CAR-ML

Bilal Saher & Sigurd Kløgetvedt, 25.10.2025

OMFANG / *SCOPE*

Formålet med prosjekt CAR-ML er å utvikle en maskinlæringsbasert webapplikasjon som estimerer markedsverdien av brukte biler basert på tekniske og visuelle egenskaper. Dette er f.eks hvilket merke, modell, årsmodell, kilometeravstand, drivstofftype og motorstørrelse som bilen har. I dag foretas slike prisvurderinger ofte gjennom private meglere, eller kan det være opp til den enkelte bileier å estimere et prisanslag. Bileier kan ta i bruk forenklede bilkalkulatorer som baserer seg på faste prisintervaller. Eller de kan sjekke nettsteder som finn.no eller autotrader for sammenligninger. Problemet er at disse metodene tar sjeldent hensyn til et stort antall bilspesifikke faktorer samtidig og oppdateres ikke automatisk av markedets endringer.
Ved å bruke maskinlæring kan vi analysere historisk data for titusenvis av biler og finne mønstre som gir mer presise og datadrevne estimater. En slik løsning ville kunne gitt:

- Privatpersoner et realistisk anslag på salgspris før publisering på bruktbilmarkedet.
- Støtte bilforhandlere til å prise brukbiler korrekt.
- Redusere risikoen for feilprising og bidra til raskere omsetning.

Business Objective

Målet er å tilby et verktøy som kan være med på å redusere prisavviket mot markedspris og gir brukeren et realistisk estimat som man kan stole på. Basert på målingene våre tenker vi en realistisk ambisjon for første versjon å stabilisere RMSLE <= 0,50, med videre forbedringspotensial ettersom vi vil få inn mer data og modellen itereres.
Forretningsmessig forventer vi:

- Økt treffsikkerhet i prissetting (mindre avvik fra markedspirs)
- Raskere salgstid på bruktbiler
- Økt kundetillit hos aktører som bruker verktøyet

Bruk og drift

Løsningen vil kunne driftes som en åpen nettside eller integreres i et eksisterende bilsalgsnettsted. Brukere kan legge inn bilens detaljer og få umiddelbart prisestimat På sikt kan systemet trenes videre etter hvert som nye data om bilsalg samlet inn.

Ressurser

- Beregningsressurser: PC med Jupyter Notebook og Python miljø (scikit-learn, pandas, numpy).
- Personell: To utviklerer (Bilal Saher og Sigurd Kløgetvedt)
- Datagrunnlag: Eksisterende bruktbil-datasett (ca. 188 000 obervasjoner)

Metrikker

For å måle modellens ytelse bruker vi Root Mean Squared Logarithmic Error (RMSLE).
Denne metrikken er godt egnet fordi den straffer store relative feil mer enn små. Dvs. en 10 000 kr feil på en bil til 50 000 kr bil teller mer enn samme feil på en bil til 500 000 kr. Metoden reduserer også effekten av ekstreme priser ved log-transformasjon.

- Baseline-modeller (lineær regresjon/beslutningstre) gir RMSLE ca. 1.60-0.70.
- Gjeldende beste modell (Random Forest) gir RMSLE til omtrent 0.50 - 0.55. Ambisjonene våre er å stabilisere RMSLE til < 0.50, noe som er realistisk når vi får mer datagrunnlag å bygge modellen videre på. Dette er i tråd med at RMSLE måler relative feil, demper ekstreme verdier og er mer meningsfull enn MSE/MAE for prisestimering på et hetrogent bruktbilmarked.

Som forretningsmodell sikter vi oss mot prisavvik under 20 % mot faktiske markedspris. Vi ønsker også at minst 80 % av brukerne skal oppleve at estimatet stemmer overens med deres egen forventning.

Datagrunnlag og representasjon

Prosjektet bygger på et bruktbil-datasett bestående av 188 000 observasjoner hentet fra et åpent bilmarked-datasett (tilsvarende Kaggle sitt "Used Cars Dataset"). Hver rad representerer en bil, og inneholder informasjon om blant annet merke, modell, årsmodell, kilometeravstand og drivstofftype, motor/girkasse, farger, ulykkehistorikk og om bilen har en "clean-title" (Stjålet..etc). Bilens pris er målet i amerikanske dollar (USD).

Datasettet består av både numeriske og kategoriske variabler, som krever forskjellig behandling i preprosesseringsfasen. Eks på variabler:

Type	Kolonner	Eksempel på verdier
Numeriske	model_year, milage, price	2017, 67 000, 45 000
Kategoriske	brand, fuel_type, transmission, clean_title	BMW, Gasoline, A/T, Yes
Tekstlige beskrivelser	engine, ext_col, int_col, accident	"2.0L Cylinder Engine", "Black, "None reported"

Vi lagrer også hjelpedata for UI i model/categories.json. Her har vi *choices* som er fullstendige valglister per kategori, kan være f.eks merker, farge og drivstoff. I tillegg har vi *brand_models* som gir en dropdown meny over bilmerker og og tilhørende dropdown meny for hvilke bilmodeller dette merkene har.

Datakvalitet og rensing

Ved første inspeksjon ble det identifisert manglende verdier i tre sentrale kolonner:

- "clean_title"- 21 4019 manglende verdier
- "fuel_type"- 5 083 manglende verdier
- "accident"- 2452 manglende verdier

Disse ble fylt med verdier "Unknown" for å bevare observasjoner uten å fjerne data. For å unngå skjevhet fra ekstreme priser ble alle biler med pris over 1 million fjernet.
Videre ble det lagt til en ny feature car_age, beregnet som: $car_{age} = 2025 - model\_year$
Dette gir en mer direkte og tolkbar sammenheng mellom bilens alder og pris.

Ytterligere tiltak vi gjorde for å forstå dataen:

- Visualisering av missing values via heatmap for å forstå datakvalitet.
- Undersøkte prisfordeling og kilometeravstand ved scatterplots for å oppdage outliers.
- Opprette en log-skalert visualisering av pris for å håndtere sterk skjevhet i data.

Personvern og etikk

Datasettet inneholder ingen personidentifiserbare informasjon. Hver observasjon beskriver kun tekniske spesifikasjoner for biler og kan brukes fritt til analyse og modelltrening.
Etiske hensyn knyttet til bruk av modellen kan imidlertid være relevant dersom systemet brukes kommersielt. F.eks kan feilaktig prisestimering påvirke brukerens økonomiske beslutninger. For å redusere slik risiko vises modellen predikasjon som et estimat med usikkerhetsmarginer, ikke som en garantert verdi.

Modellering

Modellvalg og baseline

Vi evaluerte tre modeller med 5-fold kryssvalidering:

- Lineær regresjon (baseline): RMSLE gir 1.60
- Beslutningstre: RMSLE gir 0.70
- Random Forest: RMSLE gir 0.50 - 0.57

Ensemblemetoder gir best ytelse når det er mange kategoriske variabler og ikke linearitet og noe tekstsstøy.

Feature-engineering og representasjon

- Numerisk: model_year, milage og avledet car_age = 2025 - model_year
- Kategorisk: one-hot-koding (drop_first=True) for oppgitte felt.
- Outliers: priser > 1 000 000 ble fjernet
- Manglende verdier i fuel_type, accident og clean_title ble satt til “Uknown” for å bevare datamengden
- “Id” ekskluderes som feature.

Skalering

Siden Random Forest-modellen er i praksis ganske lite følsom overfor forskjeller i variablerskala. Vi valgte derfor å trene sluttmodellen uten standardisering. Dette gjør at det blir enklere i drift og fullt samsvar mellom trening og frontend, uten noe behov for egen StandarScaler. Om fremtidige modeller skulle kreve skalering, f.eks. lineære metoder, kan dette enkelt legges til senere.
(Breiman, 2001; Pedregosa mfl., 2011)

Hyperparametersøk

For å optimalisere ytelsen kombinerte vi RandomiceCV for grove søk og GridSearchCV for finjusteringer av søket. De mest sentrale parameterne var antall trær, dybde, antall features per split og minstekravet for splitting/bladstørrelse. Den beste modellen oppnådde stabile resultater rundt: max_depth ≈ 22, max_features ≈ 0.3, min_samples_split ≈ 20, min_samples_leaf ≈ 3, n_estimators ≈ 220–250. Dette balanserer en god generalisering og lav overtilpasning.
(Bergstra & Bengio, 2012).

Feilanalyse og forklarbarhet

Residualer ble analysert mot bilader og kilometeravstand, samt per bilsegment(elbil, SUV, pickup). For å forstå hvilke egenskaper som påvirket prediksjonen mest, brukte vi permutasjon-importance og partial dependence-plottet. På sikt ønsker vi å implementere SHAP-verdier i brukergrensesnittet for å forklare enkeltpedisjoner og bygge tillit. (Friedman, 2001; Lundberg & Lee, 2017; Molnar, 2022)

Målsetting

Første versjon skal stabilisere RMSLE <= 0,55, redusere skjevheter mellom bilsegmenter og forbedre forklarbarheten for sluttbrukeren.

Deployment

Løsningen består av et modellartefekt (randoom_forest_final.pky) og kolonnedefinisjoner (feature_columns.json) for korrekt reindeksering ved predikasjon. Frontend er bygget i Gradio med kaskerende nedtrykksmenyer (merke -> modell) og valg fra categories.json. Brukerens input konverterer til et datarammeverk, deretter one-hot-kode og reindekseres slik at det samsvarer med treningskolonnene før prediksjon.

For konsistens brukes eksakte kategoriverdier uten automatisk endring av store/små bokstaver. Ukjente verdier mappes til “Unknown”, en kategori modellen allerede kjenner. Avledet felt som car_age = 2025 - model_year beregnes direkte på forhånd.
Datasettet er i USD, men appen viser det i NOK ved å multiplisere med en fast valutafaktor lik (11 NOK/USD). Dette gir et realistisk estimat for norske forhold.

Videre arbeid vil fokusere på usikkerhetsintervall i UI, databerikelse og bedre skalerbarhet gjennom lokale bidragsforklaringer.

Referanser

Breiman, L. (2001). Random Forests. *Machine Learning, 45*(1), 5–32. Springer. https://doi.org/10.1023/A:1010933404324

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830. https://jmlr.org/papers/v12/pedregosa11a.html

Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research, 13*, 281–305. https://jmlr.org/papers/v13/bergstra12a.html

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics, 29*(5), 1189–1232. https://projecteuclid.org/journals/annals-of-statistics/volume-29/issue-5/Greedy-function-approximation-A-gradient-boosting-machine/10.1214/aos/1013203451.full

Lundberg, S. M., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems (NeurIPS)*. https://arxiv.org/abs/1705.07874

Gradio (2024). Gradio Documentation – Build Machine Learning Web Apps in Python. *Gradio.app*. https://www.gradio.app/

Project Jupyter (2024). Jupyter Notebook Documentation. *jupyter.org*. https://jupyter.org/