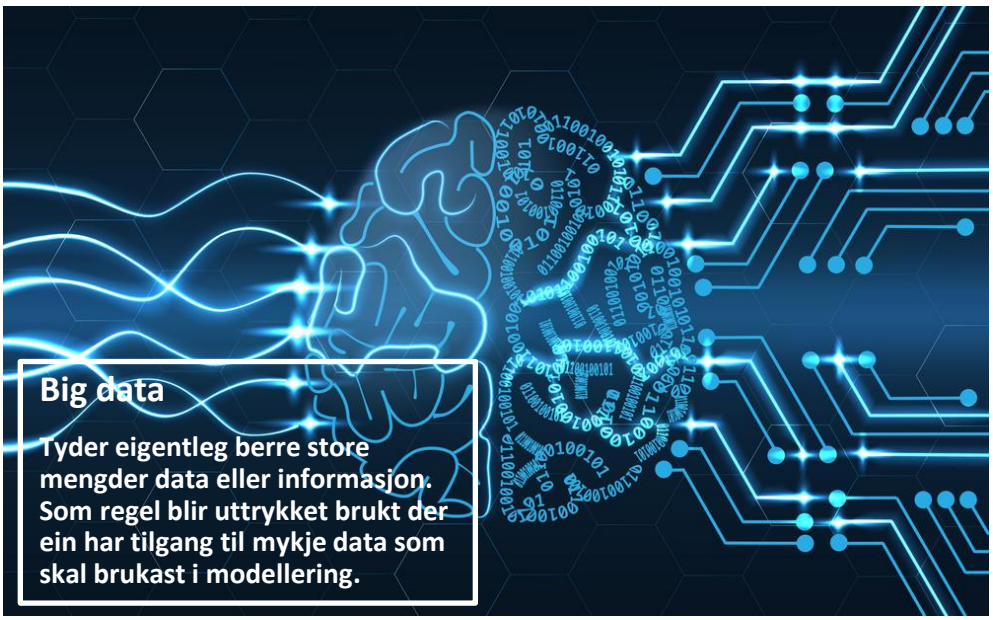
Modellering og maskinlæring med Python



Maskinlæring – kva er det?

Maskinlæring er ein del av fagområdet kunstig intelligens, men det er mindre avansert enn ein skulle tru. Den mest grunnleggande formen for maskinlæring er å lage matematiske modellar med hjelp av enkle regresjonar slik som de tidlegare har gjort i Geogebra.

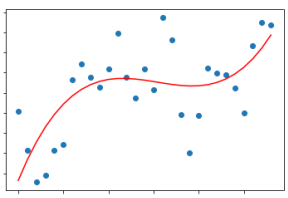
Etter kvart kan ein bruke meir og meir avanserte modellar, og gjerne system som er samansette av mange modellar som heng saman med kvarandre. Eit døme på eit slikt gigantisk system er vermodellering. Der blir det henta inn alle slags data om temperaturar, vindstyrke og lufttrykk frå mange stader i verda. Desse dataa blir nytta for å lage matematiske modellar, altså ulike matematiske funksjonar, slik de har gjort tidlegare. Vidare kan meteorologane sette inn nye verdiar i funksjonane sine for å rekne ut kva slags ver det er størst sannsyn for at det blir i ulike område på eit seinare tidspunkt. Korleis heng dette saman med kva område som er gyldig og kva som er usikkert i målingar?

Mykje modellering skjer utan at menneske aktivt programmerer alle delane, programma har «fått lov til» å lage sine eigne matematiske modellar grunna på ulike datasett. Dette kan gjere det vanskelegare å vite kva som ligg bak avgjerder som desse dataprogramma gjer, da ein del av algoritmane kan vere vanskeleg å få tilgang til. På ein måte er det fint at maskinar kan gjere dette for oss, på den andre sida kan det til dømes kunne føre til problem med diskriminering.

Diskriminering på grunn av maskinlæring har ein sett fleire døme på. Dersom eit maskinlærings-program for ansiktsattkjenning får tilgang til datasett som nesten berre er samansett av kvite menn, vil dette programmet bli mykje dårlegare på å kjenne att ansikta til kvinner og ikkje-kvite menneske. Dette har ein kunna merke på enkelte mobilmerke sin ansiktsattkjenning for å låse opp telefonen. Det er og funne døme på diskriminering i tilsettingar, til dømes der eit maskinlæringsprogram føreslo Amazon å heller tilsette menn enn kvinner, bygd på inndata om at det var veldig mange fleire menn enn kvinner tilsette der i utgangspunktet. Les gjerne meir her: https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G

Frå data

Via regresjon

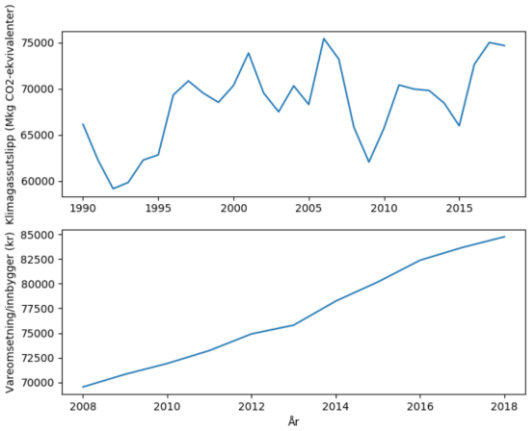


Til modellar

Lage graf med Python

Kvifor Python?
De kan alt teikne grafar med Geogebra, så kva er poenget med å bruke Python i staden?

I Geogebra kan vi plotte ein funksjon, eller legge inn punkt og få plotta ein graf frå punkta. Sidan vi må skrive inn alle punkta for hand, kan det vere tungvint dersom vi har veldig mange målingar. Dette gjeld særs om vi har ferdige filer med mange målingar. I Python kan ein lage sub-plot, det vil seie bilete med fleire grafar som viser ulike målingar i same bilete. Men hovudgrunnen til å bruke Python til plotting av grafar, er at det er byrjinga på å bruke statistikk for å analysere data. Det er slik forskarar jobbar når dei har samla inn data!



I Python kan du bruke data som ligg i ferdige filer. Du slepp å skrive dei inn! Det som er viktig, er at filene er rett type. Filene kan berre innehalde tal og lagrast i tekstformat (.txt) for dei døma vi ser på her.

Plotting av grafar i Python er samansett av seks deler:

1. Importere pylab-biblioteket i programmet
2. Importere data frå ein .txt-fil
3. Legge dataa i ein tabell (array) for x-verdiane og ein for y-verdiane
4. Skrive kva for nokre data du skal plotte
5. Skrive inn tittel og merkelappar på aksane
6. Få plottet til å visast på skjermen

I tillegg kan du skrive inn korleis grafen skal sjå ut med til dømes farge, linjestil og kor tjukk linja skal vere.

```
from pylab import *
```
→ Importerer det naudsynte biblioteket

```
plot(x, y)
```
→ Lagar eit plott av alle verdiane i tabellane x og y

```
show()
```
→ Viser figuren av plottet på skjermen

```
tabell = loadtxt("data.txt")
```
→ Importerer fila data.txt og legg henne inn i ein tabell med namn tabell

```
x = tabell[:,0]
y = tabell[:,1]
```
→ Plukkar alle x-verdiane frå tabellen og legg dei i ein tabell som heiter x. Tilsvarande for y-verdiane.

```
title("Tittel")
xlabel("x-verdier")
ylabel("y-verdier")
```
→ Lager tittel og merkelappar til aksane.

Oppgåver

1. Bruk kodelinjene på venstre side for å lage eit program som plottar ein graf frå ei fil som heiter data.txt.
2. Skriv ein omtale for kvar del av programmet ditt utan å sjå på denne sida. Legg dei gjerne inn som kommentarar i koden din. Sammenlikne så omtalane dine med dei på denne sida. Kva for nokre skilnader fann du?
3. Finn eit valfritt datasett frå statistisk sentralbyrå, og bruk Python til å lage ein graf av dataa. Kva må du endre i dømet på denne sida?

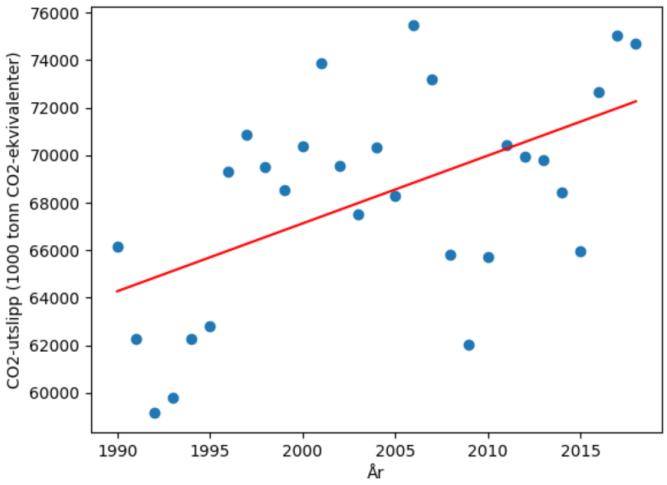
Lineær regresjon med Python

Kvifor Python?

De har at gjort mange lineære regresjonar med Geogebra, så kva er poenget med å bruke Python i staden?

I Geogebra må vi skrive inn alle målingane våre for hand, eller kopiere frå til dømes Excel. Da kan det vere vanskeleg å gjere ein regresjon dersom vi har mykje data. Tenk deg at du har 100 målingar, da tek det veldig lang tid å skrive inn alle målingane i reknearket i Geogebra. Det kan fort bli ein feil eller to undervegs og. Tenk om du har endå fleire målingar, kan hende fleire enn 1000! Da blir det ganske kjedeleg å skrive inn alle tala.

Samla klimagassutslepp for Norge 1990 – 2018



I Python kan du bruke data som ligg i ferdige filer. Du slepp å skrive dei inn! Det som er viktig, er at filene er rett type. Slik som med plotting av grafar i Python.

Ein annen fordel med å bruke Python, er at det er eit programmeringsspråk som blir nytta av forskarar som jobbar med modellering. Andre forskararere kan bruke andre programmeringsspråk som er spesialtilpassa akkurat den modelleringen dei jobbar med, men de følger dei samme prinsippa som Python. Så dersom du kan modellere med Python, er det enklare å modellere i eit anna programmeringsspråk enn om du berre har brukt Geogebra tidlegare.

Regresjon i Python er samansett av seks delar:

1. Importere dei biblioteka som trengst
2. Importere data frå ei .txt-fil
3. Legge dataa i ein tabell for x-verdiane og ein for y-verdiane
4. Lage ein programmeringsfunksjon som seier at vi skal bruke ein lineær modell i regresjonen
5. Utføre regresjonen, og vise på skjermen kva den matematiske modellen blir, stigningstalet (a) og skjæringspunktet med y-aksen (b)
6. Teikne grafen med resultatet – tilpass koden for plotting av graf
   - Plotte punkta med eit scatterplot
   - Plotte regresjonsgrafen – korleis få programmet til å rekne ut y-verdiane for modellen?

Puslespeloppgåve

Sorter kodelinjene etter nummera i tekstboksen over, om regresjon i Python. Da vil du få den rette rekkefølgen i programmet ditt.

1.
```
co2data = loadtxt("co2utslipp.txt")
```

2.
```
from pylab import *
from scipy.optimize import curve_fit
```

3.
```
scatter(x_verdier, y_verdier)
plot(x_verdier, funksjon(x_verdier, a, b), color="red")
title("Samlet klimagassutslipp for Norge 1990 - 2018")
xlabel("År")
ylabel("CO2-utslipp (1000 tonn CO2-ekvivalenter)")
show()
```

4.
```
koeffisienter, kovarianser = curve_fit(funksjon, x_verdier, y_verdier)
a, b = koeffisienter
print(a,b)
```

5.
```
x_verdier = co2data[:,0]
y_verdier = co2data[:,1]
```

6.
```
def funksjon(x, a, b):
    return (a * x) + b
```

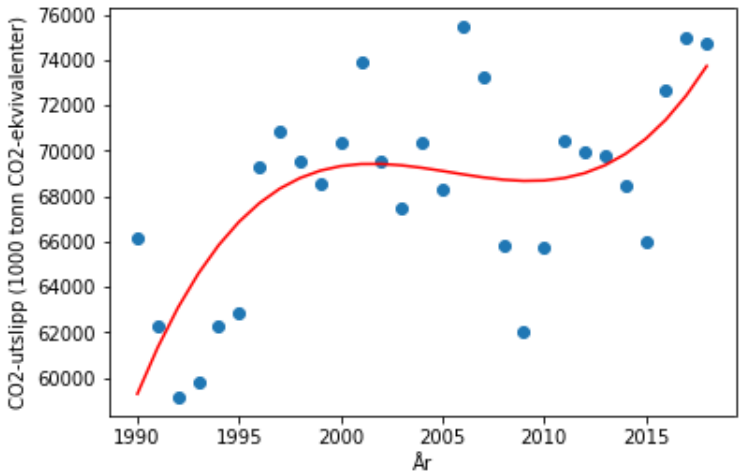
Ikkje-lineær regresjon med Python

Skilnad på lineær og ikkje-lineær regresjon i Python

Det er liten skilnad på korleis programma for lineær og ikkje-lineær regresjon ser ut. Den viktigaste skilnaden er å endre på kva for ein datasettet skal tilpassast. Det står eit oversyn over nokre funksjonstypar, og korleis ein skriv dei i python, på dei to neste sidene.

Men korleis skal vi vite kva for ein funksjonstype som passar til datasettet vårt? Går det an å finne det ut på førehand? Svaret på det spørsmålet kjem an på datasettet ditt. Du kan lage eit scatterplot av datasettet ditt, og sjå om du kjenner att ein av funksjonstypane. Som regel er det likevel vanskeleg å gisse seg til ein funksjonstype på førehand, og vi kan prøve oss fram med ulike funksjonstypar.

Samla klimagassutslepp for Norge 1990 – 2018



Programmeringa er veldig lik for lineær og ikkje-lineær regresjon. Vi må berre endre den lineære funksjonen til en ikkje-lineær funksjon. Da vil modellen vår tilpasse seg datasettet vårt best mogleg, og vi får vite koeffisientane som avgjer funksjonen for modellen vår. I tillegg må vi endre til rett tal koeffisientar i programmet vårt. Sjå dømet under på kva som må endrast.

```
def funksjon(x, a, b):
    return (a * x) + b
```

→

```
def funksjon(x, a, b, c, d):
    return (a*x**3)+(b*x**2)+(c*x)+d
```

```
koeffisienter, kovarianser = curve_fit(funksjon, x_verdier, y_verdier)
a, b = koeffisienter
print(a,b)
```

↓

```
koeffisienter, kovarianser = curve_fit(funksjon, x_verdier, y_verdier)
a, b, c, d = koeffisienter
print(a,b,c,d)
```

```
scatter(x_verdier, y_verdier)
plot(x_verdier, funksjon(x_verdier, a, b), color="red")
title("Samlet klimagassutslipp for Norge 1990 - 2018")
xlabel("År")
ylabel("CO2-utslipp (1000 tonn CO2-ekvivalenter)")
show()
```

↓

```
scatter(x_verdier, y_verdier)
plot(x_verdier, funksjon(x_verdier, a, b, c, d), color="red")
title("Samlet klimagassutslipp for Norge 1990 - 2018")
xlabel("År")
ylabel("CO2-utslipp (1000 tonn CO2-ekvivalenter)")
show()
```

Vi nytter data frå statistisk sentralbyrå over klimagassutslepp i Norge for åra 1990 – 2018, og ønsker å lage ein matematisk modell over utviklinga over tid. Vi prøver ut ulike funksjonstypar i regresjonsprogrammet, og ser kva for nokre modellar vi får.

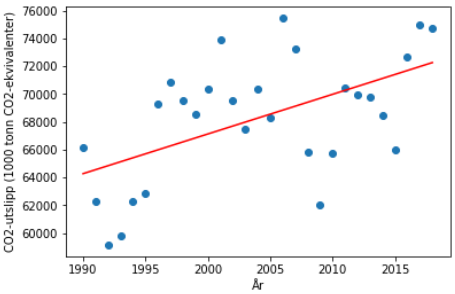
Vi bruker ein lineær modell i programmet vårt

```
def funksjon(x, a, b):
    return (a * x) + b
```

Programmet gir oss den lineære funksjonen

$$f(x) = 286x - 504059$$

som den beste tilpassinga til datasettet vårt.



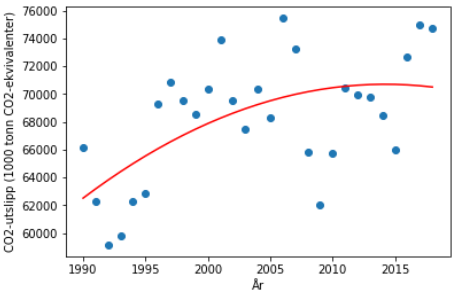
Vi bruker ein andregradsmodell i programmet vårt

```
return (a*x**2)+(b*x)+c
```

Programmet gir oss andregradsfunksjonen

$$f(x) = -14x^2 + 56298 - 56627608$$

som den beste tilpassinga til datasettet vårt.



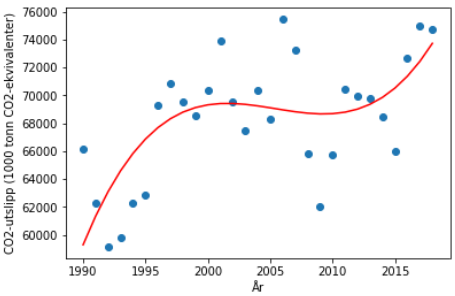
Vi bruker ein tredjegradsmodell i programmet vårt

```
return (a*x**3)+(b*x**2)+(c*x)+d
```

Programmet gir oss tredjegradsfunksjonen

$$f(x) = 3x^3 - 19753x^2 + 39612328x - 26479502215$$

som den beste tilpassinga til datasettet vårt.



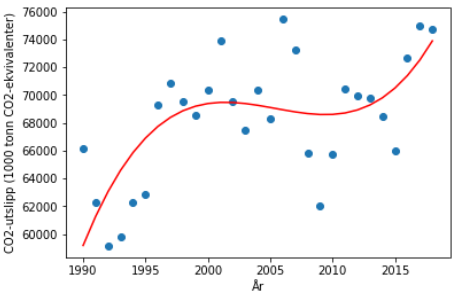
Vi bruker ein fjerdegradsmodell i programmet vårt

```
return (a*x**4)+(b*x**3)+(c*x**2)+(d*x)+e
```

Programmet gir oss fjerdegradsfunksjonen

$$f(x) = 0{,}004x^4 - 30x^3 + 79206x^2 - 92029574x + 39188643912$$

som den beste tilpassinga til datasettet vårt.



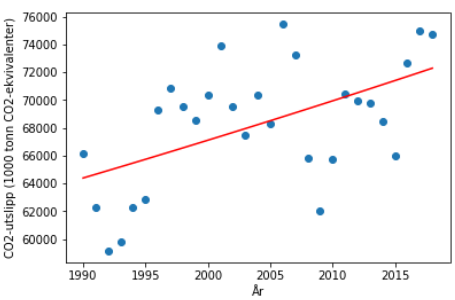
Vi bruker ein eksponentiell modell i programmet vårt

```
return a * (b**x)
```

Programmet gir oss eksponentialfunksjonen

$$f(x) = 17 \cdot 1{,}004^x$$

som den beste tilpassinga til datasettet vårt.



Vi bruker ein potensmodell i programmet vårt

```
return a*(x**b)
```

```
RuntimeError: Optimal parameters not
found: Number of calls to function has
reached maxfev = 600.
```

Når vi prøver å lage ein potens-modell for CO_2-dataa våre, får vi det ikkje til. For nokre datasett vil nokon funksjonstypar passe veldig dårlig. Da kan det hende at vi får ei feilmelding i programmet, om at vi har brote ei tidsavgrensing. Da har det teke programmet for lang tid å tilpasse modellen for oss, og vi får ikkje noko resultat. Det kan vere eit teikn på at modellen (funksjonstypen) vi har valt passer dårleg med datasettet vårt.

Vi ser på nokre andre døme med eit datasett for kumulativt tal Covid 19-smitta i Norge for veke 8 til veke 32 i 2020.

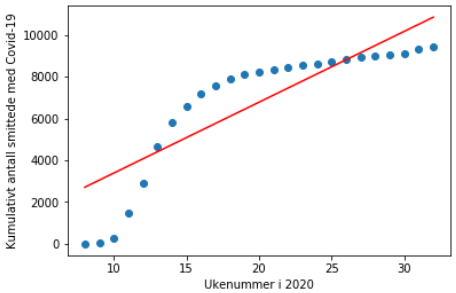
Vi nyttar ein potensmodell i programmet vårt

```
return a*(x**b)
```

Programmet gir oss potensfunksjonen

$$f(x) = 0{,}800 \cdot x^{1{,}00}$$

som den beste tilpassinga til datasettet vårt.



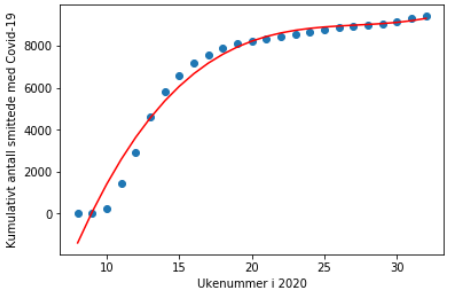
Vi nyttar ein tredjegradsmodell i programmet vårt

```
return (a * x**3) + (b * x**2) + (c * x) + d
```

Programmet gir oss tredjegradsfunksjonen

$$f(x) = 1{,}37x^3 - 112x^2 + 3080x - 19578$$

som den beste tilpassinga til datasettet vårt.



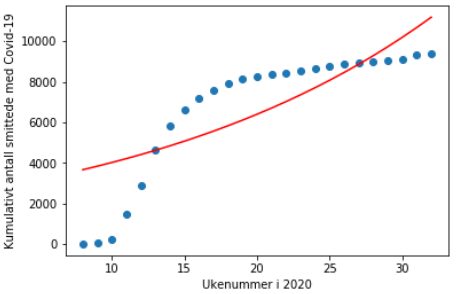
Vi nyttar ein eksponentialmodell i programmet vårt

```
return a * (b**x)
```

Programmet gir oss eksponentialfunksjonen

$$f(x) = 2521 \cdot 1{,}048^x$$

som den beste tilpassinga til datasettet vårt.



Nå veit vi korleis vi kan bruke data for å tilpasse mange ulike funksjonstypar. Men korleis skal vi vite kva for ein modell som passar best med datasettet vårt? Må vi berre sjå kva for ein funksjonstype som ser ut til å passe best med datasettet vårt, eller kan vi gjere det på ein meir objektiv måte? Bla om til neste side for å finne ut meir.

Korrelasjon i Python

Når vi skal rekne ut korrelasjonen, er det som regel Pearson sin korrelasjonskoeffisient vi meiner, og den blir kalla R. Det er og denne vi skal bruke her. Vi er mest interesserte i denne korrelasjonskoeffisienten opphøgd i andre, altså R^2, fordi den verdien seier kor stor andel av variasjonen i målingane/datasettet vårt som heng direkte saman. Verdien kan vere mellom null og ein. Jo høgare verdi, desto større del av variasjonen i y kan forklarast av variasjonen i x. R^{2b} blir oppgitt som eit desimaltal, men multipliserer vi det med 100 %, får vi svaret i prosent.

Den enklaste forma for korrelasjon er mellom x og y-verdiane i eit datasett. Da finn vi kor stor del av variasjonen i y som kan forklarast av variasjonen i x. Dette blir gjort med at Python gjer ein lineær regresjon, slik at R^2 viser kor god den lineære samanhengen mellom x og y-verdiane er. Da er R^2 eit mål på kor godt den lineære funksjonen passar med datasettet vårt.

Det går og an å gjere tilsvarande for ikkje-lineære regresjonar. Da får vi et objektivt mål på kva for ein modell som passar best med datasettet vårt. Dersom ein vil finne R^2 for ein regresjon der f(x) er ein eksponentialfunksjon, tek vi til med å utføre sjølve regresjonen. Når han er ferdig, veit vi kva koeffisientane i eksponentialfunksjonen er, altså veit vi kva f(x) er. Da bruker vi den estimerte funksjonen til å putte inn x-verdiene frå datasettet vårt, for så å rekne ut dei tilsvarande f(x) verdiane. For å finne R^2 bruker vi y-verdiane frå datasettet vårt og f(x)-verdiane som vi rekna ut ved å bruke den eksponentielle modellen vi fann. Da gjeld framleis det at den modellen som gir ein verdi nærmast R^2 = 1,00 er den beste modellen.

Korrelasjon i Python er samansett av fem delar:

1. Importere biblioteka som trengst
2. Importere data frå ei .txt-fil
3. Legge dataa i ein tabell (array) for x-verdiane og ein for y-verdiane
4. Rekne ut korrelasjonskoeffisienten
5. Vise korrelasjonskoeffisienten på skjermen

Datasetta vi bruker, kan vi til dømes hente frå statistisk sentralbyrå. Korleis ein gjer dette står i opplegg 26. Eller ein kan nytte ein micro:bit for å samle inn data. Dersom vi har mange målingar, er det fint å lagre det som ei fil på micro:biten. Korleis dette kan bli gjort står i kapittel 12.

```
from pylab import *
from scipy.stats import pearsonr

data = loadtxt("filnavn.txt")

x = data[:,0]
y = data[:,1]

R = pearsonr(x, y)[0]

print(R)
print(R**2)
```

Importerer dei naudsynte biblioteka.

Legg inn data frå fil i ein tabell.

Legg x-verdiane frå første kolonne inn i ein tabell og y-verdiaene frå den andre kolonnen i ein tabell.

Reknar ut korrelasjonskoeffisienten.

Skriv korrelasjonskoeffisienten og R^2 til skjermen. R^2 svarer til R^2-verdien i Geogebra.

Oppgåver

1. Finn eit valfritt datasett frå statistisk sentralbyrå, og bruk Python til å finne korrelasjonen for datasettet. Kva må du endre i dømet på denne sida?
2. Korleis kan ein rekne ut korrelasjonen mellom to ulike datasett? Kva for nokre kriterium må vere oppfylte for at det skal kunne gå an?

Val av regresjonsmodell i Python

Vi kan ikkje *berre* sjå på R^2 når vi skal velge modell (velge mellom dei ulike funksjonstypane). Vi bør nemlig ha ein idé ut frå teori, om kva type modell som burde fungere best. Det er til dømes slik at ein polynomfunksjon vil passe betre og betre, desto høgare grad han er. Det vil sei at ein fjerdegradsfunksjon alltid vil passe like godt eller betre enn en tredjegradsfunksjon. Eller at ein andregradsfunksjon alltid vil passe like godt eller betre enn ein førstegradsfunksjon, som er det same som ein lineær funksjon.

Regresjon med korrelasjon i Python er samansett av sju delar:

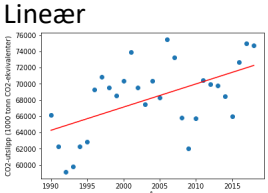
1. Importere biblioteka som trengst
2. Importere data frå ei .txt-fil
3. Legge dataa i ein tabell for x-verdiane og ein for y-verdiane
4. Lage ein funksjon som seier kva for ein modell som skal brukast i regresjonen
5. Gjere regresjonen, og vise på skjermen kva den matematiske modellen blir
6. Teikne grafen med resultatet
7. Rekne ut R og R^2 og skrive dei ut til skjermen

```
from pylab import *
from scipy.optimize import curve_fit
from scipy.stats import pearsonr
```

```
R = pearsonr(y_verdier,funksjon(x_verdier,a,b,c))[0]
print(R)
print(R**2)
```

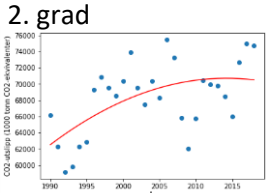
Nå kan vi sjekke kva for ein matematisk modell som skildrar datasetta våre best. Vi sjekker kva for ein modell som passar best for både $C0_2$-utsleppa og kumulativt tal på covid-19 smitta. Sjekk om du får same verdiar for dei ulike modellane.

Lineær



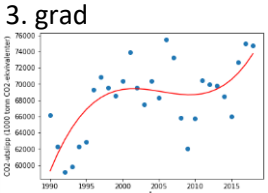
R = 0,5428
R^2 = 0,2947

2. grad



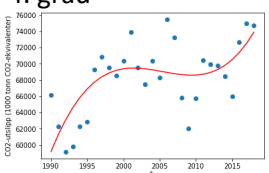
R = 0,5780
R^2 = 0,3340

3. grad



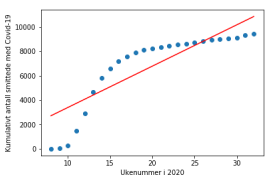
R = 0,6695
R^2 = 0,4483

4. grad



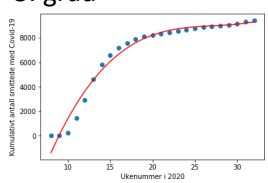
R = 0,6710
R^2 = 0,4502

Potens



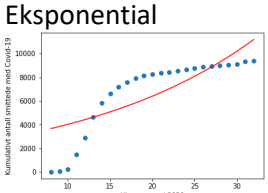
R = 0,8730
R^2 = 0,7620

3. grad



R = 0,9869
R^2 = 0,9740

Eksponential



R = 0,8004
R^2 = 0,6407

Diskusjonsoppgåver

1. Kva for ein modell skildrar CO_2-utsleppa best? Grunngi svaret.
2. Kva for ein modell skildrar kumulativt tal på covid-19 smitta best? Grunngi svaret.
3. Kan det potensielt finnast andre modellar som skildrar datasetta våre betre?
4. Dersom du har laga eit program som alt utfører ein regresjon, kva må du legge inn for at det og skal finne og skrive ut R og R^2?