# Maskininlärning och siffror


Karl Tengström

EC Utbildning

2025-03

## Abstract

In this report the process of implementing a machine learning model is explored for the purpose of identifying numbers drawn a web-application developed with Streamlit. The model used is called Random Forest Classifier and the dataset used to train it is named MNIST.

# Innehållsförteckning

# 1 Inledning

Inom data sience är machine learning ett användbart verktyg för att angripa problem kring mönster och igenkänning. Genom att ge en modell data att träna sig på lär den sig att identifiera mönster och göra förutsägelser på ny, okänd data. Träningen sker genom att använda maskininlärningsmodeller i kombination med programmering och data.

En vanlig uppgift för blivande data scientists är att försöka skapa en modell som kan känna igen siffror, vanligtvis med datasetet MNIST som träningsdata. Syftet med denna rapport är att undersöka hur denna process kan se ut, och visa hur maskininlärning går att använda för att känna igen siffror som en användare ritat i en webapplikation. För att uppfylla syftet kommer följande frågeställning besvaras:

1. Kan maskininlärningsmodeller tränas till att göra korrekta prediktioner på 95% (vanligt förekommande värde i sannolikhetslära) av testdatan i MNIST?
2. Kommer samma modell kunna känna igen siffror ritade i en web-applikationen skapad med Streamlit?

# 2 Teori & Begrepp

## 2.1 MNIST

MNIST (Modified National Institute of Standards and Technology)  är ett dataset som är vanligt att använda inom maskininlärning och bildigenkänning. Datasetet består av 70 000 bilder på handskrivna siffror (0 till 9) i gråskala (Wikipedia$_1$, 2025)

## 2.2 Maskininlärningsmodeller

Maskininlärningsmodeller delas huvudsakligen upp i två kategorier: övervakad inlärning och oövervakad inlärning (men fler sätt att kategorisera finns). För detta arbete har övervakad inlärning använts. Övervakad inlärning innebär att den data som används för att träna modellen har ”etiketter”, dvs vi berättar för modellen redan i början vad datan representerar. När modellen sen är färdigtränad så används den på okänd data för att göra förutsägelser. (EduicationalTopicsExplained, 2025)

#Antonios youtubevideo

## 2.3 Scikit-learn

Scikit-learn är ett open-source bibliotek till Python för maskininlärning. Det använder sig bland annat av Numpy, SciPy och Matplotlib för att skapa förhållandevis enkla och effektiva verktyg för att arbeta med maskininlärning. Det ses som ett av de mest populära biblioteken för maskininlärning (Wikipedia$_2$, 2025)

## 2.4 Accuracy

För att utvärdera modellerna i detta arbete används Accuracy score från Scikit-learn. Måttet beskriver hur många klassifikationer som blir korrekta i form av en kvot. Om alla klassifikationer är korrekta blir kvoten 1, om alla är fel blir den 0 (Scikit-learn, 2025)

## 2.5 Streamlit

Streamlit är ett framework för Python som möjliggör ett enkelt skapande och delande av web-applikationer. I denna rapport så kommer dess Drawable Canvas användas, vilket möjliggör att användare kan rita bilder i applikationen.

# 3 Metod

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

## 3.1 Data

För att träna en maskininlärningsmodell behövs data, och för detta arbete så användes datasetet MNIST. Det importerades som en del av pythonbiblioteket Scikit-learn. Vid import består datasetet av 70.000 bilder på handskriva siffror. Till varje bild finns också en label (”target”) som berättar vilken siffra bilden visar. Detta möjliggör att vi kan använda oss av övervakad inlärning.

Vid importering så är varje bild en array med shape 28 x 28, där varje element har ett värde mellan 0 och 255 som beskriver dess pixelvärde. För att normalisera datan och underlätta framtida processering så delas pixelvärderna på 255 och formen på arrays blir 784 x 1. Datan delas sen upp i tre delar: train/validation/test, med en 50.000/10.000/10.000-fördelning.

## 3.2 Modeller

Fyra olika modeller testades på träningsdatan: LogisticRegression, RandomForestClassifier, ExtraTreesClassifier samt en VotingClassifier (som använde sig av de tre första modellerna). För jämförelse användes en ConfusionMatrix samt Accuracy. Tre av modellerna presterade väldigt lika (RandomForest, ExtraTrees och VotingClassifier) medan LogisticRegression hade något sämre värden på Accuracy (se tabell X under resultat och diskussion). Då RandomForest hade kortare körtid än de andra två valdes denna som modell att gå vidare med.

## 3.3 Finjustering av modellen

För att ytterligare finjustera RandomForest så genomfördes en GridSearch för optimering av hyperparametrarna ”n_estimators”, ”max_depth” och ”min_sample_split”. Sedan tränades RandomForest om på tränings och valideringsdatan gemensamt och testades mot test-datan. Som ett sista steg innan modellen sparades ner inför användandet i Streamlit så tränades den om på det totala datasetet (träning, validering och test), då detta sågs som produktionssättning av modellen.

## 3.4 Streamlit och pre-processing

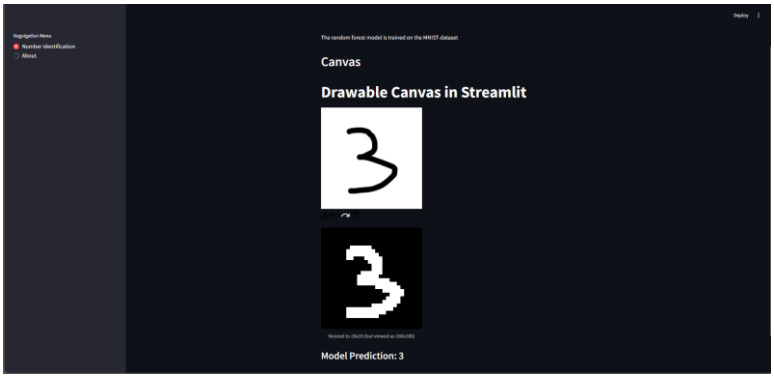
För att spara (och sen ladda in) modellen i Streamlit användes biblioteket ”Joblib”. I Streamlit så skapades en canvas som användaren av appen kan rita på. För att möjliggöra analys av bilden på canvasen med modellen så behöver det som ritas pre-processas.

För pre-processing av bilden från Streamlit-canvasen så används biblioteket OpenCV, vilket importerats som cv2. De steg som genomförs för att preprocessa bilden är följande:

1. Omvandla bilden till en Numpy array med pixelvärden. Då denna array innehåller mer information än nödvändigt så väljs endast RGB-kanalerna. Dessa konverteras sen till gråskala (cv2.cvtColor)
2. Bilden skalas ner till 28 x 28 pixlar (cv2.resize), för att stämma överens med storleken på bilderna modellen tränats på. Då MNIST är vita siffror mot svart bakgrund och vår canvas har vit bakgrund och ritar med svart så får värderna inverteras (cv2.bitwise_not).
3. Ökar kontrasten i bilden (cv2.treshhold) samt förstärker det som ritats (cv2.dilate).
4. Avsultningsvis ändras formen på arrayn så att det stämmer överens med hur modellen tränats (alla värden i en kolumn).

## 3.5 Applikationen

För att användare ska kunna rita siffror som sen modellen gör prediktioner kring så skapas en enklare Streamlit applikation med en canvas på. Användare ritar på den vita rutan och resultatet av pre-processing visas under. Längst ner skrivs prediktion ut.



# 4 Resultat och Diskussion

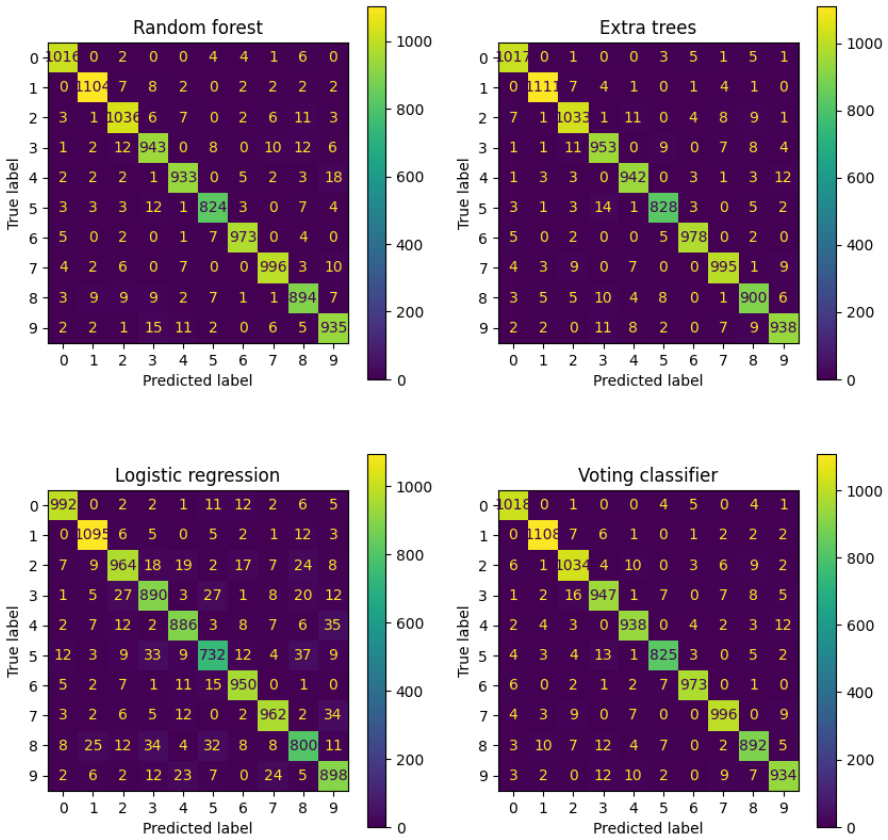
## 4.1 Modellernas prestationer

För att jämföra de fyra modellerna användes Accuracy-score för samt ConfusionMatrix. Logistic Regression valdes bort då den hade ett sämre accuracy-score. Voting Classifiers körtid valdes bort pga högre körtiden. Då ExtraTrees och RandomForest presterade så lika valdes Random Forest då skribenten kände sig mer bekväm med den modellen och tydligare underlag för hjälp fanns att tillgå.

Då det finns en tradeoff i bias-variance mellan dessa borde detta ligga till grund för framtida försök att optimera arbetet.

| **Accuracy score för olika modeller** | | |
|---|---|---|
| Modellnamn | Accuracy score | Körtid (sec) |
| Logistic Regression | 0.9168 | 54.6 |
| Extra Trees Classifier | 0.9695 | 42.5 |
| Random Forest Classifier | 0.9654 | 47.1 |
| Voting Classifier | 0.9665 | 151.8 |

*Tabell 1: Accuracy-score för de fyra valda modellerna.*




*Figur 2: ConfusionMatrix för de fyra valda modellerna.*

## 4.2 Hyperparametrar för RandomForest

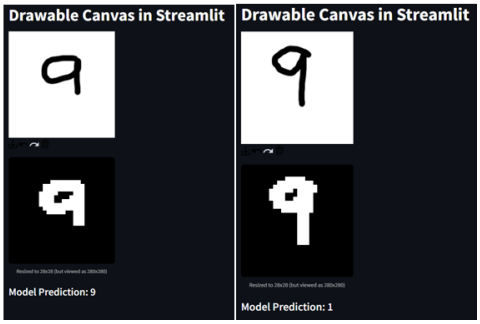
De hyperparametrar som jämfördes för RandomForest presenteras i tabell 2 nedan. Med mer tid hade dessa kunnat utforskas ytterligare.

| **Hyperparametrar RandomForest** | | |
|---|---|---|
| Hyperparameter | Alternativ | Valt värde |
| n_estimator | 50, 100 | 100 |
| max_depth | 20, 30, 40 | 30 |
| min_samples_split | 2, 5, 10 | 2 |

*Tabell 2: Hyperparametrar för RandomForest.*

## 4.3 Applikationen och pre-processing

Överlag fungerar applikationen och modellen okej. De flesta siffrorna blir korrekt predikterade men vissa av dom har den mycket svårare för (exempelvis 9, se figur X nedan).




Det finns mycket som kan förbättras kring pre-processandet som troligtvis hade lett till bättre prediktioner:

1. För att bättre efterlikna hur bilderna är sparade i MNIST borde siffran skalas ner till 20x20 (och inte 28x28) för att sen placeras i centrum på en 28x28 bakgrund.
2. Använda Otsu för att bättre känna av vilket treshhold som ger effekt när bilden ska skärpas (gråa pixlar transformeras till antingen svarta eller vita pixlar).
3. För att underlätta arbetet med koden borde de tranformationer som görs i pre-processingen varit skrivna som egna funktioner istället för raka kodanrop.

# 5 Slutsatser

Målet med denna rapport var att visa hur processen kan se ut när en maskininlärningsmodell ska tränas och sedan användas i ett faktiskt syfte (i detta fall sifferigenkänning). De frågeställningar som rapporten ämnade att besvara var:

1. Kan maskininlärningsmodeller tränas till att göra korrekta prediktioner på 95% (vanligt förekommande värde i sannolikhetslära) av träningsdatan?
2. Kommer samma modell kunna känna igen siffror ritade i web-applikationen Streamlit?

Som vi kunde se i tabell 1 så lyckads flera modeller nå upp till den nivå (95%) vi tänkte oss gällande säkerheten på prediktionerna av MNIST. RandomForest, den modell vi valde att gå vidare med, gick också att använda för att känna igen siffror som ritades av användare i Streamlit applikationen, men den hade svårare för vissa siffror. Med mer pre-processing hade det troligtvis gått att öka dess säkerhet.

# 6 Teoretiska frågor

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

   Delen ”träning” används för att träna den/de modeller du vill jämföra. När modellerna är tränade så används ”validering” för att testa hur väl modellerna fungerar mot data som inte varit en del av träningen. När den bäst fungerande modellen för ändamålet valts så tränas modellen om på ”träning” och ”validerings”-datan tillsammans, för att i ett sista steg testas mot ”test”-datan. ”Test” ska direkt plockas undan från ens dataset så att den aldrig riskerar att bli en del av träningen.

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”?*

   Förslagsvis använder Julia K-fold cross validation. Det innebär att träningsdatan delas upp i ett angivet antal ”n” lika stora slumpade bitar (vanligtvis 5 eller 10). Sen används (n-1) bitar för träning, och ”valideras” mot den återstående biten. Detta upprepas så att så att alla ”n” bitar används för validering var för sig, och medelvärdet av validerings-score blir modellens score.

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

   Regressionsproblem är en typ av problem där vi skapar modeller för att förutspå så bra värden som möjligt till en beroende variabel. Värdena som denna variabel kan anta ska vara kontinuerliga. Vanliga modeller för att uppnå detta är ”linjär regression”, ”SVM” (Support vector machine) och ”beslutsträd”. Ett regressionsproblem skulle till exempel kunna vara att vi har data med ålder och längd på en grupp barn. Då skulle vi kunna träna en modell i att förutspå ålder på barnen utifrån deras längd.

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) är ett mått på hur väl en regressionsmodell passar ett dataset. Detta gör vi genom att använda vår ”test” data som läggs undan innan vi börjar träna modellen. Eftersom vi vet dom rätta värdena ($y_{true}$) utifrån data x så kan vi jämföra dessa med värden ($y_{pred}$) som vår modell ger när den får arbeta med data x. RMSE tar och kvadrerar avståndet mellan $y_{true}$ och $y_{pred}$ (för att det inte ska spela roll om $y_{pred}$ blev högre eller lägre än $y_{true}$. Den tar sedan och beräknar medelvärdet av alla dessa kvadrerade avstånd (för att vi ska få ett ”medel-fel”) och avslutningsvis tar vi roten ur detta för att få en korrekt enhet på det jämfört med datan. Vi ska alltså tolka RMSE som ett mått på medelfelet för vår model, och genom att jämföra RMSE för olika modeller kan vi bilda oss en uppfattning över vilken som fungerar bäst.

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

Klassifieringsproblem är en typ av problem där vi delar in olika resultat i klasser. Exempelvis kan det handla om binära klassifieringar, dvs att det finns två klasser (ja/nej, man/kvinna, spam/inte spam). Med hjälp av klassifieringar kan tex företag uppnå en högre förståelse för kunder och trender hos dessa (vilka kunder handlar mer och när, eller vilka kunder riskerar att byta leverantör mm).
Vanliga modeller som används för detta är t.ex Logisk regression och beslutsträd.
Vid klassifieringsproblem så är det vanligt att visualisera resultaten med en Confusion Matrix. I en confusion matrix får faktiska och predikterade resultat vara våra dimensioner (ex faktiska resultat ses som rader, predikterade som kolumner). En confusion matrix kan också användas för att illustrera hur vi beräknar måtten ”precision” och ”recall”.

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

Kluster används inom unsupervised learning på ett liknande sätt som klassifieringar. Genom att klustra data så kan vi hitta trender och gruperingar som hjälper oss att analysera datan och dra slutsatser, det kan också hjälpa oss att identifiera avvikande datapunkter eller för att ytterligare beskriva/visualisera data.
Med K-means kan vi träna modeller för att skapa kluster. Genom att iterativt placera och optimera klustercentrum hamnar dessa snabbt på effektiva platser. Metoden är dock inte ofelbar, då K-means (om inte annat anges) börjar med att slumpa inte klusternas startpositioner (och dåliga startpositioner kan resultera i felaktiga klusercentrum). K-means kräver att vi anger hur många klustercentrum där ska finnas.

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

Kategorisk data kan delas upp i ordinal och nominal data. Ordinal data innebär att det finns en hierarki mellan kategorierna, exempelvis svårigheten på skidbackar (där den lättaste är grön, följt av blå, röd och sist svart som är svårast). Ordinal encoding innebär att vi översätter datan till siffror och behåller hierarkin. Se tabell som beskriver några skidbackar hos Idre-fjäll.

| Backens namn | Klassifiering | Ordinal encoded |
|---|---|---|
| Snövit | Grön | 0 |
| Västbacken | Röd | 2 |
| Glader | Blå | 1 |
| Toker | Svart | 3 |

Alternativa sätt att arbeta om datan på är att använda one-hot encoding och dummy variable, som är väldigt lika varandra. Vid dessa så används klassifieringen som kolumnnamn (vi får alltså en ny kolumn för varje klassifiering) och sedan används ettor och nollor för att indikera som ett record (en rad) hamnar under en viss klassifiering eller inte. I tabell 2 visas hur one-hot encoding hade sett ut.

| Backens namn | Grön | Blå | Röd | Svart |
|---|---|---|---|---|
| Snövit | 1 | 0 | 0 | 0 |
| Västbacken | 0 | 0 | 1 | 0 |
| Glader | 0 | 1 | 0 | 0 |
| Toker | 0 | 0 | 0 | 1 |

Skillnaden mellan one-hot encoding och dummy variable är att dummy variable använder sig av logiken att "om en record inte hör till någon av de första tre kategorierna så måste den höra till den fjärde". I tabell 3 ser vi ett exempel på detta, där backen toker istället för att betecknas som 0001 (enligt one-hot encoding) representeras av 000.

| Backens namn | Grön | Blå | Röd |
|---|---|---|---|
| Snövit | 1 | 0 | 0 |
| Västbacken | 0 | 0 | 1 |
| Glader | 0 | 1 | 0 |
| Toker | 0 | 0 | 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?*

   Som Julia säger så spelar kontexten roll. Utan någon kontext så är färgerna nominala, dom har ingen hierarki, men med hennes exempel (eller ännu tydligare enligt mig, med skidbackarnas svårighetsgrad) så finns blir de ordinala. Det beror alltså på vad datan representerar.

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 framework för Python som möjliggör ett enkelt skapande och delande av web-applikationer. Det fungerar bra att skapa dashboards/interaktiva grafer/interagera med maskininlärningsmodeller.

# 7 Självutvärdering

1. Utmaningar du haft under arbetet samt hur du hanterat dem.
   Under stora delar av januari och februari har det varit problem med sjukdom för mig och familjen. Det har lett till att väldigt begränsat med tid har kunnat läggas på arbetet (ca 10h/vecka har gått till studier, resten har försvunnit). Det har lett till grova förenklingar och genvägar i koden och analys.

2. Vilket betyg du anser att du skall ha och varför.
   Jag anser att jag nått målen för VG som dom är skrivna. Men jag är inte nöjd med kvalitén på det jag gjort.

3. Något du vill lyfta fram till Antonio?
   Mycket intressant kurs, och ett ämne jag hoppas kunna fortsätta att fördjupa mig inom på fritiden.

# Källförteckning


Wikipedia$_1$ (2025). *MNIST database*. Hämtat 20 mars, 2025, https://en.wikipedia.org/wiki/MNIST_database

Wikipedia$_2$ (2025). *Scikit-learn*. Hämtat 20 mars, 2025, https://en.wikipedia.org/wiki/Scikit-learn

Scikit-learn. (2025). *Accuracy score.* Hämtat 20 mars, 2025, https://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score

EducationalTopicsExplained. (2025). Introduktion till Maskininlärning [Video]. Youtube. https://youtu.be/zORv5vxrwok?si=CdfEUndOq5TKz6o9