Sifferigenkänning

Machine Learning



Alvin Lagerson

EC Utbildning

Maskininlärning

2025

# Abstract

The project explores some machine learning techniques and models to recognize handwritten numbers in the MNIST dataset. By using several classification algorithms such as Random forest, Support vector machine, Extreme gradient boosting and neural networks, the project aims to compare the their accuracy and usage. The findings highlight the strength of the Extreme gradient boosting algorithm both in performance and portability. Additionally, the project finds that by modifying the dataset to no longer having grey scale pixels the best model becomes slightly more accurate.

Innehållsförteckning

[1 Inledning 1](#_Toc193538544)

[1.1 Bakgrund 1](#_Toc193538545)

[1.2 Syfte och frågeställning 1](#_Toc193538546)

[2 Teori 2](#_Toc193538547)

[2.1 Maskininlärningsmodeller 2](#_Toc193538548)

[2.1.1 Support Vector Machine 2](#_Toc193538549)

[2.1.2 Random Forest 2](#_Toc193538550)

[2.1.3 Extreme Gradient Boosting 2](#_Toc193538551)

[2.1.4 Neural network 2](#_Toc193538552)

[2.2 Viktiga koncept 2](#_Toc193538553)

[2.2.1 Accuracy/Träffsäkerhet 2](#_Toc193538554)

[2.2.2 Confusion matrix? 2](#_Toc193538555)

[2.2.3 Grid Search 2](#_Toc193538556)

[3 Metod 3](#_Toc193538557)

[3.1 Data 3](#_Toc193538558)

[3.2 Bibliotek 3](#_Toc193538559)

[3.3 EDA 3](#_Toc193538560)

[3.4 Förberedelser 4](#_Toc193538561)

[3.5 Träning och validering 4](#_Toc193538562)

[3.6 Optimering 4](#_Toc193538563)

[3.7 Test 4](#_Toc193538564)

[3.8 Ytterligare idéer 5](#_Toc193538565)

[3.9 Streamlit 5](#_Toc193538566)

[4 Resultat och Diskussion 6](#_Toc193538567)

[4.1 Resultat 6](#_Toc193538568)

[4.2 Diskussion 6](#_Toc193538569)

[5 Slutsatser 7](#_Toc193538570)

[5.1 Frågeställning 7](#_Toc193538571)

[6 Teoretiska frågor 8](#_Toc193538572)

[7 Självutvärdering 11](#_Toc193538573)

[Appendix A 12](#_Toc193538574)

[Källförteckning 13](#_Toc193538575)

# Inledning

## Bakgrund

Att vara vuxen innebär en väldig massa saker. Man ska bland annat ansvara för sig själv, och kanske även andra. En del av detta handlar om att man en del av ett större sammanhang, ett samhälle. I samhället ställs vissa krav för att allt ska fungera som helhet, där vi bland annat har myndigheter, våra kontrollorgan. Våra myndigheter behöver information om alla invånare och denna information samlas in och uppdateras genom standardiserade blanketter. Dessa blanketter har tidigare lagts in manuellt av handläggare vilket kostar massor. Genom maskininlärning kan informationen på dessa blanketter som samlats in läsas av automatiskt och på så sätt sparas tid och pengar. Automatisk inläsning kan givetvis användas i andra sammanhang än för myndigheter, men i de allra flesta fall kommer det spara tid.

Genom effektiviseringar och automatiseringar kan vi utöka digitaliseringen av samhället och på så sätt säkra information för framtiden. Rätt sparad data ligger mycket säkrare än fysiska arkiv och är dessutom snabbare och enklare att navigera i. Genom att låta en dator lära sig känna igen handskrivna bokstäver och siffror kan man inte bara spara tid för att läsa in aktuella blanketter utan även till att digitalisera dokument för att säkra informationen.

Det finns 29 bokstäver i alfabetet, vilket blir 58 tecken när man räknar in små bokstäver, och 10 olika siffror. För att komma i gång med tolkning av tecken börjar vi med siffrorna och till detta använder vi datasetet MNIST.

## Syfte och frågeställning

Syftet med denna rapport är att kolla om man, genom maskininlärning, kan läsa in handskrivna siffror. Men, frågan är om en maskin kan läsa tecken lika bra som en handläggare. Kan en maskininlärningsmodell komma upp i över 95% träffsäkerhet?

# Teori

## Maskininlärningsmodeller

### Support Vector Machine

Support Vector Machine (SVM) handlar om att hitta den bredaste möjliga ”gatan” mellan olika klasser i ett dataset, at maximera marginalen mellan beslutsgränsen och träningspunkterna. Viktiga koncept inkluderar stödvektorer som är punkter vid gatans kant och användning av kernels för att hantera icke-linjära dataset (Geron, 2019). Vid Soft Margin Classification balanserar SVM mellan att ha så bred ”gata” som möjligt och att begränsa margin violations (Prgomet, 2023).

### Random Forest

Random Forest (RF) är en ensemble av beslutsträd som använder bagging-metoden (bootstrap aggregating) för träning och introducerar slumpmässighet vid noddelning. Detta ökar träddiversiteten och ger bättre modellprestanda. De är användbara för att analysera vilka funktioner som är viktigast och erbjuder flexibla hyperparametrar för att optimera både ensemblen och enskilda träd (Geron, 2019). Bagging är en ensemblemetod som tränar basmodeller på slumpmässiga delmängder av data och kombinerar deras prediktioner och minska variansen (Scikit-learn, 2025).

### Extreme Gradient Boosting

Gradient Boosting är en teknik för att successivt bygga en ensemble av prediktorer, där varje ny prediktor försöker korrigera residualfelen från den tidigare. Genom att använda metoder som Gradient Tree Boosting kombineras flera mindre beslutsstödsträd för att stegvis förbättra modellens noggrannhet. Tekniken kan även optimeras genom att justera parametrar som antalet träd, trädens djup och inlärningshastighet, samt tillämpa tidigt stopp för att undvika överanpassning. XGBoost (XGB) är ett bibliotek till Python utvecklat för att vara ett extremt snabbt, skalbart och portabelt alternativ av Gradient Boosting (Geron, 2019).

### Neural network

Neurala nätverk använder aktiveringsfunktioner, en funktion för hur ett neuron ska aktiveras, för att bearbeta data och lösa klassificerings- eller regressionsproblem. De tränas med hjälp av bakåtpropagering, en process för beräkning av fel mellan förväntat och faktiskt värde, går bakåt genom nätverket och mäter felets bidrag vid varje koppling och justerar vikterna för att minska felet (gradientnedstigning). Hyperparametrar som antal lager och neuroner kan anpassas för bättre resultat (Geron, 2019).

## Viktiga koncept

### Accuracy/Träffsäkerhet

Hur pass väl en modell predikterar kan man mäta med accuracy, en funktion som beräknar hur väl värdena i ett dataset predikterats gentemot de faktiska värdena. Antalet predikterade värden måste matcha antalet sanna värden exakt (Scikit-learn, 2025).

### Confusion matrix?

Efter en prediktion kan man utvärdera accuracyn genom att visa sanna negativa-, falska negativa-, sanna positiva- och falska positiva värden grafiskt (Scikit-learn, 2025).

### Grid Search

När en passande modell har valts kan man finjustera dess hyperparametrar för att förbättra modellens prestation, men annars kan man använda Grid search som gör det åt en. Grid search är en metod för att automatiskt hitta den bästa kombinationen av hyperparametrar i en modell genom att testa valda möjliga kombinationer och använda korsvalidering för att utvärdera dem (Geron, 2019).

# Metod

## Data

Datasetet MNIST har hämtats från Scikit-Learn genom funktionen fetch\_openml.

## Bibliotek

I kodningen har jag använt en lång rad olika bibliotek. Bland dessa finns numpy, pandas, matplotlib, seaborn, sklearn, tensorflow, joblib, xgboost, PIL och streamlit.

## EDA

MNIST består av 70 000 handskrivna siffror som konverterats till arrayer av värden mellan 0 och 255, där 0 är vit (bakgrund) och alla värden över 0 är gråskala upp till svart och en del av siffran. Innan jag började med föranalys delade jag upp datasetet i träning och validering, 85% och test, 15%. Jag delade även upp tränings- och valideringssetet till 85% träning och 15% validering. Sen kontrollerade jag om det fanns några saknade värden, vilket det inte gjorde. Genom att plotta några av arrayerna från träningsdata kunde jag se exempel på siffrorna från datasetet.

En bild som visar nummer, Teckensnitt, diagram, typografi

AI-genererat innehåll kan vara felaktigt.

Figur 1. Exempelsiffror plottade i graf.

Genom ett histogram kunde jag se fördelningen av värdena mellan 0 och 255, vilket föga förvånande var mest nollor.

En bild som visar text, skärmbild, skärm, Rektangel

AI-genererat innehåll kan vara felaktigt.

Figur 2. Frekvensfördelning av värden i träningsdata.

För att kolla på träningsdatasetets helhet så plottade jag ut genomsnittet för varje siffra i 10 olika grafer, mest av nyfikenhet om jag ska vara ärlig men även för att visualisera mängden värden och få en intressant grafisk bild av en stor mängd data.

En bild som visar text, Teckensnitt, skärmbild, nummer

AI-genererat innehåll kan vara felaktigt.

Figur 3. Siffrorna plottade som ett genomsnitt.

## Förberedelser

I förberedelserna skalade jag om dataseten med StandardScaler inför användning av Support vector machine, plattade till data till Random forest och normaliserade data för ett neuralt nätverk.

## Träning och validering

Först ut blev Random forest som inte presterade korrekt, följt av Support vector machine som gick fint. Därefter försökte jag göra ett neuralt nätverk (mest för att testa) men jag fick inte Tensorflow att fungera så jag kunde inte importera Keras. Eftersom neuralt nätverk inte fungera testade jag i stället en XGBoost som fungerade bra. Genom en massa felsökningar och kontroller av träningsdatasetet hittade jag att jag predikterat ett datasett och sen validerat mot ett annat och när jag hittade detta misstag fungerade Random forest. Det neurala nätverket fungerade med Google Colab så jag testkörde modellen där som referens.

## Optimering

Efter valideringen jämförde jag träffsäkerheten mellan modellerna och valde den bästa modellen. Jag gick vidare med en Grid search men begränsade mig till 10 000 sampels på grund av tid, vilket ändå tog 4 timmar. Sen tränade jag om den bästa modellen på träning och valideringsdata med parametrarna jag fick ut från Grid search.

## Test

Sluttestet genomfördes lyckat och modellen sparades ner med Joblib. För att visualisera resultatet plottade jag en Confusion matrix, se figur 4 nedan, vilket ser bra ut. För att kontrollera att allt funkar så testade jag att ladda den sparade modellen och prediktera en slumpmässig siffra, och den laddade modellen funkade.

En bild som visar text, skärmbild, Rektangel, diagram

AI-genererat innehåll kan vara felaktigt.

Figur 4. Confusion matrix.

## Ytterligare idéer

En tanke slog mig om att prova göra om datasetet för att se om resultatet förändras. Jag testade att skriva en kod som gjorde om alla värden över 10 till 255 och under 10 till 0 i arrayerna för att ”tjocka” till siffrorna. När jag ändå höll på gjorde jag likadant men med gränsen på 128 samt en gräns på 200 för att ”tunna ut” och tränade om nya XGB-modeller. I figur 5 visas en plot på de nya siffrorna.

En bild som visar diagram, linje, Graf, skärmbild

AI-genererat innehåll kan vara felaktigt.

Figur 5. Omkodade siffror.

## Streamlit

Jag jobbade samtidigt med Streamlit för att testa hur modellerna fungerade praktiskt med skrift. Där laddade jag in modellerna med joblib, och lade in en canvas där man kan rita. Bredvid ritytan lade jag in en graf med en plot av den digitala ”översättningen”, motsvarande MNIST-datasetet. Både den vanliga XGB-modellen och XGB-modellen med ”förtjockade” siffror predikterar resultat med en sannolikhetsfördelning av första XGB-modellen längst ner. Den ”förtjockade” modellen predikterar en motsvarande ”förtjockad” version av canvas-siffran.

En bild som visar skiss, skärmbild, design

AI-genererat innehåll kan vara felaktigt.

Figur 6. Streamlit Canvas med digital översättning.

# Resultat och Diskussion

## Resultat

|  |  |
| --- | --- |
| **Accuracy för olika modeller på valideringsdata** | |
| Random forest | 97,21% |
| Support vector machine | 97,40% |
| Neural network | 97,46% |
| XGBoost | 97,96% |

Tabell 1. Accuracy för de fyra valda modellerna.

Av de fyra olika modellerna som testats var XGBoost den med bäst accuracy.

|  |  |
| --- | --- |
| **Accuracy för olika varianter av XGBoost** | |
| XGB med Grid Search (GS) på valdata | 97,89% |
| XGB m GS på testdata | 97,86% |
| XGB m GS m förtjockade siffror på testdata | 97,91% |
| XGB u GS m förtjockade siffror på testdata | 98,07% |

Tabell 2. Accuracy för varianter av XGBoost.

Vidare efter genomförd Grid search blev resultatet snäppet sämre än utan, både på validerings- och testdata. Däremot blev resultatet för XGB-modellen som tränades och testades med Grid search och förtjockade siffror bättre än de utan. Genomförda tester med modifierade data med gräns 128 och 200 (förtunning) gav någorlunda bra resultat men båda var sämre än original data.

## Diskussion

Det blev generellt sett hög accuracy med alla valda modeller men XGBoost presterade bättre än övriga. Sen om man tittar på storleken på de sparade modellerna så är skillnaden mellan den minsta och största enorm. XGB sparades ner till en 4MB stor fil medan Random forest sparades ner till drygt 600MB så det stämmer ju med målet, att XGB ska vara en portabel modell, utöver att den presterade bäst. Däremot så tog det bara några få minuter att träna och testa XGB med default-parametrar medan Grid search tog 4 timmar och plockade fram hyperparametrar som egentligen gjorde modellen snäppet sämre. Detta verkade konstigt och jag dubbelkollade default-värdena i XGB och det var med i Grid search testparametrar. Däremot kördes Grid search på 10 000 sampels på grund av tidsåtgången så den kanske skulle kunna ge bättre hyperparametrar om den kördes på hela träning eller träning/validering datasetet. För skoj skull gjorde jag även en vote med Random forest och XGB, bara för att testa, men den presterade inte bättre så jag släppte den.

I Streamlit-applikationen testade jag flera av modellerna för att jämföra prediktionerna på egenskrivna siffror och de presterade olika vid olika tillfällen. Min uppfattning är att XGB med ”förtjockade” siffror presterade bättre än övriga i applikationen men jag har inte gjort någon mätning med ett större urval för att säkerställa det. I Streamlit har jag tagit bort eller kommenterat bort alla modeller utom XGB och XGB med ”förtjockade” siffror då både Random forest och vote gjorde hela applikationen väldigt långsam.

När man skriver en siffra i canvas blir det dock inte 97-98% träffsäkerhet. En brist med en modell tränad på MNIST när man kör in en canvas-skriven siffra är datasetets natur. Siffrorna i MNIST är handskrivna i en liten ruta, så det blir en annan sak när man skriver med musen då både handstilen försvinner och att man inte fyller ut rutan på samma sätt som man skulle gjort för hand. Att prediktera handskrivna blanketter skulle rimligtvis ge önskvärt resultat.

# Slutsatser

## Frågeställning

1. Kan en maskininlärningsmodell komma upp i över 95% träffsäkerhet?

Jag skapade en modell som uppnått 98% accuracy på osedda testdata.

# Teoretiska frågor

*Kalle delar upp sin data i ”Träning”, ”Validering” och ”Test”, vad används respektive del för?*

Träningsdata använd för att för att träna modeller, valideringsdata används för att utvärdera vilken av modellerna som är bäst. Sen tränar man om den bästa modellen på träning och valideringsdata för att sen testa den på testdata för att se hur bra modellen generaliserar på nya osedda data.

*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”. Hur skall hon välja vilken av de tre modellerna hon skall fortsätta använda när hon inte skapat ett explicit ”validerings-dataset”?*

Hon kan göra k-fold Cross-validation. Då delas träningsdata upp i cluster där till exempel 4 kluster används för träning och valideras mot 1 för att sen tränas på 4 andra och valideras mot 1 till dess alla kluster använts till validering. Om det är mycket data hon jobbar med så kan det ta tid, men i annat fall är det ett bra sätt att utnyttja sin data.

*Vad är ”regressionsproblem? Kan du ge några exempel på modeller som används och potentiella tillämpningsområden?*

Ett regressionsproblem är när den beroende variabeln (y) kan anta många värden. Till exempel om y är inkomst eller ett bostadspris som kan anta kontinuerliga värden (oändligt många olika värden). Några exempel på modeller är Linear regression och Lasso som kan användas för att till exempel förutsäga bostadspriser. Men du kan även använda SVM, Support Vector Machine, för att förutsäga bostadspriser.

*Hur kan du tolka RMSE och vad används det till?*

Root mean squared error är ett mått på medelavståndet mellan uppskattade värden och verkliga värden. Anledningen till att man kvadrerar och tar roten ur är att det inte har någon betydelse om avståndet är negativt eller positivt, det är storleken på felet som är intressant. Du kan använda RMSE för att jämföra olika modeller med varandra för att hitta den bästa modellen. Du behöver lära känna den data du jobbar med för att förstå vad som är ”bra” eller ”dåligt” eftersom talet du får fram är helt beroende av vad du tittar på. Lägre är bättre, men att få 20 i RMSE kan i sitt sammanhang vara riktigt uselt medan 20 000 i ett annat sammanhang kan vara fantastiskt, siffran i sig betyder inget utan sitt sammanhang.

*Vad är ”klassificieringsproblem? Kan du ge några exempel på modeller som används och potentiella tillämpningsområden? Vad är en ”Confusion Matrix”?*

Den beroende variabeln (y) kan anta två olika värden. Till exempel churn, 1 eller 0. Kan ha en eller flera andra oberoende variabler. Ett exempel på modell är logistisk regression som skulle kunna användas till exempel för att avgöra om en patient har en viss sjukdom. Decision tree eller Random Forest är fler exempel på modeller du kan använda vid klassificeringsproblem. Decision tree kan till exempel användas för att detektera skräppost och Random forest skulle kunna användas för att utvärdera kunder med hög kreditrisk. SVM, Support Vector Machines kan även användas vid klassificeringsproblem som i ett spamfilter. En Confusion Matrix är en matris för att grafiskt visa resultatet av en modells prestation, en enkel variant är en matris som visar verklig sant/falskt som rader och förutspådd sant/falskt som kolumner. Då visas korrekt sant/ inkorrekt falskt på första raden och inkorrekt sant / korrekt falskt på andra raden.

*Vad är K-means modellen för något? Ge ett exempel på vad det kan tillämpas på.*

Det är en oövervakad inlärningsmodell. Det betyder att det inte finns någon beroende variabel, något y-värde. Algoritmen används för att gruppera datapunkter i olika kluster (k-antal kluster). Det fungerar som så att man väljer antal kluster, de placeras ut i datarummet, varje datapunkt tilldelas närmaste centralpunkt, centroid, och sen beräknar man medelvärdet av datapunkterna i varje kluster och flyttar centroiden dit. Sen upprepar man de två sista stegen tills centroiderna stabiliseras. Ett exempel på användning är att placera kunder i olika segment för att kunna rikta relevant marknadsföring till respektive kundgrupp eller att upptäcka avvikelser, eller anomalier, i beteenden hos kundgrupper för att till exempel upptäcka potentiella brott.

*Förklara (gärna med ett exempel): Ordinal encoding, one-hot encoding, dummy variable encoding. Se mappen ”l8” på GitHub om du behöver repetition.*

Ordinal encoding används när kategorierna har en naturlig ordning (till exempel en ordinal variabel). Varje kategori tilldelas ett unikt numeriskt värde. Ett exempel är kundnöjdhet, 1 till 5 från missnöjd till mycket nöjd, men en tvåa inte nödvändigtvis är 50% av en fyra. En fördel är att det är ordning mellan kategorierna men en nackdel är att skillnaden mellan värdena inte är jämn.

One-hot encoding skapar en binär (0 eller 1) kolumn för varje kategori. Används för nominella variabler (ingen inbördes ordning). Ett exempel kan vara frukt där man ställer upp frukttyperna med svaren 1 eller 0, där ett äpple är 1 under kategorin äpple och då 0 på apelsin och 0 på mango etc. En fördel är att det inte är någon risk att modellen antar en ordning som inte finns. En nackdel är att det lätt kan bli många kolumner om det finns många unika värden (s.k. "curse of dimensionality").

Dummy encoding är en variant av one-hot encoding där en kategori utelämnas (referenskategori), för att undvika multikollinearitet, där variablerna kan förutsägas av varandra. Som i fruktexemplet kan man då ta bort kategorin äpple eftersom om det är 0 på apelsin och 0 på mango etcetera så betyder det att det är ett äpple. En fördel är att det blir färre kolumner än one-hot encoding, vilket minskar redundans. En nackdel är att det kräver att vi tolkar modellen med referenskategorin i åtanke.

*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?*

Data kan inte vara både ordinal och nominal samtidigt. Att en person med röd skjorta skulle vara vackrast på en fest är helt och hållet subjektivt och Julias personliga preferens. Om man räknar med personliga preferenser så kan i princip all data räknas som ordinal eftersom Julia även kanske tycker äpple är godare än apelsin. Detta skulle leda till att ingen modell kan användas för generalisering någonsin eftersom de subjektiva värderingarna av en mängd olika värden enbart passar Julia. Ska data kunna rangordnas och vara ordinal måste det vara objektivt sett en inbördes ordning. Ett datasett kan däremot innehålla både ordinal och nominal data.

*Kolla följande video om Streamlit: https://www.youtube.com/watch?v=ggDa-RzPP7A&list=PLgzaMbMPEHEx9Als3F3sKKXexWnyEKH45&index=12 Och besvara följande fråga: - Vad är Streamlit för något och vad kan det användas till?*

Streamlit är ett open source Python-bibliotek som används för att snabbt och enkelt bygga interaktiva webbapplikationer för dataanalys och maskininlärning. Streamlit kan bland annat användas för att visa data och låta användaren interagera med data direkt i webbläsaren utan att behöva bygga applikationen i till exempel HTML eller att helt enkelt skapa ett enkelt gränssnitt för att testa ML-modeller.

# Självutvärdering

1. Utmaningar du haft under arbetet samt hur du hanterat dem.

Under arbetets gång har jag stött på några problem. Bland annat fick min random forest en accuracy på knappt 8%, jag provade då att öka antal träd men det gjorde väldigt liten skillnad. Jag läste om att man kunde använda XGBoost i stället och testade köra den men fick i princip samma accuracy. Jag testade också att göra ett enkelt neuralt nätverk, men jag kunde inte få Tensorflow att fungera på min dator. Jag fick i gång det på Google Colab och fick då en accuracy på 8% och insåg då att jag måste ha fel i koden. I Colabs såg jag i loggen att accuracyn låg mellan 65% och 96% i varje epoch men att den till slut skrev ut 7% och insåg då att jag utvärderade X-valideringsdata mot y-testdata. Problem är till för att lösas och är en del av processen. Genom att stöta på problem och brister i arbetet får man fokusera extra på att lösa dessa och försöka förstå anledningen och då lär man sig bättre.

1. Vilket betyg du anser att du skall ha och varför.

På denna kurs har jag haft oturen att vara sjuk eller ha sjuka barn hemma i nästan 5 veckor. Det har gjort att jag inte kunnat lägga ner så mycket fokuserad tid som jag velat. Till en början hade jag tänkt att jag bara skulle göra G-delen och nöja mig där på grund av tidsbristen. Men under arbetets gång senaste veckan kände jag att jag ändå ville prova göra en Streamlit-applikation för att ha något att visa upp och att jag då lika gärna åtminstone kunde försöka göra VG-delen hjälpligt också. Jag tycker att jag förstår mer och mer ju mer jag håller på och tycker jag har en hyfsad förståelse. Men jag ser mycket till vad jag inte hunnit med snarare än vad jag hunnit med men är spänd på er bedömning.

1. Något du vill lyfta fram till Antonio?

Jag skulle nog säga att det vore bra med fler kortare praktiska uppgifter under resans gång skulle underlätta. I övrigt tyckte jag det var en bra kurs.

# Appendix A

Koden för projektet och streamlit-applikationen finns på följande länk: [Al-Vinci/ML: Machine Learning](https://github.com/Al-Vinci/ML)

# Tabell, och figur och källförteckning

## Figurförteckning

[Figur 1. Exempelsiffror plottade i graf. 3](#_Toc193544252)

[Figur 2. Frekvensfördelning av värden i träningsdata. 3](#_Toc193544253)

[Figur 3. Siffrorna plottade som ett genomsnitt. 4](#_Toc193544254)

[Figur 4. Confusion matrix. 5](#_Toc193544255)

[Figur 5. Omkodade siffror. 5](#_Toc193544256)

[Figur 6. Streamlit Canvas med digital översättning. 5](#_Toc193544257)

## Tabellförteckning

[Tabell 1. Accuracy för de fyra valda modellerna. 6](#_Toc193544205)

[Tabell 2. Accuracy för varianter av XGBoost. 6](#_Toc193544206)

## Källförteckning

Geron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow.* O'Reilly Media, Inc.

Prgomet, A. (Regissör). (2023). *Support Vector Machines* [Film].

*Scikit-learn*. (2025). Hämtat från Scikit-learn: https://scikit-learn.org/stable/index.html