Voorspellen van webwinkel aankopen met een Random Forest

Dorenda Slof
Erasmus Universiteit Rotterdam
Econometrie en Operationele Research

30 juni 2014

Samenvatting

In dit empirische onderzoek voorspellen we of een bezoek aan een webwinkel eindigt in een aankoop of niet. We maken gebruik van een beslissingsboom en een Random Forest om het al dan niet doen van een aankoop van een klant te voorspellen. Het idee van een Random Forest is dat een groot aantal beslissingsbomen samen een aankoop van een klant voorspelt. De voorspellingen van een Random Forest zijn over het algemeen beter dan de voorspellingen van een beslissingsboom. De voorspelprestaties van deze modellen vergelijken we met de voorspelprestaties van een Logit model. De resultaten leren ons dat het Random Forest model het al dan niet doen van een aankoop van een klant beter voorspelt, dat het schatten van een Random Forest een kortere rekentijd vergt en ongevoelig is voor de aanwezigheid van ruis, missende waarden en outliers in de dataset. Het voordeel van een Logit model is dat we de relevantie van de variabelen en de invloed van de variabelen op het al dan niet doen van een aankoop direct kunnen analyseren.

Inhoudsopgave

1 Introductie

Het correct voorspellen van het al dan niet doen van een aankoop in een webwinkel wordt steeds belangrijker voor ondernemers. Als een ondernemer in een vroeg stadium al kan inschatten of de klant daadwerkelijk een aankoop zal gaan doen, kan dit erg waardevolle informatie zijn. Webwinkels kunnen invloed uitoefenen op de klant en ze extra impulsen geven om een product te kopen, meer producten te kopen, of juist een duurder product te kopen. Ook is het voor de webwinkel lucratief om de relatie met de klant soepel te laten verlopen. Zo zal het een webwinkel vijf keer meer geld kosten om een nieuwe klant te werven, dan om een bestaande klant een product te laten kopen [1]. Tot nu toe is er door van den Poel en Buckinx een uitgebreid onderzoek gedaan naar relevante variabelen die het al dan niet doen van een aankoop in een webwinkel voorspellen met een Logit model [2]. Daarnaast bestaat er al een Logit model over de data die we in dit onderzoek zullen gebruiken en kennen we de relevante variabelen die in dat onderzoek naar voren kwamen [3]. Waar de focus van beide onderzoeken op de interpretatie van de variabelen lag, is het doel van dit onderzoek juist om een model te creëren dat een zo groot mogelijke voorspelkracht heeft.

Bij het schatten van een parametrisch model zullen er altijd incorrecte aannames gemaakt moeten worden. Om deze aannames los te laten zullen we gebruik maken van een beslissingsboom en een Random Forest om het al dan niet doen van een aankoop van een klant te voorspellen.

Het vervolg van het verslag ziet er als volgt uit. De data en methodologie worden besproken in sectie 2, in sectie 3 zullen we de beslissingsboom en het Random Forest introduceren. In sectie 4 worden de resultaten besproken, in sectie 5 geven we de conclusie en in sectie 6 bespreken we de beperkingen van dit onderzoek.

2 Data en methodologie

2.1 Data

De data zijn verkregen van een onbekende Duitse webwinkel over een periode van één of meerdere weekenden van vrijdagavond zes uur tot zondagavond zes uur. We hebben te maken met 50.000 klanten die in deze periode al dan niet een aankoop doen. De webwinkel slaat een groot aantal gegevens van elke sessie op, waar een sessie staat voor een onafgebroken bezoek van een klant aan de webwinkel. De variabelen die in dit onderzoek gebruikt zullen worden zijn af te lezen in *tabel 1*. Na het verwijderen van de observaties met een tijdsduur van 0 houden we 48.736 unieke observaties over.

Variabele	Type	Omschrijving
Klikdata		
cCount	geheel getal	Aantal aangeklikte producten
day	geheel getal	Dag waarop de sessie start
hour	geheel getal	Uur waarop de sessie start
cMaxPrice*	continu	Maximumprijs van alle aangeklikte producten
cMinPrice*	continu	Minimumprijs van alle aangeklikte producten
cSumPrice*	continu	Som van de prijzen van alle aangeklikte producten
itemsClickedPerMinute	continu	Gemiddeld aantal aangeklikte producten per minuut
totalDuration	continu	Totale duur van de sessie
rangeCPrice	continu	Verschil tussen de maximum- en de minimumprijs van alle aangeklikte producten
Winkelmanddata		
bCount	geheel getal	Aantal producten in het winkelmandje
clicksPerBasketedItem	continu	Gemiddeld aantal aangeklikte producten per product in het winkelmandje
bMaxPrice*	continu	Maximumprijs van alle producten in het winkelmandje
bMinPrice*	continu	Minimumprijs van alle producten in het winkelmandje
bSumPrice*	continu	Som van de prijzen van alle producten in het winkelmandje
itemsBasketedPerMinute	continu	Gemiddeld aantal producten in het winkelmandje per minuut
rangeBPrice	continu	Verschil tussen de maximum- en de minimumprijs van alle producten in het winkelmandje
Aankoopprocesdata		
availability*	categoriaal	Leverbaarheid van het product
maxBStep	categoriaal	Hoogst geregistreerde stap in het aankoopproces
onlineStatus*	categoriaal	Het al dan niet online zijn van de klant tijdens het aankoopproces
Klantprofieldata		
age*	continu	Leeftijd van de klant
accountLifetime*	continu	Aantal maanden dat het profiel van de klant bestaat
customerScore*	continu	Score die de webwinkel toewijst aan de klant ter evaluatie van het historisch aankoopgedrag
noOfPayments*	geheel getal	Aantal betalingen aan de webwinkel sinds het aanmaken van het klantprofiel
paymentsPerMonth	geheel getal	Gemiddeld aantal betalingen aan de webwinkel per maand sinds het bestaan van het profiel
lastOrder*	geheel getal	Aantal dagen sinds de laatste aankoop
gender*	categoriaal	Geslacht
maxVal*	continu	Maximum toegelaten aankoopprijs voor de klant
recurrentCustomer	geheel getal	Aantal keer dat een klant de webwinkel bezoekt tijdens de meetperiode

* variabele vertoont niet-structureel missende waarden

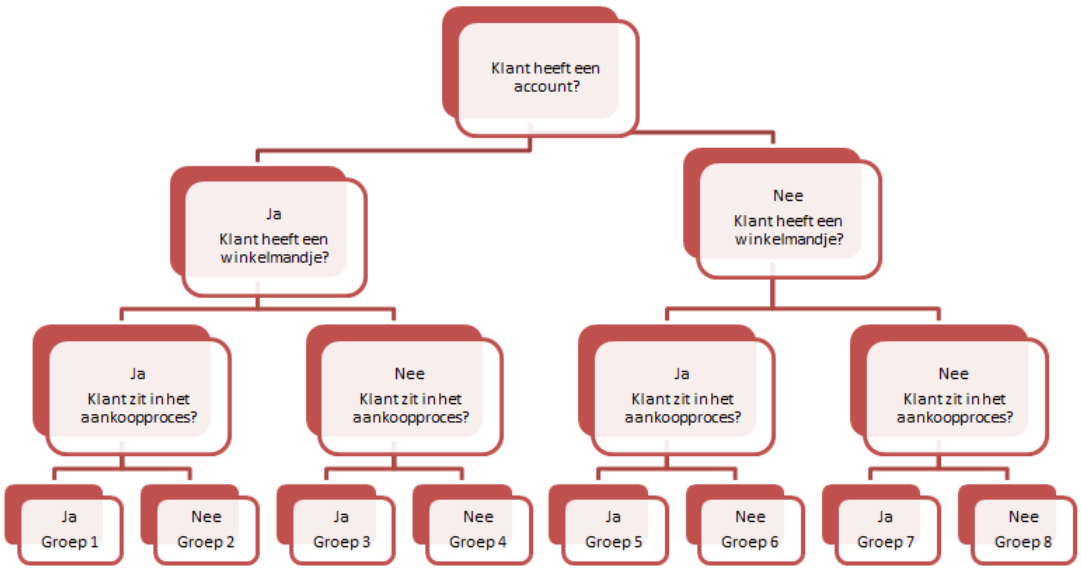
Tabel 1: Variabelen

2.2 Missende waarden

De variabelen in de dataset vertonen een groot aantal missende waarden. Beslissingsbomen en het Random Forest staan erom bekend dat ze niet gevoelig zijn voor missende waarden in de dataset [4]. Dit betekent dat de modellen vergelijkbare resultaten produceren als de missende waarden in de dataset worden vervangen door een nummerieke waarde, of als de missende waarden in de dataset blijven staan. Om dit te testen zullen we modellen schatten met de ruwe dataset die missende waarden bevat en modellen met een dataset waarin de missende waarden geïmplementeerd zijn.

De missende waarden in de dataset zijn onder te verdelen in structureel missende waarden en niet-structureel missende waarden. Structureel missende waarden komen voor wanneer de missende waarden door de datastructuur verklaard kunnen worden. Niet-structureel missende waarden zijn niet te verklaren aan de hand van de datastructuur. Uit de datastructuur blijkt dat een aantal variabelen tegelijk gemeten wordt zodra een klant een bepaalde actie doet. Als de klant deze actie tijdens een sessie niet doet, wordt een groep variabelen automatisch niet gemeten door de webwinkel. Op deze manier kunnen we onderscheid maken tussen vier groepen variabelen: klikdata, winkelmanddata, aankoopprocesdata en klantprofieldata. In *tabel 1* staat een overzicht van elke groep variabelen.

Om de niet-structureel missende waarden in de dataset te kunnen implementeren, gebruiken we de aanname van de vier groepen variabelen. Zo zal bijvoorbeeld in de i-de observatie de variabele 'gender' een missende waarde vertonen als de klant niet heeft ingelogd. Ook zal de variabele 'bCount' een missende waarde bevatten als de klant niets in zijn winkelmandje heeft gedaan. Door de data te analyseren en op te splitsen op de gedane acties van de klant verkrijgen we een boomstructuur, te zien in *figuur 1.* Aan de hand van deze boomstructuur kunnen we de data onderverdelen in acht verschillende subdatasets. De structureel missende waarden zijn door het opsplitsen van de data verdwenen, terwijl de niet-structureel missende waarden worden vervangen door het k-nearest neighbour algoritme zoals beschreven in [3].



Figuur 1: Boomstructuur

Na het toepassen van het k-nearest neighbour algoritme hebben we acht subdatasets waarin de niet-structureel missende waarden geïmplementeerd zijn. Om een model te schatten voegen we deze acht subdatasets weer samen tot een gehele dataset. De structureel missende waarden in de gehele dataset zullen niet vervangen worden door een nummerieke waarde, omdat deze missende waarden een betekenis hebben.

3 Model

De afhankelijke variabele y is een binaire variabele die de waarden 1 en 0 kan aannemen, waar de klasse 1 staat voor een aankoop en de klasse 0 voor geen aankoop. X_i is de $1 \times k$ vector met verklarende variabelen voor observatie i, waarbij k het aantal verklarende variabelen is. De verklarende variabelen worden ook wel de attributen genoemd. De matrix X met de data van de Duitse webwinkel is $n \times k$ groot, waar n staat voor het aantal observaties. De data zullen geclassificeerd worden in de beslissingsboom. Classificatie is het toewijzen van een klasse aan elke observatie i.

De beslissingsboom en het Random Forest zullen geschat worden met data uit de trainingsdataset en vervolgens getoetst worden op data uit een later verkregen testdataset. De trainingsdataset bevat 48.736 observaties, waar de testdataset 4.937 observaties bevat.

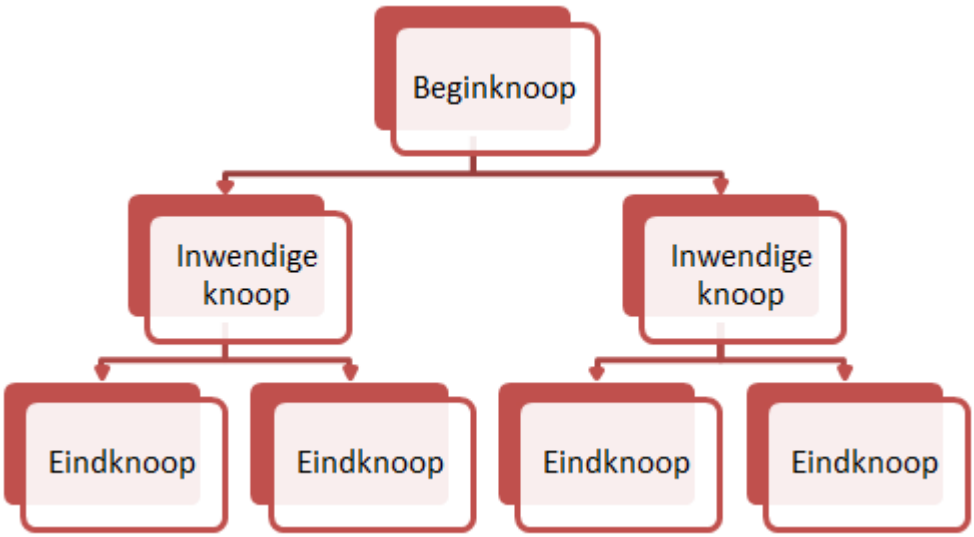
Ook het Logit model is geschat op de data uit de gehele trainingsdataset en vervolgens getoetst op de data uit een later verkregen testdataset.

3.1 Beslissingsboom

De gegevens die nodig zijn om het classificatieprobleem op te lossen zijn de matrix met verklarende variabelen X en de vector met afhankelijke variabele y. De eerste classificatietechniek die we gaan gebruiken is de beslissingsboom. Voor het beschrijven van de werking van een beslissingsboom introduceren we eerst de volgende begripsomschrijvingen:

1. De beginknoop, deze heeft geen inkomende pijlen en nul of meer uitgaande pijlen.
2. De inwendige knopen, deze hebben allen precies één inkomende pijl en twee of meer uitgaande pijlen.
3. De eindknopen, deze hebben allen precies één inkomende pijl en nul uitgaande pijlen.

Dit is grafisch weergegeven in *figuur 2*.



Figuur 2: Toelichting van de knopen in een beslissingsboom

De procedure begint uiteraard bij de beginknoop, hier voeren we alle data in de beslissingsboom in. Vervolgens bekijken we welke variabele de data het beste op kan splitsen in twee nieuwe knopen. Dan herhalen we de bovenstaande stap totdat aan een stopcriterium is voldaan en de eindknoop gerealiseerd is. Na het classificeren van de trainingsdata heeft elke eindknoop nul of meer observaties die resulteren in een aankoop en nul of meer observaties die niet resulteren in een aankoop. De vaakst voorkomende klasse in elke eindknoop kan nu worden vastgesteld. Vervolgens kan de

testdataset in de beslissingsboom geïmplementeerd worden. De klasse van observatie i uit de testdataset wordt achterhaald door de vaakst voorkomende klasse in de bijbehorende eindknoop als voorspelling te gebruiken.De ideale situatie zou zijn als de observaties die in dezelfde eindknoop zitten allen in dezelfde klasse zitten. Voor de implementatie van de bovenstaande procedure gebruiken we een Matlab functie die gebaseerd is op het CART algoritme van Breiman et al. [6].

Het CART algoritme splitst de data op in twee nieuwe knopen, zodat de twee nieuwe knopen minder variatie hebben in de afhankelijke variabele y. Bij binaire attributen, zoals de variabele 'gender', is dit vrij eenvoudig. Hier zal het algoritme in de knoop twee onderliggende knopen genereren: 'male' en 'female'. Bij nominale attributen zal het algoritme verschillende categorieën in één onderliggende knoop plaatsen, zoals in het onderstaande voorbeeld wordt uitgewerkt. Het algoritme zal bij continue attributen de knoop splitsen op een specifieke grenswaarde. Zo zou de variabele 'lastOrder' gesplitst kunnen worden op een waarde groter of gelijk aan 40 en een waarde kleiner dan 40.

Om de beste splitsing te verkrijgen gebruiken we de Gini index die als volgt gedefinieerd is [4]:

$$Gini(t) = 1 - \sum_{i=1}^{c} [p(i|t)]^2$$

waar c het aantal klassen is, i de klasse in knoop t aanduidt en $p(i|t)$ de fractie observaties uit klasse i die toebehoren aan knoop t. De Gini index meet de onzuiverheid van een voorgestelde splitsing in de knoop. Hoe kleiner deze waarde van onzuiverheid is, hoe kleiner de Gini indexwaarde en hoe schever de klassenverdeling in de onderliggende knopen is. Een kleinere Gini indexwaarde geeft dus een betere splitsing van de attributen aan. De grootste onzuiverheid van een splitsing doet zich voor bij een splitsing waar de klassenverdeling $(0,5; 0,5)$ is.

Om de bovenstaande uitleg te verduidelijken geven we een voorbeeld van een optimale splitsing met een fictieve dataset. Stel dat we de volgende dataset hebben, zoals weergegeven in *tabel 2*.

Observatie	Geslacht	Dag	Aankoop
1	Man	Vrijdag	Nee
2	Vrouw	Zaterdag	Nee
3	Vrouw	Vrijdag	Nee
4	Man	Zaterdag	Nee
5	Vrouw	Zondag	Ja
6	Vrouw	Zaterdag	Nee
7	Man	Zondag	Ja
8	Vrouw	Vrijdag	Ja
9	Vrouw	Zaterdag	Nee
10	Vrouw	Vrijdag	Ja

Tabel 2: Fictieve dataset

We hebben hier de nominale variabele 'dag', die de waarden {vrijdag, zaterdag, zondag} aan kan nemen. Als we deze nominale variabele op willen splitsen in nieuwe knopen kan dit op verschillende manieren gedaan worden. Aangezien het CART algoritme alleen gebruikt maakt van binaire splitsingen hebben we ook alleen binaire splitsingen in ons voorbeeld gebruikt. Deze splitsingen zijn weergegeven in *figuur 3*.

(a) Optie A (b) Optie B (c) Optie C

Figuur 3: Opsplitsen van nominale attributen

Om de optimale splitsing te kiezen gebruiken we de Gini index om de onzuiverheid van een splitsing te berekenen. Daarbij krijgen we de uitkomsten zoals weergegeven in *tabel 3*.

	Dag	
	{Vrijdag}	**{Zaterdag, zondag}**
y=0	2	4
y=1	2	2
Gini	0.469	

(a) Optie A

	Dag	
	{Zaterdag}	**{Vrijdag, zondag}**
y=0	4	2
y=1	0	4
Gini	0.267	

(b) Optie B

	Dag	
	{Zondag}	**{Vrijdag, zaterdag}**
y=0	0	6
y=1	2	2
Gini	0.300	

(c) Optie C

Tabel 3: Gini indexwaarden bij de fictieve dataset

Waar $y = 0$ de klasse niet-aankoop voorstelt en $y = 1$ de klasse aankoop weergeeft. De gewogen Gini index van optie A wordt op de volgende manier berekend:

$$Gini_A = (1 - (2/4)^2 - (2/4)^2) \times 4/10 + (1 - (4/6)^2 - (2/6)^2) \times 6/10 = 0.469$$

Intuïtief is optie B de beste optie, aangezien de klassen daar schever verdeeld zijn dan bij de andere opties. De Gini index geeft bij optie B ook de laagste waarde, wat betekent dat het opsplitsen in {zaterdag} en {vrijdag, zondag} de optimale splitsing zou zijn bij de variabele 'dag'.

De beslissingsboom zal stoppen met groeien als aan een stopcriterium voldaan is. Zo kan de lengte van de beslissingsboom a priori vastgesteld worden, of kan het aantal minimale observaties per knoop van te voren bepaald worden. In de uitgangssituatie zal de boom stoppen met groeien als:

1. Alle observaties in de huidige knoop behoren tot dezelfde klasse.
2. Alle attributen in de huidige knoop dezelfde waarden hebben.

Om de voorspelkracht van een beslissingsboom te kunnen schatten maken we gebruik van de generalization error. De generalization error is het verwachte aantal incorrect voorspelde observaties van de testdataset. De generalization error wordt gebruikt als de testdataset van het model nog niet voorhanden is. Om de generalization error voor de beslissingsboom te kunnen schatten hebben we uit de originele trainingsdataset een nieuwe trainingsdataset en een testdataset nodig. Deze datasets creëren we door de originele trainingsdataset in tien subsets op te delen. Vervolgens

schatten we een model op de data van negen subdatasets en wordt de overgebleven subdataset als testdataset gebruikt. Deze testdataset implementeren we in het model, waarna we het aantal incorrect voorspelde observaties berekenen. Deze procedure herhalen we nog negen keer, waardoor elke subdataset één keer als testdataset gebruikt wordt. De generalization error wordt dan geschat met de cross-validation error die gegeven wordt door:

$$\text{Cross-validation error} = \sum_{i=1}^{10} \frac{\text{aantal incorrect voorspelde observaties uit testdataset } i}{\text{aantal observaties in de originele trainingsdataset}}$$

Het voordeel van een beslissingsboom ten opzichte van een parametrisch model is dat er geen aannames nodig zijn over de verdeling van de storingstermen of de attributen. Daarnaast is het algoritme erg robuust voor de aanwezigheid van ruis in de data en verslechtert de voorspelkracht niet als er irrelevante variabelen in het model worden opgenomen. Een beslissingsboom is een intuïtief model dat gemakkelijk in gebruik is. Daarnaast vergt het schatten van een beslissingsboom weinig rekentijd.

3.2 Random Forest

Het gebruik van een beslissingsboom heeft ook enkele nadelen. Zo kan een model dat zeer goed bij de trainingsdataset past een slecht model zijn om mee te voorspellen. De bijbehorende beslissingsboom is dan erg groot en kan knopen bevatten die gecreëerd zijn door specifieke gevallen in de trainingsdataset. De beslissingsboom laat zich dan slecht generaliseren naar andere data. Ook zullen de observaties uit de testdataset met deze beslissingsboom slecht voorspeld worden. De generalization error van dit model is dan erg groot. Dit fenomeen staat ook wel bekend als overfitting van de data. Om dit probleem te vermijden zal de gegenereerde beslissingsboom een proces moeten ondergaan dat pruning heet, de boom zal dan ingekort worden.

Een andere manier om overfitting van de data te voorkomen en de generalization error te verkleinen is het gebruik van een Random Forest. Een Random Forest wordt gecreëerd door een groot aantal beslissingsbomen te schatten bij de data [5]. Nadat een groot aantal beslissingsbomen gecreëerd is, kan het Random Forest bij observatie i de klasse y_i voorspellen. De individuele beslissingsbomen in het Random Forest schatten elk de klasse y_i bij de attributen X_i, waarna de meest frequent voorspelde klasse de uiteindelijke voorspelling voor observatie i wordt.

De generalization error van het Random Forest zal kleiner zijn als de correlatie tussen de individuele beslissingsbomen in het Random Forest klein is [5]. De individuele beslissingsbomen moeten onderling dus verschillen, terwijl de voorspelkracht van elke individuele boom wel zo groot mogelijk moet blijven. Om aan deze voorwaarden te voldoen schatten we het Random Forest model iets anders dan een enkele beslissingsboom.

Ten eerste zal elke individuele beslissingsboom in het Random Forest gecreëerd worden met andere data uit de gehele trainingsdataset. Om voor elke boom andere trainingsdata te genereren maken we gebruik van bootstrapping. Een boots-

trap steekproef wordt gecreëerd door herhaaldelijk, met terugleggen, volgens een uniforme verdeling, observaties uit de originele trainingsdataset te trekken en in de subtrainingsdataset te plaatsen. Elke trainingsdataset is zo groot als de gehele dataset en heeft dus observaties die meer dan één keer voorkomen. Ook zijn er observaties in de dataset die juist niet gebruikt zijn in de trainingsdataset. Deze observaties vormen de testdataset voor de beslissingsboom en gebruiken we om de generalization error te kunnen schatten. Een bootstrap steekproef bevat gemiddeld 63% van de originele trainingsdataset, omdat elke observatie een kans van $1-(1-1/N)^N$ heeft om geselecteerd te worden in de nieuwe trainingsdataset, waar N gelijk is aan het aantal observaties in de originele trainingsdataset. Als de trainingsdataset veel ruis en specifieke gevallen kent, reduceert het trekken van een bootstrap steekproef de generalization error van het Random Forest [4].

Ten tweede zullen de beslissingsbomen in het Random Forest anders gevormd worden dan een individuele beslissingsboom. Zo zal er bij een individuele beslissingsboom in een knoop altijd gesplitst worden op de variabele die de kleinste Gini indexwaarde geeft. Bij een beslissingsboom in het Random Forest gaat dit iets anders. Bij de keuze voor een splitsing in een knoop creëren we eerst een vector Q met daarin m willekeurig geselecteerde variabelen. De knoop zal voor elke variabele in Q een splitsing genereren en de bijbehorende Gini indexwaarde berekenen. De variabele die de kleinste Gini indexwaarde geeft zal de variabele zijn die de knoop in twee nieuwe knopen opsplitst. De boom zal doorgroeien tot maximale grootte zonder gebruik te maken van een pruning methode [5]. Door het gebruik van de vector Q met willekeurig geselecteerde variabelen zal de correlatie tussen de beslissingsbomen in het Random Forest kleiner worden. Immers, als we elke knoop verplicht op één willekeurig geselecteerde variabele opsplitsen zullen de bomen in het Random Forest sterk verschillen. Terwijl de bomen in het Random Forest nagenoeg gelijk zijn als we elke knoop mogen opsplitsen op een variabele die komt uit een grote subset van willekeurig geselecteerde variabelen. Als we een kleine waarde van m kiezen worden de bomen in het Random Forest gedwongen om de knopen op te splitsen op een variabele die niet de beste splitsing zal genereren, enkel de beste splitsing van de vector met m willekeurige variabelen. Dit geeft een grotere variatie in de beslissingsbomen, zodat de correlatie tussen de beslissingsbomen in het Random Forest verminderd wordt.

De belangrijkste reden om bootstrapping te gebruiken is omdat het de accuracy van het model blijkt te vergroten als we gebruik maken van willekeurig geselecteerde variabelen in de vector Q [5], waar accuracy wordt gegeven door:

$$\text{Accuracy} = \frac{p_{00} + p_{11}}{p_{00} + p_{11} + p_{01} + p_{10}}$$

waar p_{00} staat voor het aantal correct voorspelde niet-aankopen, p_{11} voor het aantal correct voorspelde aankopen, p_{01} voor het aantal onterecht voorspelde aankopen en p_{10} voor het aantal onterecht voorspelde niet-aankopen. Deze waarden worden geschat door de testdataset in het Random Forest model te implementeren.

Ook geeft het gebruik van bootstrapping ons een schatter voor de generaliza-

tion error van het Random Forest. De generalization error is het verwachte aantal incorrect voorspelde observaties van het model en wordt geschat met de out-of-bag schatter [5]. Deze schatter wordt berekend door de met bootstrapping gecreëerde testdataset in de beslissingsboom te implementeren en de voorspelde klassen te vergelijken met de daadwerkelijke waarde van y. De out-of-bag schatter voor de generalization error wordt gegeven door [7]:

$$\text{Out-of-bag schatter} = \frac{\hat{p}_{01} + \hat{p}_{10}}{\hat{p}_{00} + \hat{p}_{11} + \hat{p}_{01} + \hat{p}_{10}}$$

waar $\hat{p}_{00}$ staat voor het aantal correct voorspelde niet-aankopen, $\hat{p}_{11}$ voor het aantal correct voorspelde aankopen, $\hat{p}_{01}$ voor het aantal onterecht voorspelde aankopen en $\hat{p}_{10}$ voor het aantal onterecht voorspelde niet-aankopen. Deze waarden worden verkregen door de voorspelde aankopen in de met bootstrapping gegenereerde testdataset te vergelijken met de gerealiseerde aankopen.

Het voordeel van een Random Forest ten opzichte van een beslissingsboom is dat het Random Forest over het algemeen betere resultaten oplevert dan een beslissingsboom [4]. Stel dat we 25 beslissingsbomen schatten die allemaal een foutpercentage van $\epsilon = 0.35$ kennen. Het Random Forest schat de klasse y van observatie i door de vaakst voorspelde klasse van de 25 individuele beslissingsbomen te gebruiken als uiteindelijke voorspelling. Als deze beslissingsbomen nagenoeg hetzelfde zijn, dan zal het Random Forest evenveel observaties verkeerd schatten als de individuele beslissingsbomen. Het foutpercentage blijft dan $\epsilon = 0.35$. Als de beslissingsbomen in het Random Forest een kleine correlatie kennen, dan zal het Random Forest alleen een foute schatting van de klasse geven als meer dan de helft van de individuele beslissingsbomen een foute schatting geeft. Het foutpercentage van het Random Forest wordt in dit geval:

$$e_{RandomForest} = \sum_{i=13}^{25} \binom{25}{i} \epsilon^i (1-\epsilon)^{25-i} = 0.06$$

welke lager is dan het foutpercentage van een individuele beslissingsboom.

3.3 Modelkeuze

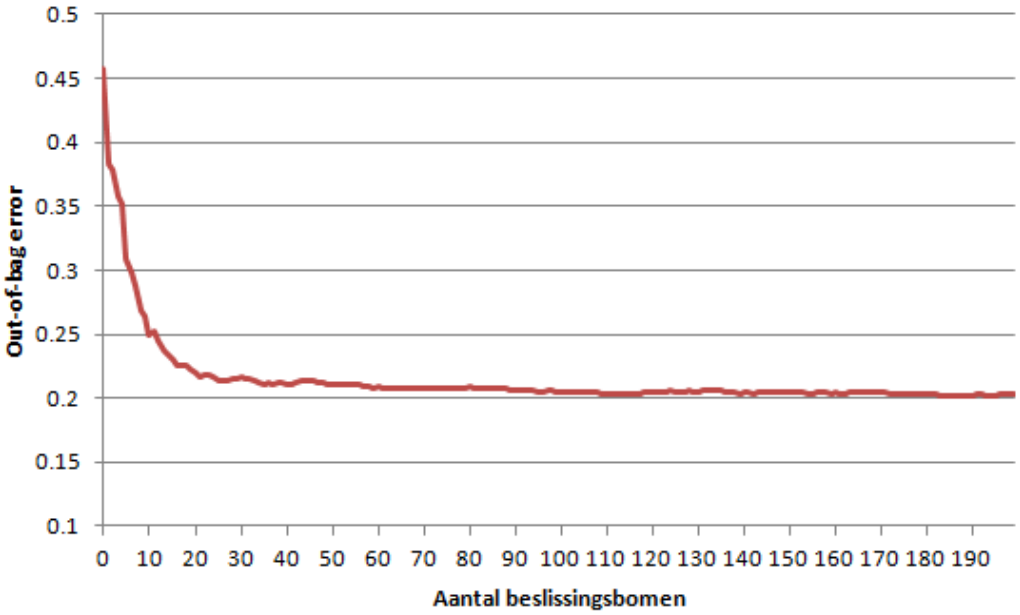
Voor het voorspellen van de klassen van de observaties in de testdataset zullen we verschillende modellen schatten. Ten eerste het Logit model waarin de missende waarden geïmplementeerd zijn dat al geschat is in [3]. Het Logit model waar nog niet-structureel missende waarden in voorkomen doet het zodanig slechter, dat we deze niet hebben meegenomen in dit onderzoek.

Ook schatten we een Random Forest met daarin *ntree* beslissingsbomen. Een Random Forest zou ongevoelig moeten zijn voor het bestaan van missende waarden in de dataset. Om dit te testen zullen we twee keer een Random Forest schatten: één met missende waarden in de dataset en één waar de niet-structureel missende

waarden in de dataset geïmplementeerd zijn.

Tot slot zullen we ook de voorspelkracht van een individuele beslissingsboom evalueren, om te analyseren of dit simpelere model eventueel een vergelijkbare voorspelkracht heeft met het Random Forest. Om een beslissingsboom te schatten gebruiken we alle trainingsdata. Om overfitting van het model te voorkomen prunen we de beslissingsboom door de generalization error te berekenen voor een voorgestelde verwijdering van een knoop. De beslissingsboom met de kleinste generalization error zal het uiteindelijke model worden. Ook hier zullen we twee beslissingsbomen schatten: één met missende waarden in de dataset en één waar de niet-structureel missende waarden in de dataset geïmplementeerd zijn.

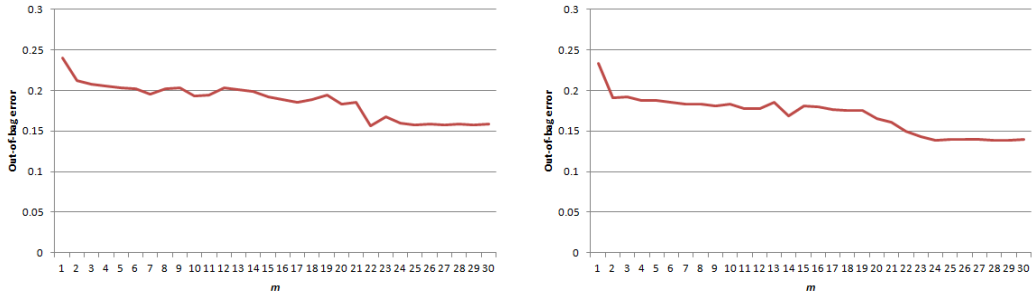
Het aantal beslissingsbomen in het Random Forest moet groot genoeg zijn om de out-of-bag error te laten convergeren. In *figuur 4* zien we dat de out-of-bag error vanaf 50 beslissingsboom convergeert naar een waarde van ongeveer 0.21. Om er zeker van te zijn dat de out-of-bag error convergeert, zullen we in dit onderzoek gebruik maken van 150 beslissingsbomen in het Random Forest. De rekentijd is nog steeds erg snel.



Figuur 4: Convergentie van de out-of-bag error

Om de out-of-bag error in het Random Forest zo klein mogelijk te houden is het belangrijk om de correlatie tussen de beslissingsbomen in het Random Forest klein te houden en de kracht van de individuele beslissingsbomen groot te houden. Om de correlatie tussen de beslissingsbomen klein te houden moet het aantal variabelen m in de vector Q klein blijven. Zo zouden we kunnen kiezen voor m gelijk aan 1 voor

een zo klein mogelijke correlatie. De keerzijde is dat de individuele beslissingsbomen dan slecht zullen voorspellen, aangezien ze niet op de beste variabelen opsplitsen, maar op een willekeurig geselecteerde variabele. Over het algemeen wordt een waarde aangeraden van $m = log_2(k)+1$, waar k het aantal verklarende variabelen is [5]. Om te onderzoeken of deze waarde daadwerkelijk de beste voorspelprestatie geeft zullen we voor verschillende waarden van m de out-of-bag error berekenen.



(a) Random Forest met missende waarden in de dataset

(b) Random Forest waar de missende waarden in de dataset geïmplementeerd zijn

Figuur 5: Out-of-bag error bij verschillende waarden van m

In *figuur 5* zien we dat het Random Forest met missende waarden in de dataset bij $m = 22$ de kleinste out-of-bag error geeft. Hier zien we ook dat een te kleine waarde van m een verslechterde kracht van het model veroorzaakt. Daarnaast zien we dat het Random Forest waarin de missende waarden in de trainingsdataset geïmplementeerd zijn de kleinste out-of-bag error heeft als $m = 24$. Deze waarden van m zijn flink hoger dan de aanbevolen waarde $m = log_2(k)+1 = log_2(30)+1 \approx 6$. In de volgende sectie zullen we de resultaten van verschillende waarden van m vergelijken.

4 Resultaten

De voorspelprestatie van het model zijn op een aantal manieren getoetst. Alle resultaten zijn verkregen door de testdataset te implementeren in de geschatte modellen. Allereerst zijn de voorspelling-realisatietabellen in *tabel 4* weergegeven.

	Voorspeld		
Waargenomen	$\hat{y}_i = 1$	$\hat{y}_i = 0$	
$y_i = 1$	1669	645	2314
$y_i = 0$	356	2155	2511
	2025	2800	4825

(a) Logit model waar de missende waarden in de dataset geïmplementeerd zijn

	Voorspeld		
Waargenomen	$\hat{y}_i = 1$	$\hat{y}_i = 0$	
$y_i = 1$	2109	216	2325
$y_i = 0$	589	2023	2612
	2698	2239	4937

(b) Beslissingsboom met missende waarden in de dataset

	Voorspeld		
Waargenomen	$\hat{y}_i = 1$	$\hat{y}_i = 0$	
$y_i = 1$	2175	150	2325
$y_i = 0$	790	1822	2612
	2965	1972	4937

(c) Random Forest met missende waarden in de dataset, m=6

	Voorspeld		
Waargenomen	$\hat{y}_i = 1$	$\hat{y}_i = 0$	
$y_i = 1$	2117	208	2325
$y_i = 0$	553	2059	2612
	2670	2267	4937

(d) Random Forest met missende waarden in de dataset, m=22

Tabel 4: Voorspelling-realisatie tabellen

Zoals we in *tabel 4* kunnen zien voorspellen de Random Forest modellen met missende waarden in de dataset beter dan het Logit model. Het Logit model voorspelt te vaak geen aankoop, terwijl er wel een aankoop gerealiseerd wordt. Het aantal correct voorspelde klassen is lager dan bij de beslissingsboom en bij beide Random Forest modellen. Het Logit model voorspelt daarentegen minder onterecht voorspelde aankopen dan de beslissingsboom en beide Random Forest modellen.

Ook zien we in *tabel 4(c)* en *tabel 4(d)* dat bij een grotere waarde van m het aantal voorspelde aankopen sterk daalt. Het aantal correct voorspelde aankopen daalt echter minimaal. Het Random Forest met een grotere waarde van m genereert betere voorspellingen dan het Random Forest met een kleinere waarde van m.

Als we *tabel 4(b)* vergelijken met *tabel 4(d)* zien we dat een beslissingsboom wel degelijk slechter presteert dan een Random Forest. Het aantal onterecht voorspelde aankopen en het aantal onterecht voorspelde niet-aankopen is kleiner bij het Random Forest dan bij de beslissingsboom. Het gebruik van een Random Forest in plaats van een beslissingsboom geeft bij deze dataset een betere voorspelprestatie.

Om de voorspelkracht van de modellen verder te evalueren maken we gebruik van de accuracy, precision, recall en de F_1-waarde van de modellen. De accuracy is het relatieve aantal correct voorspelde klassen bij de ingevoerde waarden van X uit de testdataset en is al eerder gedefinieerd. De precision is het relatieve aantal correct voorspelde aankopen ten opzichte van het aantal voorspelde aankopen en wordt gegeven door:

$$\text{Precision} = \frac{p_{11}}{p_{11} + p_{01}}$$

De recall is het aantal correct voorspelde aankopen ten opzichte van het aantal

waargenomen aankopen en wordt gegeven door:

$$\text{Recall} = \frac{p_{11}}{p_{11} + p_{10}}$$

De F_1-waarde is het harmonisch gemiddelde van precision en recall en wordt gegeven door:

$$F_1 = \frac{2 \times p_{11}}{\text{recall} + \text{precision}}$$

De voorspelprestaties van de modellen zijn in *tabel 5* en *tabel 6* te vinden.

	Boom	**Random Forest**	
		m=6	*m=22*
Accuracy	0.837	0.810	0.846
Precision	0.782	0.734	0.793
Recall	0.907	0.935	0.911
F_1	0.840	0.822	0.848

Tabel 5: Modellen met missende waarden in de trainingsdataset en testdataset

	Logit	**Boom**	**Random Forest**	
			m=6	*m=24*
Accuracy	0.793	0.834	0.823	0.837
Precision	0.824	0.833	0.765	0.843
Recall	0.721	0.812	0.901	0.805
F_1	0.769	0.822	0.827	0.824

Tabel 6: Modellen waar de missende waarden in de trainingsdataset en testdataset geïmplementeerd zijn

Zoals te zien is in *tabel 5* en *tabel 6* is de accuracy van elk Random Forest groter dan de accuracy van het Logit model. Ook de recall en F_1-waarde van de Random Forest modellen zijn hoger dan deze zelfde waarden van het Logit model. Het gebruik van een Random Forest om de data te voorspellen is dus een goed idee gebleken.

De accuracy van het Random Forest met $m = 22$ en met missende waarden in de trainingsdata en testdata is het hoogste van alle geschatte Random Forest modellen. De precision is hier kleiner dan de precision van het Logit model. Dit betekent dat het aantal ten onrechte voorspelde aankopen bij het Random Forest model groter is dan bij het Logit model. De recall is juist hoger, wat aangeeft dat het aantal correct voorspelde aankopen ten opzichte van het aantal waargenomen aankopen hoger is bij het Random Forest model dan bij het Logit model.

Als we de geschatte modellen met missende waarden in de dataset bekijken, zien we dat het Random Forest met $m = 22$ een hogere accuracy heeft dan de beslissingsboom. Ook de precision, recall en F_1-waarde van het Random Forest zijn groter dan deze waarden van de beslissingsboom. Dit duidt aan dat een Random Forest betere voorspellingen geeft dan één enkele beslissingsboom.

Als we de Random Forest modellen vergelijken op het al dan niet voorkomen van missende waarden in de dataset zien we dat de aanwezigheid van missende waarden in de dataset geen problemen geeft bij het voorspellen van het al dan niet doen van een aankoop. De accuracy van het Random Forest met $m = 22$ geeft een hogere waarde als er missende waarden in de dataset zitten, dan als de missende waarden in de dataset geïmplementeerd zijn. Hieruit blijkt dus dat het implementeren van missende waarden een model oplevert dat slechtere voorspellingen geeft. Voorhet schatten van een Random Forest model is het dus onnodig om de missende waarden eerst in de dataset te implementeren.

Als we de Random Forest modellen vergelijken op de waarde van m zien we dat bij de modellen met missende waarden in de dataset de accuracy en F_1-waarde stijgen. Ook de accuracy en de precision stijgen bij de modellen waar de missende waarden in de datasets geïmplementeerd zijn. In dit geval voorspelt het model dus beter als we een hoge waarde van m gebruiken.

5 Conclusie

Uit de resultaten kunnen we concluderen dat het Random Forest betere voorspelprestaties geeft dan het Logit model bij deze dataset. Het aantal correct voorspelde klassen is bij het Random Forest met missende waarden in de dataset en de vector Q met lengte 22 flink hoger dan bij het Logit model. Eén van de voordelen van een Random Forest is dat het weinig aannames over de data doet en weinig rekentijd vergt. Daarnaast is het een intuïtief model dat gemakkelijk te gebruiken is. Een voordeel van het Logit model is juist dat we direct uitspraken kunnen doen over de relevantie van de variabelen en de invloed van een variabele op het al dan niet doen van een aankoop.

De aanwezigheid van missende waarden in de dataset heeft geen invloed op de voorspelkracht van het model. De geschatte modellen met een dataset waar de missende waarden in geïmplementeerd zijn, doen het zelfs iets slechter. Hieruit kunnen we concluderen dat de aannames die we moeten maken voor het invullen van de missende waarden de voorspelkracht van de modellen verslechteren. Bij het gebruik van een Random Forest is het dus onnodig om de missende waarden in de dataset te implementeren.

Ook kunnen we uit de resultaten concluderen dat een Random Forest daadwerkelijk betere voorspellingen geeft dan een beslissingsboom.

Tot slot is bij deze dataset de voorspelkracht van een Random Forest met een groot aantal willekeurig geselecteerde variabelen in de vector Q groter, dan de voor-

spelkracht met een klein aantal willekeurig geselecteerde variabelen in de vector Q. Waar de lengte van de vector Q vaak klein wordt gehouden om de correlatie tussen de beslissingsbomen in het Random Forest klein te houden, levert dat bij deze dataset geen betere voorspellingen op.

6 Beperkingen

Een beperking van het onderzoek is dat onze resultaten moeilijk te generaliseren zijn voor de webwinkel. Alle data zijn afkomstig van één of meerdere weekenden, waardoor het lastig is om voorspellingen te doen over klanten die doordeweeks een aankoop doen. Het is waarschijnlijk dat het aankoopgedrag doordeweeks anders is dan in het weekend. Ook is de meetperiode van de data onbekend. Zo zou het zo kunnen zijn dat de data gemeten zijn in één weekend, terwijl we pas echt goede voorspellingen kunnen doen als we data hebben van meerdere weekenden. Voor een optimaal geschat model is het dus van belang om data te hebben van een geheel jaar.

Een andere beperking is dat we een aanname hebben gedaan over de datastructuur om de niet-structureel missende waarden in te kunnen vullen. Om deze aanname te controleren zouden we meer informatie moeten hebben over de data van de webwinkel.

Referenties

[1] Slater, S. F., & Narver, J. C. (2000). Intelligence generation and superior customer value. Journal of the Academy of Marketing Science, 28(1), 120127.

[2] Buckinx, W., & den Poel, D. v. Predicting online-purchasing behaviour. European Journal of Operational Research, 557-575.

[3] Eulderink, S., Plaatsman, T., Slof, D. (2014). Voorspellen van webwinkel aankopen.

[4] Tan, P., Steinbach, M., & Kumar, V. (2005). Introduction to data mining. Boston: Pearson Addison Wesley.

[5] Breiman, L. Random Forests. Machine Learning, 45, 5-32.

[6] Breiman, L. (1984). Classification and regression trees. Belmont, Calif.: Wadsworth International Group.

[7] Breiman, L. (1996). Bagging Predictors, Machine Learning, 24, 123-140.

Appendix

Overige resultaten

	Voorspeld		
Waargenomen	$\hat{y}_i = 1$	$\hat{y}_i = 0$	
$y_i = 1$	1669	645	2314
$y_i = 0$	356	2155	2511
	2025	2800	4825

(a) Logit model waar de missende waarden in de datasets geïmplementeerd zijn

	Voorspeld		
Waargenomen	$\hat{y}_i = 1$	$\hat{y}_i = 0$	
$y_i = 1$	2109	216	2325
$y_i = 0$	589	2023	2612
	2698	2239	4937

(b) Beslissingsboom met missende waarden in de dataset

	Voorspeld		
Waargenomen	$\hat{y}_i = 1$	$\hat{y}_i = 0$	
$y_i = 1$	1875	435	2310
$y_i = 0$	377	2204	2581
	2252	2639	4891

(c) Beslissingsboom waar de missende waarden in de datasets geïmplementeerd zijn

	Voorspeld		
Waargenomen	$\hat{y}_i = 1$	$\hat{y}_i = 0$	
$y_i = 1$	2175	150	2325
$y_i = 0$	790	1822	2612
	2965	1972	4937

(d) Random Forest met missende waarden in de dataset, m=6

	Voorspeld		
Waargenomen	$\hat{y}_i = 1$	$\hat{y}_i = 0$	
$y_i = 1$	2117	208	2325
$y_i = 0$	553	2059	2612
	2670	2267	4937

(e) Random Forest met missende waarden in de dataset, m=22

	Voorspeld		
Waargenomen	$\hat{y}_i = 1$	$\hat{y}_i = 0$	
$y_i = 1$	2081	229	2310
$y_i = 0$	639	1942	2581
	2720	2171	4891

(f) Random Forest waar de missende waarden in de datasets geïmplementeerd zijn, m=6

	Voorspeld		
Waargenomen	$\hat{y}_i = 1$	$\hat{y}_i = 0$	
$y_i = 1$	1859	451	2310
$y_i = 0$	345	2236	2581
	2204	2687	4891

(g) Random Forest waar de missende waarden in de datasets geïmplementeerd zijn, m=24

Tabel 7: Voorspelling-realisatie tabellen