Sifferigenkänning med maskininlärning



Matilda Wilhelmsson

EC Utbildning

Machine learning – Kunskapskontroll 2

2024-03-21

# Abstract

This essay explores the usage of machine learning (ML) in relation to number recognition. By testing three different ML models (*Random Forest, Extra Trees and Linear SVM Classifier*) *on* records of handwritten numbers from the MNIST database we evaluate their capacity to recognize and categorize numbers as they would look in real life, in handwritten form. We continue exploring the model’s capacity by creating a Streamlit-app where we upload our own handwritten numbers. The results show us that all three models have the capacity to correctly recognize and categorize the records of number I MNIST. The best model however is *Extra Trees* with an accuracy score of 96.91%. However, when applied on the numbers given by the Streamlit-app, Linear SVM Classifier had by far the best performance.

# Förkortningar och Begrepp

AI = Artificiell intelligens

ML = Machine learning

MNIST = Modified National Institute of Standards and Technology

SVM = Support vector machine

Innehållsförteckning

[Abstract 1](#_Toc161922294)

[Förkortningar och Begrepp 2](#_Toc161922295)

[1 Inledning 4](#_Toc161922296)

[2 Teori 5](#_Toc161922297)

[2.1 Viktiga koncept inom maskininlärning 5](#_Toc161922298)

[2.1.1 Bias 5](#_Toc161922299)

[2.1.2 Variance 5](#_Toc161922300)

[2.1.3 Bias-Variance tradeoff 5](#_Toc161922301)

[2.2 Klassificeringsmodeller 5](#_Toc161922302)

[2.2.1 Random Forest 5](#_Toc161922303)

[2.2.2 Extra Trees 6](#_Toc161922304)

[2.2.3 Linear SVM Classifier 6](#_Toc161922305)

[3 Metod 7](#_Toc161922306)

[3.1 Skapa och välja maskininlärningsmodell 7](#_Toc161922307)

[3.1.1 Importera paket 7](#_Toc161922308)

[3.1.2 Databas 7](#_Toc161922309)

[3.1.3 Preparera data 8](#_Toc161922310)

[3.1.4 Träna modeller 9](#_Toc161922311)

[3.1.5 Accuracy score 9](#_Toc161922312)

[3.2 Skapa en Streamlit-app 10](#_Toc161922313)

[4 Resultat 11](#_Toc161922314)

[5 Diskussion 12](#_Toc161922315)

[6 Slutsatser 13](#_Toc161922316)

[7 Teoretiska frågor 14](#_Toc161922317)

[8 Självutvärdering 16](#_Toc161922318)

[Appendix A 17](#_Toc161922319)

[Källförteckning 20](#_Toc161922320)

# Inledning

Maskininlärning och AI är två områden som är extremt aktuella i dagens samhälle. Komplexa modeller och maskiner programmeras och tränas med resultat som i vissa fall utklassar människors förmåga. Grunden till dessa massiva modeller är dock enklare modeller som tränats att känna igen och kategorisera data. Genom att förstå hur dessa enklare modeller funkar kan vi få en större förståelse för hur de mer komplicerade modellerna funkar. Detta ger oss värdefull förståelse för ett område som redan nu är så viktigt i vardagen men som i framtiden kommer bli ännu viktigare.

I denna rapport kommer vi kolla närmre på hur man kan träna modeller med maskininlärning för att få modellerna att känna igen handskrivna siffror. Till vår hjälp har vi MNIST-databasen med handskrivna siffror. Ingen modell är dock speciellt intressant om den inte kan generalisera på ny data. Vi skapar därför en Streamlit-app där vi matar in våra egna handskrivna siffror som vi sedan låter modellerna analysera.

Syftet med denna rapport är att undersöka potentialen hos tre olika ML-modeller när det kommer till att känna igen handskrivna siffror, för att uppfylla syftet så kommer följande frågeställning att besvaras:

1. Vilken av de tre följande maskininlärningsmodellerna är bäst på att identifiera och kategorisera de handskrivna siffrorna i MNIST-databasen: Random Forest, Extra Trees eller Linear SVC?
2. Vilken modell är bäst på att identifiera och kategorisera de nya handskrivna siffrorna som vi matar in i vår streamlit-app för sifferigenkänning?

# Teori

## Viktiga koncept inom maskininlärning

Innan vi går in på specifika maskininlärningsmodeller är det till vår fördel om vi går igenom några viktiga koncept inom maskininlärning. Generaliseringsfel är samlingsnamnet på flera olika problem som dyker upp hos maskininlärningsmodeller. Ett sådant problem är bias. Ett annat är varians. Koncepten vi går igenom i följande avsnitt är utvalda för att de är viktiga för just denna uppsats.

### Bias

Bias i en maskininlärningsmodell står för att vi har gjort modellen okänslig till variation i datan (Géron 2019, s.134). Modellen är i många fall enklare än det riktiga mönstret hos datan, t. ex. om vi gör modellen är linjär men datan egentligen är kvadratisk. En modell med hög bias tenderar att passa träningsdatan dåligt. Detta fenomen kallas underfitting. En sådan modell kan dock vara bättre på att generalisera när det kommer till nya data.

### Variance

Variansen hos en modell står i direkt motsats till dess bias. En modell med hög varians innebär att den i hög grad följer träningsdatans mönster (Géron 2019, s.134). En sådan modell är ofta mer komplex rent matematiskt. Den tenderar att leda till overfitting, alltså passa träningsdatan bra men vara dålig på att generalisera nya data.

### Bias-Variance tradeoff

Eftersom bias och varians är varandras motsatser, ökar vi ena så minskar den andra. Denna relation kallas för bias-variance tradeoff (Géron, s. 134). Hos en modell vill man försöka hitta den perfekta balansen mellan dessa generaliseringsfel.

## Klassificeringsmodeller

Vi kommer att kolla på tre olika modeller För att få en bättre uppfattning om hur bra ML generellt funkar på MNIST databasen samt på datan i Streamlit-appen.

### Random Forest

Random Forest är en typ av maskininlärningsalgoritm som bygger på logiken bakom Decision Trees (Géron 2019, s.176–177, 197). Random Forest är en ensemble av Decision Trees som man tränar genom att använda ”bagging” (Géron 2019, s.192–193). Styrkan hos Random Forest-modeller är att de vid varje förgrening inte letar efter den allra bästa ”feature” i hela datasetet utan väljer ut den bästa feature bland en slumpmässigt utvald grupp av features. Detta innebär att vi får mer variation bland våra olika träd i ”skogen” vilket i sin tur leder till en minskad varians i modellen på bekostnad av en högre bias. Detta leder oftast till en mer träffsäker modell.

### Extra Trees

Extra Trees är en variant av Random Forest. Det som skiljer de åt är att vid varje förgrening så väljer Extra Trees den bästa feature från en slumpmässigt utvald grupp features men vi sätter även vårt gränsvärde till ett slumpmässigt värde (Géron 2019, s.198). Detta innebär att vi inte automatiskt letar efter det bästa gränsvärdet. På så sätt gör vi träden i vår skog ännu mer slumpmässiga vilket leder till minskad varians och högre bias hos modellen. Extra Trees är snabbare att träna än Random Forest eftersom modellen inte behöver hitta att perfekt gränsvärde vid varje förgrening.

### Linear SVM Classifier

En Linear Support Vector Machine (SVM) Classifier är en maskininlärningsmodell som kan användas för klassificeringsproblem. Modellen identifierar olika kluster i datan och strävar efter att dra en linje som befinner sig mitt emellan dessa kluster (Géron 2019, s.153). Linjen markerar gränsen mellan en kategori och en annan. Modellen predikterar sedan vilken kategori datapunkten tillhör genom att se vilken sida av linjen punkten befinner sig. Nedan bild (Géron 2019, s.154) illustrerar hur en Linear Support Vector Machine Classifier kategoriserar datan från två olika typer av blommor.

En bild som visar text, linje, skärmbild, Graf

Automatiskt genererad beskrivning

Figure 1: Visualisering av hur en SVM fungerar.

# Metod

I detta avsnitt kommer vi redogöra för arbetsgången och metodvalen som gjorts i detta projekt. Vi börjar med att gå igenom skapandet av en maskininlärningsmodell. Sedan kollar vi även på koden och metodvalen som gjorts för att skapa en Streamlit-app för sifferigenkänning.

## Skapa och välja maskininlärningsmodell

### Importera paket

Innan vi börjar med koden behöver vi först installera några paket i Python (fig. 2). Dessa paket innehåller metoder som vi behöver för att kunna använda oss av maskininlärningsmodeller etc.

### Databas

Datan vi använder oss av kommer från MNIST (Modified National Institute of Standards and Technology). Den består av 70 000 bilder av handskrivna siffror. Bilderna är standardiserade och har storleken 28x28 pixlar. Totalt har varje bild 784 *features* (oberoende variabler) och 1 *label* (beroende variabel) där man manuellt skrivit vilken siffra bilden föreställer.

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

Figure 2: Importerade paket Python

### Preparera data

Innan vi kan använda datan i modellerna behöver vi preparera den. Vi börjar med att dela upp datan i X och y, vår oberoende och beroende variabel (fig. 3).

Därefter delar vi datan i testdataset och tränings/valideringsdataset och sätter testdatasetet till totalt 10 000 bilder (fig. 4). Vi kan i nästa steg dela på det sammansatta tränings/valideringsdatasetet så att vi får ett separat träningsdataset och ett valideringsdataset som består av 10 000 bilder. Vi gör dessa steg med hjälp av funktionen *train\_test\_split.* Det är viktigt att vi delar upp datan på detta sätt innan vi börjar jobba med den (Géron 2019, s.30). Träningsdatan är den data som vi tränar modellen på, ”modellen lär sig” denna datan. På grund av detta kan vi inte använda träningsdatasetet för att utvärdera modellen. Därför har vi även ett testdataset. Detta dataset är okänt för modellen och vi kan då få en verklighetstrogen bild av hur modellen fungerar på okända data. Valideringsdataset är viktigt att ha om vi jämför olika modeller med varandra. Efter att vi tränat alla modeller på testdatan så kan vi validera dem på valideringsdatasetet. Den modell som presterar bäst på valideringsdatan tränar vi sedan om och testar på testdatan.

En bild som visar text, Teckensnitt, linje, vit

Automatiskt genererad beskrivning

Figure 3: Definierar våra variabler X och y

En bild som visar text, Teckensnitt, linje, skärmbild

Automatiskt genererad beskrivning

Figure 4: Dela upp våra data i tränings-, validerings- och testset.

### Träna modeller

Efter att vi har preparerat datan så kan vi börja med att träna modellerna på träningsdatan. När vi tränar modeller låter vi modellen få tillgång både till alla features och labels. Vi ger den alltså även svaren. På detta sätt kan modellen lära sig känna igen vilka olika värden på features som resulterar i olika värden på labels.

Här under tränar vi datan på vår första modell, Random Forest (fig. 5). Vi väljer hyperparameter n\_estimators och sätter den till 100. Detta innebär att vi i vår ”skog” har 100 olika ”träd”. Vi använder .fit för att träna modellen på träningsdatan.

Vi gör samma med våra modeller Extra Trees och Linear SVM Classifier.

### Accuracy score

För att kunna bedöma vilken av våra modeller som presterat bäst på datan behöver vi kunna mäta prestationerna. Vi gör det här med *.score* metoden som vid default har samma funktion som *accuracy\_score* funktionen. Båda metoder visar hur stor andel av prediktionerna som är korrekta. T.ex. om modellen gissar rätt på 8 utav 10 försök så kommer *accuracy\_score* vara 0,8.

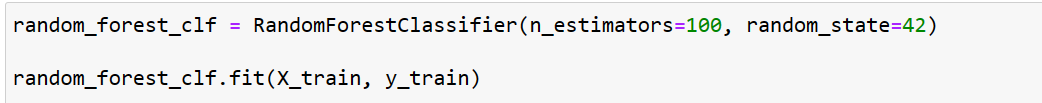


Figure 5: Tränar Random Forest på träningsdatan.



Figure 6: Beräknar score för Random Forest.

## Skapa en Streamlit-app

Efter att vi tränat våra modeller och valt ut den bästa är det nu dags för den andra delen av projektet, att skapa en Streamlit-app. Vi skapar appen för att kunna mata in egna handskriva siffror och testa modellens förmåga att prediktera dessa. Detta ger oss värdefulla insikter om hur generaliseringsbar vår modell är på data som inte kommer från MNIST-datasetet.

I Appendix A (fig. 11–13) finns hela koden för Streamlit-appen.

Första steget är som alltid att installera alla paket vi behöver för att skapa koden. Utöver olika Python-paket behöver vi även installera Streamlit vilket gjorts via Anaconda distributor. Vi laddar även in våra modeller som vi tränat genom att använda Joblib. Vi skapar en enkel layout för sidan så att det blir tydligare att se vilka funktioner som finns.

I nästa steg börjar vi med att skapa en funktion för att läsa in bilder som finns sparade på datorn. Detta är användbart om man tagit en bild med mobilen och sedan laddat upp den till datorn. Vi börjar med att skapa en st.file\_uploader som gör just detta. Sedan ser vi till att bilden laddas in som just en bild genom att använda imread. Vi gör om bilden till gråskala genom att använda rgb2gray. Vi behöver ha bilden i gråskala för att sedan kunna ha en skala av pixelvärden som vi analyserar (nyanser av grått). I nästa steg inverterar vi bilden så att det som tidigare var mörkt blir ljust och vice versa. Detta för att bilderna som maskinen är tränad på (MNIST) är i inverterat format. Maskinen skulle annars ha svårt att känna igen formatet och prediktera rätt. Vi skapar en gränsvärdesfunktion med threshold\_otsu. Denna funktion gör bilden binär, svart eller vita pixlar. Gränsvärdet bestäms av funktionen som hittar det optimala värdet för varje bild. Vi använder sedan img\_as\_ubyte för att göra om skalan hos pixelvärdet från 0–1 till 0–255. Detta för att modellen ska känna igen värdet på pixlarna. Vi gör om storleken på bilden så att den endast består av 28 gånger 28 pixlar. Detta för att efterlikna bilderna i MNIST. Slutligen så är det dags att passa modellen på bilden och prediktera vilken siffra det är. Detta gör vi med .predict.

Vi lägger till ytterligare en funktion hos appen. Istället för att ladda upp bilder skapar vi en kamera så att vi kan ta bilder direkt med kameran på datorn och prediktera dem. Vi skapar kameran med st.camera\_input. Sedan upprepar vi alla preprocess-steg som vi gjorde för bilder med vår file\_uploader.

(Genom koden används st.image för att visa bilden i appen i olika stadier av processeringen. Dessa kan ses i Appendix A (fig. 14–17).

# Resultat

Resultaten från träningen av modellerna visas nedantill:

|  |  |
| --- | --- |
| **Accuracy score för modellerna** | |
| Random Forest | 0,9692 |
| Extra Trees | 0,9715 |
| Linear SVC | 0,859 |

Tabell 1: Accuracy score för de tre valda modellerna.

Vi kan tydligt se att Extra Trees var den modell som presterade bäst på MNIST träningsdata.

När vi sedan använder denna modell i appen för nummerigenkänning så predikterar dock modellen fel på varje ny handskriven siffra som matas in. Detta demonstreras med exempel i nedanstående bild (fig. 7) som visar att en inmatad 3:a predikteras vara en 7:a av modellen. Vi får exakt samma resultat om vi istället använder vår näst bästa modell Random Forest, där en 3:a också predikteras vara en 7:a . När vi istället använder oss av den modell som presterade sämst på MNIST-datasetet, SVM så får vi korrekt prediktion, en 3:a predikteras vara en 3:a av modellen (fig. 8)! Resultatet av att lägga in fler bilder av olika siffror och låta SVM prediktera kan ses nedan i båda bilderna (fig. 9, 10). Modellen predikterar rätt även på dessa!

|  |  |
| --- | --- |
| En bild som visar skärmbild, design  Automatiskt genererad beskrivning  Figure 7: Extra Trees och Random Forest predikterar fel | En bild som visar skärmbild, Grafik, design  Automatiskt genererad beskrivning  Figure 8: SVM predikterar rätt. |
| En bild som visar skärmbild, skärm, text, cirkel  Automatiskt genererad beskrivning  Figure 9: SVM predikterar rätt. | En bild som visar skärmbild, Grafik, design  Automatiskt genererad beskrivning  Figure 10: SVM predikterar rätt. |

# Diskussion

Trots att detta projekt var uppdelat i två delar, där vi i den första tränar modellerna på MNIST data och i den andra testar modellerna på ny data, så var det inte helt väntat att det skulle vara en sådan stor skillnad i prestation hos modellerna på de olika delarna. Extra Trees och Random Forest presterade mycket bra på MNIST men funkade nästan inte alls på de handskrivna siffrorna som vi matade in i Streamlit-appen. Linear SVM Classifier presterade något sämre än de andra på MNIST-datan men fungerade nästintill felfritt på siffrorna i appen.

Detta skulle kunna bero på att modellerna Extra Trees och Random Forest lider av overfitting. D.v.s att modellerna är för anpassade till datan i MNIST för att vara bra på att generaliseras på nya egna handskrivna siffror trots att dessa siffror preparerats för att likna siffrorna i MNIST. Denna tes skulle vara intressant att utforska vidare samt om den visar sig stämma, försöka åtgärda så att modellerna blir bättre på att generalisera på ny data.

Slutligen hade det även varit intressant att kolla närmre på hur Linear SVM Classifier lyckas så bra med att generalisera från MNIST-datan till datan från appen.

# Slutsatser

Här nedan besvarar vi de frågor som ställdes i början av uppsatsen.

1. Vilken av de tre följande maskininlärningsmodellerna är bäst på att identifiera och kategorisera de handskrivna siffrorna i MNIST-databasen: Random Forest, Extra Trees eller Linear SVC?

* Extra Trees är den modell som är bäst på att identifiera och kategorisera datan i MNIST.

1. Vilken modell är bäst på att identifiera och kategorisera de nya handskrivna siffrorna som vi matar in i vår Streamlit-app för sifferigenkänning?

* Linear SVM Classifier är den modell som är bäst på att identifiera och kategorisera de handskrivna siffrorna som vi matar in i Streamlit-appen.

# Teoretiska frågor

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

- Träning är träningsdatan som vi tränar och passar modellen på. Validering är valideringsdata som vi använder för att välja bland flera modeller (som ett mini-test). Test är vår testdata, oftast 20% av totala datan, som vi slutligen testar vår bästa modell på.

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”. Hur skall hon välja vilken av de tre modellerna hon skall fortsätta använda när hon inte skapat ett explicit ”valideringsdataset”?

- Hon kan använda cross-validation och RMSE för de olika modeller och välja den som har lägst RMSE.

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

- I regressionsproblem har den beroende variabeln kontinuerliga värden. Linear regression, lasso regression, ridge regression, Support vector machines, decision trees och random forest. Tex. Om vi vill prediktera huspriser i olika områden etc så är regression bra att använda.

4. Hur kan du tolka RMSE och vad används det till: 𝑅𝑀𝑆𝐸 = √∑(𝑦𝑖 − 𝑦̂𝑖 ) 2 𝑖

- Root mean squared error (RMSE) är medelfelet hos vår modell jämfört med testdatan. Vi vill ha ett så lågt medelfel som möjligt. Detta är ett mått på hur träffsäker vår modell är.

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

- När vår data har diskreta värden och kan klassificeras in i olika klasser. Tex. Random Forest, KNNeighbors, Decision tree, Extra trees. Kan användas för att särskilja olika blommor. Confusion matrix används för att se hur träffsäker vår modell är. Den innehåller det predikterade värdet och det sanna värdet och jämför dem. Vi få fram olika mått såsom precision, recall och F1-score.

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

- K-means är en algortim som försöker hitta varje klusters centroid (observationen i mitten av klustret). Därefter klassificeras alla punkter till det kluster vars centrum är närmast.

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

- Ordinal encoding är när datan kan klassificeras genom att rangordnas (1a, 2a ….). One-hot encoding är ett sätt att göra kategorier binära. One-hot encoding körs som följande: röd (1,0,0), blå(0,1,0) och grön(0,0,1), maskinen kan på så sätt se vilken klass instansen tillhör. Dummy variable encoding innebär att en instans antagligen är (1) eller inte är (0).

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?

- Julia har rätt. Kontexten avgör om en variabel klassas som nominal eller ordinal.

9. Kolla följande video om Streamlit: https://www.youtube.com/watch?v=ggDaRzPP7A&list=PLgzaMbMPEHEx9Als3F3sKKXexWnyEKH45&index=12 Och besvara följande fråga: - Vad är Streamlit för något och vad kan det användas till?

- Streamlit är ett open-source ramverk som används i Python vid maskininlärning för att skapa dataapplikationer.

# Självutvärdering

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

* Kursen har varit krävande när det kommer till mängden material att gå igenom. Jag har lagt upp en tydlig studieplan under kursen för att kunna hänga med. Det har fungerat bra.
* Jag har haft utmaningar framförallt med skapandet av Streamlit-appen. Jag tacklade dessa utmaningar genom att felsöka och identifiera felet metodiskt, söka information online men framförallt prata med klasskamrater som själva stött på och hanterat problemet. Trots att detta var den mest utmanande delen då vi inte fick mycket guidning i hur vi skapar appen, så var det den mest lärorika delen, tack vare det goda utbytet med kursare.

1. Vilket betyg du anser att du skall ha och varför.   
   - Jag anser att jag ska ha betyget väl godkänt. Jag har genomfört både G-delen och VG-delen i kunskapskontrollen med gott resultat. Mer specifikt har jag redogjort för mina metodval samt lyft kritik och insikter mot dessa i metoddelen av rapporten men även i diskussionen som något att kolla vidare på.
2. Något du vill lyfta fram till Antonio?

* Inget speciellt. Utmanande kurs, men extremt givande.

# Appendix A

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

Figure 11 Importera paket och modeller samt fixa layout

En bild som visar text, skärmbild, programvara, Teckensnitt

Automatiskt genererad beskrivning Figure 12 Koden för uppladdning av bild

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

Figure 13 Koden för att ta en bild med kamera

En bild som visar skärmbild, text

Automatiskt genererad beskrivning

Figure 14 steg 1 i appen, ladda upp bild och visa den obehandlade bilden.

En bild som visar snö, svart och vit, vinter, monokrom

Automatiskt genererad beskrivning

Figure 15 Steg 2 i appen, visa bilden i gråskala.

En bild som visar skiss, rita, svart och vit

Automatiskt genererad beskrivning

Figure 16 steg 3 i appen, bilden i binärt format visas samt prediktion görs.

En bild som visar text, skärmbild, design

Automatiskt genererad beskrivning

Figure 17 Alternativ med kamera i appen.

# Källförteckning

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