Machine Learning task 2 - Playing the whole game

Sigvart Hatlestad / ML gruppe 29 / 18 nov 22

**Beskrivelse**

I en verden der filmer tjente anslagsvis 41,7 milliarder dollar i 2018. Men hvilke filmer ved får inn mest inntekter på billettkjøp? Hvor mye betyr en regissør? Eller budsjettet?

Her blir du presentert data om over 7000 tidligere filmer fra The Movie Database for å prøve å forutsi den samlede billettinntekten. Punkter som er oppgitt inkluderer rollebesetning, mannskap, plotnøkkelord, budsjett, plakater, utgivelsesdatoer, språk, produksjonsselskaper og land.

I notatboken vil vi være med på å forutsi filmens samlede billettinntekter.

I dette prosjektet blir det laget et maskinlærings-program som forutser hvor mye en film

kommer til å tjene på Box Office basert på data som inneholder informasjon om filmer som

er hentet fra TMDB.

Hva er det som gjør at filmer tjener mest? Klarer et

maskinlærings-program å forutse hvor mye en film kommer til å tjene bedre enn mennesket

kun basert på informasjonen om filmen? Dersom programmet klarer dette, kan det være

mulig for filmselskap og produsenter å bruke et slikt program for å avgjøre om en film burde

bli laget eller ikke. De kan også finne ut eventuelle endringer de burde gjøre med

filmprosjektet for at filmen skal gjøre det bedre ved slipp.

# **BESKRIV PROBLEMET**

## **SCOPE**

I dette prosjektet blir det laget et maskinlærings-program som forutser hvor mye en film

kommer til å tjene på Box Office basert på data som inneholder informasjon om filmer som

er hentet fra TMDB.

Hva er grunnen til at noen filmer tjener mest? Klarer en maskinlærings modell å forutse bedre en det mennesket klarer kun basert på informasjonen om filmen? Åltså hvor en film kommer til å tjene.

Om dette er tilfellet, kan det nok være at produsenter eller filmselskaper bruker denne modellen til å finne ut om filmen burde være laget. Eventuelle endringer på filmprosjekt kan også brukes i modellen

En tentativ tidslinje for prosjektet:

1. Finne ut et prosjekt som jeg vil gjøre
2. Lese og finne ut av prosjektet. Endte til til slutt opp med å bruke dataene til «TMDB Box Office Prediction» fra Kaggle. Leser informasjonen der og går også gjennom noen av prosjektene som andre har gjort. Finner også prosjekt jeg har gjort før.
3. Lager til denne rapporten. Opprette en rapport over prosjektet og skriver på denne underveis gjennom hele prosjektet
4. Oppretter notebook i jupyter som jeg henter ut dataen i prosjektet
5. Ser på dataen. Finner hvillken attributt de har og utforsker de. Finner noen har rekker som mangler data osv. Fjerner disse
6. Oppretter en notebook i jupyter og henter dataen for prosjektet. Setter også opp grafer for å vise sammenhenger.
7. Legger til data på attributt som har få rekker som mangler data. Brukte den hyppigste verdien til attributtet.
8. Fikser strenger. Noen kolonner er lister av strenger. Har kun lyst til å vise frem name. Lager derfor en funksjon som sorterer strengene som gjør at kun det vesentlige vises, som i dette tilfellet er "name".
9. Release date. Denne funksjonen er hentet fra Kaggle, fra en som hadde gjort prosjektet tidligere. Funskjonen hjelper med å sortere "release\_date" til å kun få ut året for utgivning. Enda en snarvei som er tatt.
10. Fyller inn missing values. Maskinlæringsalgoritmer kan ikke kjøre med manglende verdier. Tar hver kolonne som har "NaN" verdier og fyller disse med "None", slik at den på sikt kan bli kjørt gjennom en Pipeline.
11. Tar hånd om flere attributes. De fleste maskinlæringsalgoritmer foretrekker at alle attributtene er numeriske. Siden 'genres', 'original\_language', 'production\_companies', 'production\_countries', 'release\_year', 'spoken\_languages', 'Keywords', og 'cast' er teksbasert. Dette blir gjort numerisk.
12. Tranformer pipelines. Tidligere i prosjektet har det blitt vist manuelt hvordan man behandler missing values, tekst og kategoriske attributter i tillegg til å skalere ned verdier. Det meste av dette kan bli gjort ved hjelp av Pipelines.
13. Utforsker forskjellige modeller og finner den beste. Ved å bruke grid search
14. Velger den modellen som gir best resultat.
15. Bruker testsettet på modellen og laster opp resultatet på konkurransen på kaggle.
16. Forsøk på deployment via flask og heroku.

## **METRIKKER**

Det bør være minimum at modellen å gå over til nåværende metoder som vil oppnå det samme.

R-Square bør komme nærmest mulig 100 %. Viss den er lav, kan det føre til at brukeren ikke stoler på resultatet. Og heller ikke ta en avgjørelse utifra dette.

Det er lettere desto nærmere den er 100%. Det er ytelsen på modellen

Et minimumskrav for modellen bør være å gå over ytelsen til nåværende metoder som

oppnår det samme.

# **DATA**

Det er nå laget en webapplikasjon med de featurene jeg synest var viktigst. Denne modellen vil ikke helt enda kan konkurrere med den manuelle vurderingen, men som kan med jevnlig trening bli mer presis.

Hadde datasettet jevnlig fått inn nye data på, måtte tallene blitt overvåket slik at man trener modellen til å bli best mulig. En konsekvens av en så enkel modell vi har publisert er at modellen kan bli underfitted, men det er trodd at disse featuresene vil gjøre risikoen mindre for dette.

En løsning for dette ville selvfølgelig vært å tilrettelegge for å kunne legge inn flere og bedre features. Dette vil øke risikoen for å få en overfitted modell som gjør det bra på nåværende sett, men dårlig på nye data. Det er derfor viktig å overvåke modellen - overvåkning er også viktig for å forhindre at man får inn vilkårlige/gale verdier som i sin helhet kan ha stor innvirkning på resultatet.

Over tid kan for eksempel en faktor påvirke revenue mer enn det den gjør i dag. Etter noe tid må man da gå gjennom settet og kanskje legge til flere features, eller forandre verdien på features man ikke har i webapplikasjonen (medianverdier forandrer seg gjerne etter noe tid). Modeller har en tendens til å "råtne" over tid dersom de ikke på jevnlig basis trenes på nye data.

# **MODELLERING**

Hovedfokuset med dette maskinlæringsprosjektet har vært på deploymentdelen. Derfor har det blit tatt noen "snarveier" på veien til en ferdig modell. Blant annet er det droppet et par kolonner, og i tillegg gjort endringer på dataen i noen kolonner. En konsekvens av disse "snarveiene" er funksjonen som fikser strenger tar kun med én verdi. Et eksempel er "genres" kolonnen kan filmer ha flere sjangre. Men funksjonen som fikser strenger beholder kun én sjanger.

I dette maskinlæringsprosjektet har lært hvordan en setter maskinlæringsmodeller i drift. I punkt fem er det undersøkt ulike modeller for prosjektet. Ut fra kryssvalidering fikk to modeller ganske god score. Modellen som blei valgt var Decision Tree Regressor, ettersom den fungerte best.

I deploymentdelen ble det valgt input på kun fire "features": budget, release\_year, cast og genres. Grunnen var at disse fire har størst sannsynlighet for å påvirke inntekten. Alle de andre "features" får medianen som verdi dersom de er numeriske, og den hyppigste dersom de er kategoriske. Siden de fleste verdiene blir satt til bestemte verdier "bak kulissene" er dette ikke den mest treffsikre modellen. Den har noen begrensinger.

# **DEPLOYMENT**

*Hvordan skal modellen(e) settes i drift? Hvordan skal prediksjonene brukes? Hva er dine planer for monitorering og vedlikehold av maskinlæringssystemet? Hvis relevant, hvilke planer har du for å forbedre systemet etter at det er satt i drift?*

Uansett hva prøvde på greidde jeg ikke å deploy ved hjelp av forskjellige verktøy. Men

# **REFERANSER**

##### *Link til prosjektet vi henter koden fra:*[*https://www.kaggle.com/artgor/eda-feature-engineering-and-model-interpretation?fbclid=IwAR3wZWesOpiIi88vXgq5QzQnQVdB8lkbMDF-F3tWp2QwCVcfbIIJBlQH-1U*](https://www.kaggle.com/artgor/eda-feature-engineering-and-model-interpretation?fbclid=IwAR3wZWesOpiIi88vXgq5QzQnQVdB8lkbMDF-F3tWp2QwCVcfbIIJBlQH-1U)