Klassificering av Handskrivna Siffror med Maskininlärning

Lence Majzovska

EC Utbildning

Machine Learning

2025/03

Innehållsförteckning

1 Inledning

I denna rapport undersöks hur maskininlärning kan användas för att klassificera handskrivna siffror från bilder. För ändamålet används MNIST-datasetet, som består av 70 000 bilder av handskrivna siffror (0–9). MNIST-datasetet används ofta som referens för att jämföra och utvärdera olika maskininlärningsmodeller.

Tre modeller – ***Logistic Regression***, ***Random Forest***, ***Extra Trees*** – har valts ut för att utvärderas och jämföras med avseende på prediktionsnoggrannhet och generaliserbarhet. Dessa modeller skiljer sig åt i komplexitet och databehandling, vilket påverkar deras effektivitet och träffsäkerhet vid klassificering.

Rapporten undersöker även om den bäst presterande modellen kan implementeras i en interaktiv applikation byggd med Streamlit för realtidsklassificering av handskrivna siffror.

De centrala frågeställningarna som behandlas i rapporten är:

- Vilka åtgärder kan vidtas för att optimera modellernas förmåga att klassificera handskrivna siffror från MNIST-datasetet?
- Kan den bästa modellen ge en pålitlig realtidsklassificering i en praktisk applikation byggd med Streamlit?

2 Teori

I detta avsnitt introduceras centrala begrepp inom klassificering, optimering och modellval som använts i projektet. Fokus ligger på de valda modellerna och varför de är lämpliga för att lösa det aktuella klassificeringsproblemet.

2.1 Klassificeringsproblem

Ett klassificeringsproblem är en typ av maskininlärningsproblem där syftet är att förutsäga vilken kategori (klass) en observation tillhör, baserat på dess egenskaper, exempelvis om ett e-postmeddelande är spam eller inte.

Det finns två huvudsakliga typer:

1. Binär klassificering: Två möjliga utfall (JA/NEJ).
2. Multiclass-klassificering: Fler än två möjliga utfall (till exempel siffrorna 0–9 i MNIST).

2.2 Beskrivning av modeller

Logistic Regression är en binär klassificerare som omvandlar en linjär kombination av indata till en sannolikhet mellan 0 och 1 med hjälp av logistisk funktion.

Baserat på denna sannolikhet gör modellen en klassificering:

- Om sannolikheten är större än eller lika med 0,5, klassificeras observationen som tillhörande den positiva klassen.
- Om sannolikheten är mindre än 0,5, klassificeras observationen som tillhörande den negativa klassen.

Random Forest är en ensemblemodell som bygger på flera beslutsträd. Varje träd byggs utifrån en slumpmässig delmängd av träningsdatan. En viktig egenskap i Random är att träden inte alltid väljer den mest optimala variabeln vid varje split. Detta ökar variationen mellan träden, vilket gör modellen mer robust mot överanpassning och förbättrar dess generalisering på osedda data.

Extra Trees (Extremely Randomized Trees) är en variant av Random Forest där splitpunkterna väljs ännu mer slumpmässigt för att minska variansen och förbättra träningshastigheten. Den är också mer robust mot överanpassning genom att öka variationen mellan träden. I denna studie valdes Extra Trees för att jämföras med Random Forest, i syfte att undersöka om dess ökade slumpmässighet kan ge bättre prestanda på MNIST-datasetet.

3 Metod

Detta avsnitt beskriver projektets metodik – från datahantering till modellträning, optimering och utvärdering.

3.1 Datahantering

MNIST-datasetet (70 000 gråskalebilder, 28x28 pixlar) laddades ner från OpenML. Datasetet delades upp i träningsdata (80 %) och testdata (20 %) med ***stratifierad uppdelning*** som innebär att datasetet delas på ett sätt som säkerställer att fördelningen av målvariabeln (i detta fall siffrorna 0–9) är proportionellt representerad.

3.2 Modellval och träning

Tre modeller valdes ut för att jämföras i detta projekt. Modellerna tränades på samma dataset för att säkerställa en rättvis jämförelse.

3.2.1 Logistic Regression

En enkel, linjär modell som förutsäger sannolikheter. Eftersom modellen är känslig för skalan på funktionerna, tränades den med hjälp av en ***Pipeline*** som inkluderade:

- ***StandardScaler*** för att standardisera funktioner (medelvärde 0, standardavvikelse 1).
- *PCA* (Principal Component Analysis) för att minska datans dimensioner och snabba upp träningen.

Pipeline gör det möjligt att kedja flera steg (förbehandling + modellträning) i ett enhetligt flöde. Det säkerställer konsekvent hantering av både tränings- och testdata, samt minimerar risken för dataläckage.

3.2.2 Random Forest och Extra Trees

Som beskrivs i teoriavsnittet bygger både Random Forest och Extra Trees på beslutsträd. I detta projekt tränades modellerna direkt på den ursprungliga datan utan behov av skalning eller dimensionalitetsreduktion.

3.2.3 Hyperparameteroptimering

För att maximera modellernas prestanda användes Scikit-learns, ***GridSearchCV***, som utvärderar olika kombinationer av hyperparametrar baserat på valideringsresultat. Optimeringen genomfördes med 3-faldig korsvalidering för att säkerställa en tillförlitlig uppskattning av modellernas generaliseringsförmåga.

De bästa parametrarna som identifierades var:

- **Logistic Regression:** C = 0.1
- **Random Forest:** n_estimators = 300, max_depth = 30
- **Extra Trees:** n_estimators = 300, max_depth = 30

3.3 Modellutvärdering

Modellerna jämfördes baserat på valideringsnoggrannhet. Den bäst presterande modellen, ***Extra Trees***, valdes därefter för utvärdering på separat testdata för att mäta generaliseringsförmågan.

3.4 Implementering av Streamlit-applikationen

Extra Trees implementerades i en interaktiv Streamlit-applikation. Användaren kan rita en siffra direkt i gränssnittet och få en förutsägelse i realtid. Bilden förbehandlas genom att konverteras till gråskala, inverteras, skalas om till 28x28 pixlar, kontrasten förstärks och därefter binariseras innan den skickas till modellen.

Figur 1. Gränssnittet i Streamlit-applikationen där användaren kan rita en siffra och få en förutsägelse i realtid.

4 Resultat och diskussion

Detta avsnitt presenterar resultaten från modellernas utvärdering på valideringsdatan och testdatan.

4.1 Valideringsresultat

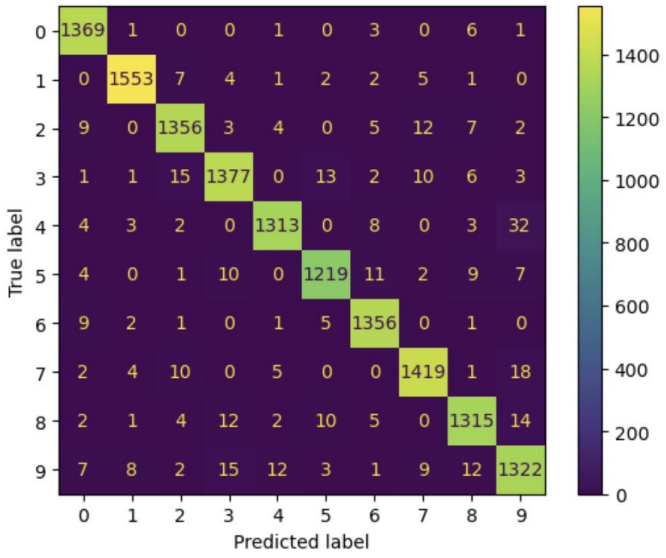
Extra Trees presterade bäst enligt valideringsresultaten och testades därför vidare. Valideringsnoggrannheten sammanfattas i tabellen nedan:

Modell	Accuracy (validering)
Logistic Regression	91,7 %
Random Forest	96,7 %
Extra Trees	**97,1 %**

Figur 2. Jämförelse av valideringsnoggrannhet mellan modellerna Logistic Regression, Random Forest och Extra Trees baserat på valideringsdata.

4.2 Testresultat för Extra Trees

Extra Trees testades på en separat testuppsättning, vilket resulterade i en testnoggrannhet på 97,1 %. Modellen visade stark generaliseringsförmåga och lyckades väl med att klassificera även tidigare osedda bilder. Ökad slumpmässighet förbättrar trädens variation och minskar överanpassning, vilket återspeglas i förvirringsmatrisen nedan.



Figur 3. Förvirringsmatris för Extra Trees på testdatan. Diagonalen visar antalet korrekta klassificeringar per siffra, medan övriga rutor visar felklassificeringar.

5 Slutsatser

Baserat på resultaten valdes Extra Trees som den mest lämpliga modellen för klassificering av handskrivna siffror i MNIST-datasetet. Den uppnådde högst noggrannhet vid både validering och testning, vilket tyder på god generaliseringsförmåga.

Logistic Regression användes som en enkel baslinjemodell och presterade stabilt men förväntat sämre. Random Forest presterade väl, men Extra Trees överträffade den tack vare sin ökade slumpmässighet och förbättrade variation.

5.1 Preprocessingens betydelse

Bildförbehandlingen spelade en avgörande roll för modellens prestanda. Genom att förbättra kontrast och reducera brus kunde modellen lättare tolka siffrornas form, vilket ökade träffsäkerheten.

5.2 Modellens praktiska tillämpning

Streamlit-applikationen demonstrerade att modellen kunde användas för snabba och tillförlitliga förutsägelser i realtid. Den användarvänliga designen underlättade interaktion och illustrerade hur modellen kan tillämpas i praktiska sammanhang.

5.3 Framtida arbete

Det finns flera möjliga förbättringsområden för fortsatt utveckling:

- Förbättrad skalning och normalisering för ökad precision.
- Utforskning av djupinlärning för att förbättra noggrannheten ytterligare.
- Förbättrad preprocessing för att tydliggöra siffrorna och minska brus.

6 Teoretiska frågor

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

Träningsdata används för att skapa och träna modeller.
Valideringsdata utvärderar modeller för att välja den som presterat bäst.
Testdata används för att testa den bästa modellen på testdatan, efter att den tränats om på tränings- och valideringsdatan, för att få en uppskattning på modellens generaliseringsförmåga på ny data.

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 ”validerings dataset”?

Cross-validation ersätter ett separat valideringsset och kan användas på träningsdatan för att jämföra modellerna och välja den som ger bäst resultat.

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

Regressionsproblem förutsäger ett kontinuerligt numeriskt värde, den beroende variabeln y. Exempel på modeller: Linjär Regression, SVM (Support Vector Machines), Beslutsträd. Tillämpningsområden kan exempelvis vara att förutsäga en persons inkomst baserat på ålder och utbildning.

4. Hur kan du tolka RMSE och vad används det till:

$$RMSE = \sqrt{\frac{1}{n}\sum_{1}^{n}(y_i - \hat{y}_i)^2}$$

RMSE (Root Mean Squared Error) mäter hur nära modellens förutsägelser ligger de faktiska värdena vid regressionsproblem.

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

Klassificeringsproblem skattar sannolikheter, modellen ska förutsäga vilken kategori (klass) en observation tillhör. **Exempel på modeller:** Logistic Regression, Random Forest, Support Vector Machines (SVM). **Exempel på tillämpning:** att identifiera om ett e-postmeddelande är spam eller inte, eller att klassificera handskrivna siffror i bilddata.

Confusion Matrix är ett verktyg för att utvärdera klassificeringsmodeller. Den visar modellens resultat i två dimensioner – verkliga och predikterade klasser – och innehåller värden för true positives, true negatives, false positives och false negatives.

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

K-means grupperar data i kluster och används för unlabeled dataset. Algoritmen försöker hitta varje klusters centrum (centroid) och tilldelar alla punkter till det kluster vars centrum är närmast. Exempel på tillämpning: Kundsegmentering där kunder grupperas efter ålder, kön, inkomst, köpbeteende för att möjliggöra riktad marknadsföring.

7. Förklara (gärna med ett exempel): Ordinal encoding, one-hot encoding, dummy variable encoding.

Ordinal encoding omvandlar kategoriska variabler till numeriska värden som kan användas för modeller som kräver numeriska data.

Exempel:

[0] - Low
[1] - Medium
[2] - High

One-hot encoding används när kategorierna inte har någon ordning, skapar en ny kolumn för varje kategori och markerar närvaron med en etta och frånvaron med nollor.

Exempel:

[1, 0, 0] - Blå
[0, 1, 0] - Grön
[0, 0, 1] - Röd

Dummy variable encoding liknar one–hot encoding men tar bort en kolumn då vissa modeller har svårt att hantera *one-hot*-kodade kategorier. Om [1, 0, 0] representerar ”blå” och [0, 1, 0] representerar ”grön”, behöver vi inte ytterligare en binär variabel för att representera ”röd”. Istället kan vi använda [0, 0] för att indikera att det varken är ”blå” eller ”grön”, vilket då innebär ”röd”.

Exempel:

[1, 0] - Blå
[0, 1] - Grön
[0, 0] - Röd

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 – Datatyper som ordinal och nominal beror på hur datan tolkas i ett sammanhang.

9. Vad är Streamlit för något och vad kan det användas till?

Streamlit är ett öppet ramverk för Python som används för att skapa webbaserade applikationer inom maskininlärning.

7 Självutvärdering

Jag känner mig stolt över vad jag har åstadkommit under det här projektet, och jag anser att jag uppfyller kraven för VG.

- Jag har besvarat de teoretiska frågorna, modellerat MNIST-datan, skrivit rapport samt implementerat en modell i en fungerande Streamlit-applikation. Under projektets gång har jag fördjupat min förståelse för maskininlärningens arbetsflöde – från databehandling och modellträning till utvärdering och praktisk tillämpning.

- Jag har reflekterat över både mina misstag och framgångar, och lärt mig mycket om den tekniska processen och mig själv under arbetets gång. Jag har också insett vikten av att börja i god tid för att ge mig själv mer utrymme att hantera oväntade utmaningar. Projektet har varit utmanande men lärorikt och lärt mig att hantera komplexa problem samt att jag behöver arbeta mer strukturerat under press.

Appendix A

A.1 Hyperparametrar för modeller

Nedan listas de hyperparametrar som testades för varje modell under optimeringen:

- Logistic Regression
  C: [0.1, 1, 2, 5]
  Regleringens styrka. Låga värden innebär starkare reglering, medan högre värden ger en mer flexibel modell.
- Random Forest
  n_estimators: [100, 200, 300]
  Antal träd i skogen. Fler träd kan förbättra modellen men ökar också beräkningstiden.
  max_depth: [10, 20, 30]
  Maximalt djup på varje träd. Djupare träd kan överanpassa datan om de inte optimeras ordentligt.
- Extra Trees
  n_estimators: [100, 200, 300]
  Antal träd i modellen. Fler träd kan förbättra precisionen men medför högre beräkningskostnader.
  max_depth: [10, 20, 30]
  Maximalt djup på träden, som styr hur komplexa mönster modellen kan lära sig.

A.2 Länkar till projektet

Streamlit-applikation
https://ds24ml-8r59cjjwdshqsdstsrp7ig.streamlit.app

Projektets GitHub-repo
https://github.com/lencemajzovska/ds24_ml.git

Källförteckning

EC Utbildning. *Kursmaterial, föreläsningar och presentationer från kursen Machine Learning.* Hämtad mars 2025 från skolans lärplattform.

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

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830. Hämtad mars 2025 från https://scikit-learn.org/stable/index.html