Dataset: Concrete

Regression

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:**

Målet til datasettet er å predikere hvor sterk en betong blanding er, basert på betong blandingen. Attributtene vi har er: cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate og age. Forklaring av kolonnene finnes i EDA notebook.

Først og fremst valgte jeg å endre navnet på alle kolonnene til sin korteste versjon ettersom navnene var opprinnelig lange og uhåndterlige. Ellers inneholder datasettet ingen feil eller verdier jeg måtte endre på.

**XGBoost:**

XGBoost er en algoritme som håndterer de aller fleste tabell datasett. For dette datasettet trengte algoritmen ingen databehandling.

**Før tuning:**

Text

Description automatically generated

**Valg av tuning strategi:**

For denne algoritmen testet jeg nok en gang ut 3 strategier: sekvensiell, mindre sekvensiell og Random Search.

**Etter tuning (Sekvensiell):**

Text

Description automatically generated

**Etter tuning (Mindre Sekvensiell):**

**Text

Description automatically generated**

**Etter tuning (Random Search):**

Text

Description automatically generated

**Diskusjon av resultater:**

Baseline modellen hadde god treffsikkerhet, men var veldig over-fitted. Med tanke på den kontinuerlige variabelen vi prøver å predikere var R2 scoren ganske bra.

Etter de 3 strategiene jeg brukte for tuning klarte jeg å redusere over-fittingen ganske mye ved å senke training score og øke test score, samt bra nedgang i R2. Etter dette var det den sekvensielle strategien som generelt gjorde det best. Dette kommer nok av ren flaks da de andre strategiene burde i teorien gjøre det bedre.

**Random Forest:**

For dette datasettet trengte ikke RF noe ekstra databehandling enn hva jeg gjorde for XGBoost.

**Før tuning:**

**Text

Description automatically generated**

**Valg av tuning strategi:**

Her testet jeg ut både min mindre sekvensielle metode, men også random search. Begge metodene sliter med at max\_depth drar seg høyt opp og fører til en veldig over-fitted model, selv etter tuning.

**Etter tuning (Mindre Sekvensiell):**

Text

Description automatically generated

**Etter tuning (Random Search):**

Text

Description automatically generated

**Diskusjon av resultater:**

Baseline modellen var en av de aller mest over-fitted jeg har sett noen gang, med hele 10%. Den hadde fremdeles ganske bra treffsikkerhet på test dataen, samt en ganske bra R2 score.

I modellene etter tuning klarte jeg å redusere over-fittingen noe ved å primært senke trening scoren. Ett problem som oppstod her, er at grid searchet ville bare ha høyere og høyere max\_depth. Dette fører vanligvis til over-fitting da man tillater treet å utvikle veldig lange grener som spesialiserer seg ekstremt på treningsdataen.

I dette datasettet slet jeg med å øke generell score fra baseline modellen.

**Support Vector Machine:**

For SVR måtte jeg skalere dataen. For dette brukte jeg RobustScaler() funksjonen. Utenom dette trengte jeg ikke å gjøre noe mer

**Før tuning:**

Text

Description automatically generated

**Etter tuning:**

Text

Description automatically generated

**Diskusjon av resultater:**

Baseline modellen presterte relativt dårlig i forhold til de andre baseline modellene, men hadde relativt lite over-fitting.

Her klarte jeg å skvise ut cirka 20% ekstra fra modellen etter tuningen. Dette er nok det beste jeg har fått til noen gang. Modellen ble noe mer over-fitted, men sier meg fremdeles fornøyd med resultatet.

**Sammenligning av alle resultater:**

Alle modellene endte opp med å gjøre det bra etter tuningen. På dette datasettet viste det seg at det er en tre-basert regresjon som var det beste. Jeg klarte å få den generelt beste treffsikkerheten med XGBoost tunet med den første, mest sekvensielle strategien. Som nevnt ovenfor, kommer dette mest sannsynlig av ren flaks, da den andre strategien, som er hakke mindre sekvensiell, burde gjøre det bedre enn den første.