Åse Grethe Fagerbakke



INF161 Prosjekt

Predikasjon av sykehusopphold.

03/11/2024

Innholdsfortegnelse

[Mål med prosjektet 2](#_Toc181519434)

[DATARENSING OG FORBEHANDLING 2](#_Toc181519435)

[Innledning 2](#_Toc181519436)

[1.1.1 Innlasting og sammenslåing av data 2](#_Toc181519437)

[1.1.2 Sammenslåing og oppdeling til en samlet DataFrame. 2](#_Toc181519438)

[2.1.1 Optimalisering 3](#_Toc181519439)

[2.1.2 Fjerner støy 3](#_Toc181519440)

[2.1.2 Negative verdier for alder 4](#_Toc181519441)

[2.1.3 Kategoriske verdier til numeriske 4](#_Toc181519442)

[3.1.1 Visualisering 7](#_Toc181519443)

[Innledning 7](#_Toc181519444)

[3.1.2 Fordeling i datasettet, visualisering og statistikk. 7](#_Toc181519445)

[3.1.7 Oppholdslengde for pasienter med gjennomsnittlig fysiologisk score 11](#_Toc181519446)

[3.1.8 Korrelasjon mellom variabler. 12](#_Toc181519447)

[3.1.9 Oppsummering 13](#_Toc181519448)

[MODELLERING 14](#_Toc181519449)

[Innledning 14](#_Toc181519450)

[4.1.1 Innfylling av NaN-verdier, NMAR. 14](#_Toc181519451)

[4.2.1 Fyller inn NaN-verdier,MAR. 17](#_Toc181519452)

[4.3.1 Identifisering av den beste modellen på hyperparameter-tuning 18](#_Toc181519453)

[4.3.4 Grunnlinjemodell log-transformasjon 20](#_Toc181519454)

[4.3.5 Hyper parameter tuning 21](#_Toc181519455)

[4.4.2 Overfitting i modeller 24](#_Toc181519456)

[5.0.1 Variabelutvinning 25](#_Toc181519457)

[Innledning 25](#_Toc181519458)

[5.0.2 Analyse av korte og lange opphold. 25](#_Toc181519459)

[5.0.3 Beholde sykdomskategori? 26](#_Toc181519460)

[5.0.3 Videre variabelutvinning 27](#_Toc181519461)

[5.1.6 Prøver å kombinere forskjellige variabelutvinninger 31](#_Toc181519462)

[5.1.7 Resultat etter variabelutvalg 32](#_Toc181519463)

[6.0.0 Valg av den endelige modell, og prediksjon på testsettet. 36](#_Toc181519464)

[6.0.1 Andre metoder jeg har prøvd, men ikke brukt 37](#_Toc181519465)

[6.0.2 Oppsummering, drøfting av resultat. 37](#_Toc181519466)

[7.0.0 Nettside 38](#_Toc181519467)

[8.0.0 Refleksjon 40](#_Toc181519468)

[Avslutning 40](#_Toc181519469)

[Bibliografi 42](#_Toc181519470)

# Mål med prosjektet

Målet med dette prosjektet er å predikere hvor lenge pasienter blir værende på sykehuset. Det generelle målet er å gjennomføre et data-science-prosjekt, ved å anvende de forskjellige metodene vi har lært i faget. Vi starter med ubehandlet pasient-data og skal ende opp med en maskinlæringsmodell som kan ta inn behandlet data å predikere hvor lenge en pasient vil bli værende på sykehuset. Jeg skal ta i bruk dataforbredelse, dataanalyse, håndtering av manglende data, variabelutvinning og modellutvinning.

# DATARENSING OG FORBEHANDLING

## Innledning

Det er viktig med god datarensing og forbehandling ved predikasjon fra et datasett. Dette må gjøres for å sikre at dataen som brukes i analysen er pålitelige, og klare for modellering. Manglende eller feil data kan svekke modellen vår og forverre det endelige resultatet. I denne delen beskriver jeg hvordan jeg konverterte kategoriske variabler til numeriske verdier, mapper enkelte verdier, og hvordan/hvorfor jeg fjerner støy og andre faktorer som jeg anså som kritiske i datasettet.

## 1.1.1 Innlasting og sammenslåing av data

For å starte analysen må jeg først laste inn datasettene. Jeg starter med å lese inn data fra 'hospital' og 'demographic' som begge er i csv-format. ‘Physiological’-dataen er lagret i en tekstfil, og jeg bruker ‘table’ for å lese den inn. ‘Severity’ er en json-fil for sykdomsalvorligheten. Json-fil er en filtype som kan inneholde komplekse datastrukturer, slik som lister, objekter, strenger og tall.

Jeg velger å eksludere den siste kolonnen ‘DNR-status’, som representerer en ‘ikke gjenopplivsordre ved innlesing av Json-filen. Selv om denne informasjonen er tilgjengelig på dag 1, er jeg ikke interessert i den. Jeg vet ikke når denne eventuelt blir iverksatt senere, og dermed ikke hensiktsmessig for min modell som da skal predikere oppholdslengde fra dag 1.

For å lese inn Json-filen uten 'DNR-status', bruker jeg indeksering til å inkludere alle kolonner fra starten (indeks 0) til den nest siste kolonnen (indeks -1). Ettersom noen av kolonnene i JSON-filen inneholder lister, må jeg "explode" disse, noe som betyr at jeg konverterer hver liste til flere rader, slik at informasjonen blir lettere å analysere. Jeg vil til slutt ha en ryddig indeks som starter på 0, og derfor må jeg bruke ‘reset\_index’.

Disse stegene er viktige for å omforme rå-dataen til en dataframe som kan bli brukt av maskinlæringsmodellene.

## 1.1.2 Sammenslåing og oppdeling til en samlet DataFrame.

Jeg velger å slå sammen datasettene tidlig i prosessen, før jeg har analysert datasettene individuelt. På denne måten kan jeg dele opp tabellen til trening-, validering- og test-sett slik jeg unngår data leakage ved optimaliseringen. Jeg vil kun fokusere på treningssettet, og vil sikre at jeg ikke ubevist drar nytte av informasjon som skal være skjult i test-settet. Sammenslåingen gjør jeg ved å bruke en felles ‘nøkkel’, altså pasient-id. Sikrer at dataen fra de ulike settene kobles riktig.

**1.1.3 Fjerner duplikater og negativ oppholdslengde.**

Etter å ha slått sammen de fire datasettene, kan det oppstå duplikater. For å unngå at modellen får samme informasjon flere ganger, noe som kan gi skjevhet og støy, fjerner jeg duplikater basert på pasient-id. Pasient-id fungerer kun som en identifikator for pasientene og har ingen verdi for selve prediksjonen, så jeg setter denne som indeks i datasettet.

Det er nødvendig å fjerne negative verdier for oppholdslengden, da slike verdier kan påvirke modellen negativt. Derfor filtrerer jeg ut rader med negativ oppholdslengde for å sikre en mer nøyaktig prediksjon.

**1.1.4 Inndeling**

Datasettet er nå klargjort for videre behandling. For å sikre en strukturert modellering, deler jeg datasettet inn i trenings-, validerings- og testsett.

Datasettet deles opp i X, som består av de uavhengige variablene, og y, som representerer målvariabelen (oppholdslengden). Treningssettet utgjør 70 % av dataene, mens validerings- og testsettet utgjør 15 % hver.

Valideringssettet brukes til å evaluere modellens ytelse underveis i treningen, mens testsettet holdes helt skjult og brukes kun til å evaluere den endelige modellens ytelse.

## 2.1.1 Optimalisering

Under optimaliseringen fjerner jeg støy og data som ikke er tilgjengelige på dag 1. Dette sikrer at kun relevante variabler blir med i analysen, slik at modellen baserer prediksjonene på riktig informasjon fra starten av. Målet er å klargjøre dataene for visualisering og modellering, som gir et godt grunnlag for presise prediksjoner.

Før jeg starter optimaliseringen, identifiserer jeg hvilke variabler i treningssettet som har NaN-verdier og hvor mange. Dette gjør jeg for å følge med på prosessen og sikre at ingen manglende verdier fylles inn feil. Slik unngår jeg at modellen trenes på ufullstendig eller misvisende data.

### 2.1.2 Fjerner støy

Jeg fjerner jeg kolonnene "sykehusdød" og "dødsfall," siden dette er opplysninger som ikke er kjent på dag 1 og dermed ikke relevante for prediksjonen. Ved å fokusere på dataene som er tilgjengelige tidlig, sikrer jeg at modellen baseres på relevante og rettferdige variabler.

Variablene adl-pasient og bilirubin er først kjent på dag 7, så jeg fjerner disse kolonnene for å holde modellen konsistent. Jeg fjerner også kolonnen dnr-status, da den har et stort antall NaN-verdier (4657 av 5013 rader mangler). For å unngå støy og forbedre modellens nøyaktighet, ekskluderer jeg denne variabelen, noe som reduserer risikoen for at ufullstendig data påvirker resultatene negativt.

Manglende verdier for dnr-status:

Et bilde som inneholder tekst, skjermbilde, Font, nummer

Automatisk generert beskrivelse

### 2.1.2 Negative verdier for alder

Jeg velger å sette eventuelle negative verdier for alder til NaN, da slike verdier ikke gir mening. Ved å erstatte de negative verdiene med NaN, kan jeg senere håndtere dem på en mer hensiktsmessig måte.

### 2.1.3 Kategoriske verdier til numeriske

**2.1.4 Kjønn/etnisitet**

**Etnisitet**

For variabelen etnisitet har jeg identifisert 28 manglende verdier. Histogrammet viser at 85 % av verdiene tilhører kategorien "white". Dette støtter at de manglende verdiene kan fylles inn med den mest vanlige kategorien, "white". Siden denne kategorien er i flertall, vil dette være en rimelig og nøyaktig tilnærming som bevarer datasettets generelle fordeling.

Øverst i skjemaet

Nederst i skjemaet

Et bilde som inneholder diagram, Plottdiagram, line, tekst

Automatisk generert beskrivelse

Mange modeller kan ikke jobbe direkte med kategorisk data, så jeg mapper inntekt til numeriske verdier. Dette gjør at modellene kan tolke og analysere informasjonen mer effektivt og korrekt.

|  |  |
| --- | --- |
| ‘Etnisitet’ | Numerisk verdi |
| ‘White’ | 0 |
| ‘Black’ | 1 |
| ‘Asian’ | 2 |
| ‘Hispanic’ | 3 |
| ‘Other’ | 4 |

**Kjønn**

For variabelen kjønn bruker jeg get\_dummies i stedet for OneHotEncoder.En fordel med get\_dummies er at den kan håndtere NaN-verdier og beholde disse i datasettet, mens OneHotEncoder ikke gjør dette. Get\_dummies konverterer de kategoriske verdiene male og female til binære verdier på en enkel og effektiv måte. Mens OneHotEncoder ville delt kjønn inn i to separate kolonner, lar get\_dummies meg holde kjønn i én enkelt kolonne, der 1 representerer male og 0 representerer female. Dette gir en ryddigere og mer kompakt databehandling.

**2.1.5 Sykdom underkategori og kreft**

Jeg ser at jeg har både sykdomskategori og sykdom-underkategori. Sykdomskategori gir en mer generell klassifisering av sykdommen, hvor flere sykdommer grupperes under én bred kategori. Derimot gir sykdom-underkategori en mer spesifikk inndeling for pasienten.

Jeg sammenlignet de to kolonnene for å identifisere eventuelle forskjeller, og fant at det var variasjoner mellom dem. Jeg velger derfor å beholde kun underkategoriene, ettersom de gir en mer presis beskrivelse av sykdommene. Samtidig beholder jeg sykdomskategori som en tilleggsvariabel, slik at jeg senere kan aktivere den og sammenligne resultatene ved behov.

Hvis jeg skal bruke sykdomskategori i modelleringen min senere, må jeg gjøre den om til numeriske verdier, jeg velger da å mappe, slik jeg får den i en kolonne.

Sykdomskategori mapping:

|  |  |
| --- | --- |
| Sykdomskategori | Numerisk verdi |
| ‘ARF/MOSF’ | 4 |
| ‘COPD/CHF/Cirrhosis’ | 2 |
| ‘Coma’ | 3 |
| ‘Cancer’ | 1 |

(Denne tilnærmingen til mappingen kommer jeg tilbake til.)

Konvertering til numeriske verdier for kreft og sykdom-underkategori:

Både kreft og sykdom-underkategori inneholder ingen NaN-verdier, noe som gjør at jeg kan bruke OneHotEncoder direkte på disse variablene. OneHotEncoder vil konvertere hver underkategori til en egen binær kolonne, der verdien er 1 hvis pasienten har sykdommen, og 0 hvis ikke. Dette gjør sykdomsinformasjonen lettere for modellen å tolke numerisk, noe som kan forbedre prediksjonene.

For kreft ønsker jeg en samlet kolonne der 1 indikerer at pasienten har kreft, og 0 at de ikke har det. Dersom en pasient har kreft med spredning (kreft-metastatic), vil kreft-kolonnen settes til 1 for å sikre at all relevant kreftinformasjon blir representert i modellen. Dette betyr at hvis en pasient har kreft med spredning, vil kreft-kolonnen også vise 1.

Sykdomskategori-id fungerer kun som en identifikator for ulike sykdomskategorier og gir ikke detaljert eller nyttig informasjon om pasientens helsetilstand. Derfor fjerner jeg denne kolonnen for å redusere støy og sikre at modellen fokuserer på variabler som faktisk er informative.

**2.1.6 Inntekt og utdanning**

For variablene "inntekt" og "utdanning" velger jeg å mappe dem direkte til numeriske verdier, siden begge har fire kategoriske nivåer som naturlig kan rangeres i stigende rekkefølge. I stedet for å bruke OneHotEncoder eller get\_dummies, som ville skapt flere kolonner, bevarer jeg den naturlige ordningen av verdiene. Dette reduserer kompleksiteten i datasettet ved å holde variablene i en enkelt kolonne, samtidig som jeg beholder relasjonen mellom de forskjellige nivåene.

Ved å tildele numeriske verdier i stigende rekkefølge reflekteres den faktiske forskjellen mellom kategoriene. Dette kan bidra til at modellen bedre fanger opp mønstre i dataene og gir mer presise prediksjoner.

Utdanning Inntekt

|  |  |
| --- | --- |
| Utdanning | Numerisk verdi |
| Grunnskole/ungdomsskole | 0 |
| Videregående | 1 |
| Høyere utdanning | 2 |
| Videre utdanning | 3 |

|  |  |
| --- | --- |
| Inntekt | Numerisk verdi |
| under $11k | 0 |
| $11-$25k | 1 |
| $25-$50k | 2 |
| >$50k | 3 |

**Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse2.1.7 Konvertere resterende variabler.**

Jeg bruker ‘train\_df.info()’ for å sjekke verdiene i datasettet og forsikre meg om at alle kategoriske variabler har blitt korrekt konvertert til numeriske verdier. Jeg oppdaget da at noen av variablene, selv om de inneholder numeriske verdier, fortsatt er lagret som object-datatype (strenger), som vist i bildet. Dette kan føre til problemer under modelltreningen, da modeller forventer numeriske typer som int eller float.

For å løse dette, konverterer jeg disse kolonnene til numeriske verdier ved å bruke ‘pd.to\_numeric()’. På denne måten sikrer jeg at dataene blir behandlet riktig i den videre analyse- og modelltreningen.

## 3.1.1 Visualisering

### Innledning

Visualisering av data er en viktig del av dataforberedelsen fordi den gir innsikt i mønstre, sammenhenger og skjevheter i datasettet. Dette gjør det mulig å justere variabelutvalg og modelltrening for bedre ytelse og nøyaktighet. En dypere forståelse av dataene øker også påliteligheten i prediksjonene.

### 3.1.2 Fordeling i datasettet, visualisering og statistikk.

**Fordeling av oppholdslengde**

Jeg skal predikere oppholdslengde, så jeg starter med å analysere hvordan denne variabelen er fordelt. Gjennomsnittet er 17.88 dager, men med en maksimal verdi på 281 dager. Dette indikerer en høyreforskjøvet fordeling, der et fåtall pasienter har svært lange opphold, såkalte outliers (ekstreme verdier). Dette bekreftes i visualiseringen av oppholdslengden, som tydelig viser at flertallet av pasientene har korte opphold, mens noen få har uvanlig lange opphold. Det er viktig å være oppmerksom på disse verdiene for å forbedre modellens ytelse.

Et bilde som inneholder tekst, skjermbilde, Font, programvare

Automatisk generert beskrivelse

Et bilde som inneholder tekst, skjermbilde, line, Plottdiagram

Automatisk generert beskrivelse

Jeg visualiserer dette i et boxplot for å illustrere den skjeve fordelingen tydeligere. Halen slutter på 41 dager, noe som indikerer at flertallet av pasientene har oppholdslengde som er mindre enn eller lik 41 dager. Da er verdier trukket ut over dette bli ansett som outliers.

Et bilde som inneholder skjermbilde, tekst, line, diagram

Automatisk generert beskrivelse

**3.1.3 Fordeling av oppholdslengde for hver sykdom**

Statistisk beskrivelse av de ulike sykdommene, de som har høyere mean vil indikere at flere pasienter har denne sykdommen.

**Et bilde som inneholder tekst, kvittering, skjermbilde, hvit

Automatisk generert beskrivelse**

Grafen under viser en viktig sammenheng mellom oppholdslengde og de forskjellige sykdommene (underkategoriene). Ved å undersøke fordelingen av oppholdslengden for hver sykdom, ser vi at visse sykdommer, spesielt sykdom\_ARF/MOSF med Sepsis, har en større andel pasienter med lange sykehusopphold, flere outliers. Dette betyr at disse pasientene ofte krever lengre opphold enn f.eks. demens eller diabetes, som viser betydelig kortere opphold.

Denne informasjonen er gir innsikt i hvilke sykdommer som har størst innvirkning på sykehusoppholdets varighet, og det kan gi en indikasjon på hvordan prediksjonsmodellen kan prestere. Kan fort bli trekket mot de ekstreme verdiene.

Et bilde som inneholder tekst, skjermbilde, diagram, Plottdiagram

Automatisk generert beskrivelse

**3.1.4 Fordeling av oppholdslengde for sykdomskategori**

Her har jeg undersøkt fordelingen av oppholdslengden for de fire overkategorier:

1. COPD/CHF/Cirrhosis
2. Cancer
3. Coma
4. ARF/MOSF

Jeg ser at sykdomskategorien ARF/MOSF følger samme mønster som underkategorien sykdom\_ARF/MOSF w/Sepsis, der ekstreme verdier (outliers) trekker gjennomsnittet opp. Dette illustrerer gjerne at noen sykdomskategorier er lik noen underkategorier, samlet sett på opphold.

Jeg har problemer med å plassere kreft og kreft-metastatic i noen av de eksisterende overkategoriene, da den sammenslåtte kategorien ikke viser like lange opphold som det disse to underkategoriene har. Dette kan indikere at de er kategorisert feil eller under andre grupper, noe som skaper en utfordring for modellens presisjon.

Derfor, hvis jeg bruker overkategorier, er det verdt å merke seg at ikke alle underkategorier kan fjernes. Selv om overkategoriene kan gi en mer generell forståelse, er det fortsatt viktig å beholde spesifikke underkategorier, spesielt de som har en betydelig innvirkning på resultatene, som kreft og kreft-metastatic. Hvis disse fjernes eller ignoreres, risikerer vi å miste viktig informasjon som påvirker modellens predikasjon. Det er viktig å finne en balanse, slik at alle sykdommene blir tatt i betraktning.

I tillegg ser jeg for COPD/CHF/Cirrhosis at COPD og CHF har svært lik oppholdstid, men Cirrhosis har gjerne kortere oppholdstid. Er det da hensiktsmessig å inkludere underkategorien for CHF? Dette er problemstillinger jeg vil sjekke ut under modellering og variabelutvalg.

Jeg vil videre vurdere om alle underkategorier bør inkluderes i modellene, eller om en kombinasjon av over- og underkategorier vil være mer hensiktsmessig. For sykdommer med svært lik oppholdslengde kan det være nyttig å slå dem sammen. Dette vil forenkle modellens prediksjonsarbeid og samtidig bevare essensiell informasjon.

Et bilde som inneholder tekst, skjermbilde, diagram, Plottdiagram

Automatisk generert beskrivelse

**3.1.5 Oppholdslengde i forhold til fysiologisk score**

Ved å visualisere sammenhengen mellom fysiologisk score og oppholdslengde, observerer vi at pasienter med høyere fysiologisk score generelt har kortere sykehusopphold. Imidlertid ser vi en interessant klynge av pasienter med en fysiologisk score mellom 20 og 40 som har både korte og lange opphold. Dette antyder at pasienter innenfor dette intervallet kan oppleve varierende behandlingsforløp, som igjen påvirker oppholdslengden deres i større grad enn hos de med svært høy eller lav score.

Et bilde som inneholder tekst, Plottdiagram, skjermbilde, line

Automatisk generert beskrivelse

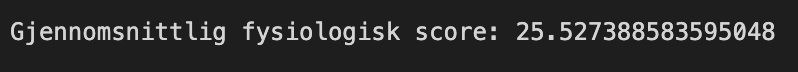
**3.1.6 Fordeling av fysiologisk score blant sykdomskategoriene.**

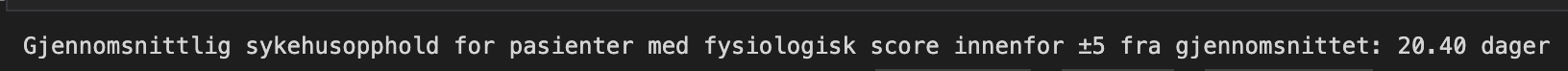
Et bilde som inneholder tekst, skjermbilde, diagram, Plottdiagram

Automatisk generert beskrivelseFor å undersøke om enkelte sykdommer har høyere fysiologisk score, har jeg visualisert fordelingen av fysiologisk score på tvers av ulike sykdommer. Alle sykdomskategoriene viser en spredning i score, og alle inneholder uteliggere (outliers). Dette tyder på at det ikke er en bestemt sykdom som systematisk gir høyere eller lavere score. Dette mangfoldet i score på tvers av sykdommer kan indikere at fysiologisk score ikke alene kan forklare lengden på sykehusoppholdet for disse sykdommene

### 3.1.7 Oppholdslengde for pasienter med gjennomsnittlig fysiologisk score

Etter å ha visualisert sammenhengen mellom fysiologisk score og oppholdslengde, beregnet jeg gjennomsnittlig fysiologisk score for hele datasettet, som ligger rundt 25. Jeg observerer at det er flere pasienter med ekstremt høye verdier som drar opp dette gjennomsnittet. Deretter filtrerte jeg ut pasienter som har en fysiologisk score innenfor ±5 poeng fra gjennomsnittet.





For denne gruppen ser jeg at gjennomsnittlig oppholdslengde på sykehuset er omtrent 20 dager. Imidlertid er det stor variasjon innenfor denne gruppen—noen har svært korte opphold, mens andre har svært lange. Denne variasjonen, inkludert noen få ekstreme verdier (outliers), indikerer at fysiologisk score alene ikke fullt ut forklarer oppholdslengden.

Selv om fysiologisk score kan gi en indikasjon på hvor syk en pasient er, viser denne analysen at selv blant pasienter med en gjennomsnittlig fysiologisk score, varierer oppholdslengden betydelig. Dette kan utfordre modellen, ettersom pasienter med gjennomsnittlig sykdomsgrad kan ha alt fra korte til lange opphold. Det tyder på at andre faktorer i tillegg til fysiologisk score spiller inn.

Et bilde som inneholder tekst, skjermbilde, line, Rektangel

Automatisk generert beskrivelse

### 3.1.8 Korrelasjon mellom variabler.

Jeg har laget en korrelasjonsmatrise mellom variablene og den predikerte variabelen vår, oppholdslengde. Denne matrise viser hvor sterk lineær sammenheng hver uavhengig variabel har med oppholdslengde, ved å bruke ‘Pearson-korrelasjonskoeffisienten’. Pearson-korrelasjonen måler kun lineære sammenhenger mellom kontinuerlige variabler. En høy positiv korrelasjon (nær 1) indikerer at når denne variabelen øker, øker også oppholdslengden. En negativ korrelasjon (nær -1) indikerer at når variabelen øker, synker oppholdslengden.

Selv om denne metoden gir en indikasjon på sammenhengen mellom kontinuerlige variabler og oppholdslengde, har den begrensninger i dette tilfellet. Datasettet inneholder flere kategoriske variabler, som ulike sykdommer, som ikke egner seg for analyse med Pearson-korrelasjon. Pearson-korrelasjon krever at begge variablene er kontinuerlige og har jevn variasjon. En kategorisk variabel som "sykdom\_ARF/MOSF w/Sepsis" er derimot binær (0 eller 1) og bryter derfor med forutsetningene for denne analysen.

Dette betyr at korrelasjonsmatrisen gir begrenset informasjon for de kategoriske variablene i datasettet. Vi må utforske andre metoder for å fange opp denne sammenhengen.

Et bilde som inneholder line, skjermbilde, Plottdiagram, diagram

Automatisk generert beskrivelse

Jeg har laget en korrelasjonsmatrise for å visualisere sammenhengen mellom variablene i datasettet. Det er viktig å tolke resultatene med forsiktighet, da datasettet inneholder både kategoriske og kontinuerlige variabler. Korrelasjonsanalyse måler kun lineære sammenhenger, så det kan være vanskelig å fange opp mer komplekse relasjoner, særlig hvis de ikke er lineære.

Matrisen gir verdifull innsikt for enkelte kontinuerlige variabler, som fysiologisk score og apache fysiologisk score, som begge måler pasientens tilstand med numeriske verdier. Høy korrelasjon (80%) mellom disse variablene tyder på at de måler lignende aspekter, noe som gjør det mulig å vurdere å slå dem sammen.

Dette viser hvordan korrelasjonsmatrisen kan være nyttig for variabler som enten er kontinuerlige eller måler svært like forhold. Slik som overlevelsesestimater

Et bilde som inneholder skjermbilde, tekst, diagram, line

Automatisk generert beskrivelse

### 3.1.9 Oppsummering

Visualiseringen har gitt en bedre forståelse av datasettet. Dataene er skjevt fordelt med mange ekstreme verdier (outliers), noe som kan påvirke modellens resultater. Det er også overlapp i informasjon, spesielt mellom sykdomskategorier og sykdom-underkategorier, som kan forverre analysen. I tillegg korrelerer enkelte variabler sterkt, som "apache fysiologisk score" og "fysiologisk score," samt "overlevelsesestimatene."

Variabler som "lege\_overlevelsesestimat\_6mnd" og "lege\_overlevelsesestimat\_2mnd" har 90 % korrelasjon, mens "overlevelsesestimat\_2mnd" og "overlevelsesestimat\_6mnd" korrelerer med 96 %. Denne sterke korrelasjonen tyder på overlappende informasjon, så jeg vurderer å kombinere disse variablene for å redusere kompleksitet.

Det høye antallet variabler, kombinert med outliers, kan skape støy som svekker modellen. Jeg vil derfor utforske ulike metoder for å redusere støy, som å velge de mest relevante variablene, og vurdere å lage nye variabler for å fange opp viktige mønstre.

Hvordan alle variablene best håndteres, gjenstår å se, men håndtering av outliers og nøye utvalg av variabler vil være viktig for å bygge en god modell.

# MODELLERING

## Innledning

Målet er å utvikle en modell som effektivt kan forutsi lengden på sykehusopphold, til tross for utfordringer som ekstreme verdier (outliers) og liten variasjon mellom sykdommene. Jeg tror dette er mulig gjennom riktig håndtering av dataene.

For å oppnå dette vil jeg nøye velge modeller og håndtere outliers på en måte som forbedrer modellens nøyaktighet og robusthet. Det store antallet variabler og små forskjeller mellom sykdommene kan gjøre det utfordrende for modellen å skille mellom ulike pasientgrupper.

Jeg starter med å håndtere manglende verdier (NaN-verdier) for å sikre en pålitelig evaluering, siden mange modeller ikke fungerer godt med slike mangler. Deretter optimaliserer jeg modellene ved å finne de beste parameterne gjennom hyperparameter-tuning. Når de beste modellene er identifisert, tester jeg variabelutvalg og andre forbedringer for å ende opp med den mest effektive modellen.

Målet er å utvikle en robust modell som overgår baseline ytelsen.

Oppsummert:

1. Fylle inn manglende verdier (NaN-verdier).
2. Optimalisere modellene gjennom hyperparameter-tuning.
3. Identifisere de beste modellene og utforske variabelutvalg.
4. Utføre variabelutvalg for å forbedre modellene.
5. Stå igjen med den beste modellen til slutt.

## 4.1.1 Innfylling av NaN-verdier, NMAR.

Jeg begynte med å analysere mønstre av manglende verdier i datasettet. Først så jeg hvor mye prosentandel hver variabel manglet, og om det var noen jeg kunne fjerne. Ingen hadde en kritisk mengde med manglende variabler, så beholder alle variabler.

Et bilde som inneholder tekst, meny, skjermbilde, design

Automatisk generert beskrivelse

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Uten medisinsk ekspertise var det mest effektivt å benytte biblioteket missingno, som muliggjør visualisering av hvor NaN-verdier oppstår i en korrelasjons matrise. (Geeksforgeeks, 2019). Visualiseringen gjorde det mulig å identifisere om to variabler manglet NaN-verdier på like plasser.

Et bilde som inneholder tekst, skjermbilde, diagram, design

Automatisk generert beskrivelse

Det er sterk korrelasjon mellom blod\_ph og lungefunksjon, bloduera-nitrogen og glukose, urinmengde og glukose, urinmengde og bloduera-nitrogen, lege-overlevelsesestimat-2mnd og 6mnd.

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelseEn annen tilnærming er å undersøke korrelasjonen mellom variabler ‘manuelt’. Dette krever en viss medisinsk kunnskap for å forstå hvilke variabler som sannsynligvis henger sammen. Ville tatt tid å få gjort dette mellom alle variabler.

**4.1.2 Blod\_ph og lungefunksjon**

Et bilde som inneholder tekst, skjermbilde, Font, nummer

Automatisk generert beskrivelseJeg så en sterk korrelasjon mellom blodets pH-verdi og lungefunksjon, noe som er logisk, siden lungefunksjonen regulerer pH-verdien i blodet gjennom kontroll av karbondioksidnivået (Hamm, Nakhoul, & Hering-Smith, 2015). Mange av verdiene for lungefunksjon i datasettet avviker fra normalen, noe som tyder på at blod-pH ofte måles ved unormal lungefunksjon.

Informasjonen om datasettet vårt oppgir at normalområdet for blodets pH-verdi ligger mellom 7,35 og 7,45. Derfor fyller jeg inn manglende verdier med 7,40, som er midtpunktet i dette området. Når det gjelder lungefunksjon, følger jeg anbefaling fra nettsiden datasettene er hentet fra, og fyller inn NaN-verdiene med 333,3. (Nickson, 2024)

Jeg vurderte å omgjøre disse variablene til binære verdier (0 for normal, 1 for unormal), men har valgt å beholde dem som kontinuerlige for å fange opp nyanser i dataene. Tester dette ved variabelutvinningen.

**4.1.3 Glukose og bloduera nitrogen**

Forhøyede glukosenivåer kan indikere diabetes, mens blodurea nitrogen (BUN) måler nyrenes evne til å filtrere avfallsstoffer. Økte glukosenivåer er en kjent risikofaktor for nyreskade, noe som kan forklare korrelasjonen i matrisen, når én verdi mangler, mangler ofte også den andre. Den normale verdien for BUN er 6,51 mg/dL, så jeg fyller inn manglende verdier med denne for konsistens i dataene. (Blood urea nitrogen, 2024)

For glukoseverdier, som normalt ligger mellom 70 og 100 mg/dL, bruker jeg midtpunktet på 85 mg/dL for å fylle inn manglende verdier. Dette gir en mer konsistent databehandling og tilrettelegger for mapping av data senere. (Blood sugar level, 2024).

Ved å følge denne tilnærmingen sikres pålitelighet i modellens resultater og bedre tilpasning til medisinske forhold.

**4.1.4 Urinmengde**

I datasettet anbefales det å fylle inn de manglende verdiene for urinmengde med verdien 2502, noe jeg har implementert for å opprettholde konsistensen i dataene.

Jeg observerer en sterk korrelasjon mellom urinmengde, blodurea nitrogen (BUN), og glukose. Dette kan skyldes at høye glukosenivåer ofte resulterer i utskillelse av glukose i urinen, som er et symptom på diabetes. Ved unormalt høye glukosenivåer er det også vanlig å vurdere nyrefunksjonen ved å måle blodurea nitrogen.

Urinmengde, BUN og glukose er viktige indikatorer i vurderingen av både diabetes og nyrefunksjon, da de gir verdifull informasjon om kroppens evne til å regulere og utskille avfallsstoffer effektivt. (nhi.no, 2024).

## 4.2.1 Fyller inn NaN-verdier,MAR.

Prosessen i korte trekk:

**4.2.2 Best imputasjonsstrategi for NaN-verdier, MAR.**

Dette steget kommer etter hyperparametertuning, men nevnes her for å sikre riktig rekkefølge og korrekt imputasjon av NaN-verdier. Ved å håndtere imputasjon etter optimalisering av parametere, sikrer vi at de manglende verdiene fylles inn på en måte som best støtter modellens ytelse og nøyaktighet.

I første omgang fyller jeg inn de gjenværende NaN-verdiene med gjennomsnittsverdiene for hver kolonne. Denne tilnærmingen gir en rask og enkel løsning, som kan fungere som et godt utgangspunkt. Denne metoden kan noen ganger føre til forenklede estimater hvis dataene har mye variasjon. Derfor vil jeg sjekke modeller med flere imputasjonsteknikker.

Deretter gjør jeg hyperparameter-tuning for å finne de beste innstillingene for hver modell. Når de optimale parametrene er satt, tester jeg ulike måter å håndtere manglende verdier på, uten å måtte justere modellene på nytt for hver metode. Dette lar meg sammenligne effektiviteten til de forskjellige imputeringsmetodene under de beste forholdene for hver modell.

Tester:

* Gjennomsnittsimputering (mean),
* Median-imputering,
* Imputering basert på den hyppigst forekommende verdien (most frequent),
* KNN-imputere med ulike antall naboer (KNN med 50, 100, og 150 naboer).

Grid Search tester systematisk ulike kombinasjoner av imputeringsteknikker og hyperparametere for modellene, slik at jeg kan identifisere den beste kombinasjonen som gir lavest rmse. For meg viste ExtraTreeRegressor sammen med simpleImputer, med gjennomsnittet den laveste rmse. Dette var også den mest effektive teknikken for Gradient Boosting.

Jeg bruker en Pipeline for å automatisere prosessen med å teste ulike imputere på forskjellige modeller. Dette gjør det mulig å effektivt sammenligne kombinasjoner av imputeringsteknikker og modeller i én samlet prosess.

Jeg vurderte å inkludere log-transformasjon for RandomForest-modellen, men det viste seg å være krevende å implementere i koden, så jeg valgte å fokusere på de andre modellene.

Øverst i skjemaet

Nederst i skjemaet

For å minimere kjøretiden har jeg delt prosessen inn i to steg. Selv om det er mulig å teste både imputering av NaN-verdier og hyperparametere samtidig med Grid Search, har jeg valgt å optimalisere hver for seg for å sikre bedre kontroll over prosessen.

Jeg ser da at den beste imputeringsprosessen er simpleMean, altså gjennomsnitt:

Et bilde som inneholder tekst, meny, skjermbilde, Font

Automatisk generert beskrivelse

## 4.3.1 Identifisering av den beste modellen på hyperparameter-tuning

**4.3.2 Grunnlinjemodell**

For å evaluere ytelsen til mer avanserte modeller trenger jeg en grunnlinjemodell som referansepunkt. Grunnlinjen fungerer som et minimumskrav – de andre modellene må prestere bedre enn denne for å anses som nyttige. Min nåværende baseline rmse er 20,6, noe som jeg ikke anser som spesielt bra. De fleste opphold er mellom 0-10 dager, så jeg anser som 20.6 som høyt. En lavere rmse, vil bety bedre nøyaktighet.

**4.3.3 Tester ulike modeller**

Jeg vil teste flere ulike modeller uten hyperparameter-tuning for å raskt få en idé om hvilken modell som fungerer best på datasettet mitt. Ved å teste flere modeller kan jeg tidlig eliminere de som presterer dårlig, siden én modell kan fungere bedre på mitt spesifikke datasett. Jeg har brukt ensemble-metoder i noen av modellene, der flere beslutningstrær blir kombinert for å lage en sterkere og mer robust modell.

Modellutvalg (Scikit learn, u.d.):

1. Decision Tree Regressor: Beslutningstrær er gode til å fange opp ikke-lineære sammenhenger og interaksjoner mellom variabler. De er enkle å forstå og tolke.
2. GradientBoostingRegressor: Bygger modellen gradvis ved å legge til nye trær som korrigerer feilene fra tidligere trær. Denne modellen er svært effektiv for datasett med komplekse strukturer og mange variabler.
3. ExtraTreesRegressor: Lager trær på en mer tilfeldig måte enn vanlige beslutningstrær. Her velges splitter tilfeldig fra et utvalg av variabler, noe som gjør modellen raskere og mindre utsatt for overtilpasning.
4. RandomForestRegressor: Kombinerer flere beslutningstrær for å skape en robust modell med lav risiko for overtilpasning ved å ta gjennomsnittet av prediksjonene fra hvert tre.
5. MLPRegressor (nevralnettverk): Kan lære komplekse mønstre i dataene, spesielt nyttig for datasett med mange ikke-lineære sammenhenger.
6. Lasso (lineær modell med L1-regulering): En lineær modell som både kan predikere og utføre variabelseleksjon ved å straffe mindre viktige koeffisienter, noe som gjør den godt egnet for datasett med mange variabler.

Jeg ser at GradientBoosting presterer best uten hyperparameter-tuning, og jeg forventer at modellen vil bli enda bedre med riktig tuning. Likevel tester jeg flere andre modeller for å undersøke om hyperparameter-tuning gir bedre resultater for noen av dem. Vil teste videre ExtraTreesRegressor, MLPRegressor og Lasso.

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

### 4.3.4 Grunnlinjemodell log-transformasjon

Prøver å bruke en log-transformasjon slik jeg kan redusere virkningen av outlierne. (Mondal, 2024). Log-transformasjonen komprimerer skalaen til dataene, slik at ekstremt høye verdier blir mindre dominerende. Dette ser man på illustrasjonen nedenfor. Bidrar til å balansere fordelingen og kan forbedre modellens ytelse ved å gjøre dataene mer håndterbare.

Jeg ser derimot at baseline for log-predikasjonen blir høyere, noe som kan skyldes at jeg får modellen til å fokusere mer på små verdier og kan bli mindre nøyaktig for de større verdiene. Dette ser jeg igjen i modellene jeg tester (samme modeller som forrige grunnlinjemodell). Dette kan være at modellen blir mer nøyaktig for lavere verdier, men gjør dem mindre presise når det gjelder å forutsi pasienter som har lengre opphold på sykehuset. ExtraTreeregressor er et unntak, hvor den faktisk blir litt lavere med log-transformasjon.

Et bilde som inneholder diagram, skjermbilde, line, Plottdiagram

Automatisk generert beskrivelseTransformasjonen forbedrer modellenes nøyaktighet for lave verdier, men den gjør dem mindre presise når det gjelder å forutsi høye verdier, noe som kan være en ulempe ettersom vi har pasienter med lengre opphold på sykehus.

Modellene fra log transformasjonen, tester videre med ExtraTreesRegressor med log-transformasjon:

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

### 4.3.5 Hyper parameter tuning

eg bruker GridSearch til å finne de beste hyperparameterne for hver modell ved å teste alle kombinasjoner av parametere. Den benytter kryssvalidering, som evaluerer ytelsen på flere deler av dataen og bidrar til å unngå overfitting. Dette gir en grundig gjennomgang av parametere, og sikrer at jeg finner de beste parameterne for å oppnå lavest mulig rmse. (Hassan, 2023).

Jeg kunne ha inkludert både GridSearch og hyperparametertesting for alle modellene i ett samlet søk, men dette ville tatt betydelig mer tid. Derfor har jeg valgt å dele opp prosessen for bedre effektivitet og mer oversikt.

Jeg fokuserer på de mest lovende modellene basert på ytelse uten tuning, men inkluderer flere for å vurdere effekten av hyperparameter-tuning. Lar de beste hyperparameterene stå ved innlevering, ettersom dette kan ta mye kjøretid. Modellene jeg velger basert på tidligere resultater:

1. ExtraTreesRegressor
2. GradientBoosting
3. Lasso
4. MLP
5. Log-transformasjon med ExtraTreesRegressor.

**4.3.6 Resultat av hyperparameter-tuning**

Oppsummering på modellene:

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Etter hyperparameter-tuningen kommer ExtraTreesRegressor (uten log-transformasjon) best ut. Altså den presterer nå bedre enn GradientBoosting. Modellen er bedre tilpasset datasettet med de optimale parameterene.

Jeg har visualisert prediksjonen mot de faktiske verdiene i et prediksjonsplott, der den røde linjen representerer en ideell prediksjonslinje. Det er tydelig at prediksjonsnøyaktigheten synker ved lengre oppholdslengder, noe som tyder på at modellen sliter med å forutsi lange opphold. Dette mønsteret går igjen på tvers av alle modellene, noe som indikerer at ingen av dem er spesielt gode til å fange opp faktorer knyttet til lengre opphold. I tillegg ser jeg at lasso predikerer negative verder, noe som er feil. Ingen kan være på sykehus i -20 dager. Det samme gjør MLP, vil ikke ta disse med videre.

Et bilde som inneholder tekst, line, Plottdiagram, skjermbilde

Automatisk generert beskrivelseEt bilde som inneholder tekst, line, Plottdiagram, Font

Automatisk generert beskrivelse

Et bilde som inneholder skjermbilde, tekst, line, Plottdiagram

Automatisk generert beskrivelseEt bilde som inneholder tekst, line, Plottdiagram, Font

Automatisk generert beskrivelse

Et bilde som inneholder tekst, Plottdiagram, line, skjermbilde

Automatisk generert beskrivelse

**Feature Importance**

Jeg gjennomførte en feature importance-analyse med modellene GradientBoosting, ExtraTreeRegressor, og ExtraTree med log-transformasjon. Analysen identifiserte hvilke variabler som anses som mest nyttige av modellene. Denne innsikten kan utnyttes til å forbedre variabelutvalget i dataforbehandlingsfasen, noe som potensielt kan øke modellens ytelse.

ExtraTreesRegressorEt bilde som inneholder tekst, meny, skjermbilde, Font

Automatisk generert beskrivelse GradientBoosting ExtraTree-log

Et bilde som inneholder tekst, meny, skjermbilde, Font

Automatisk generert beskrivelseEt bilde som inneholder tekst, meny, skjermbilde, Font

Automatisk generert beskrivelse

Et bilde som inneholder tekst, skjermbilde, Font, design

Automatisk generert beskrivelse

Et bilde som inneholder tekst, Font, skjermbilde, nummer

Automatisk generert beskrivelse

Et bilde som inneholder tekst, Font, skjermbilde

Automatisk generert beskrivelse

Disse listene viser hvilke variabler som anses som mest betydningsfulle. Samtlige modeller legger mest vekt på variabelen ARF/MOSF med sepsis, noe som kan indikere at denne sykdommen har stor innvirkning på oppholdslengden. Dette samsvarer med tidligere observasjoner, hvor ARF/MOSF-sykdommen hadde flest uteliggere (outliers), og kan forklare hvorfor modellene tillegger den så stor betydning.

Denne innsikten er nyttig for å velge ut de mest relevante variablene, slik at vi kan fokusere på de som modellen vurderer som viktige. Eventuelt kan vi også bruke denne informasjonen til å gjøre videre variabelutvinning basert på hvilke variabler som har størst innvirkning.

### 4.4.2 Overfitting i modeller

Jeg analyserer overfitting i modellene for å vurdere hvilke som generaliserer best til ukjent data.

MLP og Lasso utelukkes videre på grunn av deres tendens til å predikere negative oppholdslengder, samt at de har høyere rmse på trening enn på validering. Dette indikerer at de sliter med å modellere dataene presist, noe som gjør dem mindre egnet for å oppnå pålitelige og realistiske prediksjoner.

Selv om ExtraTree-log oppnår den laveste rmse på treningssettet, presterer den dårlig på valideringssettet. Dette tyder på overfitting, der modellen lærer seg detaljene i treningsdataene, men klarer ikke å overføre denne kunnskapen til ny, ukjent data.

ExtraTreesRegressor oppnår den laveste rmse på valideringssettet, noe som indikerer god generaliseringsevne. Modellen har likevel tegn til overfitting, ettersom feilen er lavere på treningsdataene enn på valideringsdataene.

GradientBoosting har høy rmse på både trenings- og valideringssettet, men ser ut til å generalisere godt til ukjent data, siden rmse-verdiene er nesten like for de to datasettene. Dette indikerer at modellen ikke overtilpasser treningsdataene. Likevel er rmse fortsatt høy, så jeg vil utføre ytterligere variabelutvinning og optimalisering av hyperparametere for å se om dette kan forbedre modellen.

For videre variabelutvinning og videre hyperparameter-tuning, velger jeg å fokusere på modellene ExtraTreesRegressor, ExtraTreesRegressor-log, og GradientBoosting for å undersøke om optimalisering av disse kan forbedre resultatene.

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Et bilde som inneholder tekst, line, Plottdiagram, Font

Automatisk generert beskrivelse

## 5.0.1 Variabelutvinning

### Innledning

Datasettet har mange variabler, og jeg vil nå undersøke om variabelutvinning kan forbedre modellens ytelse. Ved å fokusere på de mest relevante variablene kan jeg kanskje øke nøyaktigheten og gjøre analysen enklere. Samtidig tester jeg hvilke hyperparametere som gir best resultat.

### 5.0.2 Analyse av korte og lange opphold.

Før jeg begynner med ommapping og justeringer, vil jeg undersøke om det er signifikante forskjeller mellom pasienter med lange og korte sykehusopphold. Tidligere visualiseringer viste ingen klare mønstre, så jeg deler nå pasientene inn i grupper med opphold over og under 100 dager. Jeg ser at enkelte sykdommer kun forekommer hos pasienter med kortere opphold (se listen under), noe som kan være viktig å ta hensyn til. Målet er å identifisere muligheter for å lage nye variabler eller tilpasninger som kan styrke modellens ytelse.

Analysen viser at enkelte sykdommer kun forekommer hos pasienter med kortere opphold, som lungekreft og tykktarmskreft, samt en høyere forekomst av MOSF/w Malig. På den annen side ser vi at sykdommen ARF/MOSF med sepsis forekommer hyppigere blant pasienter med lengre opphold. Denne innsikten kan være verdifull for videre modellering og variabelutvinning, da det gir en indikasjon på hvilke sykdommer som har størst innvirkning på oppholdslengden.

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelseEt bilde som inneholder tekst, meny, skjermbilde

Automatisk generert beskrivelseLange opphold / Korte opphold:

Gjennom analysen oppdaget jeg at det er viktig å tildele sykdommer som koma og ARF/MOSF høyere alvorlighetsgrad når jeg mapper variabler i sykdomskategorien. Dette er fordi disse sykdommene har større sannsynlighet for å føre til lengre opphold. Denne innsikten gjorde at jeg måtte gå tilbake og justere alvorlighetsgraden i variabelmappingen (se Kapittel 2.1.5).

### 5.0.3 Beholde sykdomskategori?

Jeg vurderer om det er fordelaktig å beholde sykdomskategori som en samlet variabel for å gruppere lignende sykdommer, da flere sykdommer har tilsvarende oppholdslengde, som sett i visualiseringene i kapittel 3.1.3 og 3.1.4.

Når jeg beholder sykdomskategorien, observerer jeg at overfittingen i ExtraTrees har blitt redusert, og rmse har sunket litt for både ExtraTrees og ExtraTrees-log. Dette indikerer at endringene har hatt en positiv effekt. Som neste steg vurderer jeg å fjerne duplikater i datasettet før jeg går videre med variabelutvinning. Ved å gjøre datasettet renere, kan jeg potensielt forbedre den grunnleggende rmse ytterligere. Etter denne justeringen vil jeg evaluere modellen på nytt for å se om endringen faktisk gir bedre resultater.

Et bilde som inneholder tekst, skjermbilde, Font, sort

Automatisk generert beskrivelseRmse før og etter sykdomskategori er lagt ved:

|  |  |  |
| --- | --- | --- |
| Funksjon | RMSE før | RMSE etter |
| ExtraTreesRegressor | 18.63 | 18.62 |
| GradientBoosting | 18.96 | 19.09 |
| ET-log | 19.38 | 18.98 |

Når jeg skal fjerne duplikater, fjerner jeg underkategoriene som kan plasser i en sykdomskategori. Altså sykdommene med lik oppholdslengde. Jeg klarer som sagt ikke å plasser kreft, og kreft med spredning så jeg beholder disse.

For koma har jeg tre variabler: en som underkategori, en i sykdomskategori og en for koma-score. Jeg velger å fjerne koma som underkategori og beholde koma-score for en mer presis måling basert på Glasgow-skalaen. Når det gjelder ARF/Sepsis, vet jeg at dette er en viktig faktor, men siden den allerede er inkludert i sykdomskategori, fjerner jeg den fra underkategoriene. Lunge- og colonkreft representeres i den samlede kreftkategorien i sykdomskategori, som nevnt tidligere.

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelseJeg fjerner da følgende kolonner for å unngå duplikater i koden min:

Ved å gjøre disse endringene ser jeg at alle modellene nå generaliserer bedre. Dette kan skyldes at jeg har fjernet overlappende informasjon i datasettet. Når flere variabler har overlappende informasjon om sykdom, kan modellene bli forvirret av små forskjeller og dermed overtilpasse seg til enkelte variabler. Jeg merker at Gradient Boosting nå har nesten lik RMSE på trenings- og valideringssettet, men den totale RMSE er fortsatt relativt høy. Jeg vil derfor fortsette med videre variabelutvinning for å se om modellene kan forbedres ytterligere.

|  |  |  |
| --- | --- | --- |
| Funksjon | RMSE før | RMSE etter |
| ExtraTreesRegressor | 18.62 | 18.57 |
| GradientBoosting | 19.09 | 18.97 |
| ET-log | 18.98 | 18.90 |

Overfitting:

Et bilde som inneholder tekst, Font, skjermbilde

Automatisk generert beskrivelse

### 5.0.3 Videre variabelutvinning

Når datasettet er renset og forbedret, vil jeg analysere hvordan ulike variabelutvinningsmetoder påvirker rmse for dette datasettet. Jeg vil sammenligne rmse før og etter hver variabelutvinning for å evaluere effekten av hver metode. Til slutt vil jeg forsøke å kombinere variabelutvinningsmetoder som har gitt gode resultater individuelt (hvis det er noen), for å se om en sammenslåing kan gi enda bedre ytelse.

Jeg legger ved en tabell som viser hvordan de ulike variablene påvirker modellene. Uthever der rmse minker/er lik.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Variabelutvinning | ET før | ET etter | Gradient Før | Gradient Etter | ET-log før | ET-log etter |
| Binære variabler | 18.57 | 18.80 | 18.97 | **18.93** | 18.90 | 18.94 |
| Opphold > 150 | 18.57 | **18.57** | 18.97 | **18.97** | 18.90 | 19.32 |
| Merge fysiologisk | 18.57 | **18.54** | 18.97 | **18.83** | 18.90 | 19.35 |
| Merge overlevese | 18.57 | 18.72 | 18.97 | 19.01 | 18.90 | 19.39 |
| Fysi\*komorbiditeter | 18.57 | 18.60 | 18.97 | 19.00 | 18.90 | 19.03 |
| Alder\*diabetes | 18.57 | 18.59 | 18.97 | 19.01 | 18.90 | 19.34 |
| Nyrefunksjon | 18.57 | 18.58 | 18.97 | 19.01 | 18.90 | **18.77** |
| Hjertefunksjon | 18.57 | 18.60 | 18.97 | 19.00 | 18.90 | 19.30 |
| Lungefunskjon | 18.57 | 18.61 | 18.97 | 19.00 | 18.90 | **18.08** |
| Kreft, kreft-meta | 18.57 | 18.61 | 18.97 | **18.90** | 18.90 | 19.31 |
| Gj.snitt opphold | 18.57 | 18.61 | 18.97 | **18.90** | 18.90 | 19.28 |
| Mapper verdier | 18.57 | 18.8 | 18.97 | **18.92** | 18.90 | 19.10 |

**5.0.5 Binære variabler**

Siden analysen ikke viste tydelige forskjeller mellom lange og korte opphold, valgte jeg å teste binær koding, der normale verdier ble mappet til 0 og unormale til 1. Målet var å se om modellene kunne tolke data bedre med en forenklet struktur. Resultatene viste at rmse økte for ExtraTrees og ET-log, men sank for GradientBoosting, som ser ut til å kompensere bedre og generaliserer bra til usett data.

**5.0.6 Redusere påvirkning på opphold over 150 dager.**

Ettersom jeg vet at datasettet inneholder mange outliers, forsøker jeg å redusere påvirkningen av de øverste 5% av disse først. Jeg oppnår dette ved å bruke "clip" på den 95. persentilen, noe som setter en øvre grense for verdiene. Deretter gjennomfører jeg hyperparameter-tuning for å undersøke om dette fører til bedre modellresultater. Ser at den øverste oppholdslengden i treningsdataen er på 56 dager.

Dette fører til dårligere prediksjon på valideringssettet og forsterker overfitting, siden vi i praksis lærer modellen å fungere best på kortere opphold. Modellen mister evnen til å generalisere for lengre opphold, noe som svekker ytelsen.

Et bilde som inneholder tekst, skjermbilde, diagram, line

Automatisk generert beskrivelse

Derimot så prøvde jeg meg frem og fant ut at resultatet ble bedre hvis jeg fjernet alle de mest ekstreme oppholdene, som jeg anser å være over 150 dager Utliggerne er mindre ekstreme og tettere på hovedgruppen. Ved å redusere de mest ekstreme verdiene, blir modellen bedre i stand til å fange opp mønstre i dataene som gjelder for flertallet av observasjonene. Dette kan potensielt forbedre modellens generaliseringsevne, siden den ikke bruker unødvendig kapasitet på de mest ekstreme utliggerne som kanskje ikke representerer vanlige pasientopphold bra.

Et bilde som inneholder skjermbilde, tekst, line, Rektangel

Automatisk generert beskrivelse

**5.0.7 Slår sammen verdier med høy korrelasjon**

Jeg slår sammen variablene apache\_fysiologisk\_score og fysiologisk\_score, siden begge måler fysiologiske tilstander og har en sterk lineær korrelasjon. (Ser dette fra korrelasjonsmatrisen, 3.1.8). Det gir derfor mening å kombinere dem for å redusere overlapp i modellen. Denne sammenslåingen gir bedre resultater og den sammenslåtte variabelen blir en viktig faktor i ET, og gradient. Dette kan skyldes at jeg eliminerer to svært like variabler, noe som gjør at modellen blir mer effektiv og presis.

Jeg kombinerte deretter overlevelsesestimater og lege-overlevelsesestimater, men oppdaget at dette førte til en økning i rmse, noe som tyder på at viktig informasjon fra de opprinnelige variablene kan gå tapt. Jeg forsøkte også ulike strategier som min/max og median for å slå sammen verdiene. Prøvde å slå sammen 2mnd og de på 6mnd osv., men disse resulterte i ytterligere forverring av validerings-rmse. Igjen tester jeg med flere ulike parametere for modellene.

**5.0.8 Kombinerer fysiologisk score med antall komorbiditeter.**

Jeg ønsker å undersøke om det er en sammenheng mellom antall sykdommer en pasient har, og lengden på sykehusoppholdet, spesielt for de som har ekstremt lange opphold (over 135 dager) sammenlignet med kortere opphold. Jeg la merke til at pasienter med lange opphold ofte har færre registrerte sykdommer. For å utnytte denne innsikten, har jeg laget en ny variabel som kombinerer den fysiologiske scoren med antall komorbiditeter. Målet med dette er å hjelpe modellen med bedre å forutsi lengden på oppholdet basert på pasientens samlede helsetilstand. Får ikke noe særlig bedre rmse på validering i tabellen.

Dette bildet, vil kun vise ‘short stays’ etter klipping hvis jeg ender å bruke denne endringen i den endelige modellen.

Et bilde som inneholder line, diagram, Rektangel, Plottdiagram

Automatisk generert beskrivelse

**5.0.9 Kombinerer alder med diabetes**

Jeg kombinerte alder og diabetes i en ny variabel for å undersøke om dette kunne forbedre modellens ytelse, siden sannsynligheten for diabetes øker med alderen. Resultatet ble imidlertid dårligere, noe som tyder på at samspillet mellom alder og diabetes ikke har en sterk innvirkning på modellens prediksjonsevne.

**5.1.0 Kombinerer nyrefunksjoner**

Jeg identifiserte variablene blodurea\_nitrogen, kreatinin og natrium som indikatorer på nyrefunksjon. For å redusere overlapp, kombinerte jeg dem ved å ta gjennomsnittet og opprette en ny variabel kalt ‘nyrefunksjon’. Jeg testet også å inkludere ‘serumalbumin’ i denne kombinasjonen, men resultatet ble dårligere, noe som kan skyldes at serumalbumin måler mer enn bare nyrefunksjon eller at modellen min har nytte av å beholde den som en egen variabel.

**5.1.1 Hjertefunksjon**

Jeg har blodtrykk og hjertefrekvens som begge indikerer noe om hjertefunksjonen, slå derfor disse sammen. Minket ikke rmse for noen.

**5.1.2 Lungefunksjon**

Jeg prøvde å kombinere variablene lungefunksjon og blod-pH, da disse målerelatene har en sammenheng og kan gi en helhetlig indikator på pasientens lungehelse. Ved å slå dem sammen, håper jeg å forenkle modellen, så at rmse-en ble værre.

**5.1.3 Kombinerer kreft og kreft-metastatic**

Siden en pasient med metastatisk kreft alltid også har kreft, velger jeg å kombinere disse to variablene ved å ta maksimumsverdien. Dette gir en samlet kolonne som indikerer om pasienten har kreft. Jeg har observert at oppholdslengden for pasienter med kreft og metastatisk kreft er ganske lik, selv om kreft har noen få utstikkere, mens metastatisk kreft har en jevnere fordeling. Ved å slå sammen disse variablene reduserer jeg overlapp i dataene, noe som kan hjelpe modellen til å tolke informasjonen mer effektivt og forbedre ytelsen.

**5.1.4 Kolonne for gjennomsnittlig oppholdslengde for sykdommene.**

Jeg prøvde å legge til en kolonne som viser gjennomsnittlig oppholdslengde for de ulike sykdomskategoriene. Resultatet ble imidlertid en høyere rmse. Dette kan skyldes at modellen fokuserer på en generalisert verdi for hver sykdomskategori i stedet for individuelle variasjoner. Denne tilleggsinformasjonen kan introdusere støy fremfor nyttig detaljering, spesielt hvis andre variabler allerede dekker de relevante sykdomsrelaterte forskjellene. Gradient klarte å generalisere bedre til denne informasjonen.

**5.1.5 Mapper verdier**

Jeg forsøkte å mappe ulike verdier ved å dele dem inn i kategorier for å gjøre det enklere for modellen å tolke dataene. Hensikten var å gi modellen en mer strukturert tilnærming til variablene, men resultatet førte til høyere rmse. Dette kan tyde på at verdiene er bedre representert i sin opprinnelige form, og at kategorisering av verdiene i dette tilfellet kan føre til tap av detaljert informasjon som er viktig for nøyaktige prediksjoner

### 5.1.6 Prøver å kombinere forskjellige variabelutvinninger

Jeg har nå testet ulike variabelutvinninger og implementert dem som av/på-knapper for fleksibilitet. Videre vil jeg undersøke om kombinasjoner av variabelutvinninger kan gi bedre resultater sammen enn hva de gjorde individuelt. Ingen utvinning viste seg for å være veldig effektiv. Jeg viser ikke resultatene fra alle forsøk, siden dette ville blitt for omfattende. Jeg har eksperimentert med forskjellige parameterinnstillinger sammen med ulike variabelutvinninger, og gjennom disse testene har jeg identifisert de mest lovende kombinasjonene:

* Klipp ekstreme verdier ned til 150 (beholder egenskaper, bare ‘setter’ ned oppholdslengden).
* Slår sammen apache- og fysiologisk score, gjennomsnittet.
* Multipliserer sammen den sammensatte fysiologiske scoren med antall komorbiditeter.
* Slår sammen verdier for nyrefunksjon, gjennomsnittsverdien.
* Slår sammen for hjertefunksjon.
* Slår sammen kreft og kreft-metastatic.
* Legger til en vekt jeg trener modellene på.

Jeg observerte at modellene har utfordringer med å predikere lengre opphold, og at prediksjonene ofte blir høyreforskyvet. For å hjelpe modellen med å vektlegge sykdommer som oftere forekommer ved kortere opphold, tilordner jeg høyere vekter til disse sykdommene under treningen. Dette betyr at modellen vil «lære» betydningen av sykdommer knyttet til kortere opphold bedre og justere prediksjonene deretter. Ved å tildele større vekt til sykdommer assosiert med kortere opphold, kan jeg motvirke skjevheter i dataene som fører til høyreforskyvde prediksjoner, noe som igjen resulterte i en lavere rmse.

Jeg testet vektlegging av både lange og korte opphold og fant at det å vektlegge korte opphold ga best resultat. Dette skyldes at modellen fokuserer på de kortere oppholdene og reduserer påvirkningen av ekstreme verdier.

|  |  |  |
| --- | --- | --- |
| Sykdom | Vekt | Tallverdi i datasetttet |
| ARF/MOSF | 1 | 4 |
| Coma | 3 | 3 |
| Cancer | 4 | 2 |
| COPD/CHF/Cirrhosis | 6 | 1 |

### 5.1.7 Resultat etter variabelutvalg

Etter mye prøving, og feiling ender jeg opp med disse resultatene for de tre modellene mine:

Hvordan de predikerer:

Et bilde som inneholder tekst, line, Plottdiagram, bakke

Automatisk generert beskrivelseEt bilde som inneholder line, Plottdiagram, skjermbilde, tekst

Automatisk generert beskrivelse

Et bilde som inneholder line, Plottdiagram, skjermbilde, tekst

Automatisk generert beskrivelse

Et bilde som inneholder skjermbilde, Plottdiagram, tekst

Automatisk generert beskrivelse

Et bilde som inneholder tekst, Plottdiagram, line, diagram

Automatisk generert beskrivelse

Et bilde som inneholder tekst, line, Plottdiagram, diagram

Automatisk generert beskrivelse

Visualiseringene viser at ingen av modellene er gode til å forutsi lengre opphold. Prediksjonene er høyreforskjøvet, noe som er et resultat av outliersene i datasettet. Samtidig overvurderer alle modellene antallet kortere opphold. Dette skyldes da at datasettet har betydelig mange fler pasienter med korte opphold, og lærer mer fra disse.

Jeg ser også at GradientBoosting predikerer negative verdier, noe som er urealistisk for oppholdslengde. Til tross for at jeg har prøvd ulike metoder for variabelutvinning, har dette problemet vedvart. Dette tyder på at modellen ikke klarer å tilpasse seg datastrukturen godt nok og sliter med å fange opp de underliggende mønstrene i dataene.

Visuelt fremstår ExtraTree-log som den modellen som best fanger de faktiske verdiene i treningssettet sammenlignet med de predikerte verdiene. Basert på denne observasjonen, vil jeg videre undersøke ExtraTree-log for tegn på overfitting og analysere rmse-verdiene for å bekrefte ytelsen.

Ved sjekk av overfitting viser GradientBoosting seg å generalisere best, med en rmse som ikke ligger langt over ExtraTreesRegressor. Samtidig observerer jeg at ExtraTree-log, som presterte godt på treningssettet, har svakere generaliseringsevne på usett data.

Overfitting-resultatene for ExtraTreesRegressor har forbedret seg siden kapittel 4.4.2, med en reduksjon i overfitting fra 4,63 til 2,63. Selv om overfittingen var noe lavere før variabelutvalget, har rmse nå blitt litt lavere. Jeg ser dette som en positiv utvikling, da modellen nå har lavere rmse og fortsatt god generaliseringsevne, uten betydelig økning i overfitting.

Et bilde som inneholder tekst, diagram, Plottdiagram, line

Automatisk generert beskrivelse

Et bilde som inneholder tekst, Font, skjermbilde

Automatisk generert beskrivelse

Her bruker jeg en modell som viser hvordan modellene predikerer. Ser jo tydelig her at alle modellene er dårlig på lengre opphold, men ExtraTreesRegressor er litt bedre enn de andre. (Chouhary, 2021)

Et bilde som inneholder line, skjermbilde, Plottdiagram

Automatisk generert beskrivelse

**5.1.7 Sjekker Feature Importance**

Jeg har mange variabler i datasettet og har gjennomført en analyse for å identifisere hvilke variabler som er mest betydningsfulle for modellene. Deretter vurderte jeg om ytelsen til ExtraTreesRegressor eller GradientBoosting kunne forbedres ved å fjerne variabler som modellen anså som mindre viktige. Målet var å undersøke om dette kunne redusere problemet med negative prediksjoner i GradientBoosting, ettersom negative verdier ikke gir mening i sammenheng med oppholdslengde. Jeg valgte disse to modellene, ettersom disse ga lavest overfitting, og dermed mulighet til å predikere best på usett data, altså test-settet til slutt.

Visualiseringen av de viktigste variablene forteller meg da at modellene fokusere på ulike variabler som sine viktigste. Dette forekommer jo av at modellene er forskjellige, og har en forskjellig algoritme.

Viktigste variabler:

Et bilde som inneholder tekst, Plottdiagram, line, diagram

Automatisk generert beskrivelse

Et bilde som inneholder tekst, Plottdiagram, line, diagram

Automatisk generert beskrivelse

Da for å finne de verdiene som gir best rmse lager jeg en løkke for å iterere gjennom alle variabelene, og sjekke opp mot den beste rmse (den nåværende rmse).

Her beregner jeg rmse for hver kombinasjon der en variabel er utelatt, og identifiserer hvilken fjerning som gir lavest rmse. Hvis denne verdien er bedre enn den beste så langt, oppdateres funksjonslisten; hvis ikke, stopper prosessen. Den tar en del tid å kjøre, så kommenterer den ut når jeg er ferdig med den.

Jeg observerer at ExtraTreesRegressor vurderer alle variablene som like viktige, mens GradientBoosting har en tendens til å nedprioritere variablene ‘koma-score’ og ‘overlevelsesestimat 6mnd’. Etter å ha fjernet disse variablene, sank rmse fra 18.8 til 18.7, en liten forbedring. Imidlertid fortsetter GradientBoosting å predikere negative verdier, som kan sees på de blå punktene til venstre for 0 på visualiseringen. På grunn av dette velger jeg å beholde disse variablene i datasettet, da fjerning av dem ikke gir en forbedring.

Et bilde som inneholder Plottdiagram, line, diagram, skjermbilde

Automatisk generert beskrivelse

# 6.0.0 Valg av den endelige modell, og prediksjon på testsettet.

Hvis GradientBoosting ikke hadde predikert negative verdier, ville jeg nok valgt denne som min endelige modell. Den har lite overfitting, som betyr at den generaliserer bedre til usett data, og den har ikke særlig mye høyere rmse enn ExtraTreesRegressor.

Så da blir min endelige modell ExtraTreesRegressor, ettersom den har den laveste rmse på valideringssettet og den generaliserer bra til usett data.

Når jeg da predikerer på testsettet får jeg ikke en særlig bra rmse. Den kommer på 21.8.

Et bilde som inneholder tekst, skjermbilde, Font, sort

Automatisk generert beskrivelse

Denne predikerer høyere enn baseline, som er på 20.6 som forteller oss at dette er en dårlig modell. Som vist på visualiseringen under er den høyre forskjøvet og klarer ikke å predikere noen lengre opphold, samt at den overfitter de gjennomsnittlige verdiene. Dette gjør at modellen har en dårlig evne til å fange opp de ekstreme verdiene.

Et bilde som inneholder Plottdiagram, skjermbilde, line, tekst

Automatisk generert beskrivelse

Ser på fordelingen at den ikke klarer å predikere bra, kun nærme de gjennomsnittlige verdiene.

Et bilde som inneholder line, Plottdiagram, skjermbilde

Automatisk generert beskrivelse

### 6.0.1 Andre metoder jeg har prøvd, men ikke brukt

Jeg har også forsøkt flere alternative metoder for å optimalisere modellen, men disse ga ingen forbedringer i rmse. Blant annet testet jeg å sette alle variabler og sykdomsverdier til sine gjennomsnittlige nivåer, men dette resulterte i en høyere rmse. Samt å sette andre variabler til gjennomsnittet, for å teste.

Videre prøvde jeg å slå sammen flere variabler for å redusere kompleksiteten, men dette ga heller ingen merkbar effekt på modellens ytelse. Jeg har valgt å ikke inkludere detaljer om alle disse forsøkene, ettersom det ville gjort rapporten unødvendig lang uten å tilføre ny innsikt.

### 6.0.2 Oppsummering, drøfting av resultat.

Jeg hadde håpet at ved å bruke variabelutvinning, teste flere modeller, finjustere hyperparametere, prøve ulike metoder for å håndtere manglende verdier og analysere dataene grundig, ville resultatene bli betydelig bedre. Likevel ser jeg at fordelingen av de predikerte verdiene fortsatt ikke er god nok. Modellen har utfordringer med å skille mellom korte og lange opphold, noe som har vært vanskelig hele veien. Dette tyder på at modellen sliter med å fange opp de viktigste mønstrene som forklarer variasjonen i oppholdslengden, særlig for lengre opphold.

På valideringssettet ser jeg at rmse-verdien er lavere enn baseline, noe som tyder på at modellen klarer å gi bedre prediksjoner enn et rent tilfeldig gjett. Dette viser at modellen har klart å lære seg noen mønstre i treningsdataene, og den fungerer bedre enn en enkel gjennomsnittsprediksjon, som baseline.

**Utfordringer:**

Derimot får jeg en RMSE på 21,8 på testsettet, noe som antyder at modellen har overtilpasset seg treingsdataen, og ikke generaliserer godt til ny data. Dette kan skyldes at modellen ikke håndterer variasjonen i lengre opphold godt nok, noe som har vært en utfordring hele veien. Resultatet på testsettet viser at modellen trenger videre forbedringer.

Modellen har en tendens til å overtilpasse på kortere opphold og sliter med å forutsi lengre opphold, noe som gjør den mindre nyttig når nøyaktighet for lange opphold er viktig. For å forbedre modellen kunne jeg prøvd en mer presis utvelgelse av variabler og en bedre håndtering av outliers. Jeg ser også at for opphold på 0–10 dager er prediksjonene litt skjevt forskjøvet mot høyre, da modellen ofte antar at opphold på 10–20 dager er mest vanlig. Dette er gjerne forårsaket av skjevheten i dataen.

**Styrker**:

Denne modellen har gjerne en større styrke i å predikere kortere opphold, spesielt hvis man hadde utelukket lengre opphold.

**Potensielle forbedringer:**

Hvis jeg hadde hatt mer tid, ville jeg vurdert å dele treningssettet i to deler: én modell for lengre opphold og én modell for kortere opphold. På den måten kunne jeg funnet spesifikke mønstre som kan skille mellom lengre og kortere opphold. For eksempel kunne jeg undersøkt om fysiologiske faktorer har mer betydning for kortere opphold, mens demografiske faktorer kan ha større innvirkning på lengre opphold. Da kunne jeg også trent variabelutvinning separat for de to modellene.

I tillegg, kunne jeg brukt en overordnet modell til å først avgjøre om en pasient hører til i kategorien for kort eller langt opphold, og deretter valgt riktig modell for videre prediksjon.

En alternativ tilnærming kunne vært å inkludere flere pasienter med lengre opphold for å redusere andelen outliers. Dette ville gjort det lettere for modellen å lære de typiske egenskapene ved lengre opphold, slik at prediksjonene ikke blir skjevt forskjøvet mot kortere opphold. Dette kunne også redusert risikoen for overtilpasning til de korte oppholdene.

# 7.0.0 Nettside

**Klargjør**

Før jeg startet arbeidet med nettsiden, lagret jeg alle prediksjonene fra sample-dataene i en CSV-fil. Dette gjorde jeg slik at jeg senere kan lese inn dataene og bruke dem på nettsiden. Prediksjonene kommer fra den beste modellen min, ExtraTreesRegressor.

Jeg klargjorde sample-dataene i en ny fil kalt "preprocessed", slik at strukturen og innholdet blir likt som i den opprinnelige "raw" datafilen. Dette innebærer samme oppsett når det gjelder variabelutvinning, formatering og andre datajusteringer.

Den trente ET-modellen har jeg lagret som en pickle-fil, slik at den enkelt kan leses inn og brukes på nettsiden. Jeg kunne også brukt en pipeline for å automatisere hele prosessen, men valgte å gjøre det manuelt for å sikre at alt ble riktig, siden jeg har bedre kontroll på denne måten.

Grafen nedenfor viser fordelingen av prediksjonene fra predictions.csv. Som vi har sett tidligere, har modellen vansker med å predikere de lengste oppholdene og har en tendens til å overtilpasse på kortere opphold. Dette resulterer i en høyreforskjøvet fordeling hvor lengre opphold ikke blir godt fanget opp.

Fordeling av de predikerte oppholdene på sample data, ser dette stemmer overens med tidligere observasjoner fra modellen. Høyre forskjøvet slik at de fleste har opphold 10-20 dager, og ikke noen predikasjoner på mer enn 41 dag.

Et bilde som inneholder Plottdiagram, diagram, skjermbilde, line

Automatisk generert beskrivelse

**App**

Først laster jeg inn predictions.csv, som inneholder de predikerte oppholdslengdene, samt den lagrede ExtraTreesRegressor-modellen. Det er avgjørende at kolonnene i predictions.csv er i samme rekkefølge som under treningen, slik at modellen får riktig input og kan gi korrekte prediksjoner.

Jeg har manuelt lagt til standardverdier for inputfeltene, som er enten gjennomsnittlig verdi, den vanlige verdien eller medianen. Ettersom, hvis disse ikke er fylt inn må det tilsvare som en vanlig verdi.

For enkelte variabler, som overlevelsesestimater, setter jeg verdien til 1 dersom de ikke er fylt ut. Dette er basert på en antakelse om at pasienten vil overleve dersom overlevelse ikke er vurdert kritisk nok til å registreres.

På nettsiden har jeg også inkludert enkelte variabler, som fysiologisk\_komorbiditet\_mult. Jeg forsøkte å få denne variabelen til å beregnes automatisk ved multiplikasjon, men besluttet at dette ikke var en prioritet å bruke tid på.

Det kommer opp en feilmelding på nettsiden hvis brukeren prøver å legge til negative verdier, og at den har en del variabler den må fylle inn.

**Html**

HTML-strukturen er ansvarlig for å sette opp nettsidens layout og inkluderer feltene som kreves for prediksjonen. Jeg har delt variablene i de som er obligatoriske å fylle inn og de som kan være valgfrie. Dette gir en ryddigere og mer brukervennlig grensesnitt.

For å forbedre brukeropplevelsen, har jeg lagt til en JavaScript-funksjon som åpner et nytt vindu når en prediksjon blir utført. Selv om denne funksjonaliteten kunne ha vært lagt i en separat fil, valgte jeg å beholde den i HTML-filen da koden er relativt kort og oversiktlig (Pop, 2020).

Ved å legge til slike brukervennlige funksjoner og standardverdier sikrer jeg at appen gir en intuitiv og effektiv brukeropplevelse.

# 8.0.0 Refleksjon

**Tilnærminger og hovedfokus**

Underveis har jeg testet flere tilnærminger og brukt mye tid på å forstå variablene for å få en dypere innsikt i dataen. Dette prosjektet har vært utfordrende, både fordi mange av variablene var nye for meg og fordi forskjellene mellom dem var små, noe som gjorde det vanskelig å se deres fulle nytteverdi.

Hovedfokuset mitt har vært å visualisere dataene og analysere hvordan ulike endringer påvirker datasettet og fordelingen av verdiene. Jeg tror modellen kunne ha blitt mer nøyaktig dersom jeg hadde konsentrert meg om kortere opphold, da jeg da kunne unngått den høyreforskyvningen som outliers skaper. Dette kunne ha gitt mer presise resultater.

**Innsikt og overaskende resultat**

Samlet sett har prosjektet gitt verdifull innsikt i hvordan ulike variabler og datamønstre påvirker prediksjonene. Likevel ser jeg at denne modellen sannsynligvis ikke er gunstig for faktisk bruk på et sykehus. For dette formålet ville en mer avansert modell, som klarer å håndtere både lange og korte opphold uten å overtilpasse seg de korte, være bedre egnet. Dette vil bidra til en mer balansert og pålitelig prediksjon for alle typer oppholdslengder.

Med mer tid kunne jeg ha lagret prosessen for håndtering av rådata i en pipeline. Dette ville gjort det mulig å automatisk bruke den samme databehandlingen på ethvert lignende datasett. I tillegg kunne jeg ha optimalisert koden ved å redusere gjentakelser, noe som ville gjort den både kortere og mer effektiv.

Resultatene var overraskende, da ingen av variabelutvinningsmetodene førte til betydelige forbedringer i rmse. Selv når variabelutvinningen hadde en viss effekt, var forbedringen liten. Med min bakgrunn i data science forventet jeg at justeringer i variablene ville gi en mer positiv effekt på modellens ytelse. Dette tyder på at kvaliteten på dataene er avgjørende for modellens evne til å predikere nøyaktig. For å oppnå bedre resultater, må datasettet være mer stabilt, uten mange outliers, og modellen må klare å identifisere komplekse mønstre som ikke er umiddelbart synlige i dataene.

# Avslutning

Modellen har gitt nyttig innsikt i hva som påvirker lengden på sykehusopphold. Samtidig viser arbeidet at algoritmer bare er så gode som kvaliteten på dataene de bygger på. Fremover bør fokus være på å forbedre datakvaliteten og utvide modellen med flere relevante variabler eller mer avanserte teknikker. Dette kan gi enda mer nøyaktige og praktiske prediksjoner. Det kan også være nyttig å inkludere flere pasienter med lengre opphold, slik at modellen kan trenes mer på disse tilfellene for å redusere forskyvning i dataen.

# Bibliografi

*Blood sugar level*. (2024, August). Hentet fra Wikipedia: https://en.wikipedia.org/wiki/Blood\_sugar\_level

*Blood urea nitrogen*. (2024, September). Hentet fra Wikipedia: https://en.wikipedia.org/wiki/Blood\_urea\_nitrogen

Chouhary, R. U. (2021, Februar). *Analysis and feature engineering*. Hentet fra Medium: https://medium.com/analytics-vidhya/airbnb-data-exploration-analysis-and-feature-engineering-edbb47bf115

Geeksforgeeks. (2019, Juli). Hentet fra https://www.geeksforgeeks.org/python-visualize-missing-values-nan-values-using-missingno-library/

geeksforgeeks. (2019, Juli). *Visualize missing values (NaN) values using Missingno Library*. Hentet fra geeksforgeeks: https://www.geeksforgeeks.org/python-visualize-missing-values-nan-values-using-missingno-library/

Hamm, L., Nakhoul, N., & Hering-Smith, K. (2015, November). *National library of medicine*. Hentet fra https://pmc.ncbi.nlm.nih.gov/articles/PMC4670772/

Harrel, F. (2022). *UC Irvine Machine learning repository*. Hentet fra https://archive.ics.uci.edu/dataset/880/support2

Hassan, M. H. (2023, Oktober). *Using Grid Search For Hyper-Parameter Tuning*. Hentet fra Medium: https://medium.com/@hammad.ai/using-grid-search-for-hyper-parameter-tuning-bad6756324cc

Mondal, R. (2024, Juli). *Log Transformation to Mitigate the Effect of Outliers | Data Cleaning | Exploratory Data Analysis*. Hentet fra Medium: https://medium.com/womenintechnology/log-transformation-to-mitigate-the-effect-of-outliers-413cdd275495

nhi.no. (2024, februar). *Hyperosmolar hyperglykemi*. Hentet fra nhi.no: https://nhi.no/sykdommer/hormoner-og-naring/diabetes-generelt/hyperosmolar-hyperglykemi

Nickson, C. (2024, Juli). *Life in the fastlane*. Hentet fra PaO2/FiO2 Ratio (P/F Ratio): https://litfl.com/pao2-fio2-ratio/

Pop, F. (2020). *Create a Modal (Popup)*. Hentet fra YouTube: https://www.youtube.com/watch?app=desktop&v=XH5OW46yO8I

Scikit learn. (u.d.). Hentet fra https://scikit-learn.org/dev/index.html