INF219 - Prosjekt i Programmering I

Webscraping og maskinlæring for å estimere prisantydning av boliger i Bergen

Mathias Grønstad, Universitetet i Bergen, 12.12.2017

1. Oppsummering

Python sitt Scrapy verktøy ble brukt til å hente data fra alle boligannonser på Finn.no i Bergen omegn (opptil 7.8 km fra sentrum). Det ble scrapet 435 boligannonser over en periode på ca. 3 måneder. Etter databehandling/prosessering ble det laget en maskinlæringsmodell, ID3 beslutningstre for klassifisering. Denne modellen klarte å estimere boligverdier på et testsett med snittavvik på ca. 12.5%. Dette var mulig etter at utstikker-data ble fjernet (boliger med pris > 5 millioner og bruksareal > 100 mˆ2. Før dette ga modellen ca. 15% avvik. Før denne databehandlingen ga modellen ca. 16-17% avvik.

Etter dette ble datasettet brukt i Python sine maskinlæringsbiblioteker (TensorFlow & Keras) og modellen brukt var Random Forest med lineær regresjon. Med et tre ga denne modellen ca. Samme resultat som ID3 klassifiseringen. Med 30 trær fikk vi ca. 11.5% snitt avvik. Ved å legge ved energimerking ble avviket i snitt 10.2%. I disse estimeringene ble alle 435 datapunkter brukt.

Resultatene viser at det er mulig å gjøre relativt fornuftige estimater av prisantydning basert på historiske data som ligger ute på Finn.no selv med 435 datapunkter. Det begrensede antall datapunkter satt en grense for hvor bra estimatene kunne bli.

2. Datainnsamling, analyse og behandling:

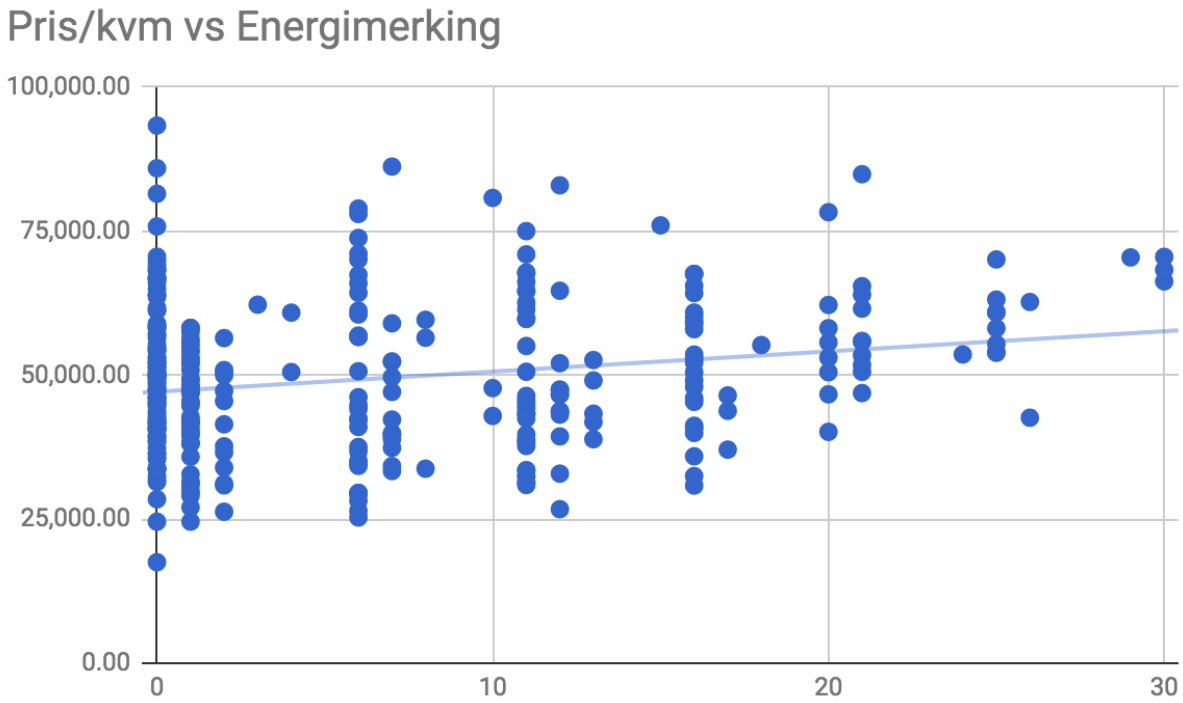
Det ble valgt å ta boliger relativt sentralt i Bergen (< 7.8 km km fra sentrum) i perioden fra 15.09 til 28.11.2017 Det ble "Scrapet" 435 boligannonser og all data som kunne tenkes å være relevant for prisantydningene:

- Bruttoareal
- Primærrom
- Bruksareal
- Adresse
- Ligningsverdi
- Fellesgjeld
- Felles formue
- Antall rom
- Antall soverom
- Etasje
- Byggeår
- Kommunale avg.
- Felleskostnader pr. Mnd
- Boligtype
- Eierform

- Energimerking

I tillegg ble avstand til sentrum og standard lagt til manuelt:
Google sitt avstands-API ble brukt til å finne distanse fra sentrum til boligens adresse[1].
Standard ble vurdert hovedsakelig utfra bilder (og delvis energimerking, byggeår og renovert år) og er basert på min subjektive mening. Dette var manuelt arbeid som ikke er skalerbart.

Energimerking er en kategorisk variabel fra rød til mørkegrønn og A til G som ble enkodet til tall fra 0 til 34 (7*5).



2.1 Manglende data:

Flere av de uavhengige variablene hadde manglende data. Disse er:

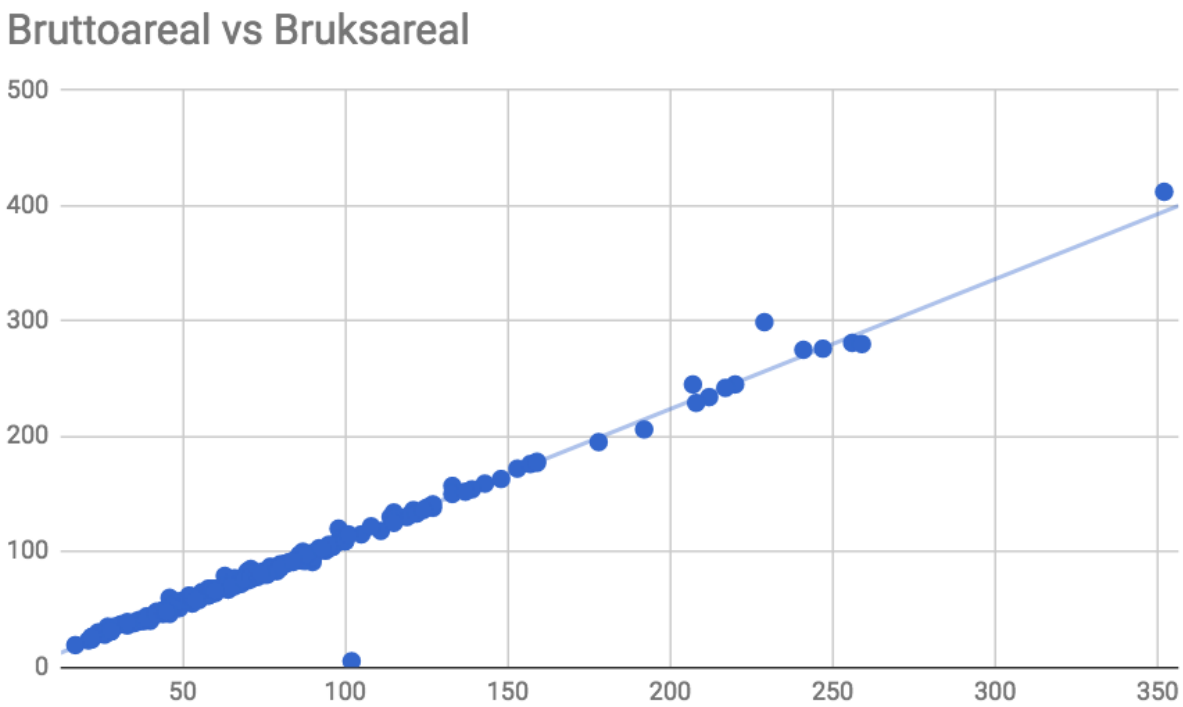
- Bruttoareal
- Ligningsverdi
- Antall rom
- Antall soverom
- Etasje
- Byggeår
- Energimerking

Et problem med å ha manglende data i så mange forskjellige kategorier er at om vi skal estimere de ut fra datasettet, så vil vi få forskjellige utfall avhengig av hvilke variable vi estimerer først, siden vi da vil bruke estimater som grunnlag for å lage andre estimater. Jeg ønsker derfor å redusere antall variable jeg estimerer.

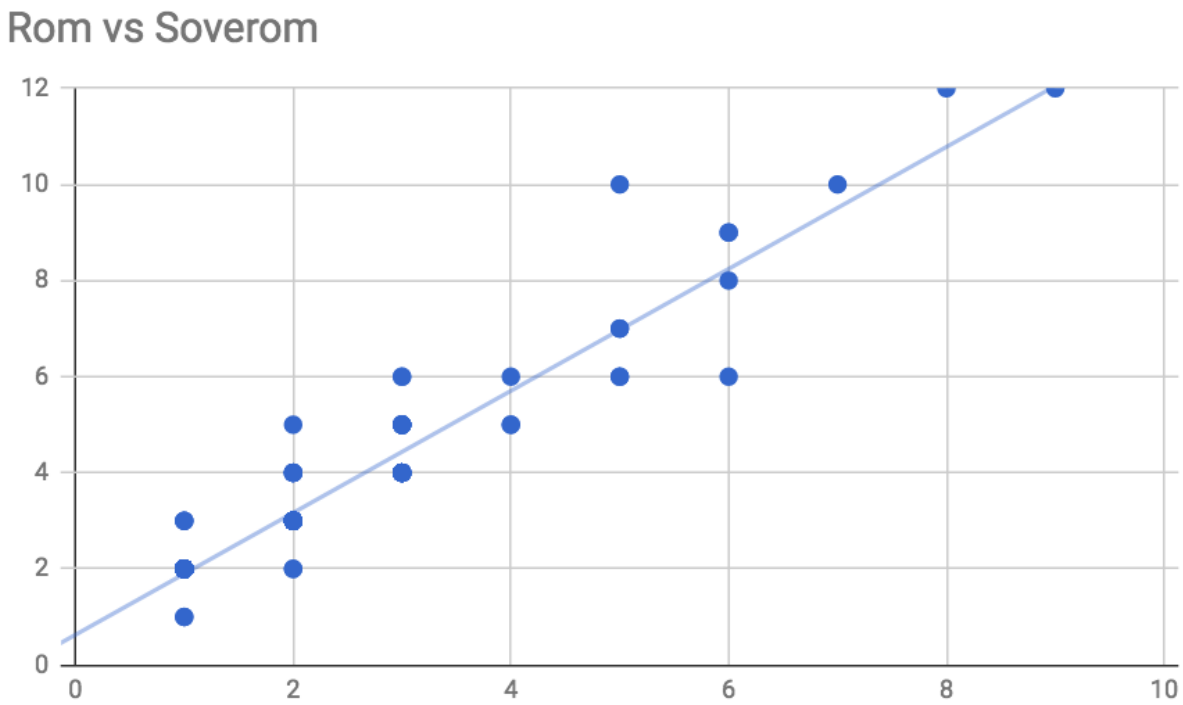
Jeg valgte å kutte ut ligningsverdi siden den er utregnet fra prisen på boligen. Typisk er ligningsverdi ca. 25% av markedsverdien på boligen. Siden det ultimat sett er

markedsverdien jeg vil estimere, og ligningsverdien er basert på denne, så er det en overflødig variabel. Dessuten mangler den for veldig mange boliger på Finn.

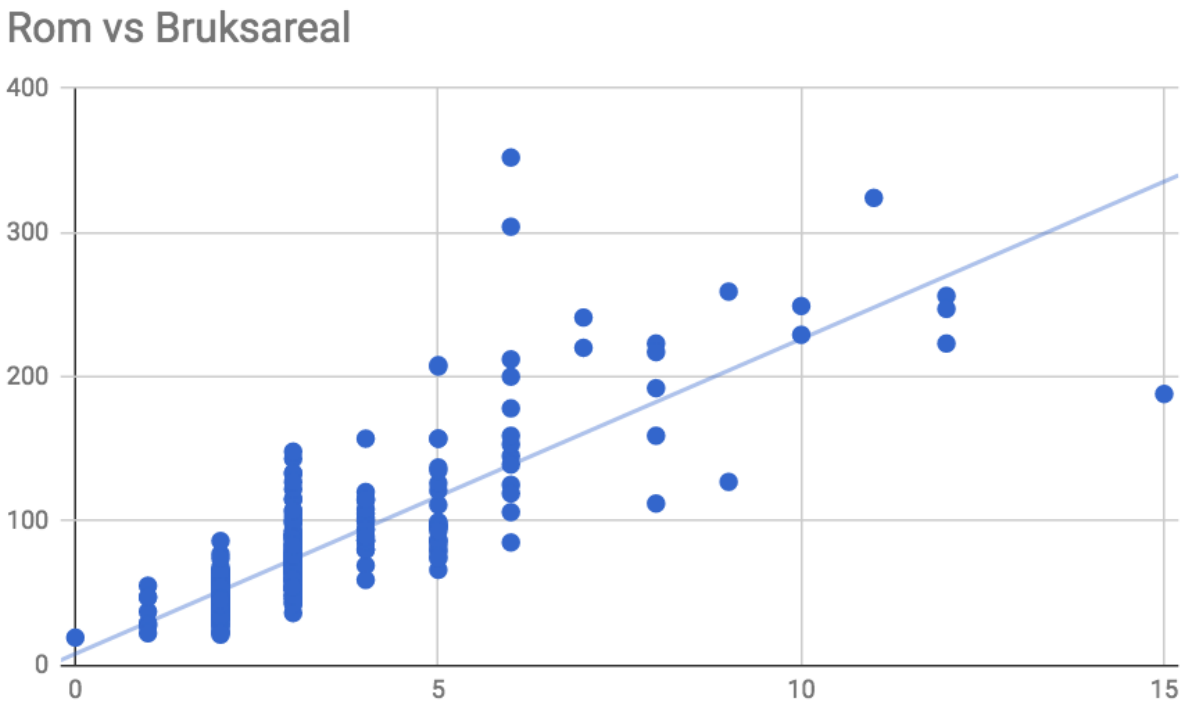
Jeg valgte også å fjerne bruttoareal, siden den er så og si lineært korrelert med bruksareal. Det blir litt overflødig å bruke begge deler som variabel. Det samme gjelder primærrom. Vi bruker kun bruksareal som areal variabel.



Antall soverom og antall rom er også nesten lineært korrelert:



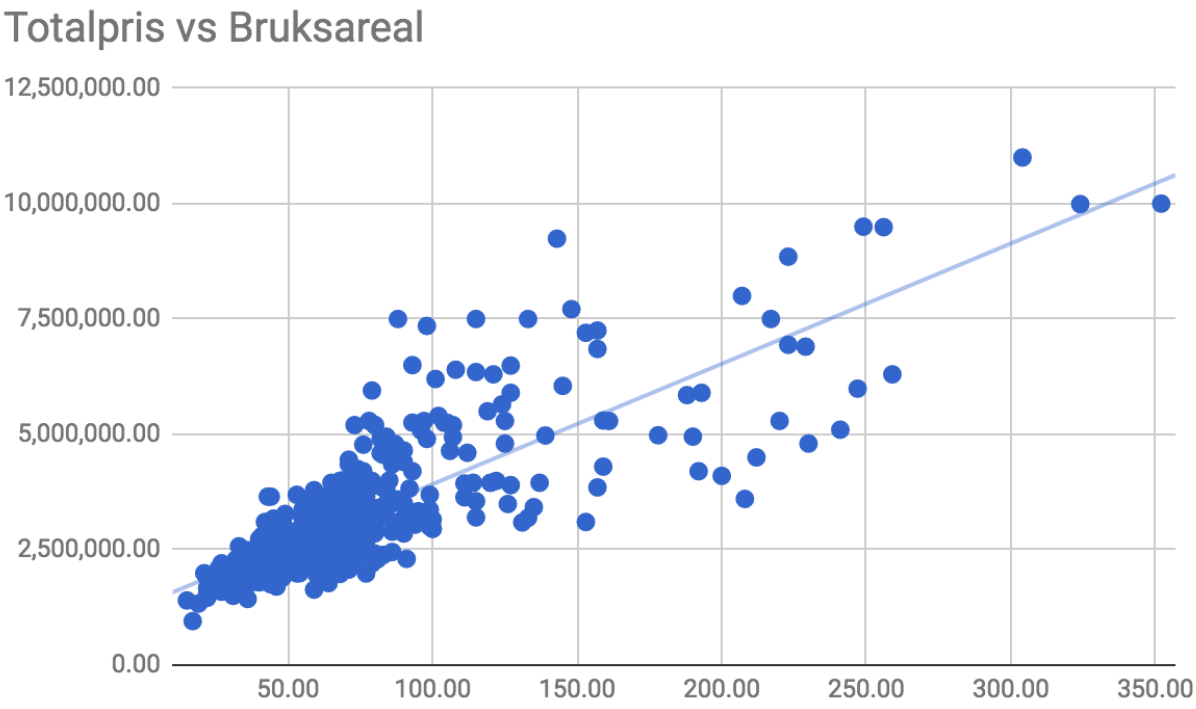
Antall rom er ca. 1.5x antall soverom. Vi bruker dette til å fylle ut manglende variable for antall rom. Siden antall soverom og antall rom er såpass korrelert, er det overflødig å sende inn begge som variable i modellen. Vi velger derfor å kun bruke antall rom.
Det er også visse boliger som mangler både antall rom og antall soverom. Vi estimerer antall rom ut fra bruksarealet, som virker nokså lineært. Utstikkerne er utleieenheter og enkelte boliger med veldig store men få rom.

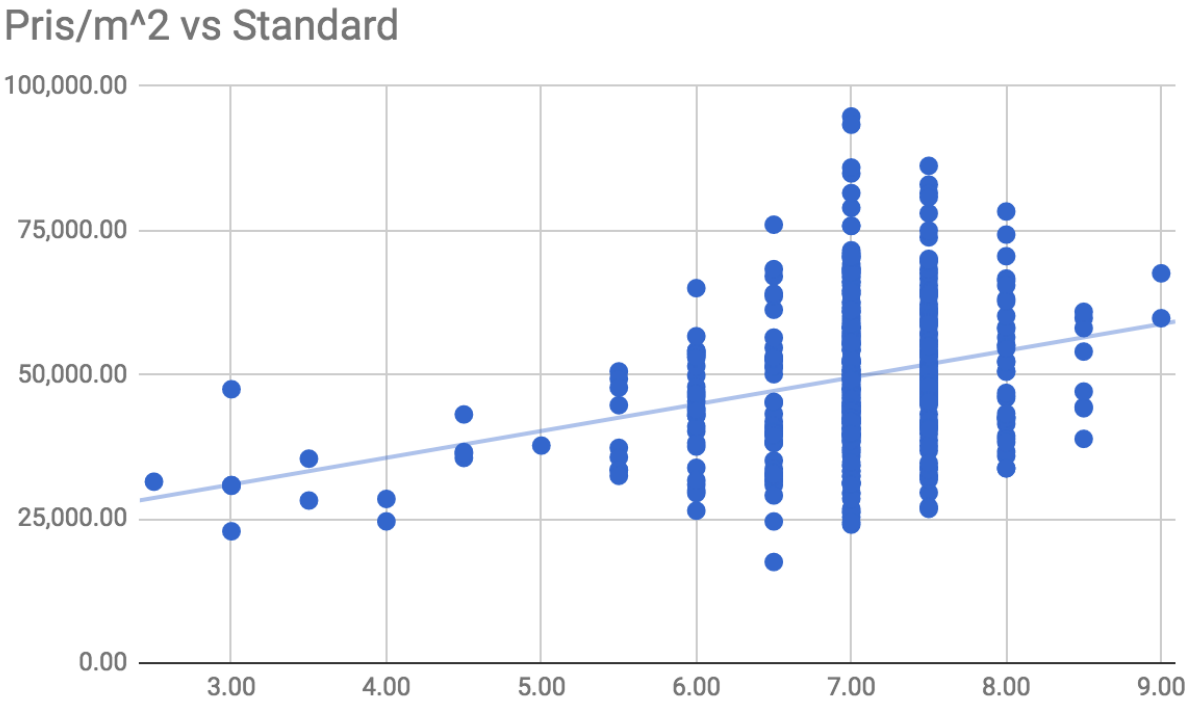


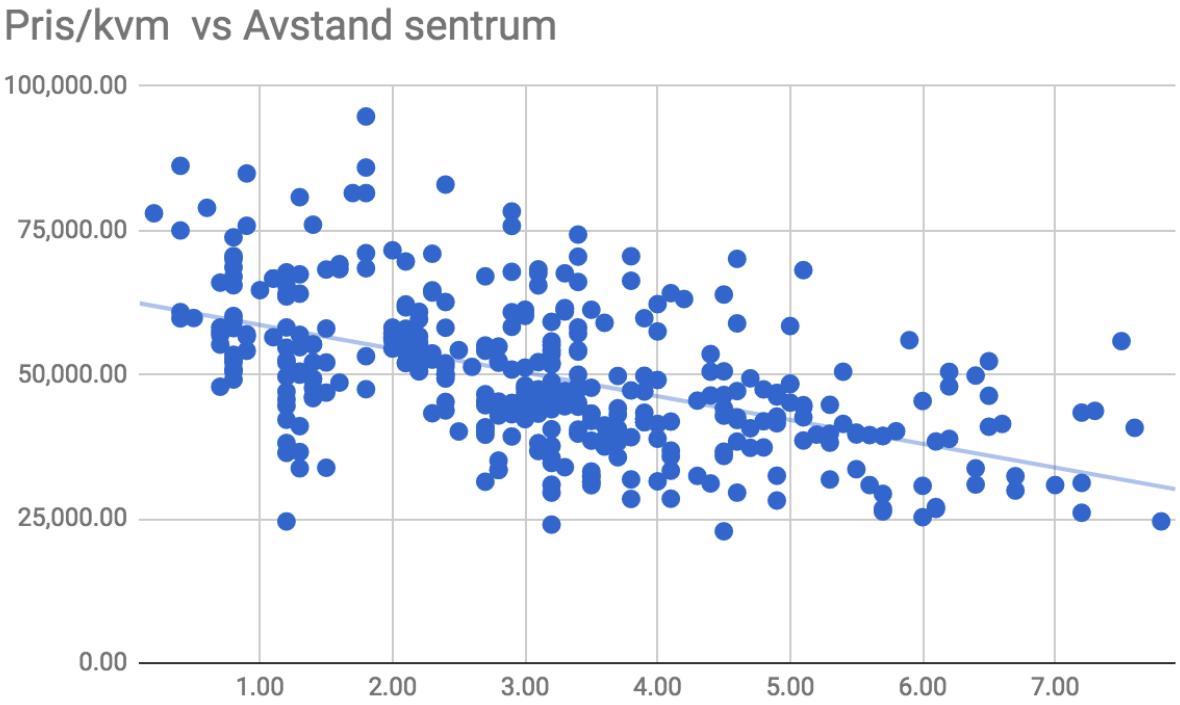
Stigningsgraden på trendlinjen er 22 kvm/rom. Jeg bruker dette til å estimere antall rom utfra bruksareal for manglende data.

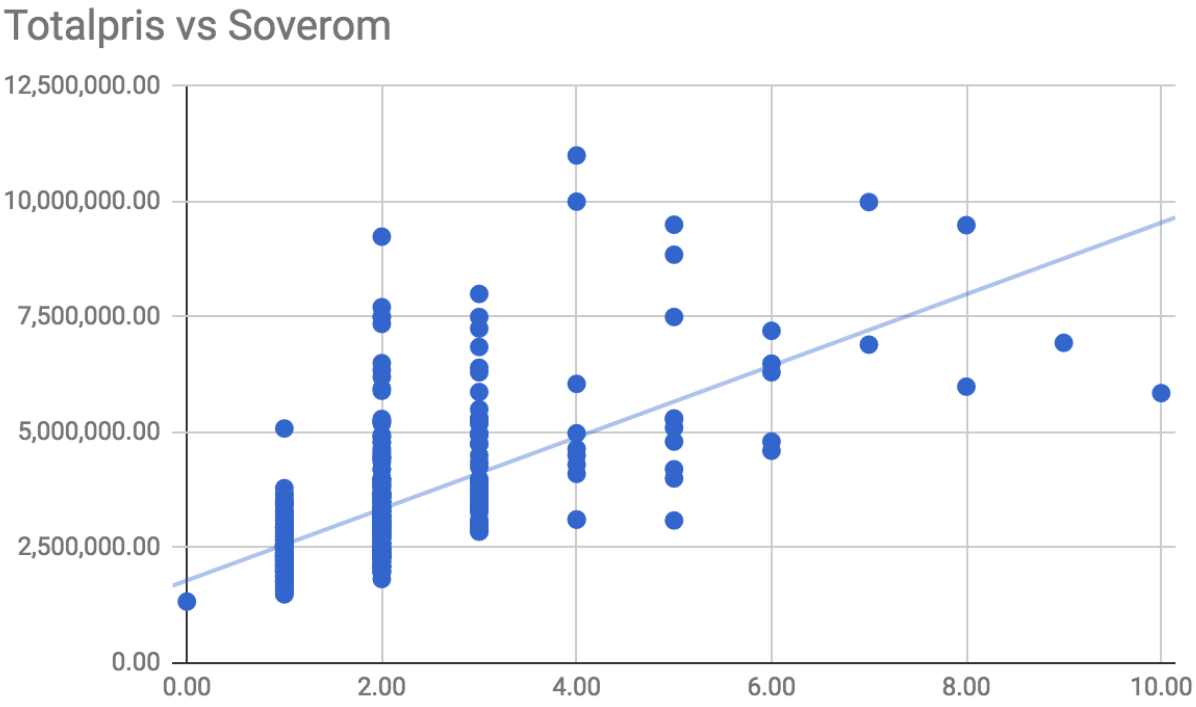
Etter dette gjenstår kun etasje og energimerking som egenskaper med manglende variabler. Grunnet allerede for mange variabler og at disse ikke er de viktigste for pris (se under), blir de ekskludert for å unngå overfitting. Energimerking blir inkludert igjen i avsnitt 5 når vi bruker Random Forest lineær regresjon.

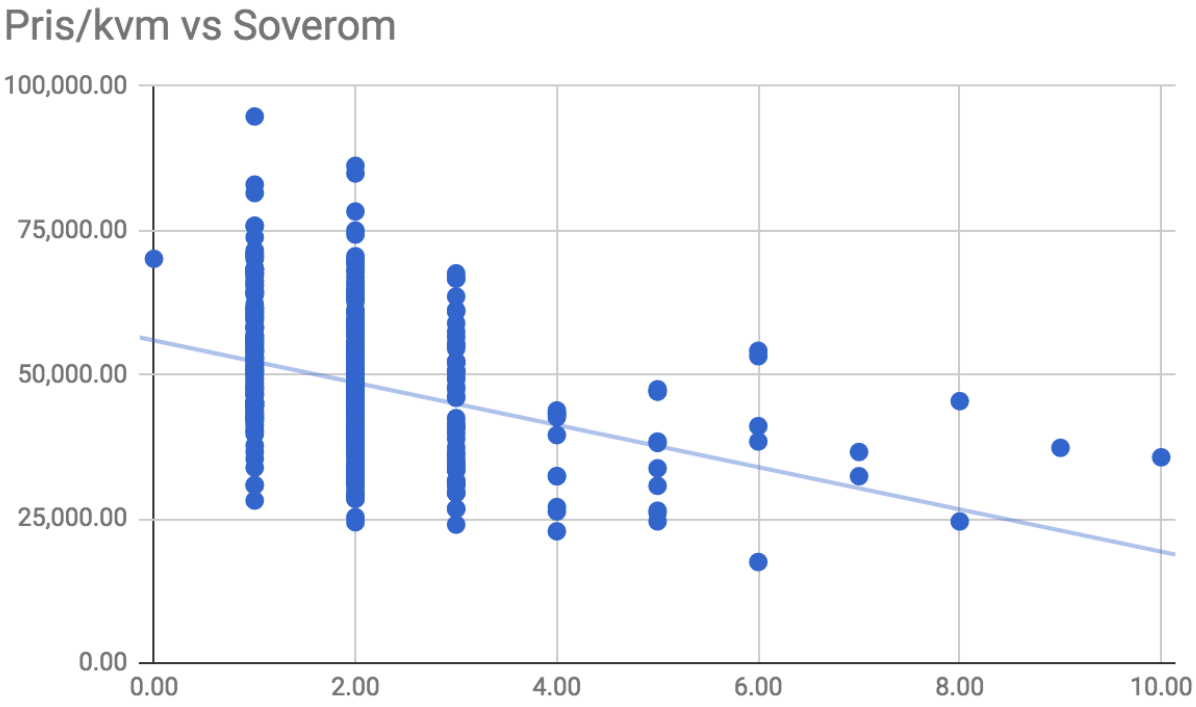
Som vi ser har vi mange uavhengige variable i forhold til antall datapunkter (435). Følger vi regelen om maksimalt 10% uavhengige variable i forhold til datasettet, bør vi ikke ha mer enn ca. 40 stk. for å unngå overfitting. Siden hver av de uavhengige variablene blir delt opp (f.eks. standard: ”bra”, ”dårlig” etc., støter fort på “curse of dimensionality” her. Vi vil først luke ut hvilke variable som har mest innvirkning på prisantydning (og pris pr. kvadratmeter):

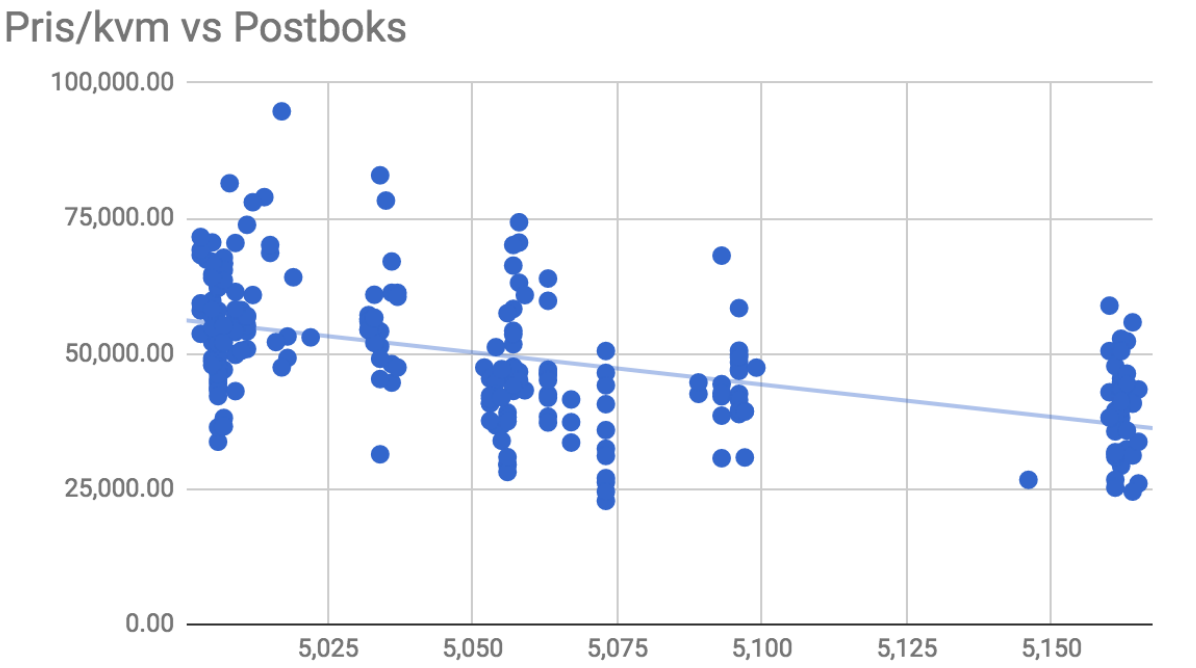

Totalpris vs Bruksareal
12,500,000.00
10,000,000.00
7,500,000.00
5,000,000.00
2,500,000.00
0.00
50.00
100.00
150.00
200.00
250.00
300.00
350.00

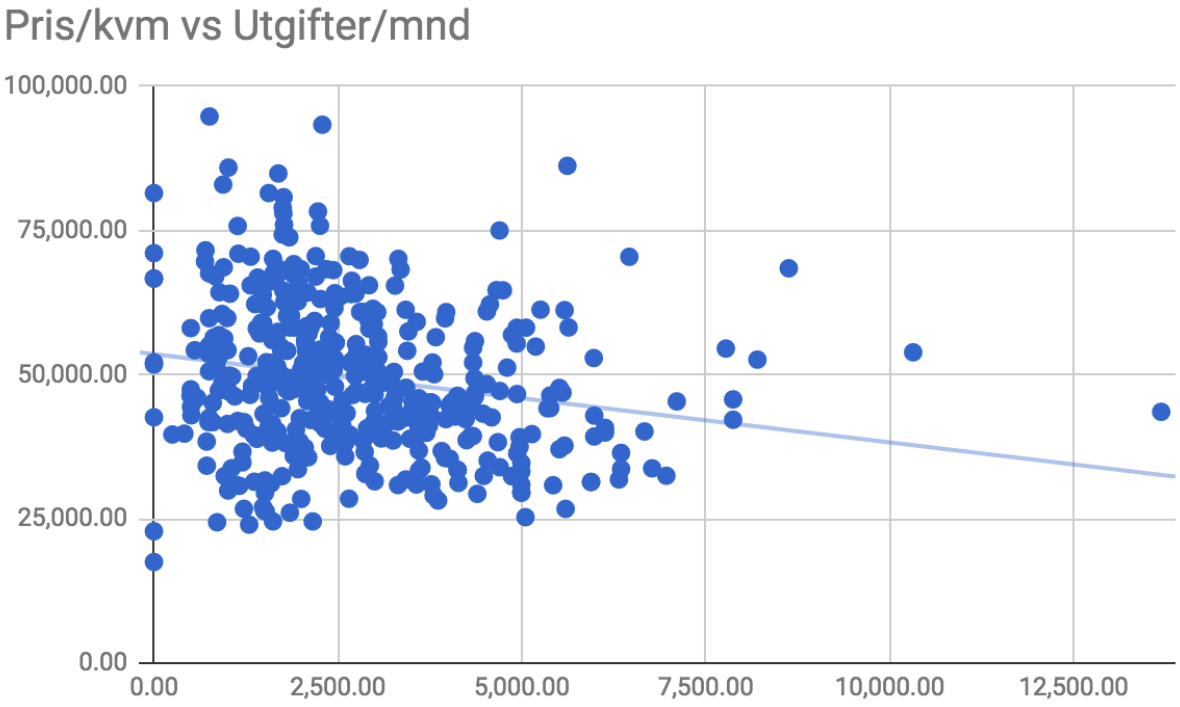

Pris/m^2 vs Standard
100,000.00
75,000.00
50,000.00
25,000.00
0.00
3.00
4.00
5.00
6.00
7.00
8.00
9.00

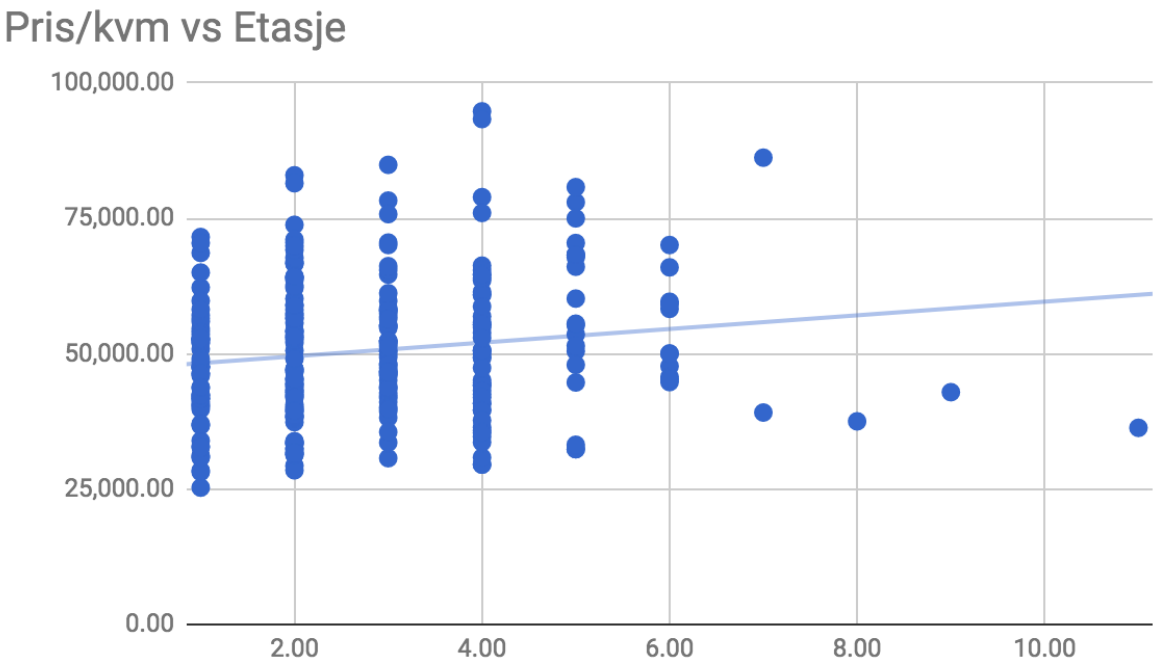

Pris/kvm vs Avstand sentrum
100,000.00
75,000.00
50,000.00
25,000.00
0.00
1.00
2.00
3.00
4.00
5.00
6.00
7.00

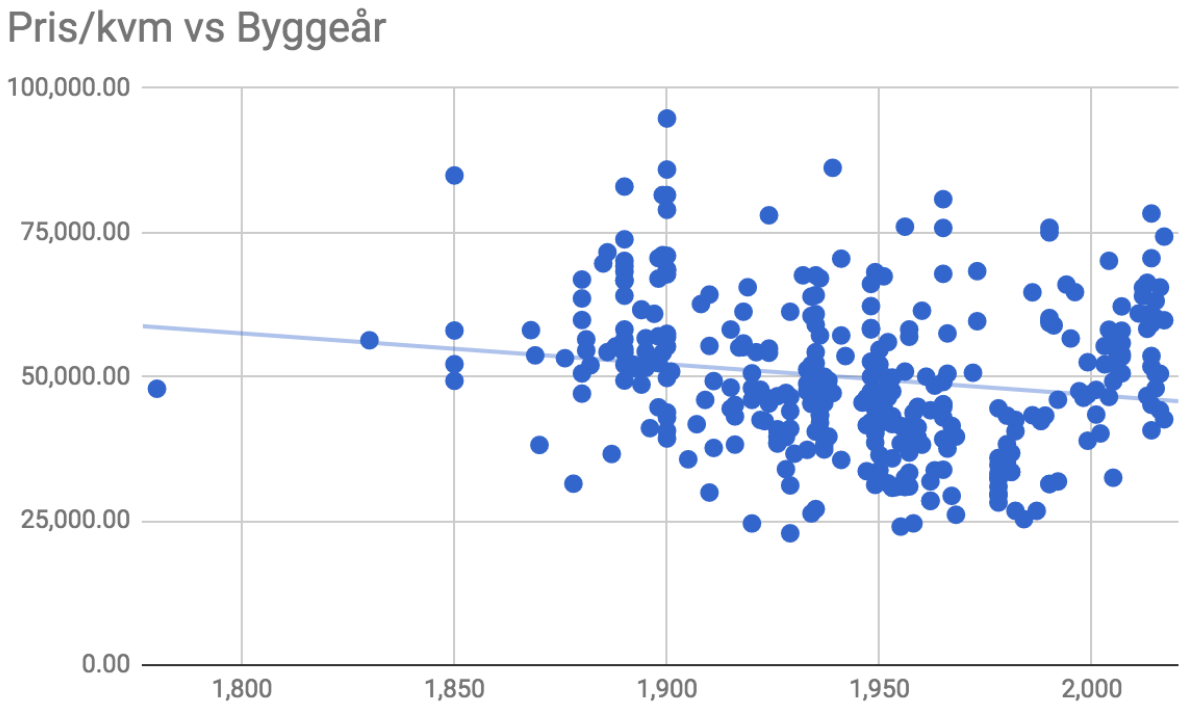

Totalpris vs Soverom
12,500,000.00
10,000,000.00
7,500,000.00
5,000,000.00
2,500,000.00
0.00
0.00
2.00
4.00
6.00
8.00
10.00


Pris/kvm vs Soverom
100,000.00
75,000.00
50,000.00
25,000.00
0.00
0.00
2.00
4.00
6.00
8.00
10.00


Pris/kvm vs Postboks
100,000.00
75,000.00
50,000.00
25,000.00
0.00
5,025
5,050
5,075
5,100
5,125
5,150


Pris/kvm vs Utgifter/mnd
100,000.00
75,000.00
50,000.00
25,000.00
0.00
0.00
2,500.00
5,000.00
7,500.00
10,000.00
12,500.00


Pris/kvm vs Etasje
100,000.00
75,000.00
50,000.00
25,000.00
0.00
2.00
4.00
6.00
8.00
10.00



Jeg endte opp med å velge følgende uavhengige variable for å teste ut en ID3 modell:

- Standard
- Totale utgifter pr/mnd. (felleskostnader + kommunale avg.)
- Postboks
- Avstand sentrum

Standard ble kategorisert:
”Super” >= 7.5
”Bra” = 7 (de fleste boliger fikk dette estimatet av standard)
7 > ”Medium > 5
”Lav” <= 5

Totale utgifter pr/mnd ble satt i 3 kategorier:
<2500
2500-5000
>5000

Postnummer (se bilde over) havnet grovt sett i 5 bokser:
<5025:
5025-5050
5050-5075
5075-5100
>5100

Avstand til sentrum ble delt i fire kategorier
<2
2-4

4-6
>6

Antall kvadratmeter ble satt som en liste fra 25 til 300 mˆ2 med intervaller på 15 mˆ2.
Dvs. vi hadde 19 kategorier for denne.

Totalt ble dette 35 (19 + 4 + 5 + 3 + 4) uavhengige variable. Ganske nær 10% grensen, så vi unngår forhåpentligvis ”overfitting” (early stopping ble også brukt i ID3 for å unngå overfitting).

3. Beslutningstre ID3

Algoritmen som ble brukt for trening og testing heter ID3 og er et ”decision tree”. Koden er en modifisert versjon av den hentet fra:

Machine Learning: An Algorithmic Perspective (2nd Edition) av Stephen Marsland (http://stephenmonika.net)

Som del av preprosessering av datasettet inkluderte jeg randomisering av datasettet før det ble splittet i klasser og ”data”.

Jeg la også til ”early stopping”. Dette gjorde jeg ved at for hvert subtre som ble generert telte jeg opp hvilken ”klasse” (dvs. pris) som forkom flest ganger i subtreet. Dersom denne overgikk 65% satt jeg hele subtreet til å være denne klassen. Vi vil ikke bygge ut et helt subtre bare for å klassifisere et punkt dersom de fleste punkter er like. Dette unngår overfitting.

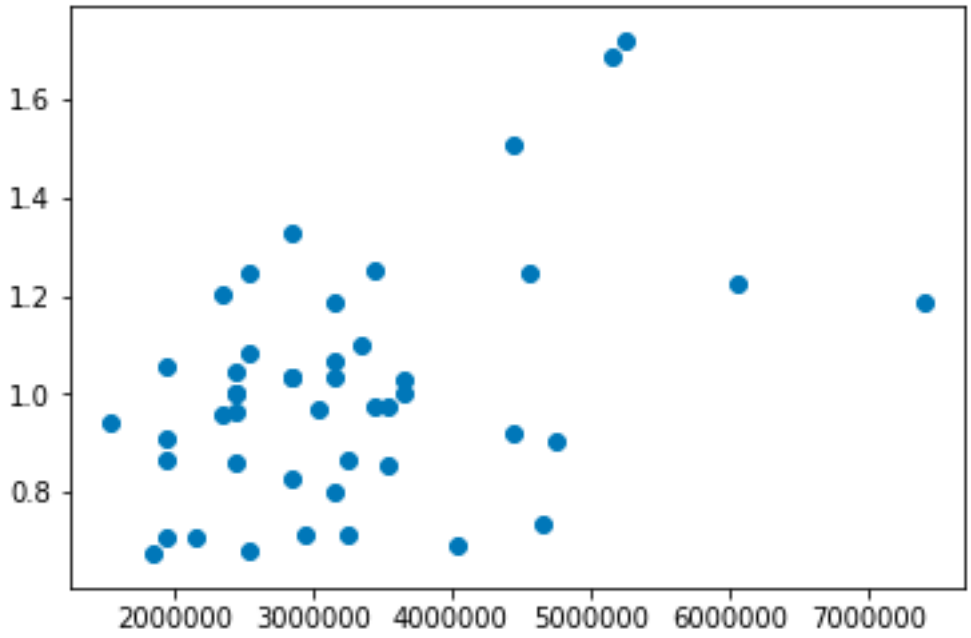
Prisene ble kategorisert inn i klasser fra 1 til 10 millioner i intervall på 100 000.

Som treningssett ble 80% av dataene brukt, mens 20% ble brukt til testing. Det var ikke nok data til at noe valideringssett ble vurdert brukt.

4. Resultater med ID3

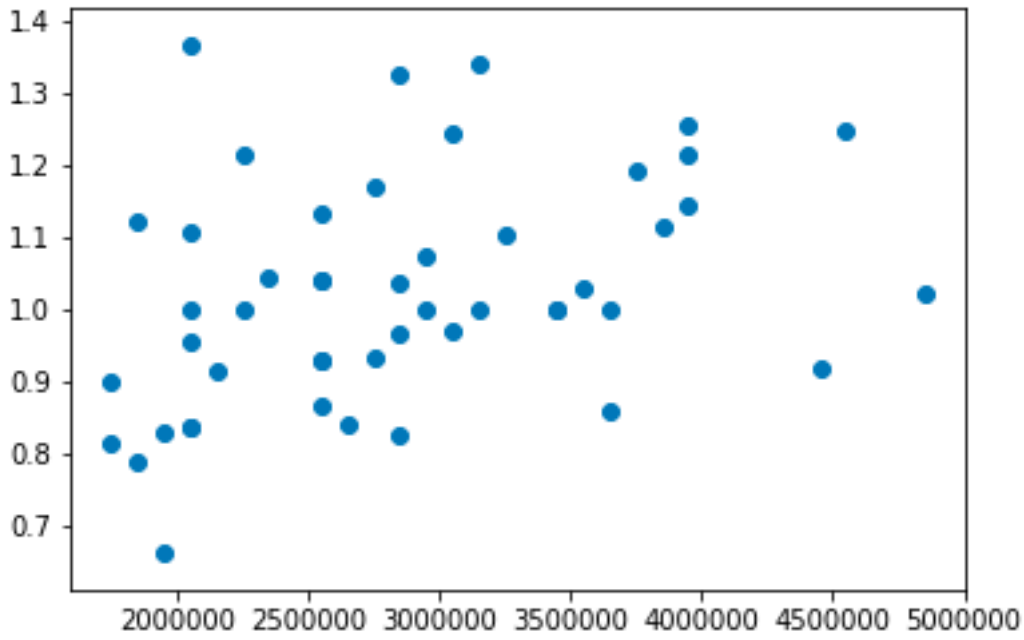
Algoritmen var ikke i stand til å gjette prisen innen gitt intervall (”spot on”) ofte, men den kom i nærheten ofte. Isteden for å bruke eksakt gjett som kriterie for suksess, regnet jeg heller ut hvor langt relativt fra riktig pris hvert estimat ble. Dvs. hvor mange prosent over/under estimatet ble. Dette ble lagret i en liste ”differences”. Snittet av denne listen gir hvor stor feil algoritmen gjør i helhet.

Hvor stor denne feilen er litt tilfeldig da vi randomiserer dataene før trening. Trener vi med 368 punkter og tester med 64 får vi en gjennomsnittlig feilestimering på ca. 15%, men det er enkelte feilestimater som utgjør en stor del av denne snitt feilmarginen:



I figuren over ser vi feilmargin på y-aksen mot riktig prisantydning på x-aksen. Det kan se ut som at feilmarginen øker for høyere boligpris. I dette tilfellet hadde vi 17% feil i snitt. De fleste boliger er priset under 5 millioner, så det kan tenkes at noen ”utstikkere” som skaper støy og ikke ”passer” inn i modellen.

For å teste hypotesen filtrerer jeg vekk boliger med total prisantydning > 5 millioner og trener/tester igjen:

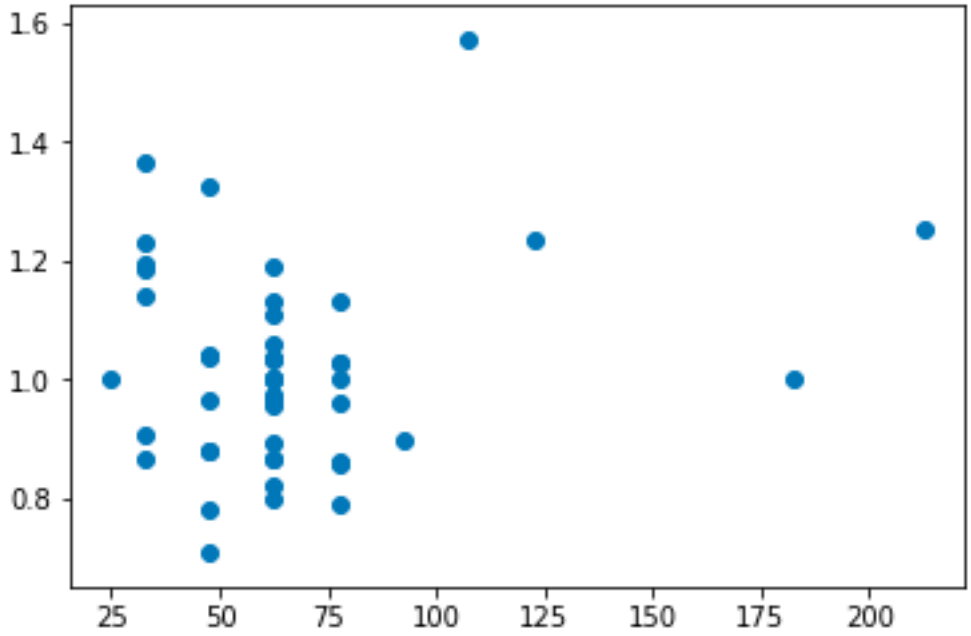


I dette tilfellet ble feilen i snitt 12.3%.

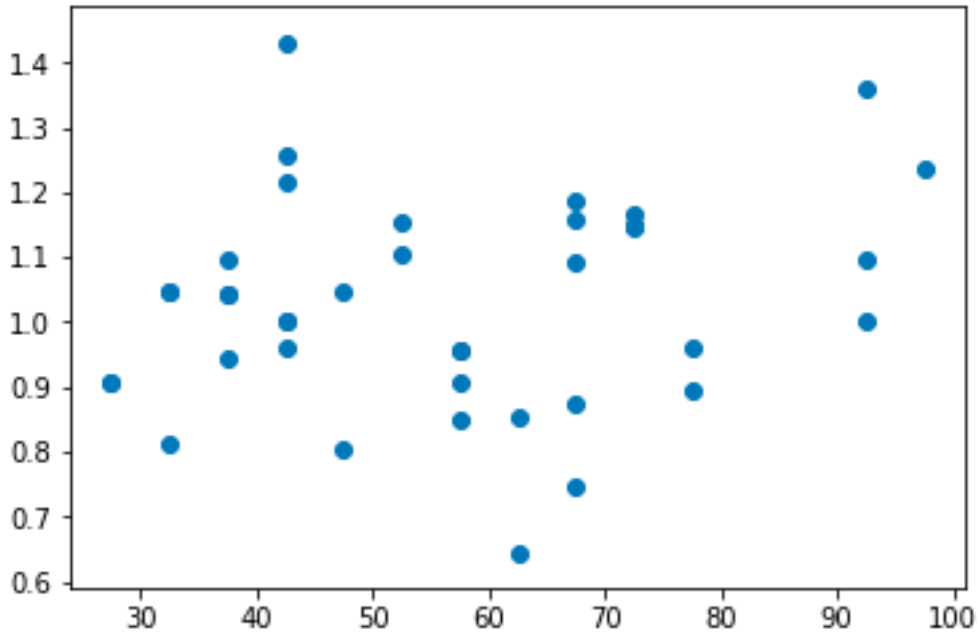
Vi kan ikke trekke konklusjoner ut fra en kjøring så jeg tar snittet av 100 kjøringer:

Når vi filtrerer vekk boliger med pris > 5 millioner får vi snitt feil på 14.1%, mens om vi inkluderer alle boliger får vi 16.6% snitt feil. Det ser altså ut som det kan være hold i teorien om at utstikkere gjør modellen dårligere. Det nye datasettet har 383 punkter i stedet for 434.

Vi kan kanskje gjøre det samme med boliger som har veldig stort areal. Vi plotter relativ feil mot areal (gitt filtreringen over):



I figuren over ser vi feilmargin på y-aksen mot areal av det vi skal estimere på x-aksen. Det kan se ut til at vi bør filtrere ut boliger over 100 mˆ2. Dette vil også innebære at vi får mye færre uavhengige variable, og vi kan kanskje kategorisere boligene i et lavere areal-intervall, f.eks. fra 20 til 100 i steg på 5 mˆ2. Merk at en del av boligene med høy mˆ2 er allerede filtrert ut siden vi har filtrert ut boliger med pris > 5 mill. Når vi har filtrert vekk boliger med > 100 mˆ2 areal sitter vi igjen med 355 data punkter.



Etter 100 kjøringer fikk vi et snitt avvik på 16.0%. Altså ikke en forbedring, men vi bruker samme kvadratmeter fordeling, så vi deler boligen kun opp i <25, 25-50 og 75-100. Her er det rom for å splitte opp finere siden vi har flere uavhengige variable å bruke.

Hvis jeg deler opp kvadratmeter variabelen i klasser fra 10 til 100 i hopp på 5 blir snitt feilen 13.8%, og vi bruker 19 variable for mˆ2. Totalt har vi 35 variable på totalt 355 datapunkter. Under er snitt etter 300 kjøringer hver:

mˆ2 intervall	Snitt feil estimat [%]
25	16.0
20	15.1
15	13.2
10	12.6
5	13.8
3	14.6

Som vi ser kan ID3 klassifisering gi ned mot 12.5% snitt avvik fra prisantydning. Det kan tenkes at ytterligere eksperimentering med dataene kan gi litt bedre estimater. Dette tar mye tid, så jeg valgte å gå videre med maskinlæringsmodeller i Python.

5. Implementering ved bruk av TensorFlow og Keras bibliotek

5.1 Random Forest Lineær Regresjon

Manuell eksperimentering slik som gjort ovenfor tar tid og det er ikke nødvendigvis optimalt. Python sine bibliotek for maskinlæring, TensorFlow og Keras kan gjøre jobben bedre og automatisk. Dette gir ikke like god oversikt over hva man gjør og er litt mer som en ”black box” men resultatet ble overraskende bra.

Jeg brukte 400 datapunkter til trening og 35 til testing. Dvs. jeg brukte all dataen og kuttet **ikke** ut utstikkere slik som jeg gjorde i ID3 klassifisering.

For hver random forrest gjorde jeg n=100 kjøringer og tok snitt av estimeringsfeilen.

Number of trees in Forest	Average error in estimates after 100 runs
1	16.4%
10	12.0%
20	11.6%
50	11.4%
100	11.4%
300	11.3%
500	11.3%

Som vi ser så får vi med kun **et** Tre i ”skogen” så og si samme feilestimat som vi gjorde med ID3 klassifiseringen. Det typer på at klassifiseringen som ble gjort er ca. like god som regresjonen for decision trees.
Vi ser også at økende antall trær i skogen gir for betraktelig bedre estimater. I dette tilfellet trenger vi veldig få trær (20 holder). Grunnen til dette er at vi har såpass få datapunkter at når antall trær blir mange, blir de svært like og deres estimater vil følgelig også blir veldig like.

5.2 Lineær regresjon med kategoriske og kontinuerlige data

5.2.1 Postnummer

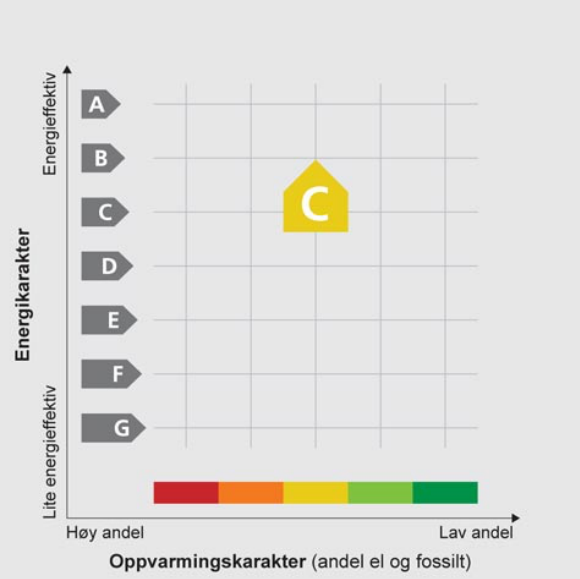
I kjøringene over har jeg brukt postnummer som en kontinuerlig variabel. Det er kan være mer riktig å bruke den som en kategorisk variabel fordi postnummeret øker ikke helt lineært ut med distanse fra sentrum. I tillegg vil visse postnummer representerer finere og dyrere strøk enn andre.

Jeg bruker 1-on-N enkoding til å enkode disse. Først omgjør jeg postnummer til strenger i fire intervaller slik som i ID3. Så omgjør jeg disse til 4 kolonner som er 0 for alle utenom det postnummer den skal representere, der har kolonnen verdi 1.

Resultatet var noe skuffende og lå på rund 13% feil mot ca. 11.5% feil ved å sette postnummer som kontinuerlig variabel.

5.2.1 Energimerking

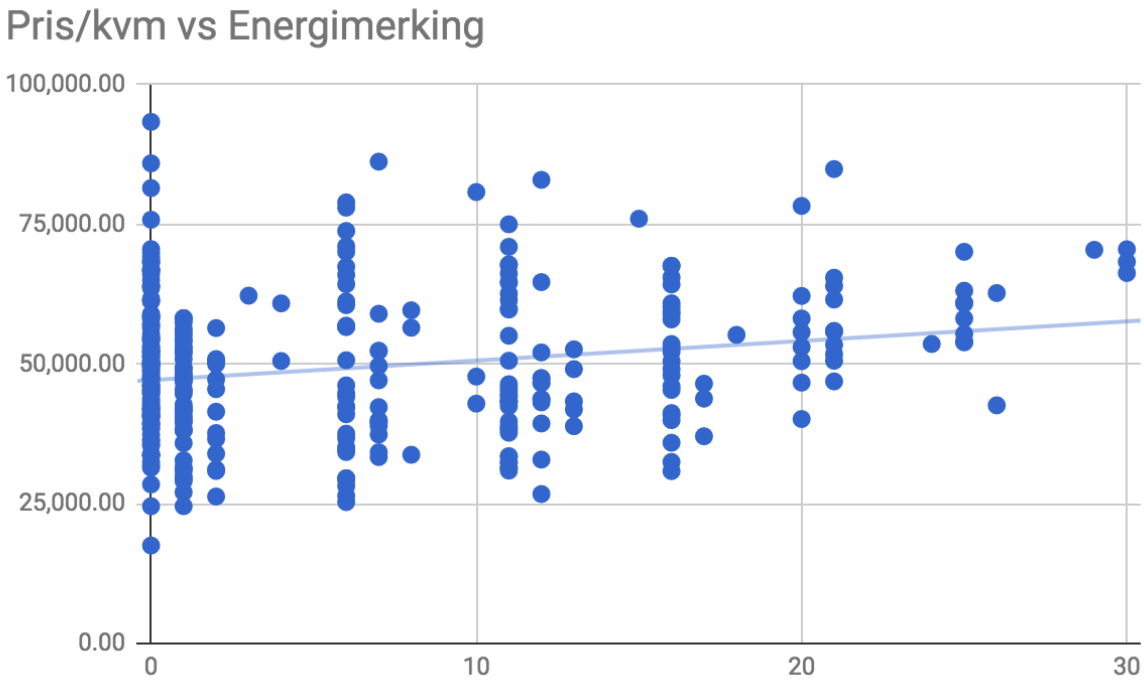
Energimerking ble ikke brukt i ID3 fordi den vel kreve ganske mange uavhengige variable (35 stk. slik jeg har gjort det). Energimerking blir delt opp i 35 kombinasjoner av A til G og Mørkegrønn til Rød:



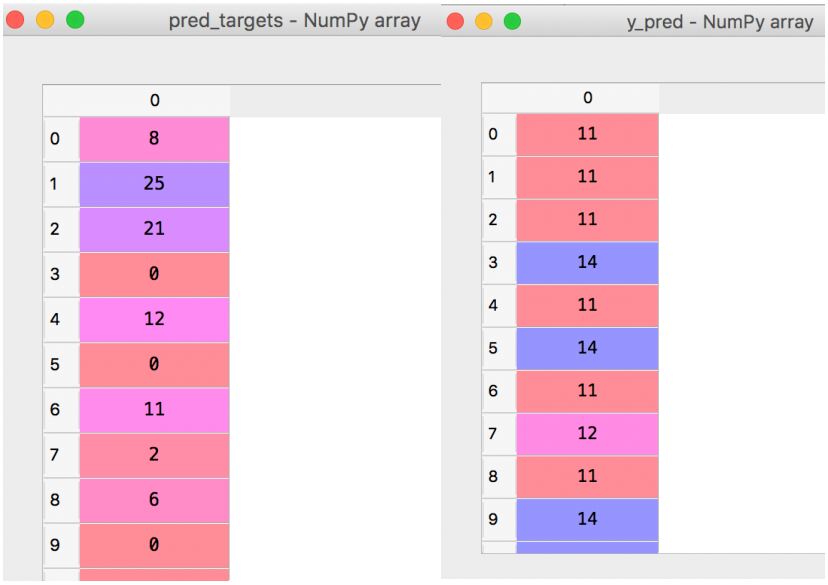
Dette ble enkodet som heltall fra 0 til 34 slik:

A	B	C	D	E	F	G	
30	25	20	15	10	5	0	**Rød**
31	26	21	16	11	6	1	**Oransje**
32	27	22	17	12	7	2	**Gul**
33	28	23	18	13	8	3	**Lyserønn**
34	29	24	19	14	9	4	**Mørkegrønn**

Når dette plottes mot pris ser vi en mulig korrelasjon høyere bedre energimerking og kvadratmeterprisen:



Det er 102 av 435 boliger som mangler energimerking i datasettet. Disse ble forsøkt tilnærmet vha. random forrest, men resultatet var svært dårlig. Skjermbildene under viser målverdier (venstre) og estimerte verdier (høyre):



Isteden satt jeg manglende celler til å være 0, som er den mest vanlige scoren boliger fikk.

Etter 1000 kjøringer ble snittfeilen på estimatene 10.2%. Dette er en forbedring fra 11.5% slik vi hadde uten energimerking.

Appendix:

1 – Google distance API function:

```
function getDirection(origin, dest, unit)
  var directions = Maps.newDirectionFinder()
    .setOrigin(origin).setDestination(dest)
    .setMode(Maps.DirectionFinder.Mode.DRIVING)
    .getDirections();


  try{
    var d = directions.routes[0].legs[0].distance.text;
  }
  catch (e){
    console.log("error",e)
  }

  //var distance, value = (this).attr('class');
//  filterClass = filterClass ? filterClass.split(' ').slice(-1)[0] : '';

  var distance, value = d.split(" ")[0].replace(",", ""), text = d.split(" ")[1];

  if(text == unit) {
    distance = value;
  } else if(text == "km" && unit == "mi") {
    distance = value / 1.6;
  } else {
    distance = value * 1.6;
  }
  return distance;
}
```