# INF161 – Data science prosjekt

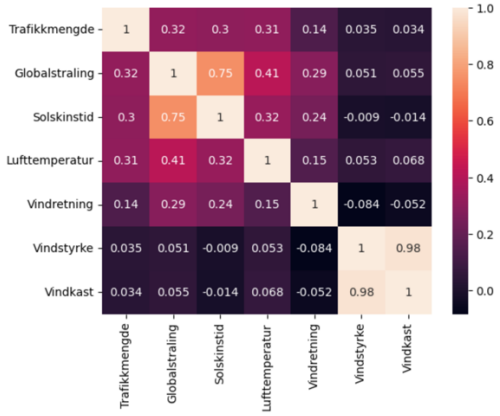
Brage Hamre Skjørestad

## Data utforskning

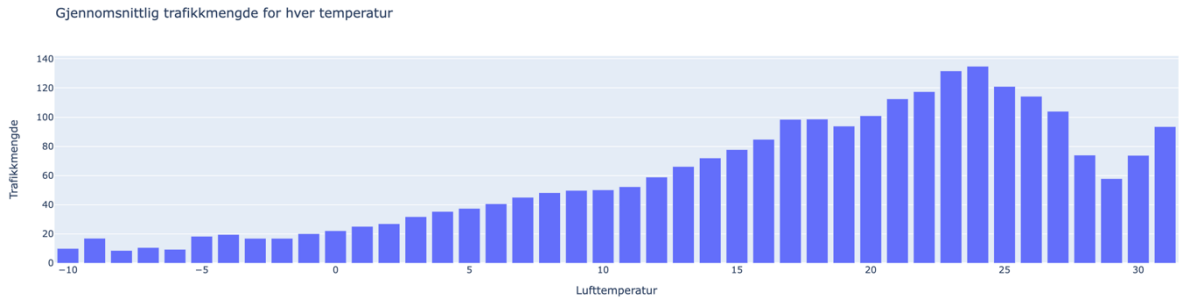
I denne oppgaven fikk vi inn 2 typer datasett, værdata og trafikkdata. Trafikkdataen kom uoversiktlig fra statens vegvesen, men vi ville ha ut total trafikkmengde for hver time. Denne dataen begynner i 2015. Værdataen kom i et sett for hvert år, men den begynte i 2010.

Det som var vanskelig i begynnelsen for meg var å slå sammen de 2 datasettene, værdata og trafikkdata. Jeg tok ikke i bruk datetime som indeks, og hadde da problemer med å få de rette værdataene på de korresponderende trafikkmengde tidspunktene. Når jeg forsto problemet var det enkelt å fikse, og jeg fikk da et komplett datasett. Jeg valgte med en gang å fjerne relativ luftfuktighet og lufttrykk, kunne aldri tenke meg at de hadde noe med sykkeltrafikk å gjøre. Dobbeltsjekket også med korrelasjonsmatrise. Det var også flere kolonner jeg kunne fjernet, for eksempel vindkast og vindstyrke som også scoret lavt i forhold til resten. Jeg tenkte derimot at vind er noe som kan påvirke antall sykler, ved storm-tilstander for eksempel. Ville i ihvertfall ta vind med videre, og ta beslutningen senere.

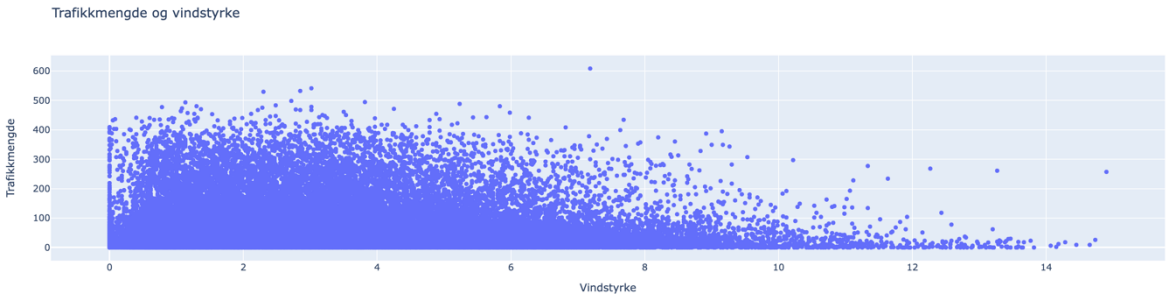
Den største utfordringen med dette datasettet er korrelasjon mellom X og y. I andre datasett kan det være litt mer åpenbart. Her må man tenke på hva som faktisk kan påvirke trafikk. For å finne ut av dette, ønsket jeg å analysere data med grafer. Grafer er alltid veldig oversiktlig, og maler et tydeligere bilde enn bare tall. Jeg startet med den nevnte korrelasjonsmatrisen. Her blir en matematisk korrelasjonsverdi beregnet mellom kolonnene, og i første kolonne er korrelasjonen med trafikkmengde.



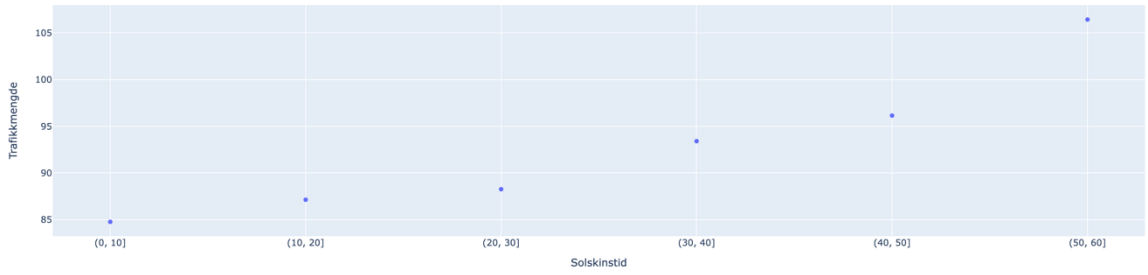
Lufttemperatur skal naturligvis med, men ville fortsatt se mønstrene.




Vi ser at trafikkmengden øker nesten helt sammen med temperaturen. Blir viktig å ha med for en god modell.

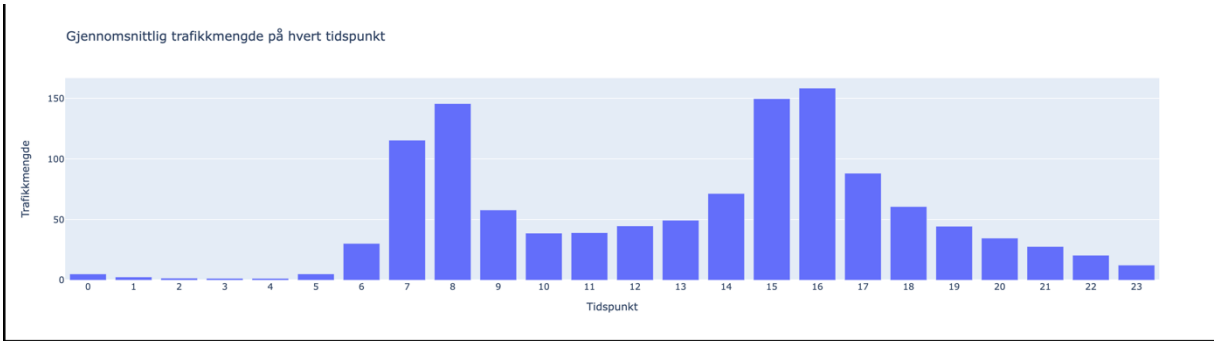



Ved høye vinder er det mer punkt med lav trafikkmengde. Den samme grafen er meget lik for vindkast.  På grunn av dette blir de ikke fjernet manuelt av meg.

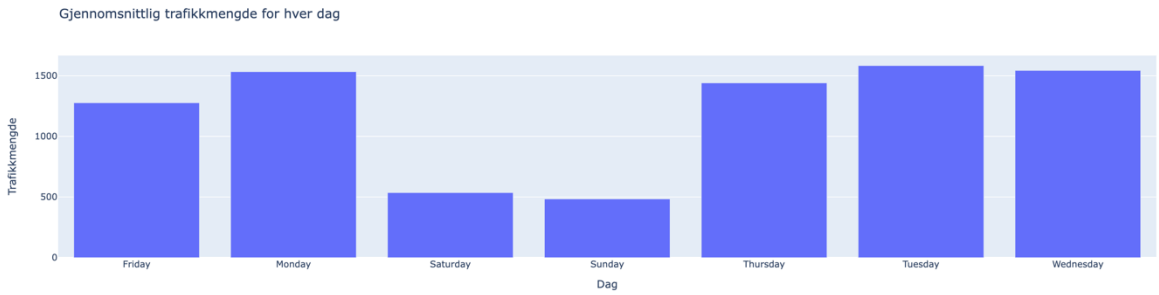



---

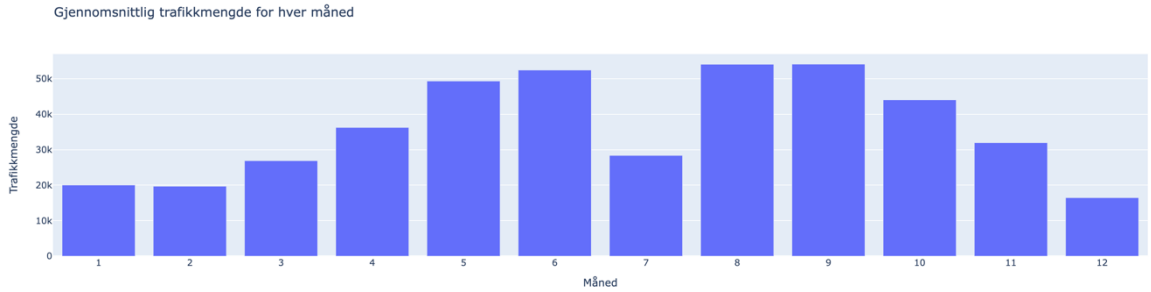
Jo mer sol det har vært den forrige timen, jo mer trafikkmengde. Solskinnstid hadde også mye korrelasjon i matrisen, men ønsket å sjekke videre. Denne variabelen og globalstråling er jo ganske like i hva de sier for noe. Globalstråling scoret enda høyere i matrisen, men jeg strevde med å få en lignende graf som ovenfor for den kolonnen. Ettersom den scoret høyest, tok jeg den naturligvis med uansett.



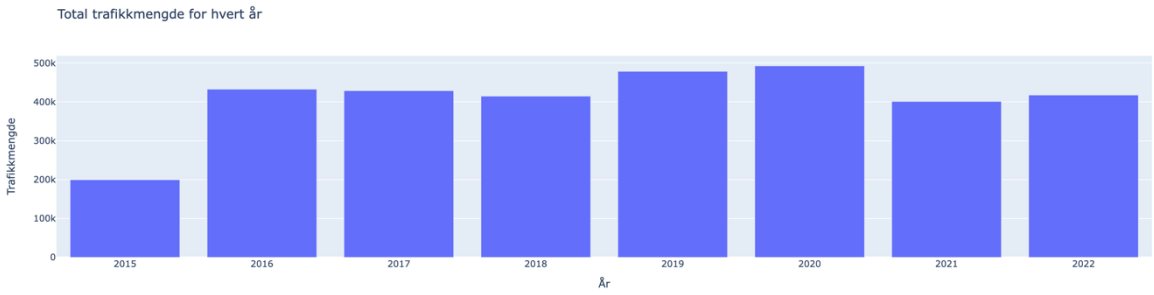

Så vil jeg gjerne få inn et par nye variabler. Tidspunkt blir en naturlig en, når man tenker på hvor mye biltrafikken varierer med tidspunkt. Vi ser her at det er ganske lik historie for syklene. Tidspunkt blir en ny variabel.




Mindre sykling i helgene, sykling til jobb er nok mesteparten av trafikkmengden. Denne variabelen kommer også med.




Samme historie, trafikk minsker i ferien og er lavere i de kalde månedene. Denne tar jeg også med.




---

Prøvde også å se på år, men var ganske jevnt. 2015 er lavere ettersom dataen startet midt i året. Tenkte kanskje vi sko se noe dypp i covid årene, men det var ikke slikt. Denne blir ikke tatt med i dataen.

## Implementasjon

5 programs:

- data_exploration.ipynb
- model_selection.py
- run_models.py
- app.py
- INF161project.py

- index.html

*data_exploration.ipynb:* Tar for seg det jeg allerede har snakket om. Den renser datasettene og setter sammen til et komplett datasett. Deretter lagrer den dataen for senere bruk, og plotter grafer. Grafene er plottet på det komplette datasettet, der NaN verdier blir fjernet i stedet for imputed. Det er også med realistiske verdier.

*model_selection.py:* Program med en klasse for å preprocesse data og velge hyperparametere. Disse 2 prosessene er i samme program fordi da kunne jeg bruke en pipeline. Jeg lagde en pipeline som først imputer, så scaler og så velger de beste variablene. Det ble totalt 12 ulike variabler i datasettet, og pipelinen velger 10. Dette tallet ble valgt ettersom vindstyrke og vindkast hadde veldig dårlig korrelasjonsverdi. Her er det en annen verdi det baseres på, mutual_info_regression. Tenkte at den bør fjerne de 2 kolonnene hvis korrelasjonsverdien var representativ. Da kan jeg legge inn realistiske verdier i værdataen, og programmet kan håndtere det. Deretter kjører jeg cross -validation

med de valgte hyperparameterne. De ble valgt etter å ha lest litt om de mest relevante parameterne for de valgte regressor-klassene.

*run_models.py:* Denne koden definerer en klasse som brukes til å kjøre mine valgte regressorer. Man skulle velge 3 fundamentalt forskjellige modeller, og jeg gikk for k-nearest neighbor (neighbor modell, Ridge (lineær modell) og Decision Tree (tree modell). Jeg tok også med Random Forest Regressor da jeg så Decision Tree gjorde det best, ettersom det er en mer kompleks utbygging av Decision Tree. Som sagt valgte jeg hyperparametere etter å ha lest om hva påvirkning de har. Denne klassen har også en metode for å velge den beste modellen (ut ifra cross-validation resultater) og kjøre den på test data, og for å lagre den beste modellen.

*INF161project.py:* Dette er koden som skal kjøres for å få ut resultatene. Programmet står for feature_engineering, som legger til de ekstra variablene jeg vil ha med. Etter å ha undersøkt mer ulike måter for variabelutvinning, la jeg til 3 ekstra kolonner. Planen var å lage 1 ekstra kolonne basert på hver av de 3 beste kolonnene (ifølge korrelasjonsmatrise). Jeg tok derfor med gjennomsnittlig solstråling de siste 3 timene, for eventuelle variasjoner med skyer. Soltid til nå er hvor mye solen har skinnet i løpet av de gjeldene dagen. Til slutt laget jeg en relasjonskolonne mellom temperatur og soltid, tenkte at høye verdier for begge disse 2 har mer trafikk. Så splitter programmet dataen i treningsdata (75 %) og testdata (25%). Splitter ikke i valideringsdata siden jeg kjører cross-validation. Deretter er det å kjøre alle modellene, kjøre den beste modellen på testdata, lagre modellen med pickle, og til slutt skrive prediksjoner av 2023 værdata til «predictions.csv».

*app.py:* Brukte min besvarelse i lab7 som template her. Dette programmet håndterer nettsiden i lag med index.html. Jeg definerer først feature_engineering funksjonen lik som i INF161project.py, slik at jeg slipper å skrive inn data man kan kalkulere seg fram til med annen data. Ut ifra lab7 la jeg til en try-except for å håndtere om ingen dato blir skrevet inn. Jeg måtte også bruke datetime sin metode «strptime» for å hente datetime objekt fra index.html inputen.

*Index.html:* Enkel html fil for å ta inn data på en lokal nettside. Det er en veldig enkel fil, brukte min lab7 som template her også. Var bare numeriske verdier som var fint. La til litt kode innenfor <style>, for å få det til å se litt finere ut.

## Resultater

Før jeg fikk se hva mine modeller skulle klare å yte på RMSE (root mean squared error), var jeg veldig usikker på hva slags verdier som kunne bli sett på som bra/dårlig. Men jeg så ikke for meg at å predikere sykkeltrafikk bare basert på tid og værdata kunne gi en feilfri modell. Jeg tenkte at under 10 RMSE ville være umulig. Jeg kan ikke si jeg hadde en spesifikk verdiprediksjon for det jeg ville få, RMSE kan være veldig mye når verdiene går fra 0 til 608. Men, de aller fleste verdier er under 100, så en RMSE som nærmer seg det tallet vil være dårlig.

```
Model: KNN      Score: 41.67

Model: DTR      Score: 30.89

Model: RDG      Score: 61.47

Model: RFR      Score: 28.2


Our best model: RFR
Hyperparameters: {'model__n_estimators': 250}


Running best model...
Our best model´s RMSE on test data: 24.92827831900305
```

(KNN: K-nearest Neighbor,
DTR: Decision Tree,
RDG: Ridge,
RFR: Random Forest)

Dette er mine resultater, score er mean RMSE i cross-validation prosessen. De er veldig varierte og overraskende. Forventet ikke en så stor forskjell i ytelse mellom modellene. Tenkte at den lineære modellen ville være dårligst, men ikke med så stor margin. Forventet også mer av KNN. Generelt trodde jeg at det ville være litt bedre resultater. Ikke veldig mye, men kanskje nærmere 20. Til og begynne med kjørte jeg uten RFR modellen, ettersom jeg først ville sjekke resultater fra enkle versjoner av 3 forskjellige regressorer.

Etter en liten skuffelse ønsket jeg bedre resultater. DTR var soleklart best, så derfor tok jeg med RFR i et forsøk på enda bedre resultater. Det funket. RFR var ikke mye bedre i valideringsprosessen, men kommer under 25 på test data, som gjorde meg fornøyd. Dummyregressoren fikk 67 i RMSE, 25 i endelig resultat er relativt bra. Likevel, jeg tror ikke at denne modellen kunne blitt brukt i det virkelige liv. Det måtte isåfall vært i kontekster der en viss mengde feil går fint. Kanskje hvis man skulle regulert hvor ofte sykler får grønt lys i forhold til trafikkmengden. Da er det ikke verdens ende om predikert verdi og faktisk verdi er litt ulike.

Vår valgte modell gjorde det dårlig på helligdager. Nettsiden predikerte 71 syklende julaften 2018 klokken 17:00, når det egentlig var 0 syklende. Med mer tid ville jeg prøvd å hente generelt mer ulik data. Gå vekk fra vær og lete etter andre faktorer som påvirker sykkeltrafikk, der jeg tror å legge inn en binær helligdag (ja/nei) kolonne ville vært veldig smart. Når det er såpass stor feil på en helligdag vil det påvirke generell RMSE, spesielt med tanke på at vi har en del helligdager i Norge.