Bruk av lineær regresjon for modellering av topografiske data

Siv Aalbergsjø

I dette prosjektet er Ordinary Least Squares (OLS), Ridge og Lasso regresjon implementert for å modellere topografiske data med en to-dimensjonal funksjon bestående av polynomer av ulik grad. Bootstrap resampling ble benyttet for å undersøke hvordan disse metodene kan gi forbedrede modeller og k-fold cross validation er benyttet for å bestemme beste parameter λog hvilken grad polynomer som er nødvendig. Metodene ble først testet ut på en kjent matematisk funksjon. Deretter ble metodene benyttet for å modellere topografiske data. For de topografiske dataene ble det funnet at OLS ga tilsvarende gode modeller som de mer avanserte dataene, men det trengs polynomer av svært høy grad for modellere terrengdata av typen som ble benyttet i denne oppgaven.

I. INTRODUKSJON

Maskinlæring går ut på å la en datamaksin modellere eller gjøre prediksjoner etter å ha trent på data for samme eller lignende fenomen. Det finnes ulike måter å dele inn maskinlæring på, men en av måtene er å dele det inn i *klassifiseringsproblemer*, *regresjon*, eller *clustering* (Hjorth-Jensen, Morten, 2021). Felles for maskinlæringsmetoder er at det inngår *datasett* som benyttes for trening, validering og testing, det lages en *modell*, som f.eks. kan være en funksjon av noen parametere, og det inngår en *kostfunksjon* som brukes til å evaluere kvaliteten til modellen (Hjorth-Jensen, Morten, 2021).

Hensikten med denne oppgaven er å undersøke om og hvordan lineær regresjon med tilpasning av polynomer i to dimensoner kan benyttes til å modellere landskapsdata. I utgangspunktet vil man kunne forvente at polynomer i to dimensjoner vil kunne modellere i det minste noen typer landskap godt. Men sannsynligvis vil en slik modell likevel ha problemer med å representere svært varieende landskap og bratt terreng på en god måte. Blant annet vil man i prinsippet trenge 2 ekstra potenser i polynomet sitt for hver bakketopp som skal modelleres(!). For å implementere og validere kode vil det gjøres modellering av Franke-funksjonen (se eget avsnitt) som i det minste visuelt kan se ut tilå ha lignende former som man kan forvente av et landskap.

I denne rapporten vil jeg i det følgende gjøre rede for metodene som er benyttet i den modelleringen og hva som er er forventet å være styrker og svakheter ved disse. Deretter vil jeg beskrive implementeringen av metodene og resultater ved uttesting på Franke-funksjonen. Til slutt vil jeg gjøre rede for resultatene jeg får når jeg benytter metodene på topografiske og diskutere styrker og svakheter ved bruk av metodene på denne type data.

Programkode, vedlegg og ytterligere plott finnes på https://github.com/sivgaa/fys-stk4155/tree/main/Project1.

II. MODELLERING GJENNOM LINEÆR REGRESJON

En lineær modell kan uttrykkes som

$$\tilde{\boldsymbol{y}}(\boldsymbol{x}) = \sum_{j=0}^{n-1} \beta_j \boldsymbol{f_j}(\boldsymbol{x})$$

hvor $\boldsymbol{f_j}(\boldsymbol{x})$ er lineært uavhengige funksjoner. I denne oppgaven benyttes polynomer for $\boldsymbol{f_j}(\boldsymbol{x})$, slik at

$$\tilde{\boldsymbol{y}}(\boldsymbol{x}) = \sum_{j=0}^{n-1} \beta_j \boldsymbol{x^j}$$

eller uttrykt på matriseform

$$\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}.$$

Her er $\tilde{\boldsymbol{y}}$ utfall som predikeres av modellen vår, mens $\boldsymbol{X}$ kalles en *designmatrise* som inneholder de ulike frihetsgradene eller egenskapene (*features*) som modellen er avhengig av og $\boldsymbol{\beta}$ er parametere frihetsgradene. Modellen er en linær modell i $\boldsymbol{\beta}$ så lenge egenskapene x_i er lineært uavhengige av hverandre.

Regresjonen består i å bestemme de beste paramterne $\boldsymbol{\beta}$ og optimeringen består i minimere en kost-funksjon $C(\boldsymbol{X}, \boldsymbol{\beta})$ som beregner feilen modellen gjør i predisjonen av data. Det er ulike måter å beregne feilen på, som gir opphav til ulike metoder for regresjon. Men vi ser at dersom modellen vår skal inne holde mange frihetsgrader, eller bygger på et stort antall datapunkter, så blir modellering regnetung. I en lineærregresjon vil dette f.eks. tilsvare å tilpasses et polynom av høy orden (mange frihetsgrader) eller å benytte mange datapunkter for å tilpasse modellen.

A. Minste kvadraters metode - Ordinary Least Squares (OLS)

Vi tar utgangspunkt i en antagelse om at dataene våre kan beskrives av en kontinuerlig funksjon $f(\boldsymbol{x})$ og en normalfordelt feil $\boldsymbol{\varepsilon} \sim N(0, \sigma^2)$

$$\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\varepsilon}$$

Deretter approksimerer vi denne funksjonen med modellen vår $\tilde{\boldsymbol{y}}$ som kommer fra løsningen av ligningene for minste kvadraters metode (OLS), slik at modellen vår approksimeres med

$$\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}.$$

Dette er en anvendbar modell fordi det medfører at $\mathbb{E}[y] = \mathbf{X}\boldsymbol{\beta}$, og at variansen til dataene våre, vil være gitt ved $\mathrm{Var}(y_i) = \sigma^2$. Den matematiske utledningen av denne og de følgende forventningsverdiene finnes i Vedlegg A.

I en lineær regresjon som benytter minste kvadraters metode, så er kost-funksjonen, det vil si feil-funksjonen som minimeres når modellen optimeres gitt ved kvadratet av feilen:

$$C(\boldsymbol{X}, \boldsymbol{\beta}) = MSE(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2,$$

hvor $\boldsymbol{y}$ er de ekte dataene og $\tilde{\boldsymbol{y}}$ er dataene som predikeres av modellen. En god modell vil ha en liten MSE-verdi for både de dataene den er trent på og nye data som brukes for å teste modellen. Tilpasning av et polynom av høy orden gi lav MSE-verdi for treningsdataene, fordi ved å øke ordenen til polynomet, kan vi lage en modell som passerer gjennom samtlige datapunkter. Derimot vil en slik modell ikke være god for data som ikke er benyttet ti å trene modellen, og vi vil se at vi får en mye høyere MSE-verdi for disse. Dette er en måte å avsløre overtilpasning av modellen.

For OLS, hvor kostfunksjonen tilsvarer MSE, er det mulig å finne uttrykket for de optimale parameterne $\boldsymbol{\beta}$ analytisk, og disse er gitt ved:

$$\boldsymbol{\beta} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}.$$

Da kan det også vises at $\mathbb{E}(\hat{\boldsymbol{\beta}}) = \boldsymbol{\beta}$, og at $\mathrm{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ se Vedlegg A.

I tillegg til å se på MSE vil vi i denne oppgaven benytte R^2 for å vurdere hvor god modellen vår er,

$$R^2(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y})^2}.$$

Mens MSE ideelt sett skal ta så lave verdier som mulig for en god modell, så vil R^2 ta verdier nær 1 hvis modellen er god.

B. Ridge og Lasso

Det vil finnes tilfeller hvor det å løse med minste kvadraters metode (OLS) ikke er hensiktsmessig. Da kan man benytte andre algoritmer, hvor kost-funksjonen er en litt annen f.eks. Ridge regresjon hvor kost-funkjonen er gitt ved

$$C(\boldsymbol{X}, \boldsymbol{\beta})_{Ridge} = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 + \lambda\boldsymbol{\beta}^2$$

eller en Lasso-regresjon hvor

$$C(\boldsymbol{X}, \boldsymbol{\beta})_{Lasso} = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 + \lambda|\boldsymbol{\beta}|,$$

hvor $|\boldsymbol{\beta}|$ betegner $\sqrt{\boldsymbol{\beta}^2}$.

Ridge og Lasso gir løsninger som ligner på OLS, men hvor noen av frihetsgradene vil vektes ned. I Ridge-regresjon vil noen frihetsgrader gis mindre vekt gjennom lavere β-parametere, mens i Lasso vil noen frihetsgrader undertrykkes helt, gjennom at parameterne settes til null (Hastie *et al.*, 2009). Dette kan være en fordel dersom problemet har høy dimensjonalitet, men vil føre til tap av nøyaktighet dersom viktige frihetsgrader undertrykkes.

Analogt med OLS kan det vises for Ridge-regresjon at (se Vedlegg A)

$$\mathbb{E}\left[\hat{\boldsymbol{\beta}}^{\mathrm{Ridge}}\right] = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_{pp})^{-1}(\mathbf{X}^\top\mathbf{X})\boldsymbol{\beta}.$$

og

$$\mathrm{Var}[\hat{\boldsymbol{\beta}}^{\mathrm{Ridge}}] = \sigma^2[\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}]^{-1}\mathbf{X}^T\mathbf{X}\{[\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}]^{-1}\}^T.$$

Disse uttrykkene ligner uttrykkene for OLS, men parameteren λ dukker opp og utgjør en forskjell. Vi ser også at dersom $\lambda = 0$ vil uttrykkene for Ridge og OLS være identiske. Men vi ser også at dersom λ blir stor, vil forventningsverdien av de tilhørende parametrne gå mot 0. Det er dette som medfører at noen av koeffisientene vektes ned i optimeringen.

For topografiske data er det i utgangspunktet naturlig å tro at alle frihetsgradene er relevante, derfor vil f.eks. Lasso-regresjon forventes å være lite hensiktsmessig til akkurat dette. Derimot vil Ridge-regresjon kunne gi gode løsninger, ettersom ikke nødvendigvis alle frihetsgradene er like viktige/representerer store variasjoner i landskapet.

C. Skalering av data, splitting av datasett og resampling

De flesete maskinlæringsalgoritmer fungerer best dersom designmatrisen ($\boldsymbol{X}$) er *skalert* (Raschka *et al.*, 2022). Hensikten med å skalere designmatrisen er å standardisere de ulike egenskapene slik at ikke noen av dem (eller sågar konstantleddet i modellen) blir skjevt behandlet i optimaliseringen fordi de tar verdier av annen skala enn de andre (Hjorth-Jensen, Morten, 2021). I denne oppgaven er det benyttet såkalt *standardskalering* hvor egenskapene er normalisert gjennom å først trekke fra gjennomsnittet i hver kolonne av designmatrisen, for deretter å dele på standardsavviket i kolonnen. Dette påvirker

ikke resultatene for OLS, men vil påvirke modellen i tilfellene med Ridge og Lasso (Hjorth-Jensen, Morten, 2021). For uttesting med Franke-funksjonen, er problemet uansett dimensjonert slik at x, y og z tar verdier mellom 0 og 1, slik at skalering vil ha liten påvirkning. Derimot forventes det at skalering blir viktig i tilfellet med ekte topografiske data.

Når modellen trenes, dvs at polynomet tilpasses til dataene våre, vil vi kunne få en svært god tilpasning til de datapunktene vi har brukt for å trene modellen dersom polynomet har nøy nok grad. Likevel vil modellen vår kunne være dårlig til å predikere verdier for punkter den ikke har trent på. Dette kalles overtilpasning, hvor vi har fått en modell som reproduserer samtlige datapunkter som er brukt til å trene modellen, men kan ha fått oppførsel mellom disse datapunktene som ikke ligner det som skal modelleres. For unngå dette, er det lurt å holde til side noen av datapunkter for å teste modellen i etterkant. Ved å teste modellen på andre punkter en den er laget på bakgrunn av, får vi et bedre mål på hvor god modellen er. Dette kalles å splitte dataene i treningssett og testsett. Hvor mye av dataene som holdes av til testsettet vil variere fra tilfelle til tilfelle, men å holde av 20-40 % av dataene er vanlig (Raschka *et al.*, 2022).

D. Bootstrap, k-fold cross validation

Det finnes ulike måter å *resample* fra samme datasett når man tilpasser modellen sin. Fordeler med dette kan være hvis datasettet i utgangspunktet ikke er så stort, men sannsynligvis også hvis datasettet er så stort at det ikke er mulig å benytte hele. Da er det nyttig med metoder for å resample som ikke favoriserer deler av datasettet, da dette vil kunne føre til bias i modellen.

Bootstraping er betegnelsen på en metode hvor man resampler fra det opprinnelige datasettet på en tilfeldig måte. Det dannes nye datasett ved å trekke datapunkter tilfeldig fra de opprinnelige treningsdataene, med tilbakelegging (Hjorth-Jensen, Morten, 2021). Deretter benyttes det nye datasettet til å tilpasse en modell, og feilen beregnes ved å benytte test-dataene. Dette gjentas så mange ganger som man ønsker. Slik lages i prinsippet et vilkårlig antall modeller (selvsagt begrenset av antall datapunkter) og man får en situasjon hvor man simulerer et større datasett. Dersom det lages mange modeller for den samme funksjonen, kan det vises at MSE kan uttrykkes som en sum av tre ledd, hvorav det siste leddet skyldes den ukjente feilen i måledataene (se Vedlegg B):

$$\begin{aligned} MSE &= \mathbb{E}\left[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2\right] \\ &= \mathbb{E}\left[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2\right] \\ &= \text{Bias}[\tilde{y}] + \text{var}[\tilde{y}] + \sigma^2 \\ &\leq \text{Bias}[\tilde{y}] + \text{var}[\tilde{y}]. \end{aligned}$$

Her representerer det første leddet biasen, altså hvor mye modellen avviker fra det vi forsøker å modellere, mens det andre leddet gir variansen i prediksjonene som modellen gjør. En kompleks modell, i vårt tilfelle et polynom av høy grad, forventes å ha lav bias. Men da for høy kompleksitet i modellen kan føre til overtilpasning, vil dette vises gjennom at vi får en høy varians isteden, noe som igjen gir en større MSE-verdi. Vi er derfor på jakt etter modellen som gir lavest mulig MSE gjennom å vinne området som har det beste kompromisset mellom bias og varians.

En annen måte å resample dataene på er gjennom det som kalles *k-fold cross validation*. I denne metoden så deles treningsdataene i k deler, og deretter gjennomføres tilpasning av modell, prediksjoner og feilestimering k ganger, hvor hver av delene får spille rollen som test-data én gang . På denne måten får man mer robuste estimat av feilen enn om man bare modellerte én gang. Denne metoden er nyttig dersom man vil f.eks. finne optimale metaparametere, som f.eks. λ i Rige- eller Lasso-regresjon, og det vanlige er å dele treningsdataene sine i omtrent 10 deler (Raschka *et al.*, 2022).

E. Testfunksjon: Franke-funksjonen

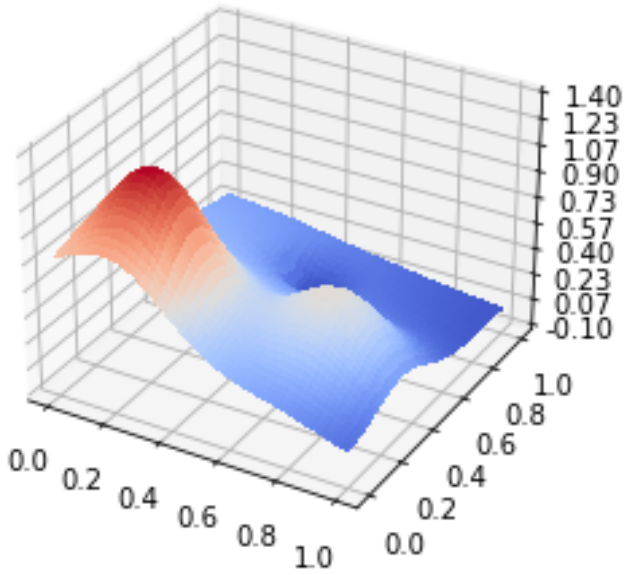
Dataene som skal modelleres i denne rapporten er topografiske data for geografiske områder i Norge. Det er derfor naturlig å anta at polynomer i to dimensjoner (x og y) kan gi en god modell for landskapsformene, men at polynomgraden som behøves for å modellere landskapet vil avhenge av hvor kupert landskapet er. For å teste ut implementeringen av metodene, benyttes derfor data generert fra en matematisk funksjon, som tilføres støy. Franke-funksjonen

$$\begin{aligned} f(x,y) = &\frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) \\ &+ \frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right) \\ &+ \frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) \\ &- \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right). \end{aligned}$$

Denne funksjonen har likheter med et landskap, se Figur 1, og kan derfor fungere som en godt datasett for å teste implementeringen av algoritmene.

III. IMPLEMENTERING I PROGRAM OG UTTESTING MED FRANKE-FUNKSJONEN

Regresjonene er implementert i python ved bruk av bibliotekene NumPy(Harris *et al.*, 2020) og Scikit-learn(Pedregosa *et al.*, 2011) og gjennomgående rapping av kodebiter fra ulike deler av forelesningsnotatene i kur-



Figur 1 Franke-funksjonen uten støy

set FYS-STK4155 (Hjorth-Jensen, Morten, 2021). Programmet er satt opp slik at det gjennomfører regresjoner med polynomer av grad 0 til nmax og starter ved at parametere settes og det velges metode for regresjon og hvilket datasett det skal kjøres på.

```
nmax = 10             #høyeste orden av polynom
N = 1000              #antall datapunkter
metode = 'Ridge'      #Ridge, OLS eller Lasso
datasett= 'franke'    #franke eller terreng
nbootstraps = 0       #0= ingen bootstrap
kfoldnumber = 10      #0= ingen k-fold cross val
kun_skalert = True
plotBeta = False
plotModell = False
```

Deretter settes datasettet opp $z = f(x, y)$, og for Franke-funksjonen er det lagt til støy $\varepsilon = 0.1 * np.random.normal(0, 1, x.shape)$. Koden i sin helhet, inkludert filen siv_funksjoner.py som bl.a. inneholder funksjoner for å sette opp designmatrise, gjøre regresjon og generere plott finnes på https://github.com/sivgaa/fys-stk4155/tree/main/Project1. Hovedprogrammet er filen proj1.py.

A. Splitting av data og skalering

I denne oppgaven er det testet ut skalering med den såkalte standardskaleringen i Scikit-learn (Pedregosa *et al.*, 2011). For at beregningen av MSE-verdiene så skal bli riktige/relevante må man da enten skalere tilbake før man sammenligner modellen med test-data, eller man må også skalere test-dataene. I dette tilfellet falt valget på det siste. Standardskaleringen innebærer at gjennomsnittet er trukket fra, og verdiene er delt på sitt standardavvik.

```
import numpy as np
from sklearn.preprocessing import StandardScaler

#NB! gir singulær matrise med OLS
scaler = StandardScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
#skalerer y manuelt
y_train_scaled =
    (y_train-np.mean(y_train))/np.std(y_train)
y_test_scaled =
    (y_test-np.mean(y_test))/np.std(y_test)
```

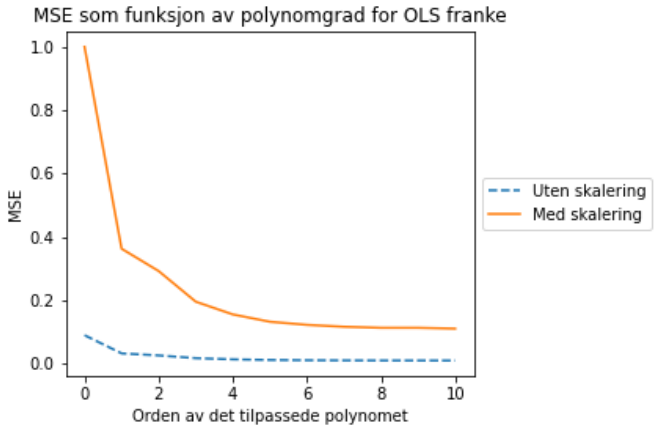
I denne oppgaven er det gjennomgående benyttet en splitting av data i treningssett og testsett i 80:20-forhold. Det er, med unntak der det er nevnt eksplissitt benyttet datasett på 1000×1000 punkter, slik at både test- og trenignssettene er store. Dataene ble splittet ved hjelp av den innebygde funksjonen i scikit-learn (Pedregosa *et al.*, 2011).

B. OLS

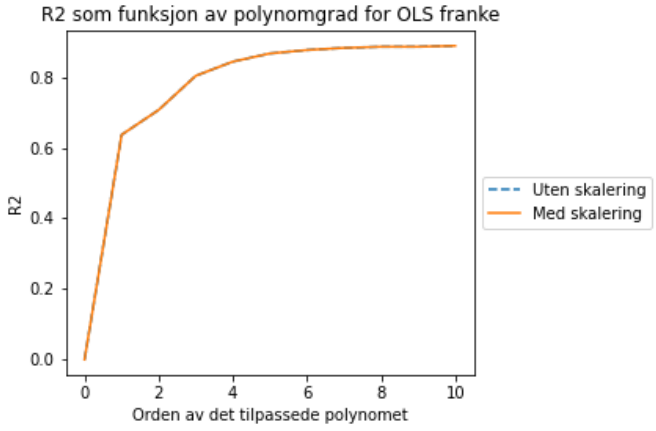
OLS er programmering direkte gjennom det analytiske uttrykket (se over). Fordi skaleringen medfører at den første kolonnen i designmatrisen blir 0, vil det ikke være mulig å invertere matrisen for å beregne $\boldsymbol{\beta}$ direkte i OLS, derfor er også NumPy sin pseudo-invers-funksjon benyttet.

Figur 2 og 3 viser henholdsvis MSE og R^2 som funksjon av kompleksitet for OLS-regresjon av Franke-funksjonen. Vi ser at MSE-verdien blir lavere i tilfellet hvor det ikke er benyttet skalering, mens R^2 er identisk i de to tilfellene. Dette skyldes at R^2 er normert, slik at skalaen på dataene ikke påvirker feilen, mens MSE-verdien vil avhenge av verdiene som dataene og modellen tar. Derimot ser vi at de to linjene følger hverandre, noe som viser at hvor god modellen er, ikke påvirkes av om dataene er skalert i dette tilfellet. Vi ser også at for polynomer av grad høyere enn 7-8, så får tilsynelatende ikke noe bedre modell ved å øke kompleksiteten ytterligere. Ser vi på β-verdiene som produseres i modellering, så ser også at det ikke er store forskjeller mellom å bruke skalerte eller uskalerte data for OLS-regresjonen, se Vedlegg C.

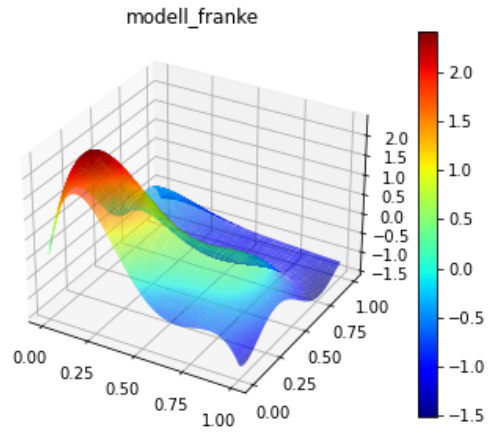
Ved å plotte modellene fra OLS-regresjonen ved polynomer av 5. go 10. grad (se Figur 4 og 5, ser vi at det med det blotte øye ikke er stor forskjell på de to modellene. Dette viser også at det å øke polynomgraden ikke



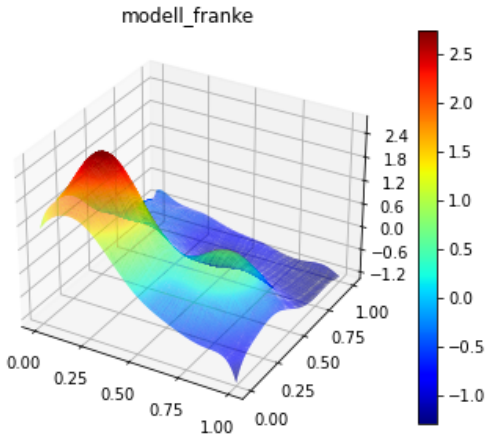
Figur 2 MSE-verdi som funksjon av kompleksitet for OLS-regresjon av Franke-funksjonen.



Figur 3 R^2 som funksjon av kompleksitet for OLS-regresjon av Franke-funksjonen.



Figur 4 Modellen fra OLS-regresjon av Franke-funksjonen med polynom av 5. grad.



Figur 5 Modellen fra OLS-regresjon av Franke-funksjonen med polynom av 10. grad.

har mye hensikt, i den forstand at det ikke gir en modell av et mer variert terreng.

Ved å gjennomføre resampling med bootstraping, blir det tydelig at selv om variansen øker med kompleksiteten, er denne variansen svært liten i forhold til biasen, selv for polynomer av høy grad, se figur 6. Dette tyder på at overfitting ikke er et problem, i det minste for de verdiene som er aktuelle her. Ved å gjennomføre modelleringen med 100×100 datapunkter blir det tydelig at selv ikke nå er et polynom av grad 10 overtilpasset, og det å redusere fra 100 til 10 bootstraps for 1000×1000 datapunkter gir ikke nevneverdig forskjelli feilen som estimeres, noe som tyder på at 100 bootstraps var overkill. Flere plot av feil som funksjon av kompleksitet finnes i Vedlegg C.

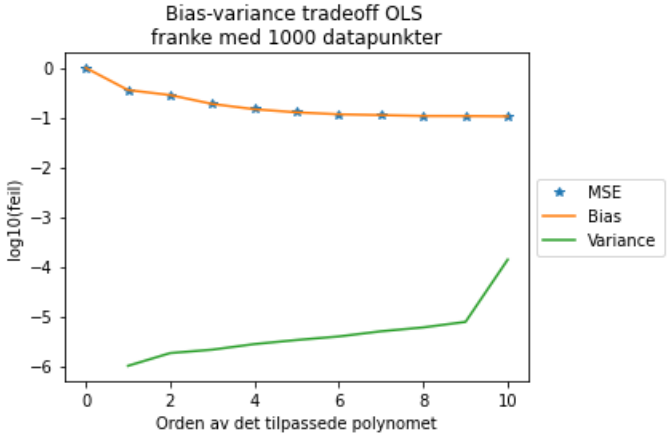
C. Ridge

For Ridge- regresjon er metodene i scikit-learn (Pedregosa *et al.*, 2011) benyttet og det er laget modeller med λ-verdier fra 10^{-8} til 10^{3}. Ved å plotte β-verdiene se Vedlegg D, blir det veldig tydelig at når λ går mot 0, sammenfaller resultatene av Ridge-regresjonen med OLS. Det blir også tydelig at når λ øker, så undertrykkes frihetsgradnee som gir store β-verdier.

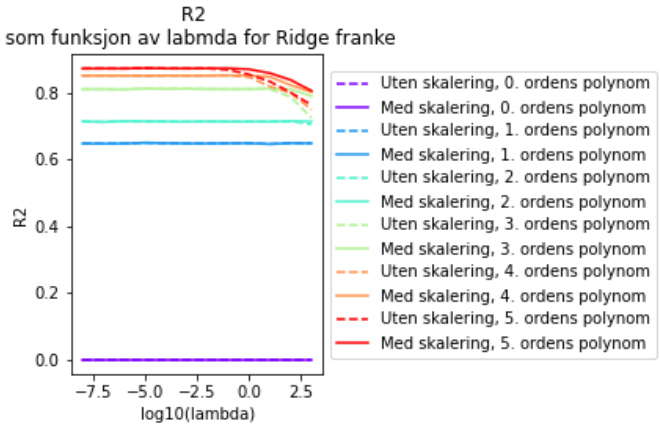
Av Figur 7 ser vi at skalering av dataene får noe å si når polynomet har grad 2 eller høyere. Derfor er det videre kun benyttet skalerte data.

k-fold cross validation er implementert i koden ved bruk av Scikit-learn-funksjonene (Pedregosa *et al.*, 2011) KFold() og $cross_val_score()hvormanenkeltkanmateinnregresjonsmodelleneson learn.Foråestimerebestevalgavpolynomgradog\lambda$ er MSE -verdien som beregnes i metoden plottet opp og øyemål er benyttet for for å se etter laveste verdi.

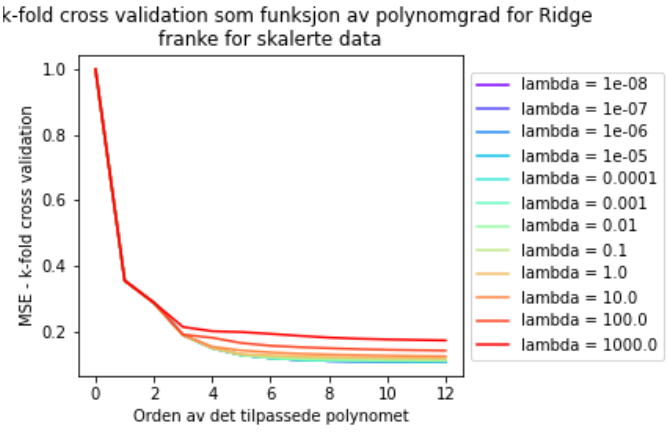
Ved å gjennomføre en k-fold cross validation på Rigdge-regresjonen får med polynomer av grad opp til 12 finner ser det ut til at, se Figurer 8 og 9, polynomer av grad 8-10 og λ-verdier under 10^{-4} er tilstrekkelig for



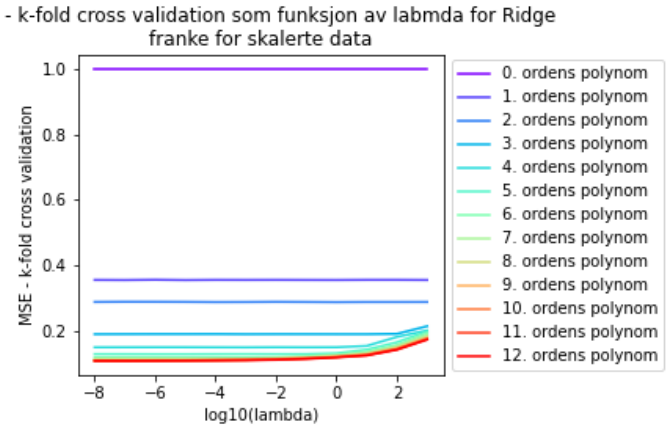
Figur 6 Feil som funksjon av polynomgrad for OLS-regresjon av Franke-funksjonen med bootstrapping i 100 iterasjoner for 1000×1000 datapunkter og polynomer av grad 0 til 10. Legg merke til logaritmisk skala på y-aksen.



Figur 7 R^2 som funksjon av kompleksitet for Ridge-regresjon av Franke-funksjonen med polynomer opp til 5. grad.



Figur 8 MSE som funksjon av polynomgrad funnet gjennom en k-fold cross validation med λ-verdier fra 10^{-8} til 10^{3} for Ridge-regresjon.



Figur 9 MSE som funksjon av pλ funnet gjennom en k-fold cross validation med polynomer av grad 0 til 12 for Ridge-regresjon.

å lage en så god modell som mulig med Ridge-regresjon. Et plott av denne modellen med polynom av 10. grad og $\lambda = 10^{-4}$ finnes i Vedlegg D, men det er ved bruk av øyet, ikke mulig å se nevneverdig forskjell på denne og de som er vist for OLS lengere oppe.

D. Lasso

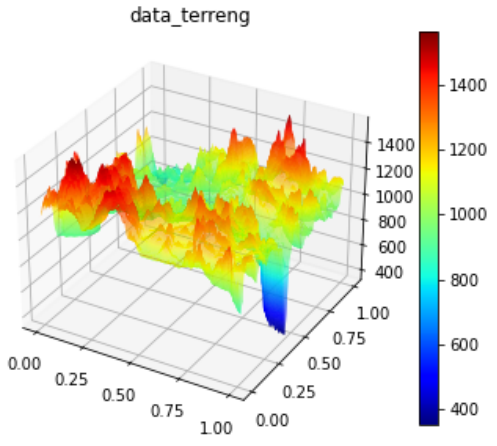
Også for Lasso-regresjonen er metodene i Scikit-learn benyttet, men på grunn av problemer med konvergens for kombinasjoner av små λ og høye polynomgrader, ble det begrenset til verdier av λ mellom 10^{-4} og 10^{3}. For å motvirke konvergensproblemer ble maksimalt antall interasjoner i Lasso-algoritmen satt til 20000, mot 1000 som er default, verdien.

Ved å se på β-koeffisientene, se Vedlegg E, så blir det tydelig at mange av disse fort blir satt til 0 etter hvert som λ nærmer seg 1. Det ser også ut, ved å se på R^2 og MSE-verdiene at for $\lambda = 10^{-3}$ er det tilstrekkelig med et polynom av 3. grad, mens det for mindre verdier av λ ser ut til at polynomer av høyere orden kan være å foretrekke. Men som nevnt oppsto det konvergensproblemer for lave verdier av λ i kombinasjon med polynomer av grad 4 og 5.

Ved gjennomføring av k-fold cross validation for intervallene nevnt over, ble det ikke nådd konvergens for $\lambda = 10^{-4}$ og polynom av 4. grad, noe som tyder på at dette er lite stabile saker. Plottene kan finnes i Vedlegg E, og der ligger også den tilstynelatende beste modellen med et 3. grads polynom og $\lambda = 10^{-3}$. Det tydelig at dette ikke er en god modell, men de groveste strukturene er på plass.

IV. RESULTATER MED TOPOGRAFISKE DATA

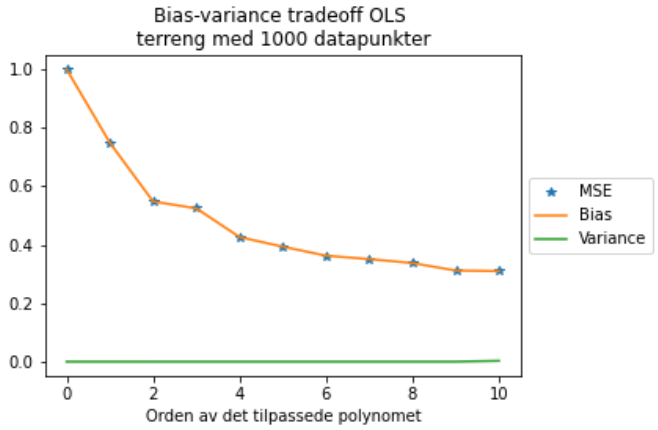
De topografiske dataene som er benyttet i denne modelleringen er data fra et fjellområde i Norge, hentet fra https://github.com/CompPhysics/MachineLearning/tree/master/doc/Projects/2024/Project1/DataFiles, og det er filen SRTM_data_Norway_1.tif som ble benyttet. Hvordan dette området ser ut, er vist i Figur 10.



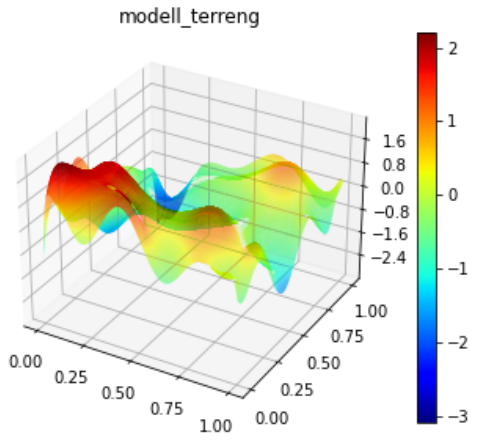
Figur 10 Modellen fra OLS-regresjon av terrengdataene med polynom av 10. grad.

Ved uttesting på dette datasettet, så ble det tydelig at det heller ikke her var noen forskjell på med og uten skalering, derimot var denne forskjellen merkbar for Ridge-regresjon med polynomer av høyere orden enn 3. Utvalgte plott vises i vedlegg F. Figur 11 viser resultatet av bootstrapping med 100 iterasjoner for OLS-regegresjonen på dette datasettet med polynomer av grad 0 til 10. Selv for polynom av 10. grad har, er denne modellen ikke stabilistert seg, og en ytterligere økning av polynomgraden forventes å kunne forbedre modellen ytterligere. Figur 12 viser modellen ved OLS og polynom av 10. grad, og det er tydelig at dette ikke er et godt bilde av terrenget, selv om vi ser at det høye området til venstre er representert, så ser vi at dalområdene er feil og det høye området til høyre i fiugren mangler.

Ved å se på Ridge-regresjon med polynomer av gra 0 til 20 (siden 10 åpenbart var litt lite) og λ-verdier fra 10^{-4} til 10^{3} ser vi at modellen vil gi best tilpasning for lave λ og høye polynomgrader. Selv ikke ved 20. grads polynom har vi nådd et bunnpunkt for MSE, se Figur 13. Det er ikke gjennomført noen k-fold cross-validation for dette datasettet, da det virker å være litt fåfengt basert på de første testene. Det fremstår rett og slett som at det beste vi kan gjøre er å kjøre opp kompleksiteten med OLS. Det virker også logisk, siden høy orden er nødvendig for å kunne reprodusere variasjonene i dette landskapet, slik at gevinsten ved å kunne undertrykke koeffisienter som vi kan oppnå for Ridge og Lasso vil være lite relevant i



Figur 11 Feil som funksjon av polynomgrad for OLS-regresjon av Terrengdataene med bootstrapping i 100 iterasjoner for 1000×1000 datapunkter og polynomer av grad 0 til 10.



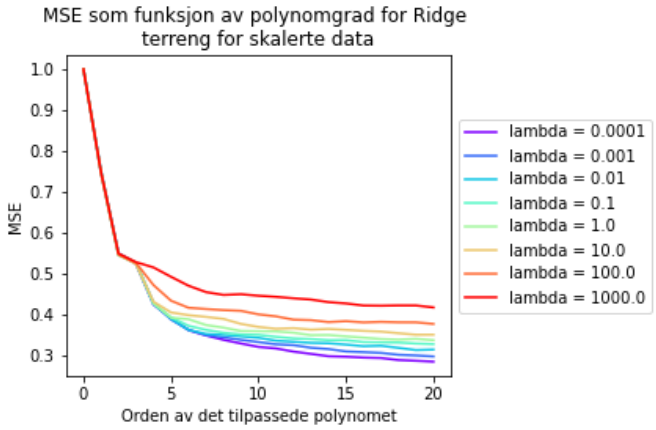
Figur 12 Modellen fra OLS-regresjon av terrengdataene med polynom av 10. grad.

dette tilfellet.

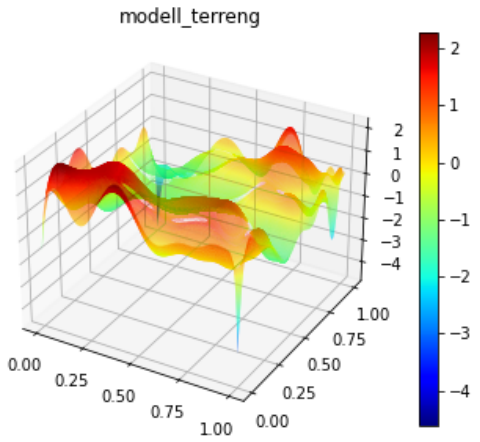
Den beste modellen jeg har laget har polynom av 30. grad for OLS. Selv for så høy grad av polynomet har ikke MSE som funksjon av kompleksitet flatet helt ut, se figur i Vedlegg F, men det ser heller ikke ut som det kan være mye å tjene på øke graden videre. Modellen med et polynom av grad 30 får MSE = 0.25, som jo ikke er tipp topp. Det kan vi også lett se av Figur 14 som viser modellen, ved inspeksjon bare med øyet, ser ikke denne bedre ut enn modellen som består av polynom til 10. grad.

V. OPPSUMMERING

Gjenta motivasjonen for å gjøre dette, og oppsummer hva du kom fram til. a.k.a. skriv abstractet på nytt, bare litt annerledes .. det blir bra. Og kanskje skriv noe om hva du synes var kløneте og vanskelig og hva som du har



Figur 13 MSE som funksjon av kompleksitet for Ridge-regresjon av terrengdata med polynomer opp til 20. grad og λ-verdier mellom 10^{-4} og 10^{3}.



Figur 14 Modellen fra OLS-regresjon av terrengdataene med polynom av 30. grad.

lyst til lå få til bedre i neste prosjekt.

Hensikten med denne oppgaven var å undersøke om og hvordan lineær regresjon med tilpasning av polynomer i to dimensoner kan benyttes til å modellere landskapsdata. I utgangspunktet forventet vi at polynomer i to dimensjoner ville kunne modellere i det minste noen typer landskap godt. Derimot ser vi at landskapet som forsøkes modellert i denne oppgaven i liten grad lar seg modellere ved polynomer av orden under 30. Etter hvert som kompleksiteten i polynomene øker, så blir det også mer regnetungt, og man kan spørre seg om polynomer er nyttige til dette formålet, hvertfall for denne typen landskap, hvor det går mye opp og ned.

A. Forbedringspotensialer

I denne oppgaven har jeg **ikke** benyttet et testsett som er holdt utenfor modellereingen helt fra starten. Jeg har brukt hele datasettet til å validere metoden. For Franke-funksjonen har dette lite å si, siden jeg hele tiden genererer nye data her. Men for høydedataene, så betyr det at jeg i praksis har brukt test-settet mitt til å bestemme hvilke parametrere jeg vil bruke i modellen, sånn at jeg ikke får gjort en reell test, noe som er uheldig. Ideelt sett så burde jeg splittet i test- og treningsdata med en gang, også splittet de treningsdataen i et trenings- og et valideringssett som ble brukt til å bestemme optimale parametere og metaparametere, før modellen ble testet med det opprinnelige test-settet (Raschka *et al.*, 2022).

Dessuten skulle jeg gjerne gjort selve koden bedre og med tid og regneressurser tilgjengelig prøvd ut polynomer av enda høyere grad for både OLS og Ridge på terrengdataene.

Jeg kunne tenke meg å forbedre programmet ved å gi parameterne for kjøringen i en input-fil istedenfor å skrive dem innn i toppen av programmet. Men det er jo tre prosjekter som skal gjøres, så jeg må jo ha noe å vokse på.

REFERANSER

C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant (2020), "Array programming with NumPy," Nature **585** (7825), 357–362.

T. Hastie, R. Tibshirani, and J. Friedman (2009), *The Elements of Statistical Learning: Data Mining, Inference and Prediction* (Springer Verlag, Berlin).

Hjorth-Jensen, Morten (2021), "Applied data analysis and machine learning," .

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011), "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research **12**, 2825–2830.

S. Raschka, Y. H. Liu, V. Mirjalili, and D. Dzhulgakov (2022), *Machine Learning with Pytorch and Scikit-Learn: Develop Machine Learning and Deep Learning Models with Python*, 1st ed. (Packt Publishing, Limited, Birmingham).