Eirik Olav Aa

Assignment 4

1 - Decision Trees

Task a

Det er ikke mulig å lage et decision tree med attributter som har veldig mange verdier. Dette vil resultere i et altfor stort tre. I denne oppgaven har jeg derfor valgt å kunne bruke de attributtene som har mindre enn 10 ulike verdier. Tanken var at med mindre enn 10 verdier er det fortsatt verdt å sjekke de ulike verdiene, men over dette så blir treet veldig stort og det blir veldig spesialisert. Valgt derfor kun å bruke attributtene: *Sex*, *Pclass*, *Embarked* og *Parch***.** Vet at egentlig *SibSp* også har mindre enn 10 ulike verdier, men valgt å ikke inkludere den når jeg kjører koden for at treet skal bli mer leselig.

Den mest naturlige attributtet å ha blant continous attributes er *Name.* Det er ingen (antageligvis) som har det samme navnet, og det vil derfor være unaturlig å prøve å lage splitter på de ulike navnene da stiene man lager aldri vil kunne teste noen andre caser enn de som heter akkurat det samme. Det samme gjelder egentlig for *Ticket.* Virker som om dette identifiserer én billett, og det vil derfor være (antageligvis) en ticket-verdi til hver passasjer.

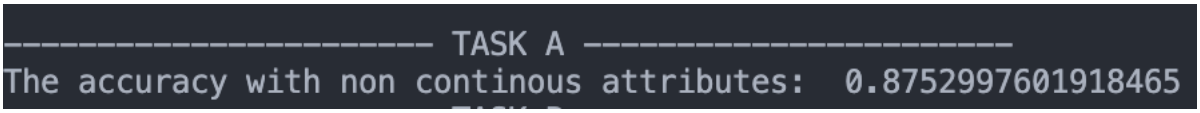
I tillegg valgte jeg å anse *Cabin* som en continous attribute av grunnen at det også her var utrolig mange forskjellige kabiner, og det er svært få som bor i samme kabin. Plasserte også *Fare* som en continous attribute da det også her fantes veldig mange ulike priser for billetter. Det var derfor ikke veldig interessant å se på hver Fare-verdi for seg selv, men kan være interessant å se på verdiene i grupper, som forskjellen mellom dyre og billige billetter. Det samme kan sies om *Age.* Det er ikke så interessant å se på hver age for seg selv, men å se på forskjellen mellom gammel og ung f.eks. er mer spennende.

Jeg valgte derfor å aldri ta med Name, Ticket og Cabin i mitt decision tree, da jeg rett og slett tror de heller gjør treet for spesialisert og bidrar ikke til et bedre tre. Det er også vanskelig å kategorisere dem. Det er lettere med Age og Fare, og vil derfor inkludere dem i min oppgave b.

Eirik Olav Aa

Vedlagt ligger PDF av kjøringen for decision tree uten continous attributes (TaskA)

Accuracy for task a



Task b

I denne oppgaven så inkluderer jeg continous attributes. Valgte her å ta med Age og Fare i tillegg til de attributtene som jeg hadde fra sist oppgave. Når jeg kjører koden kjører jeg kun med Sex, Pclass, Embarked, Age og Fare for at treet skal være så leselig som mulig.

Valgte å ikke inkludere Name da det var vanskelig å gjøre en god kategorisering her. Kunne kanskje samlet på etternavn og sett om folk fra samme familie overlevde, men det ville fortsatt hvert mange familier. Eventuelt sjekket på mr. og mrs., men den dekkes allerede av Sex. Det samme gjelder for Cabin egentlig. Her var det veldig mange tomme celler, noe som gjør at det ikke hadde blitt veldig nøyaktig uansett. I tillegg var det vanskelig å finne en fornuftig samling på disse. Kunne samlet på først bokstav: A, B, C, D eller E, men jeg vet ikke helt hva disse bokstavene forteller noe om, så valgte å ikke gjøre dette. I tillegg så droppet jeg Ticket da jeg ikke fant noe system for hvordan denne verdien var satt.

Det var greit å ta med seg Age og Fare, da man kan se på om det er stor sannsynlighet for å overleve dersom man er eldre enn et gitt år. Eller at det er stor sjanse for å dø dersom man betalte lite for billetten sin.

Vedlagt ligger PDF av kjøringen for decision tree med continous attributes (TaskB)

Accuracy for task b

Eirik Olav Aa

Task c

Tenkte originalt at dersom man har mye data, og benytter seg av mange av attributtene ved utregning burde man få et bedre decision tree. Etter å ha jobbet med oppgaven, lest pensum og testet ulike kjøringer med forskjellige variabler skjønner jeg at dette ikke nødvendigvis stemmer helt. Det er gode muligheter for at dette også kan være med på å påvirke accuracy negativt.

Dersom man inkluderer flere attributter, kan man oppleve mer støy. Treet kan bli veldig spesialisert dersom man inkluderer attributter verdier som har lite sammenheng med om en passasjer overlever eller ikke. Da kan det være at training dataen lærer treet til å se sammenhenger som egentlig ikke er der. Dette vil da gjøre utslag når vi kjører med test dataen, og påvirke accuracy negativt. Det er derfor viktig å overveie hvilke attributter man skal inkludere når man skal lage decision tree. Det er også mulig å oppleve overfitting, som vil påvirke et decision tree negativt. Overfitting innebærer at man lager veier som er så spesialisert at det ikke passer til så mye annen data, og det vil derfor dannes mange veier som aldri vil brukes/sees igjen. Ved å inkludere Name og/eller Ticket tror jeg at jeg kunne opplevd noe slik i mitt tre, men ser ikke ut som om det har vært noe problem basert på resultatene i oppgave a og b. Siden de er ganske like, så var ikke inkluderingen av Age og Fare så ille.

Det er ikke bare negativt å inkludere mange attributter. Ofte gir de oss en god måte å splitte treet vårt på. Om en node øker «information gain» med mye, så gir den oss et mer nøyaktig, og bedre tre. Det ser ut som om jeg har klart å velge fornuftige noder til dette datasettet da det ikke gjør store utslag når jeg inkluderer de gjenværende i oppgave b.

Til tross for dette er ikke treet helt perfekt. Et eksempel på dette er at vi generaliserer continous attributes til to verdier, over eller under et gitt tall. Det kan være bra det, men det er mulig at det hadde vært bedre, alltid eller til tider, å splitte det inn i grupper. Kunne da sett på grupper under 20, mellom 20 og 40 og over 40. Dette kunne ført til en økt information gain og da også resultert i et mer nøyaktig tre.

Det er også mulig å prune decision tree. Målet med decision tree er å få god accuracy, men du ønsker gjerne å oppnå dette med minst mulig dybde i treet. Dette kan du oppnå ved hjelp av pruning. Pruning går ut på å «fjerne» stier i treet som man aldri skal kunne besøke uansett. Ved å unngå å utforske disse stiene, så vil man redusere mengden støy i dataen. Ved å f.eks. droppe å gå ned stier som aldri vil øke information gain, kan man oppnå pruning. Pruning kommer ikke til å bidra med økt accuracy, da den ikke endrer resultatet noe særlig. Det den kan bidra med er en redusert sannsynlighet for dårligere accuracy.

2 – Missing Values

Det er dessverre slik at datasettet vi har tilgjengelig ikke er fullstendig. Det er flere verdier for attributter som mangler. Disse tomme feltene må behandles på en måte, hvis ikke vil oppbyggingen av treet bli feil, eller krasje. For å håndtere disse verdiene er det viktig å ha kontroll på når de dukker opp. Når man har det, så er det mulig å håndtere de på ulike måter.

Den ene måten er å gå videre, uten å inkludere denne verdien i utregningen. Dette valgte jeg å gjøre i f.eks. min gain() funksjon. Her sender jeg bare koden videre til neste verdi for attributtet for å regne ut gain videre. Dette er en veldig effektiv og enkel måte å behandle tomme attributt-verdier på. Problemet er at man mister mye data som ikke er med på å regne ut, i dette eksempelet gain. Dette kan føre til at man ikke finner det som kunne være riktig og beste splitt, og man får potensielt dårligere accuracy.

En annen måte å behandle det på er ved å bruke gjennomsnittsverdien. Dette er igjen en veldig effektiv metode, som er enkel å implementere. Her er det kun å summere opp alle verdiene i attributtet og finne snittet av disse. Man kan da erstatte alle tomme verdier med denne verdien. Jeg prøvde ikke denne metoden selv, men jeg tenker at den vil ha ganske likt resultat som det å hoppe over verdien. Det å legge til flere av gjennomsnittsverdien vil nok antageligvis ikke gjøre store utslag, men heller ikke store forbedringer.

Eirik Olav Aa