Dataset: Heart

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 har høy sannsynlighet for hjerte infarkt eller ikke, basert på disse attributtene: Age: age of patient, Exng: exercised induced angina, Ca: number of major vessels, Cp: Chest Pain type, Trtbps: resting blood pressure, Chol: cholestoral, Fbs: fasting blood sugar, Rest\_ecg: resting electrocardiographic results, Thalach: max heart rate achieved, Target: 0 = lower chance of heart attack and 1 = higher chance of heart attack.

Datasettet har ingen feil. For mer grundig gjennomgang av datasettet, se EDA i Oblig 2 mappen.

**XGBoost:**

For Heart datasettet trengte algoritmen rundt regnet ingen databehandling. Det eneste jeg valgte å gjøre var å one-hot encode alle kategoriske kolonner.

**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, 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 testet jeg ut 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. Random search blir også tunet mtp f1-score.

For dette datasettet og denne algoritmen presterte min sekvensielle metode best og har derfor kun med bilder fra den i dette dokumentet.

**Etter tuning:**

**Text

Description automatically generated**

**Diskusjon av resultater:**

Før tuning var modellen ekstremt over-fitted; den presterte altså mye bedre på trenings data enn hva den gjorde på test data, men resultatene var fremdeles generelt bra.

Etter tuning klarte jeg å få ned over-fittingen, ved at trenings resultatet gikk ned, men test resultatet gikk opp. Cross validation scoren forbedret litt, med en oppgang på rundt 0.5%. f1 scoren gikk også opp. Jeg kan dermed trygt konludere med at modellen var bedre etter tuning enn før.

**Random Forest:**

For dette datasettet fikk jeg den nøyaktig samme databehandlingen som ved XGBoost.

**Før tuning:**

**Text

Description automatically generated**

**Valg av tuning strategi:**

Her testet jeg ut to strategier: Grid search, men noe sekvensielt og Random search. Begge strategiene presterte likt og kalrte å øke scorene. For dette datasettet og denne algoritmen klarte Random Search å prestere best, jeg tar derfor kun med bilder fra denne strategien.

**Etter tuning:**

**Text

Description automatically generated**

**Diskusjon av resultater:**

Baseline modellen er svært over-fitted, men presterer generelt bedre enn baseline XGBoost modellen.

Etter tuning så presterte modellen bedre på test data enn trenings data. Dette er normalt sett et tegn på at noe ikke er som det bør være. Jeg har prøvd å fikse det, men uten hell. Cross validation scoren synker, men får en bra økning i alle andre scorer.

**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. I dette datasettet var det 5 kontinuerlige kolonner: age, trtbps, chol, oldpeak og thalachh. Jeg lagde bins for disse

**Før tuning:**

**Valg av tuning strategi:**

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

**Etter tuning:**

**Diskusjon av resultater:**

**Support Vector Machine:**

For denne algoritmen måtte jeg skalere dataen. For dette brukte jeg RobustScaler() funksjonen. Resultatet blir betydelig bedre etter. Jeg tok også å one-hot encodet alle kategoriske kolonner

**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:**

**Graphical user interface, text

Description automatically generated**

**Diskusjon av resultater:**

Baseline modellen presterte ekstremt bra, men hadde noe over-fitting.

Etter tuning ble det 0.5% mindre over-fitting, men fikk generelt dårligere prestasjon på alle scorer unntatt cross validation score.

**Neural Network Architecture 1:**

For begge neurale nettverkene å one-hot encode alle kategoriske variabler og deretter gjøre alle features om til tensors

**Før tuning:**

**Text

Description automatically generated**

**Etter tuning:**

**?????**

**Diskusjon av resultater:**

Modellen presterte veldig bra, både på trening og test data, med tilnærmet null over-fitting. Modellen presterer bedre enn XGBoost og RF på test data, men noen prosenter dårligere på trening

**Neural Network Architecture 2:**

Brukte det samme datasettet som i forrige neurale nettverk

**Før tuning:**

**Text

Description automatically generated**

**Etter tuning:**

**?????**

**Diskusjon av resultater:**

Dette neurale nettverket presterte helt likt som det forrige, men gjorde så med færre lag.

**Sammenligning av alle resultater:**

Alle modellene gjorde det ganske bra, men for dette datasettet tok SVM en soleklar seier, både før og etter tuning. Jeg tror grunnen til dette er at SVM sies å prestere bedre enn andre algoritmer på små datasett, noe dette er.