Handwritten number’s recognition

(Modellering av handskrivna siffror med maskininlärning)



Svetlana Oshcepkova

EC Utbildning

Machin learning project

March 2025

# Abstract

This project explores the classification of handwritten digits from the MNIST dataset using four machine learning models: Logistic Regression, Random Forest, XGBoostClassifier, and SGDClassifier. The dataset is split into training and testing subsets. Hyperparameters are optimized using randomized search, and model performance is measured using cross-validated accuracy scores. The models are compared based on performance and runtime, and the best-performing model is selected for final predictions. Evaluation is conducted using a confusion matrix, accuracy score, and single image prediction. The best model is saved for future use.

Table of Contents

[Abstract 2](#_Toc193651991)

[1 Inledning 1](#_Toc193651992)

[2 Teori 2](#_Toc193651993)

[2.1 RandomForestClassifier 2](#_Toc193651994)

[2.2 XGBoost 2](#_Toc193651995)

[2.3 XGBoost 2](#_Toc193651996)

[2.4 SGDClassifier 2](#_Toc193651997)

[2.5 LogisticRegression: 3](#_Toc193651998)

[2.6 Cross validation 3](#_Toc193651999)

[2.7 RandomizedSearchCV 3](#_Toc193652000)

[3 Metod 4](#_Toc193652001)

[4 Resultat 8](#_Toc193652002)

[4.1 Accuracy score 8](#_Toc193652003)

[4.2 Confusion matrix av det bästa modell: RandomForestClassifier 8](#_Toc193652004)

[4.3 Classification rapport av det bästa modell: RandomForestClassifier 8](#_Toc193652005)

[5 Slutsatser 9](#_Toc193652006)

[6 Självutvärdering 10](#_Toc193652007)

[Källförteckning 11](#_Toc193652008)

# Inledning

Syftet med denna rapport är att undersöka hur väl två klassiska modeller – Logistic Regression och Random Forest – kan klassificera handskrivna siffror från MNIST-datat.

För att uppfylla syftet kommer följande frågeställningar att besvaras:

1. Vilken av modellerna presterar bäst vid användning av cross-validation?
2. Hur bra förutsäger den bästa modellen siffror i testdatan?
3. Hur tolkar man resultatet i en confusion matrix?

# Teori

## RandomForestClassifier

**Random Forest Classifier** lämpar sig väl för **multiklassklassificering**. Och klarar av direkt.

Random Forest bygger på beslutsträd och består av en ensemble (ett stort antal) träd som tränas på olika delar av datan. Varje träd i skogen gör en egen förutsägelse, och slutresultatet bestäms genom majoritetsröstning bland träden.

När det gäller multiklassklassificering hanterar Random Forest detta **inbyggt**, utan att uttryckligen använda en One-vs-Rest (OvR) strategi. Istället bygger varje beslutsträd direkt för att skilja mellan alla klasser samtidigt. Detta gör att modellen kan fatta beslut om vilken klass en observation tillhör utan att behöva konstruera flera binära klassificerare.

## XGBoost

XGBoost Classifier-modellen fungerar bra för att hantera multiclass-klassificering. Eftersom XGBoost är en gradientförstärkningsalgoritm baserad på beslutsträd, stöder den flera strategier för att hantera multiclass-data.

För multiclass-klassificering i XGBoost används oftast strategin **One-vs-All (OvA)**, som även kallas **One-vs-Rest (OvR)**. Med denna metod tränar algoritmen en separat binär klassificerare för varje klass i datasetet, där varje modell predicerar sannolikheten att tillhörande klass är "positiv" medan alla andra klasser är "negativa". Resultaten från alla dessa modeller kombineras sedan för att välja den klass med högsta sannolikhet.

XGBoost implementerar detta internt och gör processen effektiv, så du behöver inte manuellt skapa modeller för varje klass.

## XGBoost

XGBoost Classifier-modellen fungerar bra för att hantera multiclass-klassificering. Eftersom XGBoost är en gradientförstärkningsalgoritm baserad på beslutsträd, stöder den flera strategier för att hantera multiclass-data.

För multiclass-klassificering i XGBoost används oftast strategin **One-vs-All (OvA)**, som även kallas **One-vs-Rest (OvR)**. Med denna metod tränar algoritmen en separat binär klassificerare för varje klass i datasetet, där varje modell predicerar sannolikheten att tillhörande klass är "positiv" medan alla andra klasser är "negativa". Resultaten från alla dessa modeller kombineras sedan för att välja den klass med högsta sannolikhet.

XGBoost implementerar detta internt och gör processen effektiv, så du behöver inte manuellt skapa modeller för varje klass.

## SGDClassifier

SGDClassifier avsedd i första hand för binär classifiering. När det gäller hur modellen hanterar multiklassproblem, använder **SGDClassifier** som standard strategin **One-vs-Rest (OvR).** Det innebär att modellen tränar en separat binär klassificerare för varje enskild klass. Varje sådan klassificerare försöker särskilja sin egen klass från alla övriga klasser. Vid prediktion räknar modellen ut en sannolikhet eller poäng för varje klass, och den klass som får högst värde väljs som modellens förutsägelse.

## LogisticRegression:

I Scikit-learn hanterar **LogisticRegression** multiklassklassificering med hjälp av två olika strategier: **One-vs-Rest (OvR)**, och **multinomial** (även kallad "softmax-regression"). Vilken strategi som används beror på vilken inställning man gör i parametern multi\_class.

Som standard använder modellen **One-vs-Rest (OvR)**, vilket innebär att en separat binär klassificerare tränas för varje klass för att särskilja den från alla andra. Modellen väljer sedan den klass vars klassificerare ger högst sannolikhet.

I senaste version körs automatisk identification av multi class problems multi\_class parameter ställs automatisk till 'multinomial'i den har fallet och modellen använder istället en **multinomial softmax-modell** som tränas direkt på alla klasser samtidigt. Denna metod ger ofta bättre resultat på dataset som MNIST eftersom den tar hänsyn till alla klasser på en gång och inte isolerar dem parvis.

## Cross validation

K-fold cross-validation används för att robust utvärdera modeller under träning, vilket minskar risken för överanpassning.

## RandomizedSearchCV

används för att hitta optimala kombinationer av hyperparametrar över ett fördefinierat område.

# 

# Metod

I mitt arbete ska jag följa ordningen som beskrivs nedan för att implementera LogistikRegression, Random Forest Classification, XGBoost Classifier och SGD Classifier modeller:

* 1. Ladda in datasetet MNIST (nMist\_784), innehåller 784 features (28x28 pixlar) och en målsiffra (0–9).
  2. Datasett ska delas upp manuellt enligt rekommendationen Geron (s. 105 2022)
  3. Initialisera men modell med default parameters
  4. Utföra hyperparameter tuning först med ett brett spektrum av parametrar och utföra en Random Search.
  5. Tar de optimala parametrarna från random search och lägga in ett mindre spektrum av parametrar och göra en avslutande Grid Search.
  6. Utvärdera modellen med cross\_val\_score på träningsdata.
  7. Utföra prediktioner.
  8. Utvärdera resultat med Accuracy score and Confusion Matrix
  9. Använda slumpmässigt vald bild från test datan används för att demonstrera modellens prediktion visuellt.

Först använde jag en separat fil för varje modell och testade att hitta de bästa hyperparametrarna för respektive modell. Det tog olika lång tid beroende på modell.

A screen shot of a computer program

AI-generated content may be incorrect.Jag har testat LogisticRegression i första hand. Trots att *Logistic Regression*, enligt Géron (2022), är direkt utformad för att hantera flera klasser, fungerade den inte särskilt bra i mitt fall. Det tog extremt lång tid att köra hyperparametertuning med *GridSearchCV* och *cross-validation*. När jag försökte använda parametern n\_jobs=-1 (vilket borde ha fungerat bra) kraschade min dator. Sedan när jag började om testade jag att köra *RandomizedSearchCV* först, men det fungerade fortfarande inte korrekt (se Bild 1, trots att jag till slut fick fram de bästa parametrarna som skulle kunna användas för vidare tuning med GreadSearchCV.

Bild 1: Resultat av RandomizedSearchCV på LogisticRegression

(Se Bild 2) Men jag bestämde mig att inte göra det eftersom Accurasy score efter cross validation var sämre än RandomForestClassifiger harscoreDärför bestämde jag mig för att avstå från att använda modellen vidare.

A screenshot of a computer program

AI-generated content may be incorrect..

Bild 2: Bästa parametrar för *Logistic Regression* med RandomizedSearchCV

Däremot fungerade *RandomForestClassifier*, som också är avsedd för direkt användning vid problem med flervalsklassificering (*multiclass classification*), både bra och snabbt. Modellen med de bästa hyperparametrarna uppnådde en genomsnittlig *accuracy score* på 0,97, vilket är ett mycket bra resultat (se Bild 3).A screenshot of a computer

AI-generated content may be incorrect.

Bild 3: RandomForestClassifier Best parameters med RandomizedSearchCV

A computer screen shot of a program code

AI-generated content may be incorrect.Jag har också testat *XGBoost Classifier*, och trots att *Randomized Search* för den tog ganska lång tid (se Bild 4 och 5), uppnådde modellen en genomsnittlig *accuracy score* på 0,97. Jag har inte valt

Bild 4: Tid vilken tog att köra RandomizedSearchCV på XGBoost

XGBoost som det bästa modell eftersom trots samma Accuracy score tog de mycket mer tid att utföra än RandomForestClassifiger gör.

A screenshot of a computer

AI-generated content may be incorrect.Bild 5: XG Boost parametrar med RandomizedSearchCV

Trots att SGD Classifier  är är en binär classifier och fungerar enligt *one-versus-the-rest* principen när det gäller Multiclass classification problem. Som det var forväntat , tog det fruktansvärd long tid för att tuna hyperparametrar (Se Bild 6) Enligt Geron (s. 119, 2022) “ For the MNIST problem, this means training 45 binary classifiers! When you want to classify an image, you have to run the image through all 45 classifiers and see which class wins the most duels. The main advantage of OvO is that each classifier only needs to be trained on the part of the training set containing the two classes that it must distinguish. “

A screenshot of a computer

AI-generated content may be incorrect.

Bild 6: SGDClassifier bästa parametrar med RandomizedSearchCV

# Resultat

Eftrsom jag körde prediction på test data sett bara för det bäst presterade modell anges nedan utvärdering bara för RandomForestClassifiger model.

## Accuracy score

|  |  |  |
| --- | --- | --- |
| Modell | Cross-val Accuracy | Test Accuracy |
| Logistic Regression | 0.90 |  |
| Random Forest classifier | 0.97 | 0.97 |
| XGBoost | 0.97 |  |
| SGDClassifier | 0.91 |  |

Tabell 1:Resultat av hyperparameteroptimerade modeller.

## Confusion matrix av det bästa modell: RandomForestClassifier

A graph with numbers and a bar chart

AI-generated content may be incorrect.

Bild 7:Confusion matrix RandomForestClassifier

## Classification rapport av det bästa modell: RandomForestClassifier

A screenshot of a computer

AI-generated content may be incorrect.

# Slutsatser

Modellen som fungerar bäst på MNIST\_784 data sett, när dem jämföras på prestanda och tiden för att köra modellen, är RandomForestClassifier. Modellen visade det bästa accuracy score vid cross validation på tränings och sedan visade den samma resultat på Test data vilket betyder att modellen fungerar lika bra på en ny data som på datan som den var tränad. Den också fungerar snabbt vilket är viktig vid implementering.

Från Confusion matrix ser vi samma bild: att det är ganska få tal hamnat son fals positiva och falsk negativa i jämförelse med de som var predikterade rätt. Alltså RandomForestClassifier modellen verkar fungera stabilt och konsekvent på MNIST\_784 data sett.

# Självutvärdering

1. *Utmaningar du haft under arbetet samt hur du hanterat dem.*

Den största utmaningen för mig när jag behövde börja från 0 på Tisdag eftermiddag efter min dator kraschade under körning av GreadSearchCV på LogistikRegression model när jag använde parameter n\_jobs=-1 för att kunna utföra processen snabbare. Efter en sådan erfarenhet man börjar tänka mer och använder git möjligheter mer medvetet.

1. *Vilket betyg du anser att du skall ha och varför*.

Trots att jag gjorde en hel del arbete med streamlit kan jag inte presentera någon resultat just nu så anser jag att G betyg ska vara rymlig.

1. Jag tycker att det skulle vara bra att berätta oftare och mer under kursen hur och i vilket syfte använder man Git och GitHub. Vilka möjligheter det ger och vilka nackdelar om du gör inte det.

# Källförteckning

1 Geron Aurelien (2022). ***Hands-On machine Learning with Scikit\_Learn, Keras&TensorFlow:*** *Concepts, Tools, and Techniques to Build Intelligent Systems.* Third Edition.O’Really.

2 Scikit learning documentation GreadSearch och Randomized Search https://scikit-learn.org/stable/modules/grid\_search.html