klassificering av handskrivna siffror med MNIST-datasetet

En jämförande studie av maskininlärningsmetoder för tolkning av handskrivna siffror med hjälp av MNIST-datasetet

Alamgir Rana

EC Utbildning

Maskinlärning

2025-03

Abstract

This thesis investigates the application of machine learning techniques for handwritten digit recognition using the MNIST dataset. The primary objective is to develop and compare multiple supervised learning models to classify handwritten digits with high accuracy. Utilizing the MNIST dataset comprising 70,000 grayscale images of digits ranging from 0 to 9. The study focuses on training and evaluating three models: Random Forest, Decision Tree, and Logistic Regression.

Emphasis is placed on enhancing model performance through hyperparameter tuning using GridSearchCV. The research also addresses key challenges in feature extraction, data preprocessing, and model optimization. The goal is to achieve a classification accuracy of at least 95%, demonstrating the effectiveness of traditional machine learning algorithms in the domain of image recognition.

Keywords: Machine Learning, MNIST, Handwritten Digit Recognition, Random Forest, Decision Tree, Logistic Regression, Hyperparameter Tuning, Web Application, Image Processing

Förkortningar och Begrepp

- **ML** – *Machine Learning*
- **AI** – *Artificial Intelligence* (Artificiell Intelligens)
- **CV** – *Computer Vision*
- **RF** – *Random Forest*
- **DT** – *Decision Tree*
- **LR** – *Logistic Regression*
- **SVM** – *Support Vector Machine*
- **KNN** – *K-Nearest Neighbors*
- **MSE** – *Mean Squared Error*
- **RMSE** – *Root Mean Squared Error*
- **MAE** – *Mean Absolute Error*
- **TP** – *True Positive*
- **TN** – *True Negative*
- **FP** – *False Positive*
- **FN** – *False Negative*
- **F1-score** – *Harmonic mean of Precision and Recall*
- **CV** – *Cross-Validation*
- **K-Fold** – *K-Fold Cross-Validation*
- **GridSearchCV** – *Grid Search Cross-Validation*
- **sklearn** – *Scikit-Learn*

Innehållsförteckning

1 Inledning

Med den snabba utvecklingen av artificiell intelligens har maskininlärning blivit ett kraftfullt verktyg för att lösa komplexa problem inom många områden. Ett av dessa områden är mönsterigenkänning, där maskininlärningsalgoritmer används för att analysera och tolka data på ett sätt som efterliknar människans uppfattning. Ett klassiskt exempel i detta avseende är handskriven sifferigenkänning, ett problem som länge har använts som en testbädd för att utvärdera olika maskininlärningsmodeller och metoder.MNIST-datauppsättningen (Modified National Institute of Standards and Technology) är en välkänd och allmänt använd datauppsättning inom detta område. Den innehåller 70 000 gråskalebilder av handskrivna siffror från 0 till 9 och fungerar som en idealisk referens för att testa och jämföra prestanda hos olika klassificeringsalgoritmer.

Målet med detta arbete är att undersöka hur traditionella, övervakade maskininlärningsmodeller kan användas för att klassificera handskrivna siffror. Tre modeller – decission trees classification, random forest classification och logistisk regression – implementeras och jämförs, och deras prestanda analyseras med fokus på noggrannhet och effektivitet. För att förbättra resultaten av modellerna används hyperparameterjusteringstekniker som GridSearchCV.Målet med denna forskning är inte bara att uppnå hög klassificeringsnoggrannhet, utan också att ge en djupare förståelse för hur olika maskininlärningsmodeller presterar när de tillämpas på bildigenkänning.

1.1 Syfte och Frågeställning

Syfte

Syftet med detta arbete är att undersöka hur väl olika traditionella maskininlärningsmodeller kan klassificera handskrivna siffror med hjälp av MNIST-datasetet. Genom att jämföra prestandan hos beslutsträd, random forest och logistisk regression är målet att identifiera vilken modell som ger bäst noggrannhet samt förstå vilka faktorer som påverkar deras resultat. Studien syftar även till att fördjupa förståelsen för hur hyperparametertuning, databehandling och modellval påverkar klassificeringsprestanda.

Frågeställningar

- Hur väl presterar olika maskininlärningsmodeller (beslutsträd, random forest och logistisk regression) vid klassificering av handskrivna siffror i MNIST-datasetet?
- Vilken modell uppnår högst klassificeringsnoggrannhet?

.

2 Teori

I det här kapitlet beskriver vi den teoretiska bakgrunden som är nödvändig för att förstå de maskininlärningsmodeller som används i denna studie, vikten av funktionsskalning och processen med hyperparameterjustering för att optimera modellprestanda. Vidare diskuterar vi de utvärderingsmått som används för att bedöma effektiviteten hos modellerna för att klassificera handskrivna siffror.

2.1 Decision Tree Classification

En Decision Tree Classifier är en populär maskininlärningsalgoritm som används för klassificeringsuppgifter. Det är en flödesschemaliknande struktur där varje intern nod representerar ett beslut baserat på ett attribut, varje gren representerar resultatet av beslutet och varje bladnod representerar en klassetikett eller ett kontinuerligt värde. Beslutsträd är lätta att förstå och tolka, vilket gör dem till ett kraftfullt verktyg för olika tillämpningar som ekonomi, hälsovård och marknadsföring.

2.1.1 Random Forest Classification

En Random Forest Classifier är en kraftfull ensembleinlärningsalgoritm som används för klassificeringsuppgifter. Den fungerar genom att konstruera flera beslutsträd under träning och mata ut den klass som är klassernas läge (klassificering) eller medelförutsägelse (regression) för de individuella träden. Denna metod hjälper till att förbättra den prediktiva noggrannheten och kontrollera överanpassning.

2.1.2 Logistik Regression

Logistisk regression är en statistisk modell och en vanlig maskininlärningsalgoritm som används för klassificeringsproblem. Logistisk regression vald för att den används som baslinje när man jämför andra modeller. Logistisk regression används också när de flesta beroende variabler är numeriska.

2.1.3 Hyperparametertuning och GridSearchCV

Hyperparametrar är inställningar för en modell som inte lärs från data, till exempel antalet träd i en Random Forest eller maxdjupet på ett beslutsträd. Att välja rätt hyperparametrar kan ha stor inverkan på modellens prestanda. GridSearchCV är en metod som systematiskt testar olika kombinationer av hyperparametrar genom korsvalidering (cross-validation) för att hitta den bästa inställningen.

2.1.4 Överfitting och Underfitting

En viktig aspekt vid träning av maskininlärningsmodeller är att hitta balansen mellan att modellens prediktioner är för specifika för träningsdatan (överanpassning) och att modellen är för generell (underanpassning).

- Överfitting uppstår när modellen lär sig detaljer och brus i träningsdatan som inte generaliserar väl till ny, osedd data.
- Underfitting sker när modellen är för enkel för att fånga mönstren i datan och därmed presterar dåligt både på tränings- och testdata.

2.1.5 Korsvalidering (Cross-validation)

Korsvalidering är en metod för att utvärdera modellens generaliseringsförmåga. Den vanligaste varianten är k-faldig korsvalidering där datan delas upp i k delar (t.ex. 5 eller 10). Modellen tränas på k–1 delar och testas på den återstående, vilket upprepas k gånger. Resultaten snittas för att få en rättvis bild av modellens prestanda. GridSearchCV använder denna teknik för att utvärdera varje kombination av hyperparametrar på ett tillförlitligt sätt.

2.1.6 Accuracy

Andelen korrekta prediktioner av det totala antalet.

$$\text{Accuracy} = \frac{\text{Antal korrekta prediktioner}}{\text{Totalt antal observationer}}$$

2.1.7 Confusion Matrix

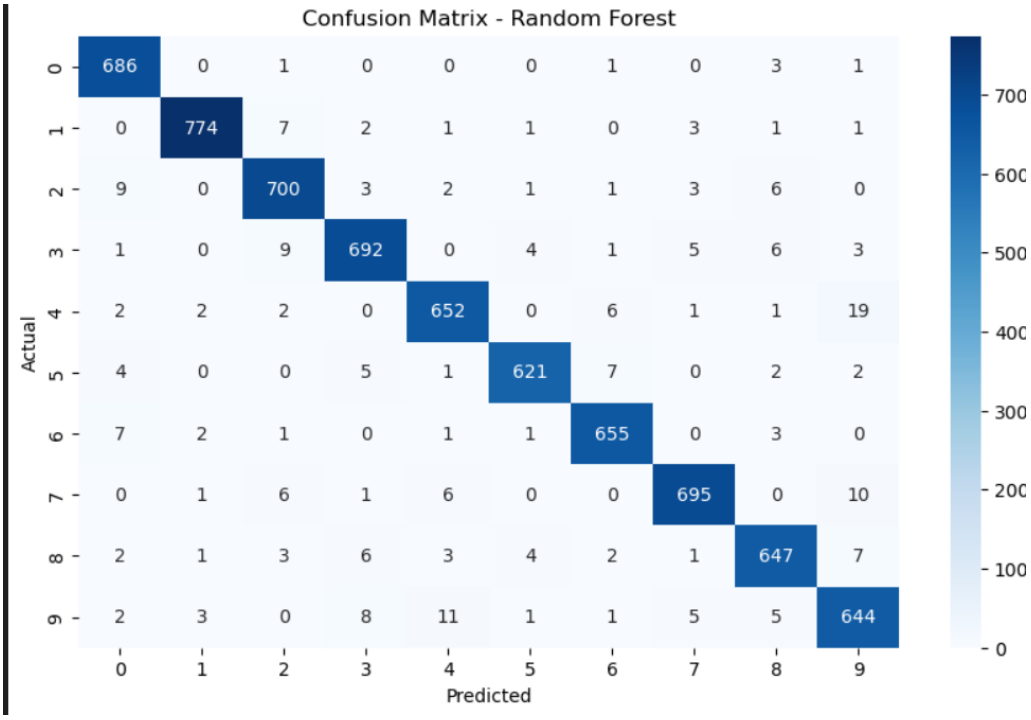
En Confusion matris är en tabell som används för att utvärdera prestandan hos en klassificeringsmodell. Den ger en detaljerad uppdelning av modellens förutsägelser jämfört med de faktiska resultaten, vilket hjälper till att identifiera var modellen gör fel. Matrisen är särskilt användbar för binära och flerklassiga klassificeringsproblem.

Confusion matrisen består av fyra huvudkomponenter:

- True positiv (TP): Modellen förutsäger korrekt ett positivt resultat.
- True Negative (TN): Modellen förutsäger korrekt ett negativt utfall.

- False positivt (FP): Modellen förutsäger felaktigt ett positivt utfall (Typ I-fel).
- False negativt (FN): Modellen förutsäger felaktigt ett negativt utfall (Typ II-fel)

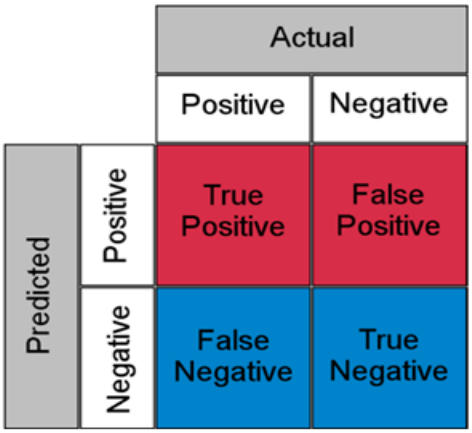
En confusion matris hjälper dig att se hur väl en modell fungerar genom att visa korrekta och felaktiga förutsägelser. Det hjälper också till att beräkna nyckelmått som noggrannhet, precision och återkallelse, vilket ger en bättre uppfattning om prestanda, särskilt när data är obalanserad.



2.1.8 Precision och Recall

Precision: Precision visar hur många av modellens positiva prediktioner som faktiskt var korrekta.

Recall: Recall visar hur många av de faktiska positiva fallen modellen lyckades fånga.



Figur 11. Exempel på en Confusion matrix (Wikipedia 2019)

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

Figur 12. Formel till Precision och Recall

2.1.9 F1-Score

Det harmoniska medelvärdet av precision och recall. Används för att väga båda dessa mått, särskilt när det är viktigt att undvika både falska positiva och falska negativa.

2.1.10 Classification Report

En klassificeringsrapport är en rapport som mäter kvaliteten på förutsägelser från en klassificeringsalgoritm. Den visar en representation av de viktigaste klassificeringsmåtten per klass, inklusive sanna positiva, falska positiva, sanna negativa och falska negativa. Rapporten består huvudsakligen av 5 kolumner och (N+3) rader. Mätvärdena i en klassificeringsrapport omfattar precision, återkallelse, f1-poäng och stöd för varje klass. Nedan finns det en exempel av classification Report för Random forest classification.

```
Classification Report for Random Forest:
              precision    recall  f1-score   support

           0       0.96      0.99      0.98       692
           1       0.99      0.98      0.98       790
           2       0.96      0.97      0.96       725
           3       0.97      0.96      0.96       721
           4       0.96      0.95      0.96       685
           5       0.98      0.97      0.97       642
           6       0.97      0.98      0.97       670
           7       0.97      0.97      0.97       719
           8       0.96      0.96      0.96       676
           9       0.94      0.95      0.94       680

    accuracy                           0.97      7000
   macro avg       0.97      0.97      0.97      7000
weighted avg       0.97      0.97      0.97      7000
```

3 Metod

Detta kapitel beskriver de steg som genomförts för att undersöka hur olika maskininlärningsmodeller kan användas för att klassificera handskrivna siffror i MNIST-datasetet. Arbetet har genomförts i följande steg: datainsamling, förbehandling, modellträning, hyperparametertuning och utvärdering.

3.1.1 Datainsamling

Den använda datakällan är det publikt tillgängliga **MNIST-datasetet** som består av 70 000 gråskalebilder (28x28 pixlar) av handskrivna siffror från 0 till 9. Datasetet är uppdelat i 60 000 träningsbilder och 10 000 testbilder. Bilderna laddades in via biblioteket sklearn.datasets

3.1.2 Förbredelse

För att modellen ska kunna bearbeta bildinformationen omvandlades varje bild till en **vektor med 784 pixlar** (28 × 28). Pixelvärden normaliserades till intervallet **[0, 1]** genom division med 255 för att förbättra inlärningsprestandan. Ingen ytterligare feature extraction användes, då pixelvärdena i sig utgör tillräcklig information för detta dataset.

3.1.3 Uppdelning av datasetet

För att säkerställa att modellen kan generalisera väl till ny data delas datasetet in i:

- **Training Set** – används för att bygga och justera modellen.
- **Test Set** – används för att utvärdera modellens prestanda på osedd data.
  En vanlig fördelning är **80 % träning och 20 % test**, men detta kan variera beroende på datasetets storlek och problemets komplexitet.

3.1.4 Normalise data

Normalise data Används standardScaler funktionen för att normalise dataset.

3.1.5 Maskininlärningsmodeller

Tre olika modeller har implementerats och jämförts:

- **Beslutsträd (Decision Tree):** En enkel modell som delar upp datan baserat på pixelvärden för att skapa beslutsklasser.
- **Random Forest:** En ensemblemetod som bygger på flera beslutsträd för att förbättra noggrannheten och minska risken för överanpassning.

- **Logistisk Regression:** En statistisk modell som predicerar sannolikheten för varje klass och väljer den med högst sannolikhet.

Modellerna implementerades med hjälp av **Scikit-learn (sklearn)**-biblioteket i Python.

3.1.6 Hyperparametertuning

För att optimera modellernas prestanda användes **GridSearchCV**, som testar olika kombinationer av hyperparametrar genom **korsvalidering (cross-validation)**.
Exempel på hyperparametrar som testades:

- För Random Forest: n_estimators, max_depth
- För Decision Tree: max_depth, min_samples_split
- För Logistic Regression: C, solver, max_iter

GridSearchCV utfördes med 5-faldig korsvalidering för att säkerställa att modellerna inte överanpassades till träningsdata.

3.1.7 Utvärdering

Modellernas prestanda utvärderades med hjälp av flera mått:

- **Accuracy (klassificeringsnoggrannhet)** på testdatan
- **confusion matrix** för att analysera vilka siffror som ofta förväxlas
- **Precision, Recall och F1-score** för att ge en mer nyanserad bild av klassificeringens träffsäkerhet, särskilt om klasserna skulle vara obalanserade

Resultaten presenteras i form av tabeller och grafer för tydlig jämförelse mellan modellerna.

4 Resultat och Diskussion

I detta kapitel beskrivs resultaten från klassificeringsexperimenten med **Decision Tree**, **random forest** och **logistisk regression** på **MNIST-784**-datasetet. Resultaten diskuteras och jämförs för att analysera modellernas prestanda och generaliseringsförmåga.

4.1 Resultat

För att utvärdera modellerna användes **Test accuracy**, **Best Score**, **Best Parameters**:

Tabell X: Jämförelse av modeller för klassificering av MNIST-784

Modell	Best Parameters	Best Score	Test Accuracy
Beslutsträd	{'max_depth': None, 'min_samples_split': 2}	85.96%	87.11%
Random Forest	{'max_depth': 20, 'min_samples_split': 2, 'n_estimators': 100}	96.49%	96.71%
Logistisk regression	{'C': 0.1}	91.75%	91.07%

- **Random Forest presterade bäst** med en **accuracy på 96.71%**, vilket visar att ensemblemetoden effektivt reducerar överanpassning och förbättrar prestanda.
- **Logistisk regression** uppnådde **91.07% accuracy**, vilket visar att modellen är effektiv men något sämre än Random Forest.
- **Beslutsträdet presterade sämst**, med **87.11% accuracy**, troligen på grund av överanpassning och begränsad generaliseringsförmåga.

4.2 Diskussion

- **4.2.1 Jämförelse mellan modellerna**
- **Random Forests höga prestanda** kan tillskrivas att modellen bygger på flera beslutsträd, vilket minskar risken för överanpassning och ger bättre generalisering.
- **Logistisk regression presterade relativt bra** trots sin enkelhet, vilket indikerar att MNIST-data har en relativt linjär separabilitet.
- **Beslutsträdet presterade sämre**, vilket kan förklaras av dess tendens att skapa överanpassade regler för träningen, vilket försämrar generaliseringen till testdata.

5 Slutsatser

Syftet med denna studie var att undersöka hur väl olika traditionella maskininlärningsmodeller kan användas för att klassificera handskrivna siffror i MNIST-datasetet. Tre modeller – beslutsträd, random forest och logistisk regression – implementerades, optimerades och jämfördes baserat på prestanda.

Samtliga tre modeller visade sig vara kapabla att hantera klassificeringen med god noggrannhet. Beslutsträdet fungerade som en grundläggande modell men presterade sämre jämfört med de övriga två. Logistisk regression uppnådde goda resultat och visade stabil prestanda, medan random forest överträffade de andra modellerna både i noggrannhet och robusthet mot överanpassning.

Random Forest-modellen visade sig vara den mest effektiva i denna studie och uppnådde den högsta klassificeringsnoggrannheten – över 96 % efter hyperparametertuning med GridSearchCV. Detta kan förklaras av modellens förmåga att kombinera flera beslutsträd och därmed minska variation och förbättra generalisering.

6 Teoretiska frågor

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

Svar: "Tränings-", "Validering" och "Test"-data används för olika ändamål.

Kalle delar upp träningsdatan i olika datauppsättningar för att lära maskininlärning modeller att känna igen mönster i data. Genom denna uppdelning kan modellen bearbeta information effektivt och justera sina parametrar beroende på datastrukturen, vilket förbättrar dess förmåga att generalisera och fatta korrekta beslut på nya, osedda data. Trängs datan omfattar 70% av den totala data.

Kalle delar upp Validerings Datan för att justera modellens hyper parametrar. Det utser en unbiased utvärdering under träningsdata. Valideringsdata omfattar 15% av den totala data

Kalle delar upp testdata för att testing skyddar modellen från overfitting till träning. Testdata är osynlig data av modellen under träning. Man kan säga att testdata mäter slutlig prestanda.

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”?

Svar: Julia använder inte valideringsdataset. Så hon kan använda cross validation på träning. Så får hon stark prestndaindikator. Hon kan även utvärdera testdatan genom använda RMSE värde och hon kan analysera bias-variance tradeoff där test och träning kan vara överpassad.

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

Svar: Regressionsproblem är typ av maskininlärning problem eftersom målet predikterar ett kontinuerlig värde baserat på input data. Målet är att prediktera kategoriska etiketter och modeller relation mellan variabler och göra numeriska uppskattningar.

Exemplar:

Linjär regression: Linjär regressions modellen är en rak linje där en beroende variabel och en eller flera oberoende variabler.

Lasso eller Ridge regression: Lassa regression är en variationer av linjär regression som använder regularisering för överpassning.

Polynomial regression: Polynomial regression använder termer för att modellera icke-linjär. Polynomial Regression förlänger linjär regression.

4. Hur kan du tolka RMSE och vad används det till: $RMSE$ =√1 n ∑(yi −$\hat{y}i$)2 1

Svar: RMSE förkortningen är root mean squared error. RMSE är en mått på hur en regressionsmodell predicerar värden jämfört med de verkliga observationerna.

$RMSE$ =√1 n ∑(yi −$\hat{y}i$)2 där yi är verkliga värde och $\hat{y}i$ är predicerad värde och n är antalet observationer.

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

Svar: Klassificeringsproblem är maskinslärnings problem där målet är att förutsäga vilken kategori tillhör en viss observationer. Klassificeringen gör att variablerna blir kontinuerligt från diskret.

Exampel:

- Logistisk regression används för binära klassificering.
- Decision trees används för att dela upp data regler för att göra klassificeringar.
- Random Forest används för flera decision trees för mer robusta förutsägelser.

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

Svar: K-means modellen man försöker dela in data i olika grupper. K-Means modellen är icke superviserad maskinslärningalgoritom som används för klustering. K-means använder för minimera avståndet av datapunkten i samma kluster och maximera datapunkten i olika kluster.

Exampel:

- Företaget kan analysera kunddata genom att använda K-means.
- Genom att använda K-means kan man dela upp bilder baserat på färg och texture.
- Gruppera texter eller nyhetsartiklar efter ämne

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

Svar:

Ordinal Encoding: Ordinal encoding konverterar kategorisk data till numeriska värde. Detta är väldigt för maskinlärnings modell för att input data ska vara numeriska värde. Till exampel: om vi har color variabler “red”,”black”, “blue”. Så vi kan tilldela “red” är 1, “black” är 1 och “blue” är 3.

One-hot encoding: One -hot encoding konverterar kategoriska data till numeriska format. Till exempel: vi överväger en dataset "fruit"-kolumn som har olika frukter(“apple”,”orange”,”grape”). Efter tillämpa hot encoding varje frukt har separat kolumn med binari varde.

Dummy variable encoding: Dummy variable encoding konverterar kategoriska variable till en serie binari variabler. Till exempel:Kategorisk variabel “color” som tre variabler “red”, “blue”, “green”. Dummy variabel encoding konverterar detta variable till tre nya binari variabler.

Color_Red

Color_Blue

Color_Green

Efter konverteringen blir det:

Color_Red: 1

Color_Blue: 0

Color_Green: 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?

Svar:

Göran har rätt i att data ofta klassificeras som nominal eller ordinal.

Julia har rätt i att kontext och tolkning påverkar om en variabel är ordinal eller nominal.

Så kan vi säga att datatypen inte alltid är objektiva, de kan ändras beroende på hur vi tolkar och använder dem!

9. 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?

Svar: Streamlit är ett python bibliotek som gör det enkelt och snabbt att bygga och dela webbapplikationer för maskininlärningsprojekt. Den är utformad för att förenkla processen att skapa användargränssnitt för datavisualisering.

Stremlit används inom datavetenskap, maskininlärning och analys till exampel: datavisualisering, maskininlärning applikation, Dataanalys & rapporter, Bild- och textanalys, IoT och realtidsdata.

7 Självutvärdering

Under arbetets gång har jag stött på flera utmaningar, både tekniska och analytiska. Nedan beskrivs några av de största utmaningarna och hur jag hanterade dem:

- Att välja optimala hyperparametrar för **Random Forest, Decision Tree och Logistik Regression** var en utmaning.
- Jag löste detta genom att använda **GridSearchCV** för att testa olika kombinationer och identifiera de bästa inställningarna.
- Random Forest med många träd och djupa beslutsträd krävde **hög beräkningskraft**, vilket gjorde träningen långsam.
- Jag hanterade detta genom att använda **n_jobs=-1** för parallell bearbetning och begränsa antalet parametrar i GridSearchCV.
- För att kunna analysera och jämföra modellerna krävdes en **förståelse för precision, recall, F1-score och konfusionsmatriser**.
- Jag löste detta genom att studera hur dessa metoder används inom maskininlärning och visualisera resultaten med hjälp av diagram och tabeller.

7.1.1 Självbedömning av betyg

Jag anser att jag förtjänar **betyget VG** baserat på följande kriterier:

- Jag har följt en tydlig arbetsmetod, från datainsamling och förbehandling till modellutvärdering.
- Jag har inte bara redovisat resultaten utan också analyserat varför modellerna presterar olika.
- Jag har även skapat en streamlit webbapplikation där man kan rita handskrivna siffror mellan 0 till 9 och sedan appen predikterar siffran.

7.1.2 Meddelande

Jag vill särskilt lyfta fram:

- Handledningen och vägledningen under arbetet har varit mycket värdefull.
- Jag uppskattar möjligheten att experimentera med olika modeller och få en djupare förståelse för hur maskininlärning fungerar i praktiken.

Källförtecning

Scikit-Learn API dokumenta□on.

EDA:

https://www.geeksforgeeks.org/exploratory-data-analysis-in-python/

train_test_split:

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

DecisionTreeClassifier:

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

RandomForestClassifier:

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

LogisticRegression:

https://scikit-learn.org/dev/modules/generated/sklearn.linear_model.LogisticRegression.html

Confusion matrix:

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

Classification Report:

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html