Klassificering av handskrivna siffror

En utvärdering av ensembelmetoder

Therese Nord

EC Utbildning

Machine Learning

Kunskapskontroll 2

202503

Abstract

This report compares the performance of two popular classification models, Random Forest and XGBoost, in predicting handwritten digits from the MNIST dataset. These models, also called ensemble methods, combine multiple decision trees to make more accurate prediction. The primary goal was to evaluate the models based on accuracy, precision, and recall determining the most effective model for this task. Both models performed similarly, with only a minimal difference in overall accuracy, with XGBoost being slightly more accurate. However, due to its faster training time, lower memory usage, and simpler workflow, Random Forest was chosen as the optimal model for this project. Future work may involve implementing an interactive Streamlit application to visualize and interact with the trained model in real-time.

Innehållsförteckning

Inledning

Maskininlärning är ett kraftfullt verktyg inom artificiell intelligens som gör det möjligt för datorer att lära sig från data och göra förutsägelser eller beslut utan att vara explicit programmerade. Tekniken används idag inom många områden, som bildigenkänning, medicinsk diagnostik och finans, för att göra automatiserade bedömningar och förutsägelser baserade på historiska data. Bland de olika typerna av maskininlärningsmodeller är klassificeringsmodeller särskilt viktiga då de används för att förutsäga vilken kategori en observation tillhör, vilket är avgörande i tillämpningar som exempelvis att identifiera handskrivna siffror, spåra kundbeteenden eller diagnostisera sjukdomar.

I denna rapport fokuserar jag på att jämföra två populära ensemblemetoder för klassificering, Random Forest och XGBoost, för att bedöma vilken modell som presterar bäst när det gäller att prediktera handskrivna siffror i ett klassificeringsproblem. Målet är att undersöka vilken av dessa modeller som ger den bästa prestandan baserat på mätbara mått som accuracy, precision och recall. Genom att göra detta hoppas jag kunna identifiera den mest lämpliga modellen för att korrekt identifiera handskrivna siffror.

Syfte och frågeställning

Syftet med denna rapport är att jämföra prestandan hos Random Forest och XGBoost när det gäller att prediktera handskrivna siffror, samt att undersöka vilken modell som ger bäst resultat i termer av klassificeringseffektivitet. För att uppnå detta syfte kommer följande frågeställningar att besvaras:

1. Vilken modell ger bäst prestanda i termer av accuracy, precision och recall när det gäller att prediktera handskrivna siffror?
2. Vilka faktorer, såsom exekveringstid och användarvänlighet, påverkar valet av modell för denna specifika uppgift?

Teori

Maskininlärning

Maskininlärning är en gren inom artificiell intelligens där datorer tränas att identifiera mönster och fatta beslut baserat på data. Det finns olika typer av maskininlärning, där *supervised learning* (övervakad inlärning) är en av de vanligaste. I övervakad inlärning tränas en modell på en dataset som innehåller både indata (features) och korrekta utfall (labels), vilket gör det möjligt att prediktera resultat på ny, osedd data.

Algoritmer för maskininlärning

Klassificeringsmodeller

Klassificeringsmodeller används för att förutsäga vilken kategori en observation tillhör. Detta innebär att modellen tilldelar varje observation till en av flera fördefinierade klasser. Klassificering kan vara binär, där observationen tillhör en av två möjliga klasser, exempelvis "JA/NEJ" eller "spam/inte spam", eller multiklass, där observationen tillhör en av flera olika klasser, exemlepvis att kategorisera blommor som olika arter.

Modellen skapar sannolikheter för varje klass och tilldelar observationen den mest sannolika kategorin. Exempel på tillämpningar är att förutsäga kundchurn, överlevnad för patienter eller om ett e-postmeddelande är spam.

För att utvärdera modeller används *Confusion Matrix* (se avsnitt Resultat och Diskussion), som visar antalet korrekta och felaktiga prediktioner, uppdelat i fyra kategorier: True Positives, False Positives, True Negatives och False Negatives. Mer om det under *utvärderingsmått för klassificeringsmodeller.*

Sammanfattningsvis används alltså klassificeringsmodeller för att kategorisera data och göra förutsägelser om nya observationer.

Random Forest

Random Forest är en ensemblemetod baserad på beslutsträd, där flera träd tränas på olika delar av datan och deras prediktioner kombineras genom majoritetsbeslut. Fördelen med Random Forest är att den minskar risken för överanpassning (*overfitting*) genom att skapa många oberoende träd. Modellen är robust och fungerar bra även utan omfattande hyperparameterjusteringar.

XGBoost

XGBoost (*Extreme Gradient Boosting*) är en avancerad gradient boosting-algoritm som bygger beslutsträd sekventiellt, där varje nytt träd korrigerar felen från det föregående. Den är känd för sin höga prestanda och används ofta för dess effektivitet och förmåga att hantera komplexa dataset. Till skillnad från Random Forest, som bygger träd oberoende av varandra, tränar XGBoost varje träd beroende på tidigare misstag, vilket gör det mer optimerat men också mer känsligt för överanpassning om det inte justeras korrekt.

Utvärderingsmått för klassificeringsmodeller

Precision, Recall och F1-score är tre viktiga mått inom klassificeringsmodeller för att utvärdera deras prestanda. De används särskilt vid obalanserade dataset där accuracy inte alltid ger en rättvis bild av modellens prestation.

Precision

Precision visar hur många av de positiva förutsägelserna som faktiskt var korrekta. Precision mäter andelen korrekt identifierade positiva fall av alla fall som modellen klassificerade som positiva. Ett högt precisionstal innebär att modellen sällan gör falska positiva förutsägelser.

$$precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Figur 1. Illustration av formel för precision (Inside Machine Learning, 2 september 2021).

Recall

Recall visar hur stor andel av de verkliga positiva fallen hittades. Recall mäter hur bra modellen är på att hitta alla verkliga positiva fall. Ett högt recall-värde innebär att modellen missar få positiva exempel men kan också innebära fler falska positiva.

$$recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Figur 2. Illustration av formel för recall (Inside Machine Learning, 2 september 2021).

F1-Score

F1-score är en kombination av precision och recall, där båda måtten vägs samman för att ge en balanserad bedömning av en modells prestanda. Det beräknas genom en formel som säkerställer att både precision och recall tas med i beräkningen samtidigt, och värdet blir lågt om någon av dem är låg. F1-score används ofta när det är viktigt att hitta en bra kompromiss mellan att fånga positiva exempel och att undvika falska positiva, särskilt vid obalanserade dataset.

$$F1\ Score = 2 \times \frac{recall \times precision}{recall + precision}$$

Figur 3. Illustration av formel för F1-score (Inside Machine Learning, 2 september 2021).

Metod

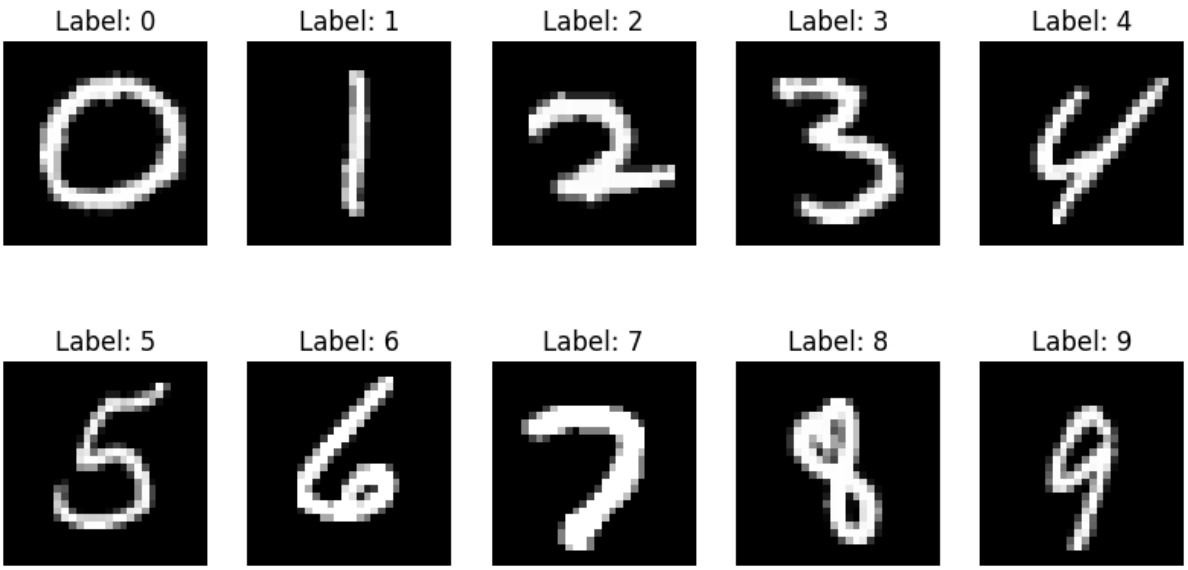
Genomförande av arbetet

Datan har erhållits från vår lärares GitHub, där jag följde den tillhandahållna instruktionen för att ladda ner den (Prgomet, 2025). Jag importerade datan till min arbetsmiljö i Visual Studio Code (VS Code). Efter att ha fått tillgång till datan har jag följt ett Machine Learning-flöde, som beskrivs i boken *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (Géron, 2019, s. 755). Genom att tillämpa dessa steg har jag systematiskt arbetat igenom uppgiften och implementerat de olika momenten för att lösa problemet.

Databeskrivning

Datasetet MNIST som används i denna studie består av handskrivna siffror från 0 till 9, och varje bild representerar en siffra i olika stil och form. Nedan förljer exempelbilder från datasetet för varje siffra för att ge en bild av variationen inom datan.

MNIST-datasetet innehåller totalt 70 000 bilder, uppdelade i 60 000 träningsbilder och 10 000 testbilder. Varje bild är 28x28 pixlar och innehåller gråskalevärden som uppstått genom anti-aliasing vid normaliseringen. Datasetet är en bearbetad version av en större samling från NIST, där siffrorna har storleksnormaliserats och centrerats för att förbättra klassificeringsprestanda. Bilderna i datasetet är även skrivna av olika personer, vilket ger en naturlig variation i handstilar.



Bibliotek

För att genomföra analysen och träna modellerna använde jag flera populära Python-bibliotek inom databehandling, maskininlärning och visualisering:

NumPy och *Pandas* användes för att hantera och manipulera data. NumPy underlättar beräkningar med matriser och vektorer, medan Pandas ger kraftfulla verktyg för att arbeta med data i tabellformat.

Joblib användes för att spara och ladda tränade modeller, vilket möjliggör effektiv återanvändning av modeller utan att behöva träna om dem från början. *Scikit-learn* användes för att hantera olika steg i maskininlärningsprocessen, inklusive:

- fetch_openml för att hämta dataset
- train_test_split för att dela upp data i träning och test
- cross_val_score för korsvalidering och utvärdering av modellernas prestanda
- accuracy_score, confusion_matrix, classification_report och ConfusionMatrixDisplay också för att utvärdera modellernas prestanda
- RandomForestClassifier och RandomizedSearchCV för att träna modeller och optimera hyperparametrar

Matplotlib användes för att visualisera data och prestandamått, inklusive confusion matrix.

XGBoost användes för att implementera och träna XGBoost-modellen, en populär och kraftfull gradient boosting-algoritm.

Dessa bibliotek valdes eftersom de är etablerade inom maskininlärning och erbjuder effektiva verktyg för både dataförberedelse, modellträning och utvärdering.

Resultat och Diskussion

Modellernas prestation

Som tidigare nämnts (se avsnitt Teori), används precision, recall och F1-score för att utvärdera modellernas prestanda. Nedan följer resultaten för Random Forest och XGBoost.

Random Forest

Class	Precision	Recall	F1-Score	Support
0	0.98	0.99	0.98	1343
1	0.98	0.99	0.99	1600
2	0.95	0.97	0.96	1380
3	0.96	0.95	0.96	1433
4	0.97	0.97	0.97	1295
5	0.97	0.96	0.97	1273
6	0.98	0.99	0.98	1396
7	0.97	0.97	0.97	1503
8	0.96	0.96	0.96	1357
9	0.95	0.95	0.95	1420
Accuracy	**0.97**			**14000**
Macro avg	**0.97**	**0.97**	**0.97**	**14000**
Weighted avg	**0.97**	**0.97**	**0.97**	**14000**
Overall Accuracy				**0.9681**

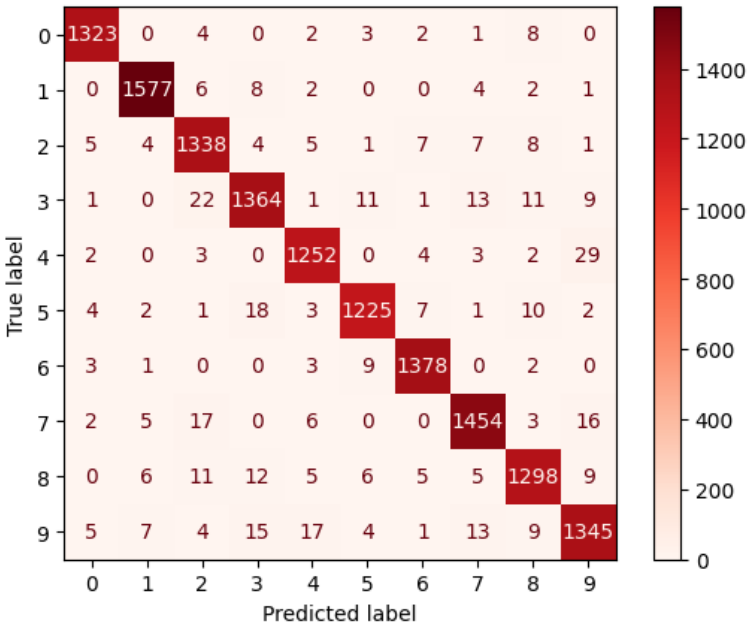
XGBoost

Class	Precision	Recall	F1-Score	Support
0	0.99	0.99	0.99	1343
1	0.98	0.98	0.98	1600
2	0.96	0.97	0.96	1380
3	0.97	0.95	0.96	1433
4	0.97	0.97	0.97	1295

Class	Precision	Recall	F1-Score	Support
5	0.97	0.97	0.97	1273
6	0.98	0.98	0.98	1396
7	0.97	0.97	0.97	1503
8	0.96	0.96	0.96	1357
9	0.95	0.95	0.95	1420
Accuracy	**0.97**			**14000**
Macro avg	**0.97**	**0.97**	**0.97**	**14000**
Weighted avg	**0.97**	**0.97**	**0.97**	**14000**
Overall Accuracy				**0.9696**

Confusion Matrix

För att utvärdera modellernas prestanda har en Confusion Matrix använts. Denna tabell ger en översikt över hur väl varje modell har klassificerat de olika siffrorna, genom att visa antalet rätta och felaktiga prediktioner för varje klass. Nedan visas Confusion Matrix för Random Forest, vilket illustrerar fördelningen av rätt och fel klassificerade siffror.



Diskussion

Val av modell

Efter att ha utvärderat både Random Forest och XGBoost och konstaterat att de gav liknande resultat när det gäller accuracy, valde jag att gå vidare med Random Forest. Här är de främsta faktorerna som vägde in i mitt beslut:

Effektivare lagring

Random Forest kräver generellt sett mindre lagringsutrymme än XGBoost, särskilt när det gäller att hantera stora datamängder och många träd i skogen. XGBoost använder en mer komplex algoritm, vilket gör att den ofta kräver mer minne och lagringsutrymme för att hantera alla parametrar. Random Forest bygger träd oberoende av varandra, vilket gör den mer minnesvänlig.

Snabbare exekvering

Random Forest är också snabbare än XGBoost både vid träning och prediktion. XGBoost kräver mer finjustering av hyperparametrar och kan vara långsammare på stora dataset, då varje steg optimeras individuellt. Random Forest tränar parallellt och oberoende, vilket gör den snabbare och mer effektiv i detta avseende.

Ett enklare arbetsflöde

Random Forest erbjuder ett enklare arbetsflöde, vilket var avgörande för mig. XGBoost kräver mycket finjustering av hyperparametrar, vilket kan vara tidskrävande. Random Forest fungerar bra även utan omfattande justeringar och är därmed mer användarvänlig, särskilt när min dators kapacitet är en begränsning.

Både Random Forest och XGBoost ger nästan identiska resultat, men Random Forest var det bästa valet för mig, då det erbjuder snabbare träning, enklare hantering och mindre krav på datorns kapacitet. För mitt specifika scenario, där snabb implementation och effektivitet var avgörande, var Random Forest det mest praktiska valet. Båda modellerna presterar bra, men Random Forest erbjuder en bättre balans mellan prestanda och effektivitet.

Slutsatser

Syftet med denna rapport var att jämföra prestandan hos de två klassificeringsmodellerna Random Forest och XGBoost för att prediktera handskrivna siffror. Baserat på de observerade resultaten för precision, recall och F1-score var båda modellerna mycket jämförbara och gav nästan identiska resultat, med XGBoost som något bättre i termer av overall accuracy (0,9696 jämfört med 0,9681 för Random Forest).

Efter noggrant övervägande av både prestanda och praktiska faktorer, såsom exekveringstid, lagringsutrymme och användarvänlighet, valde jag att använda Random Forest som den mest lämpliga modellen för mitt klassificeringsproblem. Random Forest visade sig vara mer effektiv när det gäller lagring, snabbare vid både träning och prediktion, och erbjöd ett enklare arbetsflöde, vilket gjorde den till ett mer praktiskt val för detta specifika projekt.

Jag konstaterar också att båda modellerna är väl lämpade för att hantera handskrivna siffror och ger pålitliga resultat. Om mer tid hade funnits, hade jag gärna implementerat en användarvänlig applikation med Streamlit för att visualisera och interagera med den tränade modellen i realtid. Det hade gjort det möjligt att testa få resultat från modellen direkt. Det hade varit roligt att skapa en interaktiv app med Streamlit och jag planerar att göra det efter inlämning av projektet.

I framtida arbete skulle det vara intressant att testa ytterligare optimeringar, som exempelvis hyperparameter-tuning för XGBoost, för att kanske förbättra prestandan något ytterligare. Men i detta scenario, där snabb implementation och effektivitet var centrala faktorer, var Random Forest det bästa valet.

Teoretiska frågor

1. Kalle delar upp sin data i *träning*, *validering* och *test*.” Träning” används för att, precis som ordet redan avslöjar, träna olika modeller. Detta betyder alltså att denna del av datan, som ofta är (men behöver inte vara) omkring 60-70% av datan, tränar olika moedeller på data för att lära sig identifiera mönster, relationer och strukturer i datan. ”Validering” används för att finjustera och justera modellens hyperparametrar, och för att hjälpa till att förhindra överanpassning (overfitting). Valideringsdatan, som oftast är omkring 10-20% av datan, ger en indikation på hur bra modellen presterar på osedd data under träningsfasen, här kan man justera modellens parametrar för att förbättra prestandan innan man går vidare till testfasen. ”Test” är den sista delen av datan, som ofta är omkring 10-20% av datan, som har varit isolerad för att kunna testa den tränade modellen på ny, osedd data och därefter göra en bedömning av hur bra moedllen presterar.

2. Julia delar upp sin data i *träning* och *test*. På träningsdatan så tränar hon tre modeller; ”Linjär Regression”, ”Lasso regression” och en ”Random Forest modell”. Eftersom Julia inte har ett explicip ”valideringsdataset” kan hon använda c*ross validation* för att bedöma modellernas prestanda och identifiera de bästa hyperparametrarna. Efter att hon genomfört cross-validation kan hon justera modellerna och välja den som presterar bäst, enligt exempelvis *accuracy score*.

3. Ett regressionsproblem handlar om att förutsäga ett numreriskt värde baserat på en eller flera egenskaper (*features).* Detta betyder alltså att modellen lär sig hitta samband mellan indata och en kontinuerlig målvariabel (*target label)*. Exempel på modeller för att lösa regressionsproblem är: Linjär Regression, Lasso Regression och Support Vector Regression. Potentiella tillämpningsområden kan vara t.ex. att prediktera huspriser, risk för hjärtinfarkt eller finansiell analys. För att bedömma hur bra en regressionsmodell presterar använder vi RMSE, *Root Mean Square Error*, mer om det i fråga 4.

4.

$$RMSE = \sqrt{\sum_{i} (y_i - \hat{y}_i)^2}$$

Root Mean Square Error är ett mått för att utvärdera regressionsmodeller som gjorts på regressionsproblem och används för att mäta hur mycket fel en modell gör i sina prediktioner. Vi kan tolka RMSE som *våra prediktioners medelavstånd till de sanna värdena*.

Vi får *square error* när vi kvadrerar skillnaden mellan det faktiska y-värdet och det predikterade värdet på y. När vi beräknar medelvärdet på *square error* får vi *mean square error.* För att sen kunna tolka måttet tar vi roten ur MSE för att få samma skala som för y istället för y-kvadrat. När vi här har tagit roten ur, får vi RMSE.

5. Klassificeringsproblem innebär att prediktera vilken kategori en observation tillhör, antingen binär klassificering (JA/NEJ) eller multiklassklassificering (flera klasser). Modellen skattar sannolikheter för varje klass och placerar observationen i den mest sannolika kategorin.

Exempel på tre olika tillämpningsområden för klassificeringsmodeller är att prediktera om en kund kommer ”chruna”, om en patient kommer överleva eller om mejlet du får i din e-post inkorg är spam eller inte spam. Modeller som används för klassificering är exempelvis *Logistisk regression*, *Random Forest* och *Support Vector Machines.*

Confusion Matrix är ett verktyg för att utvärdera klassificeringsmodeller. Det är en tabell som har två dimensioner: den faktiska och den predikterade dimensionen. Den visar hur många prediktioner som var korrekta och felaktiga enligt fyra kategorier: True Positives, False Positives, True Negatives och False Negatives.

6. K-means är en algoritm för klustring, där målet är att dela upp ett dataset i ett angivet antal **k**luser. Algoritmen fungerar genom att slumpmässigt placera centrum för varje kluster (*centroids*), fördela datapunkterna till närmaste centroid och sen uppdatera centroids baserat på medelvädet av punkterna i varje kluster. Det upprepas tills centroids inte rör sig längre.

Exempel där k-means används är i kundsegmentering, bilder och i textbearbetning, eller för att identifiera mönster i stora datamängder. En nackdel är att algoritmen kräver att man specificerar antalet kluster på förhand och att den inte fungerar bra när klustren har olika storlekar eller form.

7. *Ordinal encoding, one-hot encoding* och *dummy variable encoding* används för att omvandla kategoriska variabler till numeriska format som maskininlärningsmodeller kan arbeta med.

Ordinal encoding används för ordnade kategorier och tilldelar varje kategori ett heltal baserat på dess rangordning. Exempelvis för "storlek" kan vi tilldela värdena 0, 1 och 2 för ”liten, medium och stor”.

One-hot encoding används för kategorier utan inbördes ordning. Här skapas en ny binär variabel för varje kategori. Till exempel, för färgerna ”röd, blå, och gul” skulle vi skapa tre kolumner och markera med 1 den kategori som gäller, t.ex. [0, 1, 0] för "blå".

Dummy variable encoding är en variant av one-hot encoding där en kategori tas bort för att undvika multikollinaritet, eftersom en kategori blir överflödig och kan uteslutas. I exemplet med färger kan vi ha "röd" och "blå", och representera "gul" genom att sätta båda variablerna till 0. Detta fungerar eftersom om värdet inte är varken "röd" eller "blå", innebär det att det måste vara "gul", alltså [0, 0] betyder "gul".

8. *Göran påstår att datan antingen är ”ordinal” eller ”nominal”. Julia säger att detta måste tolkas. Hon ger ett exempel med att färger såsom {röd, grön, blå} generellt sett inte har någon inbördes ordning (nominal) men om du har en röd skjorta så är du vackrast på festen (ordinal) – vem har rätt?*

Jag anser att Julia har rätt. Hon har rätt i att röd, grön och blå generellt sett skulle tolkas som *nominala*, eftersom de inte har någon naturlig rangordning. Men det är också viktigt att ta hänsyn till sammanhanget. Om exempelvis en röd skjorta betyder att du är "vackrast på festen", handlar det om en subjektiv bedömning som gör att röd anses vara "mer vacker". I det fallet får färgerna en *ordinal* betydelse, där skjortans färg tolkas som "bättre/mer vacker" och därmed finns en rangordning i det sammanhanget.

9. *Streamlit* är ett gratis open-source ramverk för att snabbt bygga och dela snygga webbaserade applikationer inom maskininlärning och data science. Det är ett Python bibliotek som är specifikt designat för maskininlärning, vilket gör det enkelt att skapa och dela interaktiva verktyg och visualiseringar utan att behöva skriva en massa webbutvecklingskod.

Självutvärdering

1. Utmaningar jag har haft under arbetet har varit tid. Jag har tyvärr inte haft möjlighet att tillsätta så mycket tid som jag velat för detta projekt, vilket är synd eftersom jag sett fram emot det som en rolig kurs och spännande projekt att ta mig an. Jag har däremot fått kunskap att ta mig vidare och fullborda projektet med Streamlit på egen hand längre fram, och det känns värdefullt.

2. För detta projekt anser jag att jag tar mig in på en G nivå.

3. Tack för en bra kurs med energi och kunskap, det tar jag med mig!

Källförteckning

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow.* 2:a uppl. O'Reilly Media, Inc.

YouTube (n.d.). *Maskininlärning* [YouTube-spellista]. Tillgänglig: https://www.youtube.com/playlist?list=PLgzaMbMPEHEx9Als3F3sKKXexWnyEKH45 [Hämtad: 17 mars 2025].

Prgomet, A. (2025). *ds24_ml* [GitHub-repository]. Tillgänglig: https://github.com/AntonioPrgomet/ds24_ml.git [Hämtad: 18 mars 2025].

Inside Machine Learning (2021, 2 september). *Recall, precision & F1-score: Simple metric explanation in machine learning.* Tillgänglig: https://inside-machinelearning.com/en/recall-precision-f1-score-simple-metric-explanation-machine-learning/ [Hämtad: 18 mars 2025].