

EC-Utbildning DS2023  
Malmö-Helsingborg

# Maskininlärning Inlämningsuppgift Rapport

---

Xiaowen Chen  
Date: 2024-03-22

---

---

# Contents

<b>Chapter: 1</b>	<b>Inledning . . . . .</b>	<b>1</b>
<b>Chapter: 2</b>	<b>Datauppsättningsbeskrivning: . . . . .</b>	<b>2</b>
<b>Chapter: 3</b>	<b>Dataförbehandling: . . . . .</b>	<b>3</b>
<b>Chapter: 4</b>	<b>Val av modell . . . . .</b>	<b>4</b>
	Logistisk regression: . . . . .	4
	Random Forest: . . . . .	4
	Support Vector Machine (SVM): . . . . .	5
<b>Chapter: 5</b>	<b>Modellutbildning och utvärdering: . . . . .</b>	<b>6</b>
	Modellträning: . . . . .	6
	Modellutvärdering: . . . . .	6
<b>Chapter: 6</b>	<b>Resultatanalys: . . . . .</b>	<b>7</b>
	Modellprestanda: . . . . .	7
	Felanalys: . . . . .	7
<b>Chapter: 7</b>	<b>Använder Streamlit som ett visuellt testverktyg . . . . .</b>	<b>8</b>
	Streamlit operation: . . . . .	8
	Förutsägelse och jämförelse resultat av olika modeller . . . . .	10
<b>Chapter: 8</b>	<b>Sammanfattningsvis . . . . .</b>	<b>12</b>

# 1. Inledning

**Bakgrund:** Den här uppgiften handlar om handskriven sifferigenkänning. Handskriven sifferigenkänning är ett klassiskt bildklassificeringsproblem med ett brett spektrum av tillämpningsområden, inklusive datorseende, maskininläring och artificiell intelligens.

**Syftet med denna uppgift:** Denna uppgift syftar till att utveckla ett system som automatiskt kan känna igen siffror i handskrivna siffror. Givet en bild av handskrivna siffror bör systemet kunna identifiera siffrorna i bilden exakt och klassificera dem i kategorier från 0 till 9. Genom att lösa detta problem hoppas jag kunna tillhandahålla tillförlitliga lösningar för automatiska identifieringssystem och ge support för applikationer inom relaterade områden.

**Uppgiftstillämpning:** Dess huvudsakliga mål är att konvertera bilder av handskrivna siffror till motsvarande digitala etiketter, förverkliga designen och tillämpningen av automatiserade igenkänningssystem, förbättra arbetseffektiviteten och noggrannheten och främja utvecklingen och tillämpningen av relaterade områden. Tillhandahålla tillförlitliga lösningar för automatiserade igenkänningssystem. Handskrivna sifferigenkänningsuppgifter används i stor utsträckning inom många områden, inklusive automatiserad bearbetning (som logistik), digital bildbehandling, intelligenta transportsystem, medicinsk bildbehandling och utbildningshjälp.

I denna uppgift kommer jag att använda tre modeller: logistisk regression, slumpmässig skog och stödvektormaskin för datauppsättningsträning, utvärdering och jämförelse.

## 2. Datauppsättningsbeskrivning:

**Datakälla:**Datauppsättningen som används i denna uppgift är MNIST handskrivna siffrorsdatauppsättningar. För att tillhandahålla ett standardriktmärke för bildklassificering för maskininlärningsalgoritmer.

**Dataegenskaper:**MNIST-datauppsättningen innehåller bilder av handskrivna siffror från 0 till 9. Varje bild är en gråskalebild på 28x28 pixlar. Värdet för varje pixel sträcker sig från 0 till 255, vilket indikerar pixelns grånivå, där 0 representerar svart och 255 representerar vitt. Därför kan varje bild representeras som en 784-dimensionell vektor ( $28 \times 28 = 784$ ), där varje element representerar gråvärdet för en pixel.

**Etikettbetydelse:** Varje bild i datamängden har en motsvarande etikett som representerar numret i bilden. Etiketter är heltal mellan 0 och 9, motsvarande de handskrivna siffrorna 0 till 9 respektive.

**Datauppsättningens storlek och distribution:**MNIST-datauppsättningen innehåller 60 000 träningsbilder och 10 000 testbilder. Antalet bilder i tränings- och testseten är relativt balanserat, med cirka 6 000 träningsbilder och 1 000 testbilder för varje kategori. Fördelningen av datamängden är slumpmässig, det vill säga bilderna i varje kategori väljs slumpmässigt. När det gäller datamängdsdelning står träningsuppsättningen vanligtvis för 80 % av den totala datamängden, som används för modellträning och parameterinställning, och valideringsuppsättningen står för 20 %, som används för att utvärdera modellens prestanda. .

## 3. Dataförbehandling:

- Dataladdning: Ladda först datauppsättningen, inklusive bilddata av handskrivna siffror och motsvarande etiketter.
- Bildnormalisering: Under bildnormaliseringsstadiet skalas pixelvärden till ett intervall mellan 0 och 1. Detta uppnås vanligtvis genom att dividera pixelvärdet med dess maximala värde (vanligtvis 255) för att säkerställa att alla pixelvärden är på samma skala.
- Datauppdelning: För att träna modellen och utvärdera dess prestanda delas datauppsättningen upp i en träningsuppsättning och en testuppsättning. Vanligtvis används slumpmässig uppdelning, där en del av datamängden används för modellträning och den andra delen används för att utvärdera modellens prestanda. En vanlig splitkvot är 80 % av data för träning och 20 % av data för testning.

## 4. Val av modell

Valda modeller: För den här uppgiften valde jag tre olika maskininlärningsmodeller, inklusive linjär regression, slumpmässig skog och stödvektormaskin (SVM).

### 4.1. Logistisk regression:

- **Anledning till val:**

Logistisk regression är en klassisk klassificeringsalgoritm som fungerar bra på binära klassificeringsproblem och som enkelt kan generaliseras till klassificeringsproblem med flera klasser.

Logistisk regression har låg beräkningskomplexitet, snabb träningshastighet och är lämplig för bearbetning av storskaliga datamängder.

Den har bättre anpassningsförmåga för det linjära förhållandet mellan funktioner och kan bättre hantera linjärt separerbara datamängder.

- **Fördel:**

Träningshastigheten är snabb och lämplig för storskaliga datamängder.

Implementeringen är enkel, lätt att förstå och förklara och kräver inte för mycket parameterjustering.

Den fungerar bra för linjärt separerbara datamängder och kan mata ut sannolikheten att ett urval tillhör en viss katego

- **Nackdelar:**

För olinjära datamängder kräver dålig prestanda funktionsteknik eller användning av polynomegenskaper för transformation.

Känslig för avvikelser i funktionsutrymme.

Oförmåga att hantera komplexa relationer och interaktioner.

### 4.2. Random Forest:

- **Anledning till val:**

Random forest är en ensembleinlärningsmetod som utför klassificerings- eller regressionsuppgifter genom att kombinera flera beslutsträd. Den har god robusthet och generaliseringsförmåga och lämpar sig för bearbetning av högdimensionell data och storskaliga datamängder, så den är mycket lämplig för handskrivna sifferigenkänningsuppgifter.

- **Fördel:**

Bra robusthet, inte lätt att överanpassa och kan hantera storskaliga datamängder.

Den kan mata ut viktig information om funktioner för att hjälpa till att förstå data.

- **Nackdelar:**

Överanpassning kan förekomma för bullriga datamängder. Modellträning kräver mycket tid och minne. Det är svårt att förklara modellens förutsägelseprocess och inte särskilt intuitivt.

## 4.3. Support Vector Machine (SVM):

- **Anledning till valet:**

Stödvektormaskin är en klassisk övervakad inlärningsalgoritm som vanligtvis används i klassificerings- och regressionsuppgifter. Den utför klassificering genom att hitta det maximala marginalhyperplanet i funktionsutrymmet. Den har god generaliseringsförmåga och bearbetningsförmåga för högdimensionella data, och är lämplig för komplexa olinjära relationsmodelleringar.

- **Fördel:**

Fungerar bra i högdimensionella utrymmen och kan hantera linjärt och olinjärt separerbar data.

Den har god robusthet mot små exempeldatauppsättningar och brus i datamängden.

Du kan anpassa dig till olika datastrukturer genom att välja olika kärnfunktioner.

- **Nackdelar:**

Beräkningskomplexiteten är hög för storskaliga datamängder och högdimensionella funktionsutrymmen.

Kraven på parameterintervall och inställning är relativt höga.

Möjlig prestandaförsämring vid klassobalans

## 5. Modellutbildning och utvärdering:

### 5.1. Modellträning:

- **Logistisk tillbakagång:**

För den logistiska regressionsmodellen laddas datasetet först och bilderna normaliseras, och sedan delas datasetet upp i en träningsuppsättning och en testuppsättning. Använd sedan träningsuppsättningen för att träna den logistiska regressionsmodellen och justera modellens hyperparametrar, såsom det maximala antalet iterationer, genom korsvalidering och andra tekniker för att förbättra modellens prestanda.

- **Random Forest:**

För den slumpmässiga skogsmodellen laddas även datamängden och bilderna normaliseras och datamängden delas upp i en träningsuppsättning och en testuppsättning. Sedan tränas en slumpmässig skogsmodell på träningssetet, och modellens hyperparametrar, såsom antalet träd och trädens djup, ställs in genom korsvalidering och andra tekniker för att förbättra modellens prestanda.

- **Support Vector Machine:**

Innan utbildningen av stödvektormaskinmodellen utfördes också dataladdning och bildstandardisering, och datamängden delades upp i en träningsuppsättning och en testuppsättning. Träna sedan stödvektormaskinmodellen på träningssetet och justera modellens hyperparametrar, såsom typen av kärnfunktion, regulariseringsparametrar, etc., genom korsvalidering och andra tekniker för att förbättra modellens prestanda.

### 5.2. Modellutvärdering:

För varje modell utvärderas dess prestanda på tränings- och testseten.

Noggrannhet används som huvudutvärderingsindex, det vill säga andelen korrekt predikterade prover.

Utvärdera prestandan för varje modell genom att skriva ut indikatorer som noggrannhet på träningssetet och testsetet för att jämföra prestandaskillnaderna mellan olika modeller.



## 6. Resultatanalys:

### 6.1. Modellprestanda:

Prestanda för logistisk regressionsmodell: På testsetet uppnådde den logistiska regressionsmodellen en noggrannhet på 0,9164285714285715. Detta visar att modellen kan känna igen bilder av handskrivna siffror relativt exakt och korrekt klassificera dem i motsvarande siffror.

Slumpmässig skogsmodellprestanda: Den slumpmässiga skogsmodellen uppnådde en noggrannhet på 0,9672857142857143 på testsetet. Detta visar att modellen har hög noggrannhet och effektivt kan klassificera bilder av handskrivna siffror.

Stödvektormaskinmodellens prestanda: Stödvektormaskinmodellen uppnådde en noggrannhet på 0,9764285714285714 på testsetet. Detta visar att modellen har god generaliseringsförmåga och effektivt kan klassificera bilder av handskrivna siffror.

### 6.2. Felanalys:

För logistiska regressionsmodeller: Scenarier där fel kan uppstå inkluderar dålig prestanda på vissa bilder som är suddiga eller har dålig handstil, vilket leder till felklassificering eller feligenkänning. Som svar på dessa situationer kan jag överväga att ytterligare optimera modellens funktionsextraktionsmetod eller öka mångfalden av datamängden för att förbättra modellens robusthet.

För slumpmässiga skogsmodeller: Situationer där fel kan uppstå inkluderar prestandaförsämring i närvaro av bullriga data och klassobalans, och känslighet för brus och redundanta egenskaper i datamängden. För dessa situationer kan jag överväga att använda dataökningstekniker eller ytterligare optimera modellens hyperparametrar för att förbättra modellens prestanda och stabilitet.

För stödvektormaskinmodeller: Möjliga fel inkluderar högre beräkningskomplexitet i storskaliga datamängder och högdimensionella funktionsutrymmen, såväl som högre krav på parameterintervall och inställning. För dessa situationer kan jag överväga att optimera parameterintervallet och inställningsmetoderna för modellen, eller försöka använda andra modeller för att lösa dessa problem.

## 7. Använder Streamlit som ett visuellt testverktyg

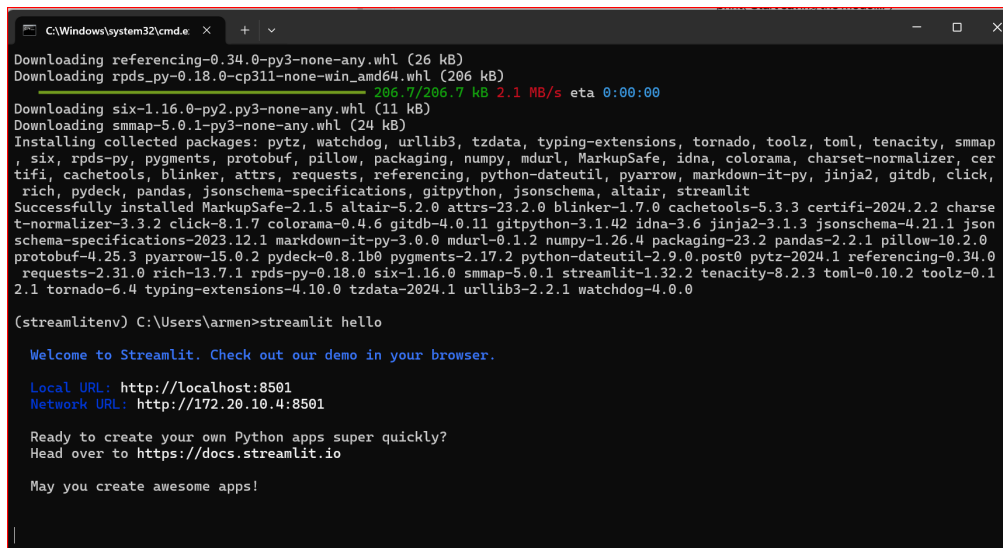
### 7.1. Streamlit operation:

#### 1. Lagra modelldokument:

Använd funktionen `joblib.dump()` för att spara logistisk regression, slumpmässig skog och SVM-modeller i två olika dokument.

Ett av dokumenten innehåller logistisk regression och slumpmässiga skogsmodeller, medan det andra dokumentet innehåller logistisk regression, slumpmässig skog och SVM-modeller.

#### 2. Installera Streamlit:



```
C:\Windows\system32\cmd.exe
Downloading referencing-0.34.0-py3-none-any.whl (26 kB)
Downloading rpds_py-0.18.0-cp311-none-win_amd64.whl (206 kB)
206.7/206.7 kB 2.1 MB/s eta 0:00:00
Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Downloading smmap-5.0.1-py3-none-any.whl (24 kB)
Installing collected packages: pytz, watchdog, urllib3, tzdata, typing-extensions, tornado, toolz, toml, tenacity, smmap,
six, rpds-py, pygments, protobuf, pillow, packaging, numpy, mdurl, MarkupSafe, idna, colorama, charset-normalizer, cer
tifi, cachetools, blinker, attrs, requests, referencing, python-dateutil, pyarrow, markdown-it-py, Jinja2, gitdb, click,
rich, pydeck, pandas, jsonschema-specifications, gitpython, jsonschema, altair, streamlit
Successfully installed MarkupSafe-2.1.5 altair-5.2.0 attrs-23.2.0 blinker-1.7.0 cachetools-5.3.3 certifi-2024.2.2 charse
t-normalizer-3.3.2 click-8.1.7 colorama-0.4.6 gitdb-4.0.11 gitpython-3.1.42 idna-3.6 Jinja2-3.1.3 jsonschema-4.21.1 json
schema-specifications-2023.12.1 markdown-it-py-3.0.0 mdurl-0.1.2 numpy-1.26.4 packaging-23.2 pandas-2.2.1 pillow-10.2.0
protobuf-4.25.3 pyarrow-15.0.2 pydeck-0.8.1b0 pygments-2.17.2 python-dateutil-2.9.0.post0 pytz-2024.1 referencing-0.34.0
requests-2.31.0 rich-13.7.1 rpds-py-0.18.0 six-1.16.0 smmap-5.0.1 streamlit-1.32.2 tenacity-8.2.3 toml-0.10.2 toolz-0.1
2.1 tornado-6.4 typing-extensions-4.10.0 tzdata-2024.1 urllib3-2.2.1 watchdog-4.0.0

(streamlitenv) C:\Users\armen>streamlit hello

Welcome to Streamlit. Check out our demo in your browser.

Local URL: http://localhost:8501
Network URL: http://172.20.10.4:8501

Ready to create your own Python apps super quickly?
Head over to https://docs.streamlit.io

May you create awesome apps!
```

Installera Streamlit-biblioteket via kommandoraden eller ett pakethanteringsverktyg som pip för att köra Streamlit-applikationer i din lokala miljö.

### 3. Skapa Streamlit-applikation:

```
C:\Windows\system32\cmd.exe
exec(code, module.__dict__)
File "C:\Users\armen\Downloads\EC-utbildning\2024-V.7-Machine learning\Xiaowen_Chen inlämningsuppgifter\R+F_app.py", line 50, in <module>
    svm_prediction = svm_model.predict([processed_image])[0]
                      ^^^^^^^^^
NameError: name 'svm_model' is not defined
forrtl: error (200): program aborting due to control-C event
Image                PC                Routine                Line                Source
libifcoremd.dll      00007FFAC382DF54      Unknown              Unknown             Unknown
KERNELBASE.dll       00007FFB0F4A7E57      Unknown              Unknown             Unknown
KERNEL32.DLL         00007FFB1159257D      Unknown              Unknown             Unknown
ntdll.dll            00007FFB11DEAA58      Unknown              Unknown             Unknown

(base) C:\Users\armen\Downloads\EC-utbildning\2024-V.7-Machine learning\Xiaowen_Chen inlämningsuppgifter>streamlit run R+F_app.py

You can now view your Streamlit app in your browser.

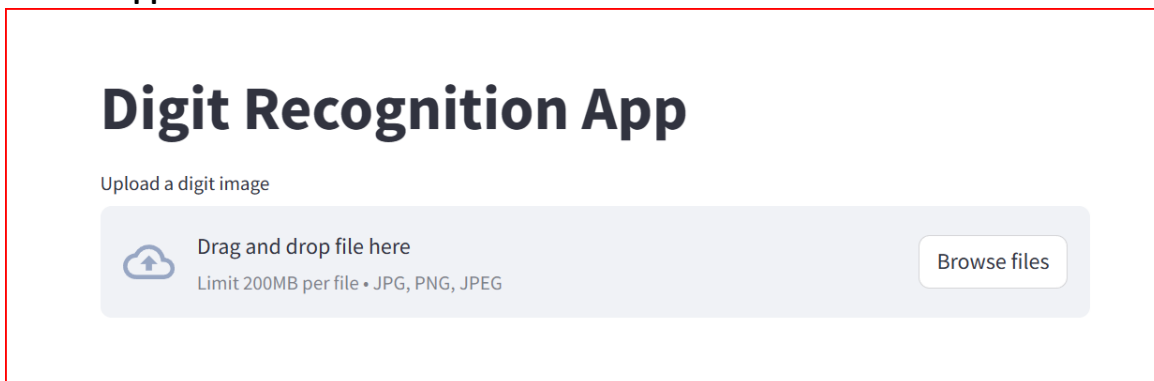
Local URL: http://localhost:8502
Network URL: http://172.20.10.4:8502

forrtl: error (200): program aborting due to control-C event
Image                PC                Routine                Line                Source
libifcoremd.dll      00007FFAC382DF54      Unknown              Unknown             Unknown
KERNELBASE.dll       00007FFB0F4A7E57      Unknown              Unknown             Unknown
KERNEL32.DLL         00007FFB1159257D      Unknown              Unknown             Unknown
ntdll.dll            00007FFB11DEAA58      Unknown              Unknown             Unknown

(base) C:\Users\armen\Downloads\EC-utbildning\2024-V.7-Machine learning\Xiaowen_Chen inlämningsuppgifter>streamlit run Mnist(3Models)R+S+F_app.py
```

Använd Streamlit-biblioteket för att skriva Python-skript för att skapa en interaktiv webbapplikation. I applikationen laddas det lagrade modelldokumentet och prediktionsfunktionalitet implementeras via användaruppladdade bilder.

### 4. Streamlit-applikationen körs:



Starta Streamlit-applikationen i en lokal miljö och låt användare ladda upp bilder och göra förutsägelser.

Observera applikationsprestanda, inklusive förutsägelser och gränssnittsinteraktivitet.

## Förutsägelse och jämförelse resultat av olika modeller

### 1. Jämförelse av förutsägelseresultaten för de två modelldokumenten Mnist(2Models)R+F\_app:



Jämför resultaten när du gör prognoser med hjälp av två modelldokument, inklusive prediktionsnoggrannhet och andra prestandamått. Det finns skillnader i prediktionsresultat mellan de två modelldokument.

## 2. Jämförelse av förutsägelseresultat för tre modelldokument Mnist(3Models)R+S+F\_app:



Jämför resultaten när du gör prognoser med hjälp av tre modelldokument, inklusive prediktionsnoggrannhet och andra prestandamått. Det finns också skillnader i prediktionsresultat mellan de tre modelldokumenterna.

## 3. Slutsats och förslag:

Enligt resultaten av jämförande analys dras slutsatsen att det finns skillnader i prediktionsresultat mellan Mnist(2Models)R+F\_app och Mnist(3Models)R+S+F\_app, även Mnist(3Models)R+S+F\_app är bättre än Mnist(2Models)R+F\_app Sämre när det gäller förutsägelser.

För att förbättra modellens prediktiva prestanda och noggrannhet måste jag lära mig mer.

## 8. Sammanfattningsvis

Baserat på ovanstående forskningsresultat drar jag följande slutsatser:

Modeller som logistisk regression, slumpmässiga skogar och stödvektormaskiner har alla en viss tillämpningspotential i handskrivna sifferigenkänningsuppgifter, men var och en har vissa begränsningar och utrymme för förbättringar.

Framtida arbete kommer att ytterligare utforska avancerade modeller som djupinlärning. Samtidigt som jag förbättrar modellens prestanda bör jag också fokusera på modellens beräkningskomplexitet och tolkningsbarhet.

Dessutom kan metoder som dataförbättring och funktionsteknik också övervägas för att ytterligare optimera modellen och förbättra modellens generalisering och robusthet.