Machine Learning



Anthony Perez Andersson

EC Utbildning

Kunskapskontroll Machine Learning

2025-03

# Abstract

In this project I have developed a classifier to identify pictures of handwritten Arabic numerals from the MNIST database with a 97% accuracy. This I have achieved by utilizing machine learning techniques and python code that I have accrued through this course, training and testing this model with attention to risks of overfitting to training data and data leakage.

Innehåll

[Abstract 2](#_Toc193567549)

[1 Inledning 1](#_Toc193567550)

[2 Teori 2](#_Toc193567551)

[2.1 MNIST 2](#_Toc193567552)

[2.2 Random Forest Classifier(RFC) 2](#_Toc193567553)

[2.3 Extra Trees Classifier(ETC) 2](#_Toc193567554)

[2.4 Bootstrap sampling 2](#_Toc193567555)

[2.5 K-Nearest Neighbour (KNN) Classifier 2](#_Toc193567556)

[2.6 GridsearchCV 2](#_Toc193567557)

[2.7 Hyperparameter 2](#_Toc193567558)

[3 Metod 3](#_Toc193567559)

[4 Resultat, Diskussion och Slutsatser 4](#_Toc193567560)

[5 Teoretiska frågor 5](#_Toc193567561)

[6 Självutvärdering 7](#_Toc193567562)

[Källförteckning 8](#_Toc193567563)

# Inledning

Maskininlärning betraktas som en del av AI(artificiell intelligens), då det är en process där man använder matematiska datamodeller för att en dator ska kunna lära sig utan direkt instruktion(kod) för att kunna göra korrekta förutsägelser i ny osedd data genom att träna att känna igen mönster i träningsdata. ” Det handlar om metoder för att med data "träna" datorer att upptäcka och "lära" sig regler för att lösa en uppgift, utan att datorerna har programmerats med regler för just den uppgiften.”(Wikipedia). Maskininlärning är en teknik som har utvecklats över många decennier och som fått en ordentlig boost i senare år när man kunnat skapa mer invecklade modeller och hitta nya tillämpningsområden för dessa, samt att datan som finns tillgänglig online i världen har ökat exponentiellt över det senaste decenniet. Från att tillämpas i sjukvårdsdiagnostik till chattbotar till bildigenkänning, det sistnämnda är vad vi kommer att fokusera på i denna rapport. MNIST databasen är en stor databas av handskrivna siffror som används ofta för träning av olika bildigenkänningssystem. Den innehåller 70000 bilder totalt och med dessa vill vi träna några modeller, se vilken som presterar bäst och sen testa på ny data för att se hur väl den faktiskt lärt sig att känna igen mönster och kunna förutsäga vad som är korrekt. Vi kommer att använda tre modeller för denna utvärdering, Random Forest Classifier, Extra Trees Classifier och k-Nearest Neighbor Classifier. Anledningen till att det blev just dessa modeller är pga resurs och tidsbrist. Support Vector Machines och XGBoost var jag intresserad av att ersätta Extra Trees med, men dessa modeller tog för lång tid att träna på min dator.

Syftet med denna rapport och medföljande jupyter notebooks är att visa hur man tränar och utvärderar en klassificeringsmodell, från början till slut.

# Teori

## MNIST

Mnist är en välkänd databas som innehåller totalt 70 000 bilder av handskrivna siffror från 0-9 i 28x28 pixelformat som vanligtvis används i träningssyfte för olika bildigenkänningsprogram. (Wikipedia) Dessa används för att träna och testa vår bildklassificerare.

## Random Forest Classifier(RFC)

RFC är en ensemblemetod som bygger på att kombinera flera beslutsträd för att göra föutsägelser, där varje träd tränas på en slumpmässigt (med återläggning) vald del av datan och sen baseras resultat på majoritetsröstning för klassificering. Detta för att förbättra noggrannhet och minska risk för överanpassning kontra att använda ett beslutsträd. (Wikipedia)

## Extra Trees Classifier(ETC)

Som RFC är detta också en ensemblemetod men har större slumpmässighet, då den tränas på olika slumpmässiga delar samt splittringarna görs slumpmässigt. Det leder till snabbare träning och ofta bättre generalisering, särskilt på stora dataset.

## Bootstrap sampling

Detta är när man kör flera träningsuppsättning genom att slumpmässigt (med återläggning) välja data.

## K-Nearest Neighbour (KNN) Classifier

Detta är en enkel klassificeringsmetod som hittar de ”k” närmaste grannarna till en given datapunkt, där klassificeringen görs genom att majoriteten av de k närmaste grannarna bestämmer klassen en ny datapunkt tillhör. Modellen är känslig för val av k och hur stor datan är. (Wikipedia)

## GridsearchCV

En metod för att optimera hyperparametrar i masinkinlärningsmodeller. Den gör det genom att gå igenom datan med alla möjliga listade hyperparametrar och korsvaliderar dessa för att se vilka uppsättningar av hyperparametrar som ger de mest effektiva resultat. Väldigt kraftfull men samtidigt resurskrävande metodik.

## Hyperparameter

Detta är inställningarna som visar hur modellen lär sig, man kan nästan tänka det som inställningarna i ett mixerbord för få bästa ljud från sångaren(datan) i en studio.

# Metod

Jag importerade MNIST dataset genom fetch\_openml från sklearn.datasets och delade upp i X variabel för själva datan(bilderna) och y variabel för labels, som jag sen delade upp i test och träningsdata. Efter att jag gjorde detta normaliserade jag X datan genom att dividera både test och träningsdatan med 255.0, för att göra varje pixel ett float värde mellan 0 och 1. Detta för att underlätta träningen av datan då modellen kan lägga mycket vikt vid 240 kontra 2 t.ex., men det är inte vad vi vill.

När detta är gjort, valde jag att dela upp träningsdatan i hälften, för att underlätta tiden för min dator att träna de olika modellerna med GridsearchCV ett par gånger, så jag kan finjustera hyperparametrar innan kör en GridsearchCV på hela träningsetet. Detta var endast pga tidsbrist och inte tillräckligt kraftfull dator. Jag skapade en dictionary med modellerna och hyperparametrarna som jag körde GridsearchCV på i en for loop.

Jag testade först några olika modeller som SVM och Logistic Regression, men dessa tog för lång tid så jag slopade dem och gick med Extra Trees som min tredje modell utöver Random Forest Classifier och k-Nearest Neighour Classifier.

Problemet var att hitta spannet på variabler som gav bäst resultat men samtidigt minimera överanpassning. Efter konsultation med ChatGPT angående vart smärtgränser för överanpassning och underpassning går, kom jag fram till att det bästa var att öka n\_estimators, sänka max\_depth och höja min\_samples\_split och min\_samples\_leaf på RFC och ETC, samt höja n\_neighbors på KNN. Detta sänkte träningsnoggrannheten på RFC och ETC lite, men höjde KNN.

Efter att jag hade finjusterat, tränade jag om dem alla på hela träningssetet, med X\_train\_normalized istället för X\_train\_partial och sparade den model som hade bäst träningsnoggrannhet med dem hyperparametrarna som gav bäst resultat. För att säkerställa att inget data läckage skulle hända under träningen hänvisades aldrig kod tillbaka till test datan i samma jupyter notebook, utan jag skapade en ny jupyter notebook som jag skulle utföra själva testet på.

Där laddade jag in min sparade modell, testade på testdatan och skapade lite illustrationer för att visa vilka siffror som den hade svårt för och visualisera för läsaren att även för mänskliga ögon är det svårt att särskilja ibland, för att illustrera hur kraftfull maskininlärning är.

# Resultat, Diskussion och Slutsatser

Reultatet var att KNN var den mest effektiva, som fick efter finjusteringar .9701 på träningsdatan, de andra modellerna fick inte högre än 0.9657. När jag sen testade KNN modellen på den osedda testdatan fick den högre score på 0.9720. Detta visade att KNN var den bästa modellen utav de tre för att prediktera korrekt siffrorna.

När jag fick bättre resultat på testdatan blev jag lite förvånad och orolig, men jag bortsåg från mina farhågor för jag var väldigt noga i hela processen att undvika dataläckage och tog mig tid att komma fram till de optimala hyperparametrarna genom att köra Gridsearch på flera modeller och flera olika spann av hyperparametrar. Hyperparametrarna på RandomForest tenderade Gridsearch att favorisera låga split och leaf samt hög max djup, vilket är risk för överanpassning. Här tog jag ett beslut att sätta dessa högre min\_sample\_leaf och högre min\_sample\_split än den föreslog och lägre max\_depth för att kontra risken för överanpassning till träningsdatan, då syftet är att den ska generalisera bra, inte memorisera träningsdatan.

Insåg efter att ha gjort modellerna och påbörjat skrivande av rapporten att min pipeline var helt meningslös, då den inte fyller ett syfte pga att jag redan normaliserat datan, jag kunde ha skippat den delen helt (då den var en förlegad bit från när jag skalade datan i min pipeline innan jag insåg att jag kunde normalisera genom att dividera med 255.0) och bara refererat direkt till min model dictionary i GridsearchCV. Ineffektiv kodning från min sida, så jag gick tillbaka och rengjorde detta.

En finurlig sak som jag inte riktigt förstod, var hur GridsearchCV föreslog n\_neighbors 3 och weight distance som bäst hyperparameter när det endast fick en training accuracy på .9600, när n\_neighbors 5 gav en training accuracy på .9701. Men efter reflektion, tänker jag att detta kan ha berott på min metod av att ha kört CV=3 i min träning av X\_train\_partial, istället för att ha kört CV=5, samt att om jag tränat hela träningssättet på cv=5 hade det kunnat visa helt andra resultat på de olika parametrarna, men det tog för lång tid. Min metod hade bra intentioner, men jag tror det fallerade lite i utförandet och introducerade lite förvanskning av vad som var bästa modellen från mig pga begränsningarna jag införde för att göra det snabbare och lättare för min dator att hantera.

# Teoretiska frågor

1. Träningsdelen används uppenbarligen för att träna modellen, visa den data som den ska försöka förstå ett mönster från. Valideringsdelen är ett ”förtest” där man får en liten inblick på hur den har lärt sig och i denna del finjusterar/kalibrerar man regulatorer och hyperparametrar för att minimera risk för underfitting och overfitting, efter denna del måste man träna modellen på både tränings- och valideringsdatan innan man går över till att testa modellen på testdatan som varit helt separerat från träningsflödet och detta säger om modellen är bra eller dålig.
2. Julia kan använda crossvalidation tekniker, t.ex. k-fold cross validation, där man delar upp träningsdatan i t.ex. 3 ”folds”, var en av dem är validering och de andra två är träning. Detta gör man lika många gånger som man har ”folds”, för att all data ska ha agerat valideringsset och all data träningsset. För varje iteration får man en Mean Square Error. Sen tar man medelvärdet av alla dessa värden och får där ett korsvalideringsvärde. Ett annat exempel på corrsvalidation är GridsearchCV , där man kan göra detta över en mängd olika hyperparametrar per modell och därmed få fram den inställning av hyperparameter som ger bäst resultat per modell.
3. Regressionsproblem är när en beroende variabel har kontinuerliga värden, där i dess simplaste form en variabel y är beroende av en eller flera oberoende x variabler och kan anta oändligt många kontinuerliga värden. Linjär regression, support vector machines, och beslutsträd är några exempel på modeller som kan hantera regressionsproblem. Potentiellt tillämpningsområden är bevakning av aktiekurs.
4. RMSE kan man tolka som medelvärdes avstånd mellan faktiska värdet och predikterade värdet. Är det litet värde, betyder det att den predikterar bra. Om det är stort, predikterar den dåligt. Alltså genomsnittlig storlek på felprediktering.
5. Klassificeringsproblem är är när en beroende variabel kan anta diskreta värden, oftast binärt, där beroende av de oberoende x variablerna kan y värdet anta en ”klass” utav två (eller flera). Exempel är t.ex. där man klassificerar y variabeln som om en kund churnar(lämnar företaget) eller inte churnar (binär klassificering), eller om en bild är av en häst, en hund, en katt eller annat(flervalsklassificering). Exempel på modeller är k-Nearest Neighbor Classifier, Random Forest Classifier, Support Vector Classifier. En confusion matrix är en tabellvisualisering av en modells prestationsförmåga.
6. K-means modellen är unsupervised klustringsmodell. Man bestämmer hur många grupper(K) man vill ha och så kommer den att slumpmässigt välja ”K punkterna” som kommer vara fokusen för grupperna, och varje punkt tilldelas gruppen som är det närmaste klustercenter. Det är användbart för anomalidetektion, då punkter som inte passar till något kluster betraktas som anomalier, vilket är avvikande beteende jämfört med datan, vilket kan innebära kriminell aktivitet t.ex.
7. Ordinal, one-hot och dummy variable encoding är alla tekniker för databehandling inom maskininlärning, där man ändrar kategorisk data till numerisk data för att modellen ska kunna tolka och lära sig. Ordinal är när den kategoriska datan har en naturlig ordning, t.ex. om vi har flera personer med olika nivåer av utbildning, då rangordnas det med siffror (Grundskola=1, Gymnasie=2, Universitet=3), eller med temperatur från kallast till varmast. One-hot och dummy är när det itne finns någon naturlig ordning. One-hot ändrar varje kategori till sin unika sträng av 0 och 1, t.ex. Volvo(1,0,0,0), BMW(0,1,0,0), VW(0,0,1,0) och Saab(0,0,0,1). Dummy variable gör samma sak, men slopar en av kolumnerna så det blir såhär Volvo(0,0,0), BMW(1,0,0), VW(0,1,0) och Saab(0,0,1).
8. Göran har rätt att det är antingen ordinal eller nominal, det kan inte vara båda två, det är motsägelsefullt. Det Julia säger handlar mer om att man kan sätta ett ordinalt värde på en nominal variabel, men det krävs att det har en logisk förklaring som är lätt att förstå. Färger är naturligt nominala, men färger i kontext av mina favoritfärger, då blir det ordinalt.
9. Streamlit är en open-source Python bibliotek som gör det möjligt att skapa interaktiva webbapplikationer som går att dela online. Det är användbart inom maskininlärning för att kunna dela med andra för att testa ens modells förmåga att prediktera osedd data som en ny användare skapar, som var VG delen i denna kurs.

# Självutvärdering

1. Utmaningar du haft under arbetet samt hur du hanterat dem.   
   Tidsuppskattning och begränsning. Det var så många olika modeller som är sjukt tidskrävande speciellt med GridsearchCV(t.ex. SVM) och jag insåg inte riktigt hur tid- och resurskrävande det är att träna modellerna och jag har inte en jättebra dator. Jag fick slopa modeller som jag ansåg tog för lång tid, spåna på andra och börja om med dem. Stötte på problemet med överanpassningsrisk när den föreslog hyperparametrar som var för riskabla, så jag tog beslut att gå emot förslag och sätta en nivå jag tyckte borde vara mellan underanpassning och överanpassning.
2. Vilket betyg du anser att du skall ha och varför.

Anser att jag har uppfyllt godkänt nivå pga att jag har skapat ett maskininlärningsflöde från start till slut som tar hänsyn till dataläckage, överanpassning och underanpassning samt tidsbristen jag hade då jag påbörjat så sent pga personliga omständigheter. Jag är nöjd med min insats under dessa omständigheter, jag har lyckats komprimera 4 veckors lära på en vecka och fått fram resultat.

1. Något du vill lyfta fram till Antonio?

Maskininlärning är väldigt intressant, dock väldigt invecklat koncept att förstå. Krävs en pedagogiskt lagd person att lära ut och tyvärr kände jag att Linus föreläsningar inte gav mig något utan förvirrade mig mer, det var först när du tog över som jag fick någon form av aha-moment.

# Källförteckning

https://sv.wikipedia.org/wiki/Maskininl%C3%A4rning

<https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm>

<https://en.wikipedia.org/wiki/MNIST_database>

https://en.wikipedia.org/wiki/Random\_forest