Utforska maskininlärningsmodeller för dataklassificering med

MNIST-databasen



Ana Banic

EC Utbildning

Kunskapskontroll 2

202503

# Abstract

En kort sammanfattning över ditt arbete och de viktigaste resultaten skrivet på engelska, cirka 5 meningar totalt.

This report presents a comparative evaluation of four machine learning models for handwritten digit recognition using the MNIST dataset. The models considered include Random Forest, Extra Trees, Support Vector Machines (SVM) and Linear Support Vector Classifier, each trained on standardized data. Performance was measured in terms of both accuracy and computational efficiency. The results indicate that while the ensemble methods offer competitive accuracy with faster training times, SVM achieves high accuracy at the expense of longer training durations. Additionally, a Streamlit application was developed to allow users to obtain real-time predictions. These findings provide valuable insights into the trade-offs between accuracy and efficiency, supporting informed model selection for practical applications in computer vision.

# Förkortningar och Begrepp

AI = Artificiell Intelligens.

ML = Machine Learning.

RMSE = Root Mean Squared Error.

LinearSVC = Linear Support Vector Classifier.

MNIST = dataset with 70,000 handwritten digit images used to train and test machine learning models.

Random Forest Classifier = A technique that builds a collection of decision trees using random subsets of the data and features.

Extra Trees Classifier = A variation of the Random Forest method that introduces additional randomness when choosing split points for each node

SVM= Support Vector Machine

**Skapas automatiskt i Word genom att gå till Referenser > Innehållsförteckning.**

Innehållsförteckning

[Abstract 0](#_Toc160654338)

[1 Inledning 1](#_Toc160654340)

[1.1 Syfte 1](#_Toc160654341)

[2 Teori 2](#_Toc160654342)

[2.1 Random Forest 2](#_Toc160654343)

[2.2 Extra Trees 2](#_Toc160654344)

[2.3 Support Vector Machines 2](#_Toc160654345)

[2.4 Linear SVC Linear Support Vector Classifier 2](#_Toc160654346)

[3 Metod 3](#_Toc160654348)

[4 Resultat och Diskussion 4](#_Toc160654349)

4.1 Streamlit app……………………………………………………………………………………………………………………………..5

[5 Slutsatser……………………………………………………………………………………………………………………………………6](#_Toc160654350)

[6 Teoretiska frågor…………………………………………………………………………………………………………………………7](#_Toc160654351)

[7 Självutvärderin 9](#_Toc160654352)

[Appendix A 10](#_Toc160654353)

[Källförteckning 11](#_Toc160654354)

# Inledning

Ge en bred bakgrund till ditt arbete och varför det är relevant. Smalna successivt av och koppla bakgrunden till detta arbete. Fortsätt med syfte och frågeställning (syfte och frågeställning kan du ha i en egen underrubrik om du önskar det).

I dagens digitala tidsålder utgör maskininlärning och datorseende viktiga verktyg för att bearbeta och analysera stora datamängder, där handskriven sifferigenkänning är ett välkänt exempel på praktisk tillämpning. MNIST-datan, med sina tusentals bilder av handskrivna siffror, har länge använts som ett standardtest för att utvärdera och jämföra olika maskininlärningsmetoder. I den här arbetet har jag valt att fokusera på tre modeller:

I detta arbete fokuserar vi på **MNIST-datasetet**, ett klassiskt exempel på handskrivna siffror (0–9), där målet är att utveckla och jämföra olika modeller för automatisk klassificering av siffror. Genom att undersöka hur väl olika maskininlärningsmetoder – såsom Random Forest, Extra Trees, Support Vector Machines och Linear SVC – presterar på detta dataset, kan vi få en uppfattning om vilken metod som lämpar sig bäst för just den här typen av uppgift. Resultaten kan även ge en bredare insikt i hur val av algoritm och förbehandling påverkar klassificeringsprestanda i praktiken.

## Syfte

Syftet med denna rapport är att **utveckla** och **utvärdera** flera modeller för handskriven sifferigenkänning med hjälp av MNIST-datan. Genom att **träna**, **validera** och **testa** modellerna vill vi fastställa vilken metod som uppvisar bäst kombination av noggrannhet, beräkningstid och robusthet mot överanpassning.

Dessutom undersöks hur en **praktisk implementation** kan se ut, exempelvis i form av en Streamlit-applikation där användaren **ritar** en siffra direkt i webbläsaren, varefter modellen utför en förutsägelse.Genom detta arbete avser vi att besvara frågor såsom:

* Vilken av de testade algoritmerna (Random Forest, Extra Trees, SVM, Linear SVC) presterar bäst på MNIST när det gäller noggrannhet?
* Hur påverkar valideringsmetod (exempelvis korsvalidering) och datadelning (train/val/test) modellernas prestanda?
* Vilka är de viktigaste faktorerna att ta hänsyn till vid implementering av en praktisk lösning för handskriven sifferigenkänning?

Genom att besvara dessa frågor kan vi inte bara dra slutsatser om vilken modell som är mest lämplig för just MNIST, utan även ge vägledning för liknande klassificeringsproblem i andra sammanhang.

# Teori

## Random Forest classifier

#### En **Random Forest** konstruerar ett stort antal beslutsträd, där varje träd tränas på en slumpmässig underuppsättning av träningsdatan samt en slumpmässigt vald delmängd av variabler (features). Därefter aggregeras besluten från alla träd genom majoritetsröstning (majority voting). Detta tillvägagångssätt leder ofta till god robusthet mot överanpassning, eftersom olika träd kan kompensera för varandras fel.

#### I våra experiment på MNIST-datan uppnådde Random Forest en hög noggrannhet, särskilt på valideringsmängden där den nådde runt 97–98%. Detta visar att metoden lämpar sig väl för klassificering av handskrivna siffror.

## Extra Trees classifier

**Extra Trees** (Extremely Randomized Trees) påminner om Random Forest, men går ett steg längre i slumpmässigheten. Vid konstruktionen av varje träd väljs tröskelvärden för uppdelning (split points) helt slumpmässigt, vilket kan snabba upp träningen och ibland ge ytterligare robusthet mot överanpassning.

I våra korsvalideringsresultat på MNIST presterade Extra Trees mycket bra (kring 96–97% noggrannhet). På den separata valideringsmängden låg den dock strax under Random Forest, vilket antyder att även små skillnader i hur slumpen faller eller datan fördelas kan påverka den slutgiltiga modellen.

2.3 Support Vector Machines (SVM)

En **Support Vector Machine** försöker hitta den mest effektiva beslutsgränsen (ett hyperplan) i ett högdimensionellt rum, så att marginalen mellan klasserna maximeras. Vid användning av en icke-linjär kärna, som RBF (Radial Basis Function), kan modellen anpassa sig till mer komplexa beslutsytor.

I våra experiment var SVM (RBF) något mer tidskrävande att träna än Random Forest och Extra Trees. Noggrannheten låg omkring 96–97%, vilket är konkurrenskraftigt men aningen lägre än ensemblemetoderna på just vår valideringsmängd.

2.4 Linear SVC Linear Support Vector Classifier

**Linear SVC** är en variant av SVM som använder en linjär kärna för att skilja klasserna åt. Denna förenklade ansats är **snabbare** att träna jämfört med SVM med icke-linjära kärnor, men kan ha något lägre förmåga att fånga upp komplexa, icke-linjära samband i datan.  
I våra försök på MNIST gav Linear SVC en acceptabel noggrannhet (kring 96%) och snabbare träningstid än SVM (RBF). Dock nådde den inte riktigt upp till ensemblemetoderna Random Forest och Extra Trees i termer av slutgiltig prestanda

# Metod

Hur har du genomfört ditt arbete? Exempelvis, hur har datan erhållits?

Fyra olika modeller implementerades och jämfördes. Samtliga tränades på den **standardiserade** träningsmängden och utvärderades först med **3-faldig korsvalidering** (CV) för en preliminär jämförelse, och därefter på en separat **valideringsmängd** på 10 000 exempel.

1. **Random Forest (RF)**
   * En ensemblemetod som bygger flera beslutsträd (med n\_estimators=100) och aggregerar deras resultat för att minska variansen och öka träffsäkerheten.
   * Parameterinställningar som användes är random\_state=42 för reproducerbarhet och n\_jobs=-1 för att utnyttja flera processorkärnor.
2. **Extra Trees (ET)**
   * Liknar Random Forest men introducerar ytterligare slumpmässighet vid val av klyvningspunkter, vilket kan påskynda träningen och minska risken för överanpassning.
   * Även här användes n\_estimators=100 och random\_state=42, och data standardiserades på samma sätt som för Random Forest.
3. **Support Vector Machine (SVM)**
   * En kraftfull metod som försöker hitta en optimal beslutsgräns genom att projicera data i ett högdimensionellt rum med hjälp av en RBF-kärna.
   * Detta kan vara tidskrävande för stora dataset, men kan ge mycket god träffsäkerhet om parametrarna (t.ex. C och gamma) är väl inställda.
4. **Linear SVC (LSVC)**
   * Linear SVC (Linear Support Vector Classifier) försöker skilja klasserna åt med hjälp av ett linjärt hyperplan. Modellen använder en linjär kärna, vilket gör träningen mindre beräkningskrävande jämfört med en SVM som använder icke-linjära kärnor (t.ex. RBF). Har sattes C=0.1, max\_iter=1000, dual=False och random\_state=42 för att balansera överanpassning och generaliseringsförmåga.Precis som för de andra modellerna användes standardisering av features för att förbättra inlärningen, och parameterinställningar.

# Resultat och Diskussion

|  |  |
| --- | --- |
| **Validation Accuracy – noggranhet** | |
| Random forest | 0.9709 |
| Extra Trees | 0.9731 |
| Support Vector Machine | 0.9656 |
| Linear SVS | 0,9142 |

Tabell 1: Validation Accuracy – noggranhet för de fyra valda modellerna.

|  |  |
| --- | --- |
| **K fold Cross Validation ( k=5)** | |
| Random forest | Mean CV Accuracy = 0.9658 (+/- 0.0019) |
| Extra Trees | Mean CV Accuracy = 0.9684 (+/- 0.0014) |
| Support Vector Machine | Mean CV Accuracy = 0.9603 (+/- 0.0031) |
| Linear SVS | Mean CV Accuracy = 0.9082 (+/- 0.0027) |

Tabell 2: K fold Cross Validation för de fyra valda modellerna.

|  |  |
| --- | --- |
| **Testset accuracy score för bästa modell (Extra Trees)** | |
| Extra Trees | 0,9706 |

så är **ExtraTreesClassifier** bäst på testdatan med en noggrannhet (accuracy) på **0.9706**.

A graph with numbers and a number in the middle

AI-generated content may be incorrect.

A graph of a test

AI-generated content may be incorrect.Voting Classifier en **valideringsnoggrannhet** på cirka **0.9709** och en **testnoggrannhet** på **0.9680**, vilket ligger i linje med den bästa individuella modellen.

## Implementation av Streamlit-applikation

För att demonstrera och utvärdera vår tränade modell i praktiken skapades en **interaktiv Streamlit-applikation**. Syftet var att ge användaren möjlighet att **rita** en siffra direkt i webbläsaren och låta applikationen förutsäga vilken siffra det rör sig om.

För att visa en praktisk tillämpning av den tränade modellen byggdes en Streamlit-app där användaren kan rita en siffra direkt i webbläsaren. När användaren klickar på “Predict” behandlas den ritade bilden enligt följande steg:

1. Normalisering och omformning  
   Bilden från ritytan (192×192 pixlar) skalas ned till 28×28 pixlar och konverteras till gråskala. Därefter normaliseras pixelvärdena till intervallet [0,1][0,1][0,1].
2. Klassificering  
   Den förbehandlade bilden matas in i den förtränade modellen (exempelvis en CNN). Modellen returnerar sannolikheter för samtliga siffror (0–9), varpå den mest sannolika klassen utses till “Predicted Digit”.
3. Resultat och visualisering  
   Användaren får omedelbart se vilken siffra modellen förutspår. Dessutom visas en enkel stapeldiagram som illustrerar sannolikhetsfördelningen över de tio klasserna.

Genom denna interaktiva lösning kan man på ett intuitivt sätt testa och utvärdera hur väl modellen klarar av att känna igen handritade siffror, utan att behöva ladda upp separata bildfiler. Detta visar hur en maskininlärningsmodell kan integreras i ett användarvänligt webbgränssnitt, vilket underlättar för både demonstration och fortsatt utveckling.

**Sammanfattning**

För att demonstrera och utvärdera den tränade modellen på MNIST-data har vi utvecklat en **interaktiv Streamlit-applikation**. Användaren kan rita en siffra direkt i webbläsaren och få en **omedelbar** klassificering, utan att behöva ladda upp separata bildfiler. Lösningen bygger på en **konvolutionell neuronnätverksmodell** (CNN), vilket visar hur maskininlärning kan integreras i ett **användarvänligt** webbgränssnitt. På så sätt blir det enkelt att både **demonstrera** och **vidareutveckla** en praktisk tillämpning för handskriven sifferigenkänning.

A screenshot of a computer

AI-generated content may be incorrect.

# Slutsatser

Här besvarar du bl.a. frågeställningarna.

I detta projekt har vi använt MNIST-datamängden för att undersöka och jämföra prestandan hos flera maskininlärningsmodeller för handskriven sifferigenkänning. Genom att dela upp datan i tränings-, validerings- och testmängder samt använda StandardScaler för att normalisera pixelvärdena, skapades en robust pipeline som möjliggjorde en rättvis utvärdering av modellerna.

Följande modeller testades:

* **Random Forest**
* **Extra Trees**
* **SVM (RBF-kärna)**
* **Linear SVC**

Resultaten från både korsvalidering och utvärdering på en separat valideringsmängd visade att ensemblemetoderna (Random Forest och Extra Trees) generellt uppnådde högre noggrannhet och bättre robusthet mot överanpassning än SVM-baserade metoder. Bland dessa framstod Random Forest som den bästa modellen med avseende på noggrannhet.

Det är även värt att notera att enklare modeller ofta hamnar runt 80 % noggrannhet på MNIST-datan, vilket understryker vikten av att använda mer avancerade metoder och noggrann dataförbehandling för att nå de högre prestanda som krävs i praktiska tillämpningar.

För att demonstrera den praktiska tillämpningen av den tränade modellen har en interaktiv **Streamlit-applikation** utvecklats. I applikationen kan användaren rita en siffra direkt i webbläsaren. Den ritade bilden konverteras till en 28×28-gråskalebild, normaliseras och matas in i modellen, som omedelbart returnerar en förutsägelse. Detta visar hur en maskininlärningsmodell kan integreras i ett användarvänligt webbgränssnitt, vilket förenklar både demonstration och vidare utveckling.

Sammanfattningsvis visar studien att:

* Korrekt dataförbehandling och en tydlig uppdelning av datan är avgörande för att uppnå hög prestanda.
* Ensemblemetoder, särskilt Random Forest, är mycket effektiva för handskriven sifferigenkänning på MNIST.
* Den interaktiva Streamlit-applikationen illustrerar hur ML-modeller kan implementeras i realtid för att ge omedelbar feedback, vilket ökar användbarheten i praktiska applikationer.

Dessa slutsatser ger en solid grund för fortsatt arbete med både optimering av modeller och utveckling av användarvänliga ML-baserade applikationer.

# Teoretiska frågor

Besvara nedanstående teoretiska frågor koncist.

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

**Träningsdata** är den del av datan Kalle använder för att lära modellen hur den ska känna igen eller förutsäga det han är intresserad av. Modellen justerar sina interna vikter/parametrar med hjälp av denna data.

**Valideringsdata** är den del av datan Kalle använder för att kontinuerligt utvärdera modellen under träning. När han testar olika inställningar eller hyperparametrar (till exempel antalet neuroner i ett nätverk) kollar han hur modellen presterar på valideringsdatan. På så sätt kan han undvika att överanpassa sig till träningsdatan.

**Testdata** är den sista delen av datan som Kalle använder först när modellen är helt färdigtränad och vald. Den används för att göra en slutlig utvärdering och se hur modellen klarar sig på ny, ”osedd” data. Detta ger en rättvis bild av modellens faktiska prestanda.

1. 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 köra korsvalidering på träningsdatan för alla tre modellerna och sedan välja den som presterar bäst i snitt. På så sätt får hon en rättvis utvärdering även utan ett separat valideringsdataset.

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

Regressionsproblem innebär kortsagt att förutsäga ett kontinuerligt numeriskt värde. Vanliga modeller är linjär regression, lasso regression och random forest regression. Tillämpningar kan vara prissättning (t.ex. bostadspriser), ekonomiska prognoser och riskbedömning.

1. A math equation with numbers and symbols

   AI-generated content may be incorrect.Hur kan du tolka RMSE och vad används det till:

RMSE är ett mått på hur fel modellen förutspår i snitt. Ju lägre värde, desto bättre stämmer förutsägelserna med de verkliga värdena. Man använder det ofta för att jämföra olika modeller och se vilken som är mest träffsäker.

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

Klassificeringsproblem handlar om att förutspå vilken kategori eller klass något ska hamna i, som ”spam” eller ”inte spam”. Vanliga modeller är logistisk regression, beslutsträd, random forest och neurala nätverk. Du kan använda dem för allt från spamfilter och bildigenkänning till diagnosstöd i vården. En Confusion Matrix är en tabell som visar hur många exempel som klassades rätt respektive fel, så man tydligt ser om modellen blandar ihop vissa klasser.

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

K-means är en klustringsalgoritm som delar in data i grupper baserat på hur nära datapunkterna är varandra. Ett exempel är att segmentera kunder efter deras köpbeteende.

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

**Ordinal encoding**: Mappar kategorier till siffror baserat på en naturlig ordning. Exempelvis kan betyg "dålig", "medel", "bra" kodas som 0, 1, 2.

**One-hot encoding**: Skapar en separat binär kolumn för varje kategori. Om vi har färgerna röd, grön och blå blir "röd" = [1, 0, 0], "grön" = [0, 1, 0] osv.

**Dummy variable encoding**: Liknar one-hot encoding men man tar bort en kategori (referenskategori) för att undvika överflödig information. Om vi tar bort "röd" och bara har "grön" och "blå" så blir "röd" = [0, 0], "grön" = [1, 0] och "blå" = [0, 1].

1. 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, jag håller med henne. Det beror på hur du tolkar datan. I sig själva är färgerna nominala, men med en specifik tolkning kan de betraktas som ordinala.

1. 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 Python-ramverk som låter dig snabbt bygga interaktiva webbappar för dataanalys och ML-prototyper med bara några få rader kod.

# Självutvärdering

1. Utmaningar du haft under arbetet samt hur du hanterat dem.
2. Vilket betyg du anser att du skall ha och varför.
3. Något du vill lyfta fram till Antonio?

# Appendix A

# Källförteckning