Bildigenkänning med MNIST databasen

Identifierar handskrivna siffror med hjälp av maskininlärning



Filip Östlund

EC Utbildning

Maskininlärnings projekt

202403

# Abstract

This project explores digit recognition using the MNIST dataset, implemented through Python and the scikit-learn library. Utilizing Streamlit, users can either draw or upload images of handwritten digits for prediction. The study compares various machine learning models, including Support Vector Machines, Random Forest, and Logistic Regression, to determine the best performer in digit recognition tasks. The selected model serves as the primary classifier for the project's image classification. Additionally, significant attention is given to preprocessing uploaded images to enhance model accuracy. The results show that a high percentage accuracy is attainable in digit recognition tasks but also that when it comes to predicting from pictures or drawing the accuracy is not as high.

Innehållsförteckning

[Abstract 2](#_Toc162015335)

[1 Inledning 1](#_Toc162015336)

[1.1 Syfte 1](#_Toc162015337)

[1.2 Frågeställningar 1](#_Toc162015338)

[2 Teori 2](#_Toc162015339)

[2.1 Logistic Regression 2](#_Toc162015340)

[2.1.1 Hyperparametrar för Logistic Regression 2](#_Toc162015341)

[2.2 Random Forest Classifier 2](#_Toc162015342)

[2.2.1 Hyperparametrar för Random Forest Classifier 2](#_Toc162015343)

[2.3 Support Vector Machines 2](#_Toc162015344)

[2.3.1 Hyperparametrar för Support Vector Machines 3](#_Toc162015345)

[2.4 Grid Search (GridSearchCV) 3](#_Toc162015346)

[2.5 Scaler (StandardScaler) 3](#_Toc162015347)

[2.6 Deskewing 3](#_Toc162015348)

[2.7 Evalueringsmetoder 3](#_Toc162015349)

[2.7.1 Accuracy 4](#_Toc162015350)

[2.7.2 Confusion Matrix 4](#_Toc162015351)

[2.8 Scikit-Learn 4](#_Toc162015352)

[2.9 Streamlit 4](#_Toc162015353)

[3 Metod 5](#_Toc162015354)

[3.1 MNIST Dataset 5](#_Toc162015355)

[3.2 Exploratory Data Analysis (EDA) 5](#_Toc162015356)

[3.3 Data Preparation 6](#_Toc162015357)

[3.4 Modellträning, validering och test 6](#_Toc162015358)

[3.5 Förbehandling av bilder 7](#_Toc162015359)

[3.6 Streamlit applikation 7](#_Toc162015360)

[4 Resultat 8](#_Toc162015361)

[4.1 Modellernas resultat 8](#_Toc162015362)

[4.2 Streamlit resultat 8](#_Toc162015363)

[5 Slutsatser 11](#_Toc162015364)

[5.1 Frågeställning 1 11](#_Toc162015365)

[5.2 Frågeställning 2 11](#_Toc162015366)

[5.3 Frågeställning 3 11](#_Toc162015367)

[5.4 Förbättringar och slutord 11](#_Toc162015368)

[6 Teoretiska frågor 12](#_Toc162015369)

[7 Självutvärdering 14](#_Toc162015370)

[Källförteckning 15](#_Toc162015371)

# Inledning

Maskininlärning har blivit en viktig del av modern datavetenskap, och används ofta för att automatisera uppgifter som traditionellt kräver mänsklig intelligens. I detta projekt utforskar vi tillämpningen av maskininlärning på ett konkret problem - att identifiera handskrivna siffror.

## Syfte

Syftet med denna rapport är att undersöka implementeringen av sifferigenkänning med hjälp av MNIST-datasetet, med fokus på användning av Python och scikit-learn-biblioteket. Vi kommer även undersöka Streamlit-bibloteket och dess förmåga att enkelt bygga webbapplikationer för maskininlärning. För att uppnå detta syfte kommer följande frågeställningar att besvaras:

## Frågeställningar

1. Går det att uppnå 95% accuracy med hjälp av olika maskininlärningsmodeller, såsom Support Vector Machines, Random Forest och Logistic Regression, vid sifferigenkänning baserat på handskrivna bilder?
2. Går det att använda förbehandlingsmetoder för att få de uppladdade bilderna till ”MNIST-format” och finns det några andra förbehandlingsmetoder man kan använda för att optimera prestanda för maskininlärningsmodellerna och hur påverkar detta resultat?
3. Går det att använda Streamlit för att skapa en interaktiv plattform där användare kan rita eller ladda upp bilder av handskrivna siffror för att förutsäga vilken siffra det är?

# Teori

Denna sektion består av relevant teori för att förstå projektets kontext.

## Logistic Regression

Logistisk regression är en linjär modell som använder logistikfunktionen för att förutsäga sannolikheten för att en instans tillhör en specifik klass. (Géron, 2019, s. 142)

En bild som visar linje, skärmbild, Graf, text

Automatiskt genererad beskrivning

Figur 1. Logistiska funktionen

### Hyperparametrar för Logistic Regression

Det finns flera hyperparametrar för logistisk regression men den som användes i detta projekt var parametern C. Desto högre värde av C, desto mindre regulariserad blir modellen (Géron, 2019, s. 149).

## Random Forest Classifier

Random forest klassificerare är en ensemble learning algoritm som konstruerar flera beslutsträd under träningen. Varje träd är byggt på en slumpmässig delmängd av funktioner och bidrar till den slutliga prediktionen genom en majoritetsröstningsmekanism. För klassificeringsproblem är det klassen som väljs av flest träd. För regressionsporblem returneras medel- eller genomsnittsprognosen från de individuella träden (Géron, 2019, s. 199).

### Hyperparametrar för Random Forest Classifier

De hyperparametrar som användes för random forest var n\_estimators och max\_depth.

N\_estimators styr antalet träd som skapas. Ju fler träd, desto mer komplex blir modellen och desto bättre kan den potentiellt generalisera över data. Dock finns det en punkt där att lägga till fler träd inte längre ger signifikant förbättring i prestanda och kan även öka tränings- och beräkningskostnader (Géron, 2019, s. 206).

Max\_depth specificerar maximala djupet av varje träd. Ett djupare träd kan bättre anpassa sig till träningsdata, men kan också leda till överanpassning. Att begränsa max\_depth kan hjälpa till att förhindra överanpassning och göra modellen mer generaliserbar till nya data (Géron, 2019, s. 184).

## Support Vector Machines

Support Vector Machines (SVM) fungerar genom att konstruera beslutsgränser, representerade som hyperplan, för att separera olika klasser i funktions utrymmet. I klassificeringsuppgifter strävar SVM:er efter att maximera marginalen mellan klasser, vilket är avståndet mellan beslutsgränsen och de närmaste datapunkterna från varje klass. Vid regressionsproblem försöker SVM passa in en "gata" mellan datapunkter. Målet är att hitta den bredast möjliga gatan samtidigt som man säkerställer att de flesta instanser faller inom den (Géron, 2019, s. 157).

### Hyperparametrar för Support Vector Machines

Hyperparametrarna som användes för SVM var kernel, gamma och C.

Kernel specifierar vilken kärnfunktion som ska användas för att separera data. De vanligaste värdena och också de som vi använde är 'rbf' (radial basis function) och 'poly' (polynomial). Både rbf och poly används för att hantera icke-linjära problem. Rbf är vanligtvis mer flexibel medans poly passar bättre för att modellera icke-linjära relationer med polynomiella funktioner (Géron, 2019, s. 160-163).

Gamma fungerar som en regleringshyperparameter. Om man ökar gamma blir kurvan som skapats runt träningsinstanserna tightare. För högt gamma kan därför leda till overfitting, vilket innebär att den presterar bra på träningsdatan men inte generaliserar väl till nya data (Géron, 2019, s. 162).

C styr toleransen för felklassificeringar. Högre C innebär mindre tolerans och en hårdare marginal. Men för högt C kan precis som Gamma leda till overfitting medan ett lägre värden på C leder till en mjukare marginal och tillåter vissa felklassificeringar, vilket kan resultera i en jämnare kurva som generaliserar bättre till nya data (Géron, 2019, s. 162).

## Grid Search (GridSearchCV)

För att få en så bra modell som möjligt behövs det testas olika hyperparameterkombinationer för att hitta vilka som fungerar bäst. För att detta inte ska ta lång tid kan man i stället använda sig av grid search. Man anger vilka hyperparametrar man vill experimentera med och vilka värden de ska testa, och GridSearchCV utvärderar sedan alla möjliga kombinationer av hyperparameter-värden med hjälp av korsvalidering (Géron, 2019, s. 79).

## Scaler (StandardScaler)

StandardScaler är en metod för att standardisera egenskaper genom att ta bort medelvärdet och skala dem till enhetsvarians. Det innebär att varje egenskapsvärde omvandlas till ett z-värde baserat på dess medelvärde och standardavvikelse. Detta är användbart för att säkerställa att alla egenskaper har liknande skala, vilket kan förbättra prestandan för många maskininlärningsalgoritmer (Pedregosa, F. et al., 2011, Scikit-learn documentation).

## Deskewing

”Deskewing” eller på svenska snedställning kan användas för att vrida upp och centrera objekt i bilder. Denna ”deskewing” som används i detta projekt är baserad på artikeln A Guide to MNIST skriven av Dibya Ghosh och Alvin Wan (2016) där de beskriver metodiken som:

1. Hitta masscentrum för bilden för att ta reda på hur mycket vi behöver justera bilden
2. Hitta kovariansmatrisen för bildens pixelintensiteter (vi kan använda detta för att uppskatta vinkelsnedstället)

## Evalueringsmetoder

För att se hur de olika modellerna har presterat har accuracy använts till största delen. Men för att få en bättre inblick i modellen så användes även confusion matrix.

### Accuracy

Accuracy är ett prestandamått som används för att bedöma hur väl en klassificeringsmodell presterar. Det beräknas genom antalet korrekta prediktioner dividerat med det totala antalet prediktioner, uttryckt som en procentandel (Géron, 2019, s. 4).

### Confusion Matrix

Confusion Matrix är ett verktyg för att utvärdera prestandan hos en klassificeringsmodell. Den ger en översikt över hur modellen har klassificerat exempel i förhållande till de faktiska klasserna. Varje rad i confusion matrixen representerar en faktisk klass medan varje kolumn representerar en predikterad klass av modellen (Géron, 2019, s. 92).

## Scikit-Learn

Scikit-learn är ett öppen källkods Python-bibliotek för maskininlärning som erbjuder användarvänliga verktyg för att bygga och analysera maskininlärningsmodeller. Det innehåller algoritmer för klassificering, regression, klusteranalys och mycket mer, vilket gör det till ett populärt val för maskininlärningsprojekt. (Pedregosa, F. et al., 2011, Scikit-learn documentation)

## Streamlit

Streamlit är ett öppen källkods Python-bibliotek som gör det enkelt att skapa och dela vackra, anpassade webbapplikationer för maskininlärning och datavetenskap. Streamlit låter dig köra din applikation på din dator eller ladda upp den online (Treuille, A. et al., 2018, Streamlit documentation)

# Metod

Detta kapitel beskriver steg för steg projektet i sin helhet.

## MNIST Dataset

MNIST-datasetet är en samling av 70 000 bilder med siffror handskrivna av gymnasieelever i Amerika och anställda på US Census Bureau. Varje bild har en label med den siffra som bilden representerar. Det finns som sagt 70 000 bilder, och varje bild har 784 funktioner. Detta beror på att varje bild är 28x28 pixlar, och varje funktion representerar helt enkelt en pixels intensitet, från 0 (vitt) till 255 (svart) (Géron, 2019, s. 87-88). Nedan ser vi hur en siffra är uppbyggd i datasetet.

En bild som visar text, skärmbild, mönster, Rektangel

Automatiskt genererad beskrivning

Figur 2. En MNIST siffra i "raw" form

Efter att ha importerat datasetet splittrar vi det i 3 olika delar. En träningsdel, en valideringsdel och till sist en testdel. Eftersom datasetet är ganska stort och vi till en början vill hålla nere träningstiderna för våra modeller tar vi endast tio tusen observationer till träningsdelen, och två tusen var till valideringsdelen och testdelen.

## Exploratory Data Analysis (EDA)

Efter att ha delat in datan i sina respektive delar kan vi göra en analys av våra träningsdelar. Vi börjar med att kolla på formen på X (features) och y (labels) för att se om det stämmer överens med den infon vi redan har om MNIST. Som vi ser nedan stämmer det överens, 10 tusen rows i både X och y + 784 columns i X precis som vi vet är mängden pixlar bilden har.

Shape of X\_train: (10000, 784)

Shape of y\_train: (10000,)

Efter att ha sett formen på datan kollade vi på hur många av varje label som finns i datan. Nedan ser vi resultatet av det. Som grafen visar har träningsdelen ungefär 1000 av varje siffra. Det finns flest av siffran 1 och minst av siffran 5 i träningsdelen.

En bild som visar text, skärmbild, diagram, Teckensnitt

Automatiskt genererad beskrivning

Figur 3. Mängd av olika siffror i träningsdatan

Nästa del av EDA:n tittades det på hur 10 st slumpmässigt valda siffror faktiskt ser ut och jämförde dem med sina labels för att se att de stämmer. Nedan ser vi plottade siffror och deras labels. Det vi kan se från denna data är att det finns flera versioner av samma siffra. Vissa är skrivna snett, vissa rakt osv.

En bild som visar text, typografi, kalligrafi, design

Automatiskt genererad beskrivning

Figur 4. MNIST labels och siffror plottade

## Data Preparation

Utifrån EDA:n skulle man nu egentligen direkt kunna gå vidare till modelltestning men eftersom en stor del av bokstäverna både är snea eller inte är i centrum av bilden valde jag att centrera och vrida upp alla siffror så dem hamna bättre i bilden vilket gör det enklare för modellerna att läsa av siffrorna. Detta gjordes med hjälp av Deskewing som beskrevs i teorin. Deskewing gjordes på hela träning, validerings och testdelen. Sista steget i dataförberedelsen är att använda StandardScaler för att standardisera alla features. Modellen SVM som vi använder är speciellt känsligt för data som inte är standardiserad. I detta fall vet vi redan innan att all data i MNIST databasen är uppbyggt på samma sätt men valet att standardisera datan gjordes ändå.

En bild som visar Grafik, Electric blue, Teckensnitt, grafisk design

Automatiskt genererad beskrivning

Figur 5. MNIST orginal vs deskewed

## Modellträning, validering och test

Nästa steg var att träna de valda modellerna och sedan testa dem på valideringsdatan. Modellerna som valdes var Logistic Regression, Random Forest Classifier och Support Vector Machines. Varför just dessa modeller valdes var utifrån tidigare erfarenhet, jag hade innan projektet jobbat med alla tre som är välkända modeller i scikit-learn när det kommer till just klassificeringsproblem som detta.

Först valdes olika hyperparametrar ut för varje individuell modell. Dessa hyperparametrar kunde man sedan tillsammans med GridSearchCV träna modellen och samtidigt se vilka hyperparametrar som fungerade bäst. Efter att modellerna tränats så testades de på valideringsdelen av datan för att se vilken som fick bäst accuracy. Resultatet som man fick ut var, vilken accuracy + vilka hyperparametrar som var de bästa. I detta steg justerades även hyperparametrar för se om det kunde påverka resultat.

Utifrån resultaten i föregående steg hade vi nu en modell som presterat bäst som vi kunde gå vidare med. Nästa steg var så att testa denna modell på testdatan och se att den även där presterade ett bra resultat. Efter testning av den valda modellen tränades modellen sedan på en större del av datan. Tidigare tränade vi bara på 10 tusen siffror, nu skapade vi i stället en träningsdel på 56 tusen siffror och en testdel på 14 tusen siffror. Efter att ha testat modellen på större del och även testat här för att se en accuracy tränades till slut modellen på hela datasetet för att få maximal uppträning inför de bilder som vi ska prediktera på.

## Förbehandling av bilder

Efter att ha tagit fram en färdig modell var nästa steg i projektet att se hur modellen kan prediktera egna bilder som vi skriver för hand och fotar. Men när vi tar ett foto med telefon eller dator blir formatet ett helt annat än den data som vi tränat modellerna på.

För att förbereda bilden för modellen genomgår bilderna en serie förbehandlingssteg. Först konverteras bilden till gråskala för att förenkla bearbetningen och minska beräkningskomplexiteten. Därefter ändras bildstorleken till 28x28 pixlar för att passa modellens krav. Efter storleksändringen normaliseras pixlarnas värden till intervallet [0, 1] genom att dela varje pixelvärde med 255, vilket är den maximala pixelfärgintensiteten. Med denna nya normalisering kan vi ytterligare förbättra bildens kontrast och göra den mer lämplig för modellträning genom att invertera den, vilket innebär att de ljusa områdena blir mörka och vice versa. Samtidigt som vi inverterar multiplicerar vi även tillbaka pixlarna med 255 för att det är det formatet som MNIST är i. Efter inversion beräknas medelvärdet för pixelfärgerna, och alla pixlar med värden lägre än eller lika med medelvärdet sätts till noll. Detta steg hjälper till att minska bakgrundsljudet och förtydliga de viktigaste funktionerna i bilden. Här var det mycket fram och tillbaka med hur högt värde som skulle sättas till noll. Det slutade med ett högre värde, men mer information om det i slutsatsen.

## Streamlit applikation

Sista steget i processen var att bygga en applikation för att användaren ska kunna gå in och ladda upp egna bilder på siffror eller rita egna siffror och utifrån det kunna få prediktioner på vilken siffra som modellen känner av. Här var det också viktigt att få med att bilderna inte kan vara tagna hur som helst, utan måste var skrivna med en helt vit bakgrund. Samt att prediktionerna inte alltid kommer vara korrekta.

# Resultat

## Modellernas resultat

De tre valda modellerna som tränades gav detta resultat som vi ser nedan. Den bästa modellen var SVM-modellen som hade en accuracy på 97%. En sak att tillägga här är att innan deskewing fick alla modeller en accuracy som i snitt var 0,3% lägre än efter deskewing. Detta resultat är också efter lite olika justeringar kring hyperparametrar, men eftersom frågeställningen nöjde sig med 95% accuracy valde jag att gå vidare utan att lägga för mycket tid på hyperparametersjustering.

Tabell 1. Modellernas resultat

|  |  |  |
| --- | --- | --- |
| Modell | Bästa hyperparametrar | Accuracy |
| Logistic Regression | {'C': 0.01} | 0.944 |
| Random Forest Classifier | {'max\_depth': 20, 'n\_estimators': 200} | 0.9585 |
| Support Vector Machine | {'C': 0.025, 'gamma': 0.25, 'kernel': 'poly'} | 0.97 |

SVM är modellen som vi kunde gå vidare med. Den tränades på testdatan som gav ungefär samma resultat. Det blev en accuracy på 96,7% och här skapades även en confusion matrix som vi ser nedan. Modellen presterade väldigt bra, det enda man kan se att den hade lite problem med var att den predikterade ett fåtal 2:or som 8:or, samt ett fåtal 3:or som 2:or.

En bild som visar skärmbild, kvadrat, text, Färggrann

Automatiskt genererad beskrivning

Figur 6. Confusion Matrix på valda modellens prediktioner

Efter att ha tittat på den valda modellens prestation på endast en liten del av testdatan så testades den på mer data och där blev resultatet precis samma som innan 96,7% accruacy.

## Streamlit resultat

Själva Streamlit applikationen gick väldigt enkelt att bygga. Nedan ser man startsidan för applikationen. Allt man behövde kunde enkelt hittas i Streamlit dokumentationen.

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

Figur 7. Streamlit startsida

Det skapades ett val mellan att rita siffror och uppladda egna bilder på siffror. När man ritar siffror fungerar de flesta siffror utan problem. Jag gjorde ingen dokumenterad testning men ritade en stor mängd siffror och skulle estimera cirka 80% accuracy. Nedan är varje siffra ritad i streamlit applikationen och predikterad korrekt.

En bild som visar text, skärmbild, Teckensnitt, nummer

Automatiskt genererad beskrivning

Figur 8. Korrekt predikterade ritade siffror

De siffrorna som det ibland kunde bli problem med var siffran 7 och 9. Nedan ser ni exempel på när modellen predikterat fel. Just Det kan vara att nians övre del blivit lite för stor eller att sjuan blivit för rak. Det vet man ej utifrån detta resultat och man skulle kunna lägga mer tid på att undersöka bakomliggande anledningar.

En bild som visar text, skärmbild, Teckensnitt, nummer

Automatiskt genererad beskrivning

Figur 9. Felaktiga prediktioner från ritade siffror

Nästa steg var att prediktera en egen uppladdad bild. Det som var problem med denna prediktion var hur bra fotoförbehandlingen fungerade. I början testade jag använda kamera direkt i Streamlit för att ta bilderna men det fungerade ej då det alltid blev någon sorts skugga i något hörn eller något annat som störde modellen i sin prediktion. På grund av detta valde jag att gå vidare med att man laddar upp en bild i stället för att använda kameran direkt i applikationen. Då får man bäst förutsättningar för prediktionen. Det lades till en text (For the prediction to work use a picture of a digit with a white background, preferably as little shadows and background "noise" as possible) på kraven för att modellen skulle kunna prediktera på bäst sätt, för att personen inte skulle ladda upp vilken bild som helst. Nedan är tre prediktioner som är korrekta samt en felaktig prediktion där bilden hade för mycket bakgrundsbrus.

En bild som visar text, skärmbild, nummer, Teckensnitt

Automatiskt genererad beskrivning

Figur 10. Fototestning med modellen

Jag testade olika handstilar och experimenterade även med olika bakgrunder. Som vi ser på resultatet kan skugga ställa till det ordentligt och var även det som blev problem när jag tidigare använde mig direkt av kamera. Jag testade alla siffror och fick rätt prediktion minst 2-3 gånger per siffra men denna variant är inte lika stabil som när vi istället ritar siffrorna. För att testa den riktiga accuracyn på egen tagna bilder skulle det behövas mer tid och fler bilder än de jag testade med.

# Slutsatser

## Frågeställning 1

Går det att uppnå 95% accuracy med hjälp av olika maskininlärningsmodeller, såsom Support Vector Machines, Random Forest och Logistic Regression, vid sifferigenkänning baserat på handskrivna bilder?

Ja, som resultatet visar var det egentligen inte särskilt svårt att uppnå 95% accuracy på de handskrivna siffrorna från MNIST datasetet. Efter att ha använt deskewing på databasen uppnådde både Random Forest och Support Vector Machines en 95 % accuracy. Logistic Regression var inte långt ifrån med sin 94,4% accuracy. Jag tror även att det går att uppnå 95% även med den om man lägger ner mer tid på att undersöka olika hyperparametrar.

## Frågeställning 2

Går det att använda förbehandlingsmetoder för att få de uppladdade bilderna till ”MNIST-format” och finns det några andra förbehandlingsmetoder man kan använda för att optimera prestanda för maskininlärningsmodellerna och hur påverkar detta resultat?

Även här blir svaret ja. Som vi såg i resultatet kunde de uppladdade bilderna göras om till ”MNIST format” för att modellen skulle kunna prediktera fram ett resultat. Här testade jag väldigt mycket olika tekniker för att komma fram till det som fungerade bäst. Ett stort problem för mig var att bilderna hade för mycket skuggor och bakgrunds ”brus”. Det som jag gjorde för att lösa det nämnde jag i metoden om att höja värdet på gränsen där pixlar skulle nollställas. Det slutade med att jag använde medelvärdet multiplicerat med 1,5 för att få minimera bakgrundsbrus. Det gick även att gå högre men då kunde det också bli problem om siffrorna som testades hade smalare stil.

En förbehandlingsmetod som användes på MNIST datan innan träning var deskewing som nämndes i metoden. Jag testade även använda deskewing på de uppladdade bilderna innan de skickades till modellen men jag fick inte någon större skillnad på resultatet och det blev ibland problem i stället och jag valde då att inte inkludera det i min Streamlit applikation. När man får valet att ladda upp en egen bild så ser jag oftast att man inte tar en helt sned bild utan en rak bild så jag valde att gå vidare med det i åtanke. Om jag skulle lagt ner mer tid på projektet skulle jag velat titta mer på deskewing och testat ladda upp helt sneda bilder och se om prediktionerna fungerar.

## Frågeställning 3

Går det att använda Streamlit för att skapa en interaktiv plattform där användare kan rita eller ladda upp bilder av handskrivna siffror för att förutsäga vilken siffra det är?

Sista svaret blir också ja. Streamlit var superenkelt att använda och följa dokumentationen. Som resultatet visade var det inga problem alls att skapa det man ville. Med lite mer tid med biblioteket tror jag man kan bli riktigt vass och bygga applikationer som liknar många av dagens hemsidor.

## Förbättringar och slutord

Projektet blir ganska ”strypt” när man håller sig till vanlig maskininlärning och Scikit-learn biblioteket när det kommer till modeller. Under min egen forskning innan projektet stötte jag på en stor mängd neural networks som utan problem löste dessa uppgifter med nästan 100% accuracy. Jag når också en väldigt fin accuracy med modellerna på MNIST datasetet, men när det kommer till accuracyn på egna uppladdade bilder och ritade bilder har jag inte tagit ett exakt mått på accuracyn men jag är inte i närheten av 95%. Jag tror att samma modell som nu används kan bli ännu bättre om man lägger ner mer tid men om jag skulle ta mig an samma problem igen skulle jag välja en annan väg än gammal hederlig maskininlärning.

# Teoretiska frågor

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

Svar: Man använder ”tränings datan” för att skapa modeller. Man använder ”validerings datan” för att utvärdera modellerna och välja den bästa. Sedan använder man ”test datan” för att testa den valda modellen och även ta fram ett estimat över hur bra modellen fungerar.

1. 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 kan använda sig av Cross Validation. Vanligast är k-fold cross validation där man delar upp träningsdatan i k delar och sedan tränas modellen på alla k-delar utom en som används som validering. Detta upprepas tills varje del av tränings datan har använts som validering och man har då k prediktionsfel. På dessa prediktionsfel räknas medelvärdet för att få ett estimat på prediktionsfelet för modellen.
2. Vad är regressionsproblem? Kan du ge några exempel på modeller som används och potentiella tillämpningsområden? Svar: Ett regressionsproblem är supervised learning och är när den beroende variabeln har kontinuerliga värden. T.ex. . Olika modeller för regressionsproblem kan vara Linjär regression, till exempel som tidigare i frågan. Predicta en persons inkomst med hjälp av ålder. En annan modell för regressionproblem är Support Vector Machines som kan användas till exempel för att räkna ut en students betyg med hjälp av deras nedlagda tid på att studera.
3. Hur kan du tolka RMSE och vad används det till?

Svar: RMSE kan vi tolka som våra prediktioners medelavstånd till de sanna värdena. RMSE används ofta för att utvärdera en modells prediktioner.

1. Vad är ”klassificieringsproblem? Kan du ge några exempel på modeller som används och potentiella tillämpningsområden? Vad är en ”Confusion Matrix”? Svar: Klassificeringsproblem är supervised learning och är när man har en beroende variabel som kan anta två eller n-mängd klasser. En modell för klassificering är Beslutsträd, kan användas t.ex. om man ska dela in en grupp i olika färger (röd, grön, blå). En annan modell man kan använda för klassificering är Support Vector Machines, kan användas om vi t.ex. ska dela in blommor i två olika klasser (versicolor, virginica). En confusion matrix är en tabell som används för att utvärdera prestandan hos en klassificeringsmodell. Den visar antalet korrekta och felaktiga förutsägelser gjorda av modellen jämfört med de verkliga klasserna i testdatan.
2. Vad är K-means modellen för något? Ge ett exempel på vad det kan tillämpas på.

Svar: K-means modellen är unsupervised learning och delar in data i k-mängd kluster. Vi väljer k själva. K-means kan t.ex. användas om vi har en kundbas och vill dela in kundbasen i olika delar för att kunna marknadsföra bättre utifrån varje grupps attributer.

1. Förklara (gärna med ett exempel): Ordinal encoding, one-hot encoding, dummy variable

Svar: Kategorier: Bullpadel, Varlion, Stiga

Ordinal encoding så ger man varje kategori en variabel i ordning. Så t.ex. om man tittar på en professionell padelspelares lön så kan man säga att lönen beror på vilket märke man har som sponsor. I detta fall är Stiga ”minsta” märket och det blir då Stiga = 1. Osv Varlion = 2 och Bullpadel = 3.

One-hot encoding så spelar ordningen ingen roll. T.ex. vilken padelspelare är bäst? Då ska ordningen på märkena egentligen inte spela någon roll. One-hot encoding tittar då på har padelspelaren bullpadel, antingen 0 som inte har eller 1 som betyder att spelaren har det märket, sen samma sak med varlion, 0 eller 1 och Stiga 0 eller 1.

Dummy variable encoding fungerar ungefär som one-hot encoding. Enda skillnaden är att man vet att spelaren antingen har Bullpadel 0 eller 1, sen Varlion 0 eller 1 då om det är 0 på båda dem så vet man spelaren har Stiga. Då behöver man inte skapa en variable för Stiga utan det är Stiga när båda de andra variablerna är 0.

1. 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: Julia har rätt. Det måste tolkas. Man får utgå ifrån vad ens ”target” är för något. Är det inkomst kanske inte färg på skjorta spelar roll. Medans om man som i exemplet har ”target” som snygghet kanske fler tycker röd skjorta är ”snyggare” och den måste därför rankas högre än en grön skjorta.
2. Vad är Streamlit för något och vad kan det användas till?

Svar: Streamlit är ett gratis och ”open source” ramverk för machine learning och data science. Som kan användas för att snabbt transformera python scripts till appar eller dashboards.

# Självutvärdering

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

Den största utmaningen var i alla fall till en början att göra alla val. Vilken modell, vilka hyperparametrar osv. Men tillslut valde jag tre ganska simpla modeller som vi jobbat med tidigare. Nästa utmaning var att processa bilden och det tog ett tag att få rätt. Även om det inte blev 100% är jag nöjd med resultatet på den korta tiden.

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

Jag anser att jag ska ha VG då jag tycker jag skaffat mig bra koll på machine learning i denna kursen. Känner mig säker på modellval, hyperparametrar och modellutvärdering. Rapportskrivandet känns också bra men jag tror att jag har lite svårt att korta ner de bitar som inte behövs. Men det är någonting som jag vill jobba vidare med.

1. Något du vill lyfta fram till Antonio?

Super intressant kurs och grymt lärorikt som vanligt!!

# Källförteckning

Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow (Andra upplagan). O’Reilly Media, Inc.

Dibya Ghosh, Alvin Wan. (2016). A Guide to MNIST, <https://fsix.github.io/mnist/>, Hämtad 2024-03-17

Pedregosa, F. et al., (2011). Scikit-learn documentation, <https://scikit-learn.org/0.16/documentation.html>, Hämtad 2024-03-18.

Treuille, A. et al., (2018) Streamlit documentation, <https://docs.streamlit.io/>, Hämtad 2024-03-20.