

Klassificering med maskininlärning

Att göra maskininlärningsmodell med Scikit-learn



1.1 Abstract

The aim of this project is to develop and evaluate a classification machine learning model with Scikit-learn. The model is designed to identify handwritten digits by being trained on the MNIST dataset. Through the use of a voting classifier combining Stochastic Gradient Descent, Random Forest, and Extremely Randomized Trees the model achieved an accuracy of 96.73%. The model was integrated into a web application where users can take a screenshot on a digit with their webcam and let the model identify it. Ideas for further improvement are handling non-digit inputs and exploring advanced models like neural networks.

Innehållsförteckning

| | | |
|----|--|----|
| 1. | PROJEKTETS SYFTE | 3 |
| 2. | SJÄLVREFLEKTION | 4 |
| 3. | TEORI | 5 |
| | 3.1 Maskininlärning | 5 |
| | 3.2 Systematiska fel och varians | 5 |
| | 3.3 Modeller | 5 |
| | 3.4 Mått..... | 6 |
| 4. | <i>Metod</i> | 7 |
| | 4.1 Bibliotek och mjukvara | 7 |
| | 4.2 Datamängd | 7 |
| | 4.3 Utvärdering av modellerna | 7 |
| | 4.4 Förbehandling av bilder..... | 8 |
| 5. | <i>Resultat</i> | 9 |
| | 5.1 Utvärdering och tillämpning av modellen..... | 9 |
| | 5.2 Vidare undersökning och förbättring..... | 12 |
| 6. | <i>Svar på "de teoretiska frågorna"</i> | 13 |
| | <i>Referenslista</i> | 15 |
| | <i>Appendix</i> | 16 |

1. Projektets syfte

Syftet med det här projektet är att med hjälp av *Scikit-learn* undersöka och utveckla en maskininlärningsmodell som korrekt kan identifiera handskrivna siffror från en bilddatamängd. Modellens prestanda ska utvärderas med avseende på möjligheten att implementera den i en webapplikation.

I och med det vill jag besvara följande frågor:

Hur kan *Scikit-learn* användas för att skapa en modell som med hög precision korrekt identifierar handskrivna siffror?

Hur kan modellens prestanda utvärderas och vilka metoder kan användas för att optimera modellen?

I vilken utsträckning kan modellen integreras i en webapplikation och vilka förbehandlingsmetoder av bilder behövs för att modellen ska kunna användas praktiskt?

2. Självreflektion

I en tradition av situerad kunskapsproduktion inleder jag rapporten med ett självreflekterande avsnitt. Kunskap produceras *in medias res* och att tydliggöra mitt perspektiv är ett sätt att kontextualisera projektet, vilket ökar transparensen och reproducerbarheten. Som svenskfödd kvinna född i slutet av 80-talet med bakgrund inom samhällsvetenskap samt vård och omsorg är maskininlärning ett nytt fält för mig.

Det jag fann vara mest utmanande i projektet var att sätta modellen i produktion. Det vill säga att hitta hur jag på ett effektivt sätt ska förbehandla bilder så att modellen kan göra en rimlig kategorisering av siffran på bilden. Jag har skrivit om förbehandlingsfunktionen många gånger och testat olika saker och mått samt felsökt genom att skriva ut varje förändring av bilden stegvis. Det svåraste visade sig vara att förändra storleken på bilden till rätt antal pixlar (med siffran i rätt storlek) och behålla skärpan.

Jag anser att mitt projekt håller VG-nivå då jag har utvärderat modellen på relevanta sätt samt gjort mitt bästa i att få den att fungera i webapplikationen.

Projektet har varit utmanande men oerhört roligt och gett mig mersmak på maskininlärning.

3. Teori

I det här avsnittet ger jag en översikt av maskininlärning samt de algoritmiska modeller och mått jag har använt i projektet.

3.1 Maskininlärning

Maskininlärning kan beskrivas som tekniken att iterativt upptäcka mönster i datamängder med hjälp av algoritmiska processer. Olika maskininlärningsmodeller som är utformade på olika sätt kan, beroende på hur datamängden är beskaffad, användas och ”tränas” att finna mönster. Dessa mönster kan modellen sedan applicera på ny mängd data och göra förutsägelser utifrån. Möjligheten att göra korrekta förutsägelser baserat på ny data ställer framför allt höga krav på den data som används för träningen. Den måste vara av tillräckligt stor mängd och representativ för ändamålet, det vill säga det som ska predikteras. (Géron 2019 s. 2)

Det värde som modellen ska prediktera kallas målvärde (eng. *target*) och de variabler eller egenskaper som modellen använder för att beräkna målvärdet kallas *features* (ibid s. 8). En direkt svensk översättning av termen *features* saknas. Målvärde och *features* kan vara av kategorisk eller numerisk art (ibid s.).

3.2 Systematiska fel och varians

Två centrala begrepp inom maskininlärning är systematiska fel (eng. *bias*) och varians (Fortmann-Roe 2012). Kortfattat kan systematiska fel beskrivas som förenklade antaganden om datan, medan varians syftar på variationen i prediktionerna (ibid). Stora systematiska fel leder till att modellen blir underanpassad (prediktionerna är för generella) medan stor varians leder till hög komplexitet i modellen vilket ofta gör den överanpassad (det vill säga den presterar mycket bra på den datamängd den är tränad på men sämre på ny datamängd) (ibid).

3.3 Modeller

Jag jämfört och utvärderat de tre klassificeringsmodellerna *Stochastic Gradient Descent (SGD)*, *Classifier*, *Random Forest classifier* samt *Extremely Randomized Trees (Extra Trees)*.

Stochastic Gradient Descent (SGD) Classifier är en linjär modell som fungerar som binär klassificerare (Géron 2019 s. 88). Algoritmen väljer slumpmässiga instanser i datamängden

och i varje instans finner den ett minimivärde i funktionen och på så vis identifierar den de mest optimala parametrarna (ibid s. 124). Modellen är effektiv på stora datamängder (ibid).

Random Forest Classifier är en samling av beslutsträd (ibid s. 197). Beslutsträd fungerar så att de skapar slumpmässiga delmängder av datamängden och i dessa identifieras kriterier för hur datan ska delas i kategorier (ibid). Algoritmen ger större andel systematiska fel men en lägre varians (ibid).

Extremely Randomized Trees (Extra Trees) gör inte bara slumpmässiga urval i datamängden som *Random Forest* gör, den använder dessutom slumpmässiga tröskelvärden för klassificeringen vilket gör den snabbare än *Random Forest* (ibid s. 198). Extra Trees ger större andel systematiska fel men en lägre varians (ibid).

Randomized Search CV används för att optimera hyperparametrar (ibid s. 321). Antal hyperparametrar som ska utvärderas bestäms manuellt, vilket är en fördel om datamängden har väldigt många hyperparametrar (Scikit-learn 2025b).

Därefter en läggs de ihop i en Voting Classifier som väljer den klassificering som fått flest ”röster” (numeriskt eller i medeltal) (Géron 2019 s. 189–192).

3.4 Mått

Jag har använt två mått för att utvärdera min klassificeringsmodell: träffsäkerhet (eng. *accuracy*) samt en förväxlingsmatris (eng. *Confusion Matrix*).

Träffsäkerhet är ett mått som betecknar andelen korrekta prediktioner genom totala antalet prediktioner (ibid s. 2).

Förväxlingsmatriser används för att visualisera och utvärdera hur modellen predikterar de olika klasserna (ibid s. 90). De predikterade klasserna ställs upp kolumnvis mot de faktiska klasserna radvis och för varje klass syns det tydligt hur många korrekta samt felaktiga prediktioner modellen gör (ibid).

4. Metod

I det här kapitlet redogör jag för mitt tillvägagångssätt under projektet.

4.1 Bibliotek och mjukvara

Jag har använt python-biblioteken *Scikit-learn* för modellutvärdering och -träning, *Pillow* för bildbehandling och *Streamlit* för att konstruera web-applikationen.

4.2 Datamängd

Datamängden jag använder heter *Modified National Institute of Standards and Technology* (MNIST) och är skapat av amerikanska *US Census Bureau* (Géron 2019 s. 85). Den består av 70 000 bilder på handskrivna siffror, skrivna av amerikanska skolbarn (ibid). Varje bild innehåller en siffra från 0 till 9. Bilderna är i standardiserat format av 28 gånger 28 pixlar och varje pixel utgör en *feature*. Varje målvärde (bestämning av enskild siffra) beräknas alltså fram av 784 variabler. Siffrorna är vita mot en svart bakgrund (bild 1).

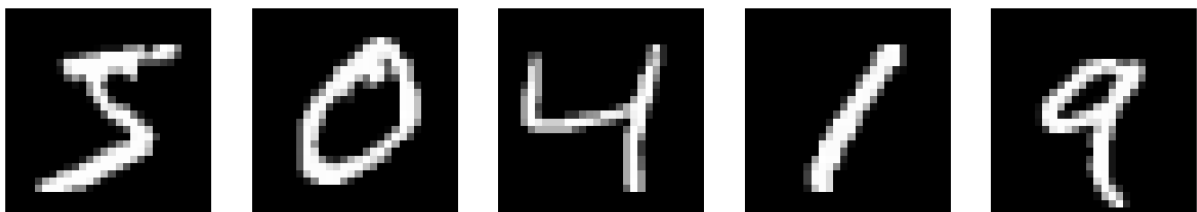


Bild 1, exempel på bilder i MNIST.

Problemet att identifiera pixlar är ett binärt klassificeringsproblem då modellen ska lära sig att kategorisera varje pixel som en del av en siffra eller inte (ibid s. 86). Datamängden är också relativt stor, med 70 000 datapunkter.

Jag delade datamängden slumpmässigt i en träningsdel om 50 000 bilder, en valideringsdel om 10 000 bilder och en testdel om 10 000 bilder.

4.3 Utvärdering av modellerna

För att lösa ett klassificeringsproblem med en stor datamängd är både *Random Forest*, *Extra Trees* och SGD passande (Géron 2019 s. 88, 197 - 198). Genom att använda dessa tre algoritmer i en *Randomized Search CV* utvärderade jag vilka hyperparametrar som gav högst träffsäkerhet. Jag använde 100 träd (*n_estimators*) och maxdjup 20 i *Extra Trees* och *Random Forest*

Classifier, SGD med logistisk regression, regularisering multiplicerat med 0,001 och 1000 iterationer över träningsdatamängden.

Vidare undersökte jag om alla tre modellerna i en *voting classifier* skulle kunna ge en högre träffsäkerhet.

Denna *voting classifier* tränade jag sedan om på hela datamängden, undantaget testdelen. Därefter utvärderade jag den på testdelen med hjälp av att mäta träffsäkerheten och göra en förvirringsmatris.

Den färdigtränade modellen kunde jag sedan applicera på ny data via *Streamlit*. Genom att göra ett interface där en användare kan ta ett foto med sin webbkamera kan den algoritmiska modellen ta emot nya bilder på siffror och prediktera vilken siffra som är avbildad.

4.4 Förbehandling av bilder

Foton på siffror som modellen tar emot via webapplikationen måste behandlas så att de liknar bilderna i den datamängd modellen är tränad på. För att åstadkomma det gör jag om bilden till gråskala, ökar kontrasten, inverterar färgerna så att siffran är vit mot en svart bakgrund och binariserar (gör den svart och vit) för att göra bilden tydligare. Dessutom klipper jag ut siffran, gör en kant runt den och omvandlar den till 28*28 pixlar.

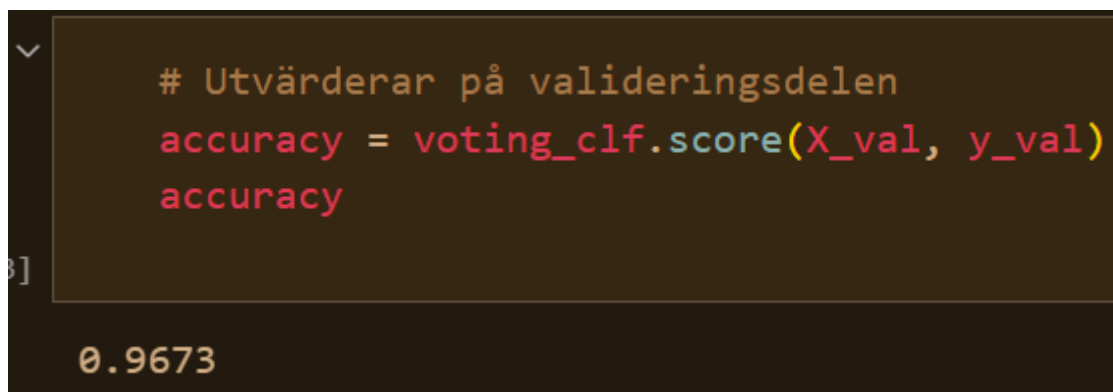
Jag lade tillfälligt till två *sliders* med värdena för kontrastfaktor och tröskelvärde för binarisering på websidan så att jag enkelt kunde testa olika värden och omgående se vad som gav bäst resultat på förbehandling av bilden. Jag fann att tröskelvärde 180 och kontrastvärde 1,55 gav bäst resultat.

5. Resultat

I den här delen av rapporten presenterar jag resultaten av projektet och resonerar kring vidare undersökningar och förbättringar.

5.1 Utvärdering och tillämpning av modellen

Voting classifier-modellen innehållande *Stochastic Gradient Descent*, *Random Forest* samt *Extra Trees* fick 96,73 % träffsäkerhet på valideringsdelen av datamängden (bild 2).



```
# Utvärderar på valideringsdelen
accuracy = voting_clf.score(X_val, y_val)
accuracy
```

0.9673

Bild 2

En förväxlingsmatris visar hur modellen predikterar på valideringsdelen av datamängden (bild 3). Matrisen visar vilka siffror som modellen oftast klassificerar fel genom att förväxla. Den blandar ihop siffran 7 med både 3 och 9 och den förväxlar 1 med 8 ibland (men aldrig 8 med 1). Den stora majoriteten av prediktionerna är dock korrekta (diagonalen från övre vänstra hörnet till nedre högra i matrisen).

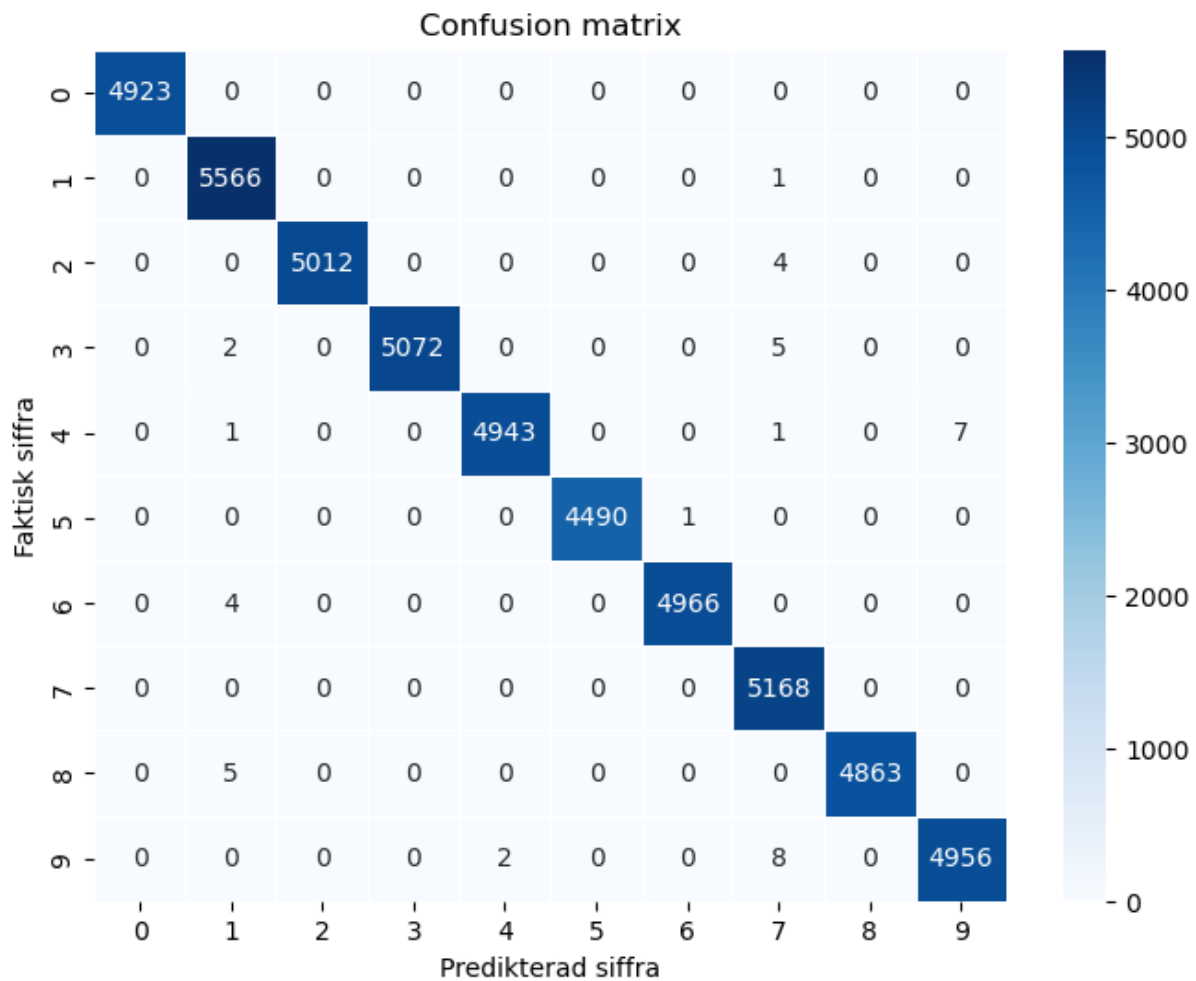


Bild 3

Genom att använda klassificeringsalgoritmer från *Scikit-learn* och MNIST-datamängden kan en modell tränas att med över 90% träffsäkerhet identifiera handskrivna siffror.

Modellens prestanda kan utvärderas genom att mäta träffsäkerheten i prediktionerna samt genom att studera prediktionerna i en förvirringsmatris. Modellen kan optimeras genom att exempelvis använda *RandomizedSearchCV* för att testa olika värden på hyperparametrarna.

Genom att förbehandla bilder på siffror för att få dem att likna bilderna i den datamängd modellen är tränad på, kan modellen integreras och användas i en webapplikation. Förbehandlingen av bilden består av att konvertera till gråskala, öka kontrasten, binarisera samt klippa och centrera och omvandla till det pixelantal som överensstämmer med bilderna i MNIST. Se bild 4 för exempel på tillämpning.

Maskininlärningsmodell som identifierar siffror.

Håll upp en bild med en siffra på och klicka på 'Take photo'.

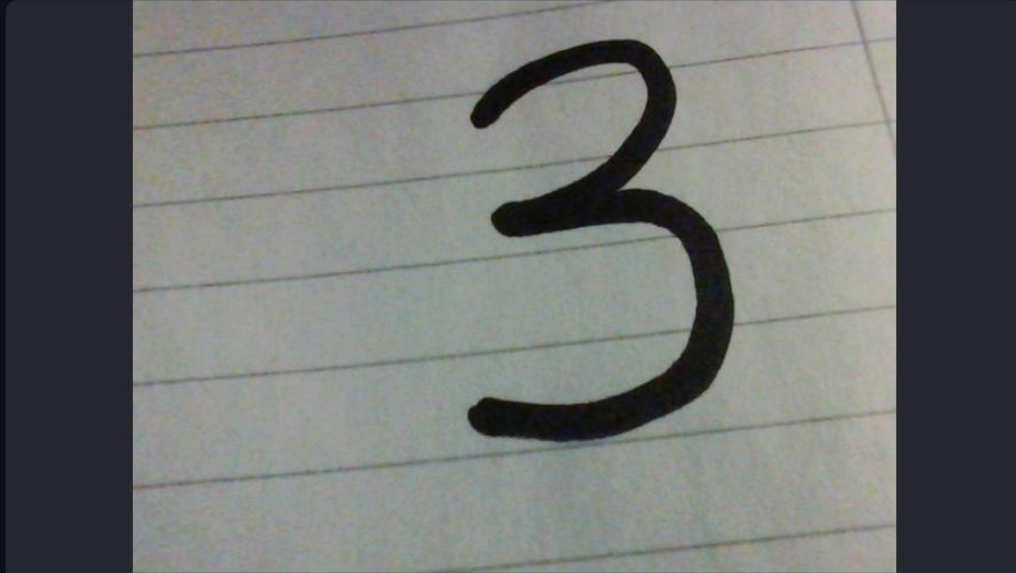


Bild 4 Exempel på hur modellen har identifierat en av mig handskivren och fotad siffra.

5.2 Vidare undersökning och förbättring

Det finns potential att göra en modell som presterar ännu bättre. Mer avancerade modeller som neurala nätverk kan ha större träffsäkerhet.

En vidare utveckling av modellen skulle kunna vara att lära den att känna igen när det *inte* finns en siffra i bilden (en form av felhantering). Får den i nuvarande skick en bild på till exempel en bokstav eller en kvadrat kommer den att kategorisera den som någon siffra mellan 0 och 9, eftersom den inte känner till något alternativ.

Förbehandlingen av bilden med siffran som ska identifieras kan troligen förbättras ytterligare så att siffran blir ännu tydligare för modellen.

6. Svar på ”de teoretiska frågorna”

Här besvarar jag de nio frågorna i numerisk ordning.

1. Träningsdelen av datasetet används för att träna modellen. Valideringsdelen används för en preliminär utvärdering av modeller. Testdelen används till en slutlig utvärdering av den modell som fick bäst resultat på valideringsdelen. (Géron 2019 s. 31)
2. Julia kan använda klassen `GridSearchCV` som automatiskt utför *cross validation* vilket innebär att (tränings)datan internt delas upp i ”veck” (eng. *folds*) som fungerar som valideringsdata (Scikit-learn 2025a).
3. Regressionsproblem innebär att målvärde och parametrar är numeriska. Syftet med modellen är att beräkna ett eller flera numeriska värden. (Géron 2019 s. 39). *Linear Regression*, *Ridge Regression*, *Least Absolute Shrinkage and Selection Operator Regression (LASSO)* och *Polynomial Regression* är exempel på modeller som kan användas för detta (ibid). Tillämpningsområden kan exempelvis vara att förutspå aktiekurser, beräkna huspriser eller beräkna effektivitet i kollektivtrafiken med avseende på tid.
4. *Root Mean Square Error (RMSE)* kan användas som ett mått för modellers prestation då den visar medelfelet mellan modellens prediktion och målvärdet (kvadrerat för att hantera negativa värden) (ibid s. 39).
5. Klassificeringsproblem innebär att målvärde och parametrar är kategoriska. Syftet med modellen är att kunna klassificera instanser. Exempel på klassificeringsmodeller är *Random Forest*, *Decision Tree* och *Support Vector Machine Classifier* (ibid s. 100). Tillämpningsområden kan till exempel vara automatiska spamfilter eller program som kan artbestämma växter via foton. Modellens prestation kan utvärderas med hjälp av en *confusion matrix* som ställer upp de predikterade kategoriseringarna mot de sanna kategorierna (ibid s. 19).

6. *K-means* är en algoritmisk modell som grupperar (eng. *clustering*) (ibid s. 238 - 249). Det kan till exempel tillämpas på bildkomprimering där antalet färger kan minskas med gruppering.

7. *Ordinal encoding* betyder att kategoriska värden konverteras till numeriska värden baserat på dess inbördes ordning, exempelvis "dålig" blir 0, "ganska bra" blir 1, "bra" blir 2 osv. (ibid s. 738). *One-hot encoding* innebär att de kategoriska värdena görs om till binära vektorer (ibid s. 431 - 432). *Dummy variable encoding* fungerar som *one-hot encoding* men en av kategorierna representeras av en vektor med enbart nollor.

8. Jag ser inte att Görans och Julias påståenden står i motsats till varandra. Kategoriska data kan vara ordinal eller nominal, precis som Göran påstår (Wikipedia 2025). Huruvida kategorierna har en inbördes ordning eller ej beror på om de *tillskrivs* inbördes ordning eller ej. Givetvis är denna tillskrivning en tolkningsfråga, precis som Julia säger. En eventuell inbördes ordning är alltid beroende av sammanhang. Ur ett visst perspektiv, i ett visst sammanhang är datan antingen ordinal eller nominal då den som hanterar och analyserar datan gör ett val och behandlar den som antingen eller.

Tanken att ordinalitet inte skulle var en tolkningsfråga implicerar att det skulle finnas ett sant objektivt perspektiv – ett Guds öga. Denna föreställning om den sanna objektiviteten är ett inom naturvetenskapen traditionellt epistemologiskt perspektiv som har kritiserats, utmanats och söndersmulats av flertalet postmoderna och feministiska vetenskapsteoretiker (se till exempel Lykke 2009 s.141, Haraway 2004, 2008 samt Harding 2004).

9. *Streamlit* är ett python-bibliotek som används för att skapa webb-applikationer (Streamlit 2025). Det kan bland annat användas för att göra så att en *machine learning*-modell kan ta emot *input* från en användare.

Referenslista

Fortmann-Roe, S (2012) *Understanding the Bias-Variance Trade Off* <https://scott.fortmann-roe.com/docs/BiasVariance.html> [hämtad 2025-03-20]

Géron, A. (2019) *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. Andra upplagan. O'Reilly

Haraway, Donna (2008) *Apor, cyborger och kvinnor – att återuppfinna naturen* Symposium

Harway, Donna (2004) "Situated Knowledges: The Science Question in Feminism and the Privilege of Partial Perspective" i Harding, Sandra (red.) *The Feminist Standpoint Theory Reader* Routledge

Harding, Sandra (2004) "Introduction: Standpoint Theory as a Site of Political Philosophic and Scientific Debate" i Harding, Sandra (red.) *The Feminist Standpoint Theory Reader* Routledge

Lykke, Nina (2009) *Genusforskning – En guide till feministisk teori, metodologi och skrift* Liber AB

Scikit-learn (2025a) *GridSearchCV* https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html [hämtad 2025-03-06]

Scikit-learn (2025b) *RandomizedSearchCV* https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html [hämtad 2025-03-20]

Streamlit (2025) *Streamlit Documentation* <https://docs.streamlit.io/> [hämtad 2025-03-07]

Wikipedia (2025) "Nominal category" https://en.wikipedia.org/wiki/Nominal_category [hämtad 2025-03-07]

Appendix

Länk till projektets GitHub-sida:

https://github.com/SisselN/ds24_ml_kurs

Länk till applikationen:

<https://ds24mlkurs-zdmypm95wougvv9afkaahe.streamlit.app/>