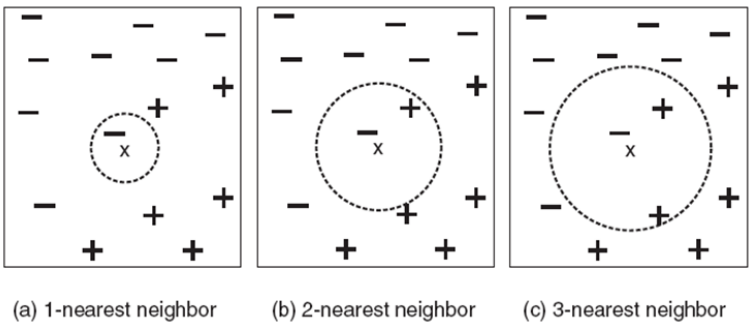
Een manier om dit op te lossen is alle training records te vinden die relatief gelijkaardig zijn aan de attributen van het test record. Deze voorbeelden, wat wij **nearest neighbors** noemen, kunnen gebruikt worden om het klasse label van het test record te bepalen.

De verantwoording voor deze strategie wordt het best weergegeven door volgende uitspraak:
"If it walks like a duck, quacks like a duck, and looks like a duck, then it's probably a duck."

6.4.2 Het KNN algoritme

Een **nearest neighbor classifier** stelt elk voorbeeld voor als een datapunt in een d-dimensionale ruimte, met d het aantal attributen. Vertrekkend van dit test record berekenen we zijn nabijheid tot de andere datapunten (records) in de training set, wij gebruiken hier een nabijheidsmaat zoals b.v. de Euclidische of Manhattan afstand. De **k-nearest neighbors** van een gegeven voorbeeld z refereren naar de k punten die het dichtst bij z liggen.

Onderstaande figuur illustreert de 1-, 2- en 3-nearest neighbors van een datapunt in het centrum van de cirkel.



Figuur 1 - 1-, 2- en 3-nearest neighbors van een instantie. *Overgenomen uit Introduction to Data Mining (p. 224) door Tan P., Steinbach M., Kumar V., 2006, Pearson Education. Copyright 2006 Pearson Education Inc.*

Het datapunt wordt geclassificeerd op basis van de labels van zijn neighbors. In het geval dat de neighbors verschillende labels hebben wordt het label met de hoogste frequentie genomen.

In figuur 1-c zijn er drie verschillende labels (twee + en één -), het datapunt krijgt de + toegekend als label want + komt het meest voor.

Het spreekt voor zich dat de juiste waarde voor k kiezen belangrijk is. Als k te klein wordt gekozen, dan zal de classifier gevoelig zijn voor overfitting. Indien k te groot wordt gekozen bestaat het risico dat een verkeerd label wordt toegekend.

We besluiten met het algoritme in pseudocode, wens je zelf aan de slag te gaan met het implementeren dan vind je tal van codevoorbeelden online (zie ook links op Leho).

Het algoritme berekent de afstand (of similariteit) tussen elk test record $z = (\vec{x}', y')$ en alle training records $(\vec{x}, y) \in D$ om de nearest neighbor te bepalen, D_z. Zo'n berekening kan computationeel veel rekenkracht kosten als het aantal training records groot is. Maar er bestaan efficiënte indexeringstechnieken om het aantal bewerkingen drastisch te verminderen.

Het k-nearest neighbor classifier algoritme.

```
Stel k het aantal nearest neighbors en D de verzameling training records.
for each test record z = (x', y') do
      Bereken d(x', x), de afstand tussen z en elk training record, (x, y) ∈ D.
      Bepaal D_z ⊆ D, de verzameling van k dichtste training records tot z.
      y' = max_v Σ_{(x_i, y_i) ∈ D_z} I(v = y_i)
end for
```

Eenmaal de lijst met nearest-neighbors is bepaald zal het test record geclassificeerd worden op basis van de "**majority class**" van zijn nearest neighbors.

$$\text{majority voting: } y' = \max_v \sum_{(\vec{x}_i, y_i) \in D_z} I(v = y_i)$$

met v een klasse label, y_i is het klasse label van een van de nearest neighbors, en $I(\cdot)$ is een indicatorfunctie die de waarde 1 teruggeeft als het argument waar is en 0 anders.

In deze "majority voting" aanpak heeft elke neighbor dezelfde impact op de classificatie. Dit maakt het algoritme gevoelig voor de keuze van k.

6.5 Performantie van een classifier

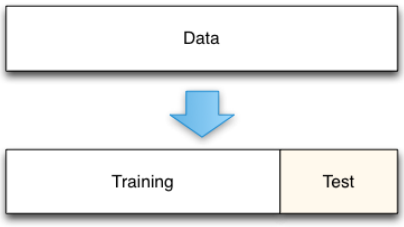
6.5.1 Evaluatiemethoden

Bij het bespreken van evaluatiemethoden gaan wij uit van twee belangrijke maatstaven, de '**accuracy**' of nauwkeurigheid en de '**error rate**' of foutenpercentage.

$$\textbf{accuracy} = \frac{\text{\# correcte voorspellingen}}{\text{Totaal aantal voorspellingen}} \qquad \textbf{error rate} = \frac{\text{\# foutieve voorspellingen}}{\text{Totaal aantal voorspellingen}}$$

6.5.1.1 Holdout methode

Bij de **holdout** methode wordt de originele dataset met gelabelde (geclassificeerde) records gepartitioneerd in twee disjuncte verzamelingen, de training set en de test set. Een classificatiemodel wordt dan geïnduceerd uit de training set en zijn performantie gemeten op de test set.

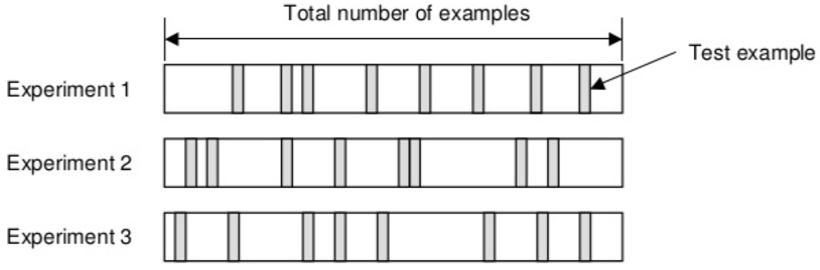


Figuur 2 - Holdout methode.

Doorgaans kiest de analist zelf welke verhouding genomen wordt, dit kan b.v. 50-50 zijn of twee derde training en één derde test.

6.5.1.2 Random Subsampling

Wordt de holdout methode meerdere keren na elkaar uitgevoerd om de geschatte performantie te verbeteren, dan spreekt men van **random subsampling**.
Stel hiervoor $\mathbf{acc_i}$ de nauwkeurigheid van het model tijdens iteratie $\boldsymbol{i}$ en $\boldsymbol{k}$ het **aantal splits**. De globale nauwkeurigheid is dan gegeven door $\mathbf{acc_{sub}} = \sum_{i=1}^{k} \mathbf{acc_i}/\boldsymbol{k}$. Random subsampling heeft echter nog steeds gelijkaardige problemen zoals de holdout methode. Zo wordt per iteratie niet alle mogelijke data gebruikt voor training. Het heeft ook geen controle over het aantal keer elk record gebruikt wordt voor test en training set, sommige records kunnen vaker gebruikt zijn dan andere.



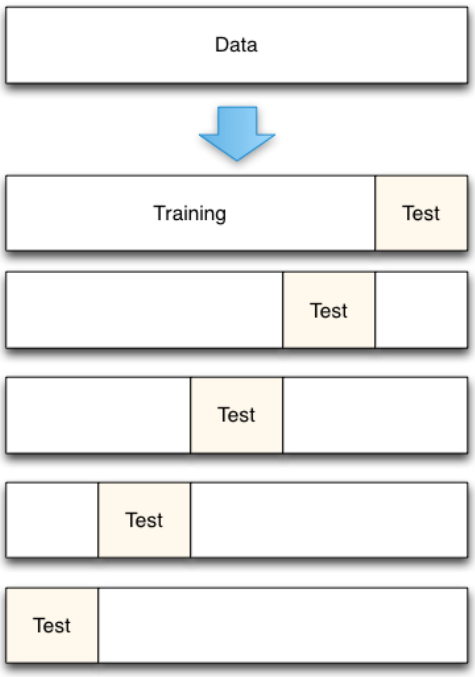
Figuur 3 - Voorbeeld Random subsampling.

6.5.1.3 Cross-Validation

Een alternatief voor random-subsampling is cross-validation. In deze benadering wordt elk record evenveel keer gebruikt voor training en precies één keer voor test set.

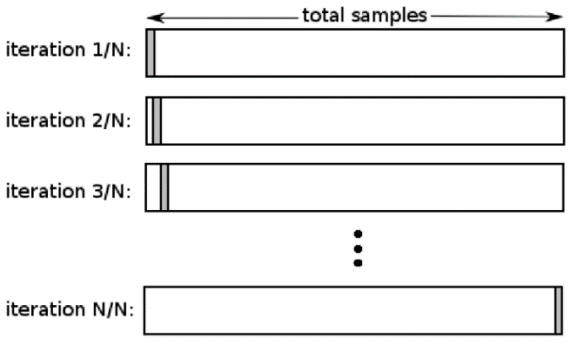
Om deze methode te illustreren veronderstellen we dat de data wordt opgesplitst in 5 even grote partities. We kiezen één van de partities voor test set en d. e overige vier voor training set. We wisselen dan de rollen zodat een andere (nog niet als test set gebruikte) partitie test set wordt. Deze benadering wordt in totaal 5 keer uitgevoerd, vandaar ook de naam **5-fold cross-validation**.

De totale foutenratio wordt verkregen door de fouten per run op te tellen.



Figuur 4 - Voorbeeld 5-fold cross-validation.

Veralgemenen we dit voor k even grote partities, dan spreekt men over **k-fold cross-validation**. Een bijzonder geval is de zogenaamde **leave-one-out** benadering, hier bevat elke test set slechts één record van de dataset, deze benadering heeft het voordeel dat het alle mogelijke data gebruikt voor training, maar deze methode is computationeel zeer kostelijk. De variantie zal hier ook hoog zijn.



Figuur 5 - Leave one out methode.

6.5.1.4 *Bootstrap*

De methodes tot dusver veronderstellen dat men een steekproef neemt uit de training records zonder teruglegging. Hierdoor zijn er geen dubbele records in training en test set. In de bootstrap methode worden training records opgedeeld met teruglegging. Elk gekozen record gaat vooraleer er een nieuwe steekproef wordt genomen, terug in de vijver van alle records. Hierdoor heeft elk record een even grote kans om gekozen te worden.

Als de dataset N records bevat, dan kan wiskundig aangetoond worden dat gemiddeld gezien een bootstrap steekproef van grootte N ongeveer 63,2% van alle records bevat.

Figuur 6 - De .632 bootstrap methode.

6.5.2 Het 'Class imbalance' probleem

Datasets met niet-uitgebalanceerde klasseverdelingen komen in de werkelijkheid vaak voor, denk bijvoorbeeld aan credit-card fraudedetectie. Er zijn veel minder frauduleuze transacties dan legitieme. De maatstaven accuracy en error rate zijn niet geschikt voor dit type van verdelingen. Als bijvoorbeeld in een dataset 1% van alle transacties frauduleus zijn dan kan een model dat voorspelt of een transactie al dan niet frauduleus is een accuracy hebben van 99%, maar toch geen enkele fraudetransactie voorspellen. Om dit probleem tegen te gaan hebben we nood aan een nieuwe metriek, de zogenaamde 'confusion matrix' en ROC-curve.

6.5.3 Confusion matrix

Bij binaire classificatie wordt de klasse die het minst voorkomt, maar doorgaans wel de belangrijkste te voorspellen is, aangeduid als de positieve klasse '**+**', terwijl de meerderheidsklasse aangeduid wordt met '**-**'.

		Voorspelde klasse	
		+	-
Werkelijke klasse	+	f_{++} (TP)	f_{+-} (FN)
	-	f_{-+} (FP)	f_{--} (TN)

Een **confusion matrix** vat het aantal correct voorspelde records vs. het aantal incorrect voorspelde records samen zoals in bovenstaande tabel.

We gebruiken volgende terminologie:

- **True Positive** (TP) of f_{++}, correspondeert naar het aantal positieve records die correct positief werden geclassificeerd.
- **False Negative** (FN) of f_{+-}, correspondeert met het aantal positieve records die foutief als negatief werden geclassificeerd.
- **False Positive** (FP) of f_{-+}, correspondeert met het aantal negatieve records die foutief als positief werden geclassificeerd.
- **True Negative** (TN) of f_{--}, correspondeert met het aantal negatieve records die als negatief werden geclassificeerd.

De aantallen in een confusion matrix kunnen ook in percentages worden uitgedrukt.

De **true positive rate (TPR)** of gevoeligheid is gedefinieerd als

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

De **true negative rate (TNR)** of specificiteit is gedefinieerd als

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

De **false positive rate (FPR)** is

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

De **false negative rate (FNR)** is

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

Recall en precision zijn twee metrieken die gebruikt worden in toepassingen waar het succesvol detecteren van één van de klasses significant belangrijker is dan de andere klasse. Een formele definitie is

$$\textbf{precision}, \boldsymbol{p} = \frac{TP}{TP + FP} \qquad \textbf{recall}, \boldsymbol{r} = \frac{TP}{TP + FN}$$

Hoe hoger de precision, hoe lager het aantal vals-positieve fouten gemaakt door de classifier. Hoe groter de recall, hoe minder positieve records die als negatief geclassificeerd zijn. Recall is in feite equivalent met de TPR.

Het is vaak mogelijk om een model op te stellen die een metriek maximaliseert, maar de andere niet. Precision en recall kunnen samengevat worden in een andere metriek, nl. de F_1-metriek

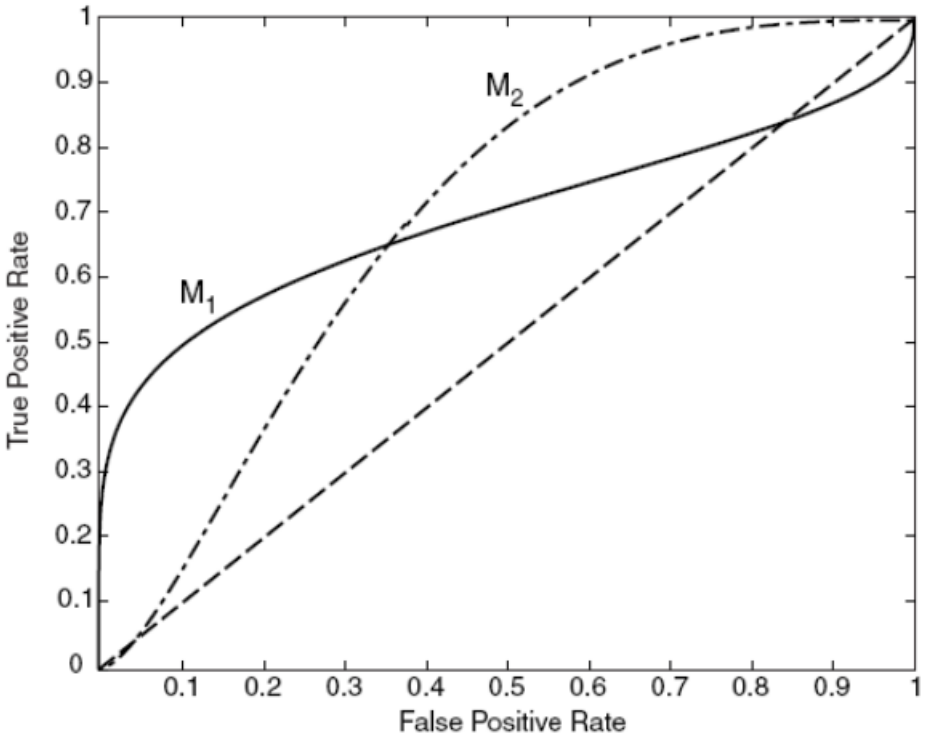
$$F_1 = \frac{2rp}{r + p} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

In principe representeert F_1 het harmonisch gemiddelde tussen recall en precision.

6.5.4 De ROC-curve

6.5.4.1 Principe

De **Receiver Operating Characteristic Curve** (ROC) is een grafische benadering voor de tradeoff tussen TPR en FPR van een classifier. In een ROC curve wordt de TPR weergegeven langs de y-as en de FPR langs de x-as. Elk punt op de kromme correspondeert met een van de modellen, geïnduceerd door de classifier.



Figuur 7 - ROC curve voor twee verschillende classifiers.

In figuur 66 vind je de ROC curves terug voor een paar classifiers, M_1 en M_2. Er zijn nogal wat kritieke punten langs de ROC curve die interpretatie verdienen:

- TPR=0 en FPR=0: het model voorspelt elke instantie als negatief.
- TPR=1 en FPR=1: het model voorspelt elke instantie als positief.
- TPR=1 en FPR=0: het ideale model.

Een goede classifier moet zich zo dicht mogelijk bij de linkerbovenhoek bevinden, terwijl een model dat willekeurig gokt zich meer langs de diagonaal moet bevinden die de punten (TPR=0, FPR=0) en (TPR=1, FPR=1) verbindt. In de figuur moet je opvallen dat M_1 beter is dan M_2 als FPR kleiner is dan 0,36, terwijl M_2 superieur is als FPR groter is dan 0,36. Maar geen van de twee is echt dominant boven de andere.
De oppervlakte onder de ROC curve (AUC) levert een andere benadering voor evaluatie. Als het model perfect is, dan is de oppervlakte onder de ROC curve gelijk aan 1.

6.5.5 Oefeningen

1) Gegeven volgende één-dimensionale data set

x	0,5	3,0	4,5	4,6	4,9	5,2	5,3	5,5	7,0	9,5
y	-	-	+	+	+	-	-	+	-	-

Classificeer het datapunt $x = 5{,}0$ op basis van de 1-, 3-, 5- en 9-nearest neighbors (maak gebruik van majority voting).

2) Maak gebruik van onderstaande dataset en de nearest neighbour classifier met k=1 om voor onderstaande testrecords de waarde te bepalen van de klasse "`GOOD SURF`".

ID	Wave Size (ft)	Wave Period (secs)	Wind Speed (mph)	Good Surf
1	6	15	5	yes
2	1	6	9	no
3	7	10	4	yes
4	7	12	3	yes
5	2	2	10	no
6	10	2	20	no

Veronderstel dat het model gebruik maakt van de Euclidische afstand om de dichtste buren te vinden, welke voorspelling geeft het model dan voor volgende nieuwe records?

ID	Wave Size (ft)	Wave Period (secs)	Wind Speed (mph)	Good Surf
Q1	8	15	2	?
Q2	8	2	18	?
Q3	6	11	4	?

3) In volgende tabel vind je voorspellingen terug (`Prediction`) voor een bepaalde categorie (`Target`) op basis van een model. Bepaal de gevraagde performantiematen.

ID	Target	Prediction	ID	Target	Prediction	ID	Target	Prediction
1	false	false	8	true	true	15	false	false
2	false	false	9	false	false	16	false	false
3	false	false	10	false	false	17	true	false
4	false	false	11	false	false	18	true	true
5	true	true	12	true	true	19	true	true
6	false	false	13	false	false	20	true	true
7	true	true	14	true	true			

a. Confusion matrix en misclassificatieratio (aantal verkeerd geclassificeerde t.o.v. totaal aantal geclassificeerde records).
b. Precision, recall en f_1-maat.

4) Veronderstel dat je werkt op een spam detectiesysteem. Het probleem heb je geformuleerd als een classificatietaak waar "Spam" de positieve klasse is en "not-Spam" de negatieve klasse. Je trainingdataset bevat 1000 e-mails, 99% van deze worden als "not-Spam" geclassificeerd en 1% als "Spam".
   a. Wat is de nauwkeurigheid (accuracy) van een classifier die altijd "not-Spam" geeft?
   b. De fractie van spam e-mails die correct als "Spam" geclassificeerd worden, wordt gemeten door de "recall" waarde. Bereken de recall van de classifier uit (a).