Dataset: Titanic

Classification & NN Classification

Markus Kinn

I dette dokumentet kommer jeg til å diskutere valg jeg har tatt og sammenligne resultater fra ulike algoritmer. Jeg har ett dokument for hvert datasett.

For å forbedre resultatene på modellene har jeg prøvd å implementere så mye so mulig fra *Machine Learning Performance Improvement Cheat Sheet.*

**Generelt om datasettet:**

Datasettet går ut på å predikere om en person kom til å overleve Titanic ulykken, basert på disse attributtene: PassengerId, Pclass, Name, Age, SibSp, Parch, Ticket, Fare, Cabin og Embarked.

Datasettet har lite feil, noe som gjør at man ikke trenger å gjøre mye data pre-Processing. Det har derimot noen manglende verdier og noen kolonner som ikke trengs. For mer grundig gjennomgang av datasettet, se EDA i Oblig 1 mappen.

Felles for alle algoritmene var at jeg gikk videre med binning av alder. Dette gjorde jeg for å gjøre denne kontinuerlige verdien om til en kategorisk. Dette gjorde jeg fordi resten av kolonnene jeg trente algoritmene på er kategoriske.

Fare og Embarked ble også fjernet fra datasettet. Fare ble fjernet på grunn av ekstrem over-fitting og Embarked ble fjernet ettersom kolonnene ikke bidro noe til modellene.

**XGBoost:**

XGBoost er en algoritme som håndterer de aller fleste tabell datasett. For Titanic datasettet trengte algoritmen rundt regnet ingen databehandling. Algoritmen håndterer null-verdier selv, men jeg valgte selv å håndtere disse, da jeg fikk hakke bedre resultat på min måte.

Fra EDA så vi at både Fare og Embarked har dårlig distribusjon og jeg vurderte derfor å fjerne disse. Etter å ha testet ut å lage baseline modeller med ulike kombinasjoner å ha med og fjerne disse kolonnene fant jeg ut dette:

* Ha med begge:

Text

Description automatically generated

* Fjerne Embarked, beholde fare:

Text

Description automatically generated

* Fjerne Fare og Embarked:

Text

Description automatically generated

Fra dette kan vi se at det å beholde begge fører til mye over-fitting og generelt ok resultat. Fjerne Embarked, men beholde Fare fører til ekstrem over-fitting, men bedre test score. Fjerne begge fører til lite over-fitting, men generelt litt dårligere resultat. Her er det altså Fare som fører til så stor over-fitting.

Jeg valgte å gå videre med datasettet hvor jeg fjernet både Fare og Embarked da jeg mener at 2 ekstra prosent på test score ikke er verdt nesten 10% ekstra over-fitting. **Dette endte jeg opp med å gjøre for alle modellene.**

**Før tuning:**

**Text

Description automatically generated**

**Valg av tuning strategi:**

Opprinnelig testet jeg ut en svært sekvensiell tuning strategi hvor jeg kjørte griden på kun en og en variabel. Denne strategien brukte betydelig kortere tid enn den litt mindre sekvensielle strategien jeg brukte senere, men presterte som regel litt dårligere.

Jeg testet også ut Halving grid search, men denne virker som fungerer kun bedre på store datasett, noe jeg ikke har.

Til slutt falt valget på Random Search. I random serach kan jeg dytte inn ekstremt mange parametere og verdier, samtidig som resultatet blir ofte bra og på kort tid. Det er derfor kun denne strategien jeg velger å bruke i de fleste notebookene.

(tar kun den beste scoren gjennom mange gjennomganger for Random search)

**Etter tuning (Random search):**

**Text

Description automatically generated**

**Diskusjon av resultater:**

Baseline modellen var bra og ga meg et godt utgangspunkt å jobbe videre fra. Den var noe over-fitted.

I tuning modellen klarte jeg å fjerne tilnærmet all overfitting. Dette skjedde ved at training scoren gikk noe ned, samtidig som test scoren gikk opp. F1-scoren var også hakke høyere enn hva baseline modellens f1-score var. Ettersom både test, f1 og cross val scoren økte, sier jeg meg fornøyd med dette.

**Random Forest:**

For dette datasettet gjorde jeg det samme som for XGBoost.

**Før tuning:**

**Text

Description automatically generated**

**Valg av tuning strategi:**

Her prøvde jeg både min egne litt mindre sekvensielle metode og Random search, men klarte ikke å få bedre resultat med Random search enn med min egen metode. Jeg har derfor valgt å la begge strategiene være i notebooken, men viser kun resultatet av den beste metoden i dokumentet her.

**Etter tuning:s**

**Text

Description automatically generated**

**Diskusjon av resultater:**

Relativt OK baseline modeler, men noe over-fitting.

Etter tuning klarte jeg å øke alle scorene relativt mye, samtidig som over-fittingen ble helt eliminert. En observasjon er at nå scorer test scoren bedre enn training. Siden alle scorene ble økt, sier jeg meg fornøyd med resultatet.

**Gaussian Naive Bayes:**

For denne algoritmen valgte jeg å gjøre noen endringer i datasettet. Primært gikk disse endringene ut på å fikse opp i kontinuerlige verdier og fjerne høyt korrelerte kolonner. I dette datasettet var det to kontinuerlige kolonner: Fare og Age. Det var også to høyt korrelerte kolonner: Plcass og Fare. Dette endte opp med at jeg lagde AgeGroups og fjernet Fare.

**Før tuning:**

**Text

Description automatically generated**

**Valg av tuning strategi:**

Her brukte jeg kun et vanlig grid search da algoritmen ikke har mange parametere å tune

**Etter tuning:**

**Text

Description automatically generated**

**Diskusjon av resultater:**

Denne algoritme slet ikke med over-fitting på baseline modellen eller den ferdig tunet modellen. Resultatet var for så vidt dårligere enn de to forrige algoritmene.

Både baseline og tuning modellen presterte nesten helt likt, men tuning modellen klarte å øke test og training scoren litt, samt en minimal økning i cross val. F1-scoren var helt lik, men siden tuning modellen presterte bedre på de andre scorene, vil jeg fremdeles si at denne var best.

**Support Vector Machine:**

For denne algoritmen valgte jeg å fjerne Fare og AgeGroup, da disse to forverret resultatet med opptil 2%. Jeg måtte også skalere dataen for denne algoritmen. For dette brukte jeg RobustScaler() funksjonen. Grunnen til at dette er viktig for SVM er at den bruker distansen mellom observasjoner til å predikere (veldig kort fortalt).

**Før tuning:**

**Text

Description automatically generated**

**Valg av tuning strategi:**

Her brukte jeg kun et vanlig grid search da algoritmen ikke har mange parametere å tune

**Etter tuning:**

**Text

Description automatically generated**

**Diskusjon av resultater:**

Tilnærmet ingen over-fitting på noen av modellene og bra resultater på baseline modellen. Her slet jeg med å øke resultatet etter tuning og modellene ble helt like. Dette kan komme som et resultat av at jeg har gjort en dårlig jobb eller at dette er rett og slett det beste modellen får til.

**Neural Network Architecture 1:**

For begge neurale nettverkene valgte jeg å fjerne kolonnen Fare grunnet høy korrelasjon med Pclass, samt kolonnen førte til kraftig over-fitting. Embarked ble også fjernet ettersom den ikke bidro til predikasjonen. Jeg valgte å ikke normalisere verdiene da datasettet kun består av kategoriske verdier. Her måtte jeg også one-hot encode alle kategoriske variabler og deretter gjøre alle features om til tensors. Jeg kjørte modellene mange ganger for å finne de beste verdiene.

**Før tuning:**

**Text

Description automatically generated**

**Etter tuning:**

Jeg møtte på mange problemer med Automatisk hyperparameter tuningen når det gjaldt NN og endte derfor opp med å gjøre dette manuelt. Jeg dokumenterer derfor prosessen her:

For dette datasettet og denne arkitekturen klarte jeg ikke å forbedre scoren jeg fikk på baseline modellen. Jeg fulgte den samme arbeidsflyten jeg har brukt i de andre NN-modellene, men etter MYE testing ble ikke scoren forbedret.

Den beste modellen etter tuning ble derfor baseline modellen. Under har jeg med et bilde av den beste modellen jeg klarte å lage etter at jeg begynte å tune:

Text

Description automatically generated

**Diskusjon av resultater:**

Baseline modellen fikk et svært bra resultat, men med litt over-fitting. Etter å ha prøvd på MYE hyperparameter tuning klarte jeg ikke å forbedre scoren. Dette kan komme av at jeg hadde flaks på baselinemodellen og traff ganske bra med engang.

**Neural Network Architecture 2:**

Brukte det samme datasettet som i forrige neurale nettverk. Jeg kjørte modellene mange ganger for å finne de beste verdiene.

**Før tuning:**

**Text

Description automatically generated**

**Etter tuning:**

Nok en gang klarte jeg ikke å forbedre resultatet etter tuning, i forhold til baseline modellen. På denne arkitekturen testet jeg også MANGE ulike verdier for de ulike parameterne, men resultatet ble fremdeles dårligere enn resultatet til baseline modellen.

Den beste modellen etter tuning ble derfor baseline modellen. Under har jeg med et bilde av den beste modellen jeg klarte å lage etter at jeg begynte å tune:

Text

Description automatically generated

**Diskusjon av resultater:**

Baseline modellen fikk et nokså bra resultat, men med noe over-fitting. Nok engang klarte jeg ikke å forbedre resultatet etter tuning. Jeg tipper grunnen til dette er den samme som ved forrige arkitektur.

**Sammenligning av alle resultater:**

Alle modellene gjorde det ganske bra, men de tre som skiller seg ut er XGBoost etter tuning, SVM og baseline modellen til det første neurale nettverket. Disse tre modellene presterte ganske likt, men baseline modellen til det første neurale nettverket tok seieren over alle algoritmene. Jeg prøvde å få en f1-score for NN, men fikk en error som jeg ikke klarte å finne ut av. Jeg belager derfor avgjørelsen kun på accuracy score for trening og test data.

Etter å ha sett på andres modeller, så ser det ut som det er rundt den scoren jeg fikk, som andre også som regel får. Dette kan bety at dette er så høy score som dataen klarer å gi.