MNIST Dataset

Utveckling av en ML-applikation med ramverket Streamlit

Dogukan Dogru

EC Utbildning

Data Science

ML-Rapport

2025-03

Abstract

This project focuses on utilizing pre-trained models on the MNIST dataset to classify handwritten digits. I develop algorithms to preprocess real-world data and subsequently classify them as digits. It includes the implementation of an application that processes and analyzes data in real-time, demonstrating how machine learning models can be applied to real-world problems.

Innehållsförteckning

1 Inledning

Datorseende eller computer vision (CV) på engelska är ett stort område med många tillämpningar. De används i till exempel självkörande bilar eller kameror med ansiktsigenkänning. Till och med din telefon kan vara utrustad med ansiktsigenkänning för att låsa upp enheten.

I detta arbete kommer jag att utveckla egna algoritmer för att prediktera handskrivna siffror. Jag kommer att förarbeta bilderna och sedan använda dessa som indata till modeller från det framstående biblioteket scikit-learn för att prediktera vilken siffra som avbildas.

Följande frågeställningar ska besvaras:

1. Kan vi utveckla modeller som uppnår en accuracy på över 80% på osedda data?
2. Kan vi skapa ett användarvänligt gränssnitt med Streamlit och produktionsätta applikationen så att användarna enkelt kan dra nytta av den.

2 Teori

De vanligaste metoderna inom supervised learning är regression, som används för att förutsäga kontinuerliga värden, och klassificering, som används för att förutsäga kategorier. Jag kommer att fokusera på klassificering, eftersom MNIST-datasettet representerar ett klassificeringsproblem.

2.1 Klassificeringsmodeller

Klassificeringsmodeller är en grundläggande del av maskininlärning och används för att kategorisera data i fördefinierade klasser.

Binär klassificering innebär att förutsäga en av två möjliga utfall. Ett exempel är att avgöra om en kund kommer att "churna" (lämna) eller inte.

Multiklass-klassificering används när det finns tre eller flera möjliga klasser. Ett exempel är vår uppgift, där det finns tio möjliga utfall eller klasser.

2.2 Metrics

För att bedöma hur våra modeller presterar har vi tillgång till flera olika mått. Till exempel accuracy, precission, recall och confusion matrix.

2.2.1 Accuracy

Det vanligaste och enklaste måttet är accuracy score. Accuracy score kan dock vara missvisande om datasettet du arbetar med inte är jämnt fördelat, det vill säga om en klass är mycket mer frekvent än andra. MNIST-datasettet är dock inte ett sådant datasett, vilket vi kan se i figuren där vi visar fördelningen mellan klasserna. (Figur 1)

$$Accuracy = \frac{Korrekta\ prediktioner}{Antalet\ prediktioner}$$

2.2.2 Precision

Mäter andelen korrekta positiva prediktioner. (Figur 2)

$$\Pr e\,cision = \frac{TP}{TP + FP}$$

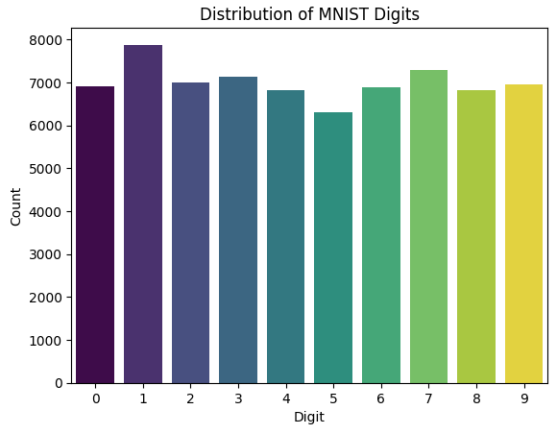
2.2.3 Recall

Mäter andelen korrekta positiva prediktioner i förhållande mot faktiska positiva instanser. (Figur 2)

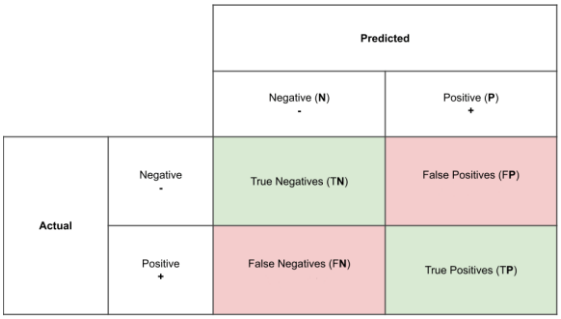
$$Recall = \frac{TP}{TP + FN}$$

2.2.4 Confusion Matrix

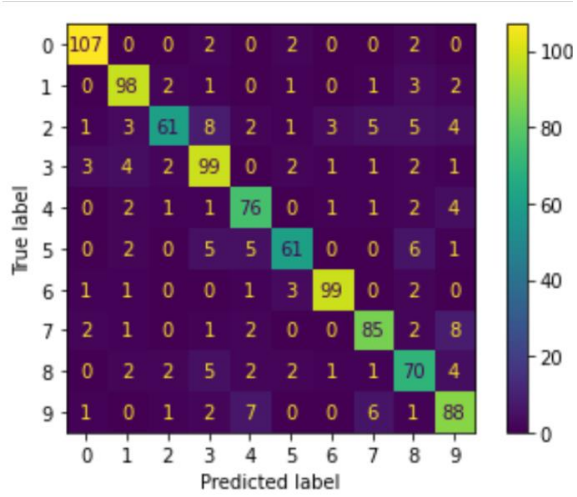
Kraftfullt sätt att visualisera prestandan på en modell. Prediktioner som ligger i det diagonala är de korrekta. (Se Figur 3.)



Figur 1. Diagram som visar fördelningen mellan de olika klasserna (github, mnist_model_eval notebook)



Figur 2. Confusion Matrix för ett binärt klassifikations problem (Prgomet, 02_klassificering.pptx)

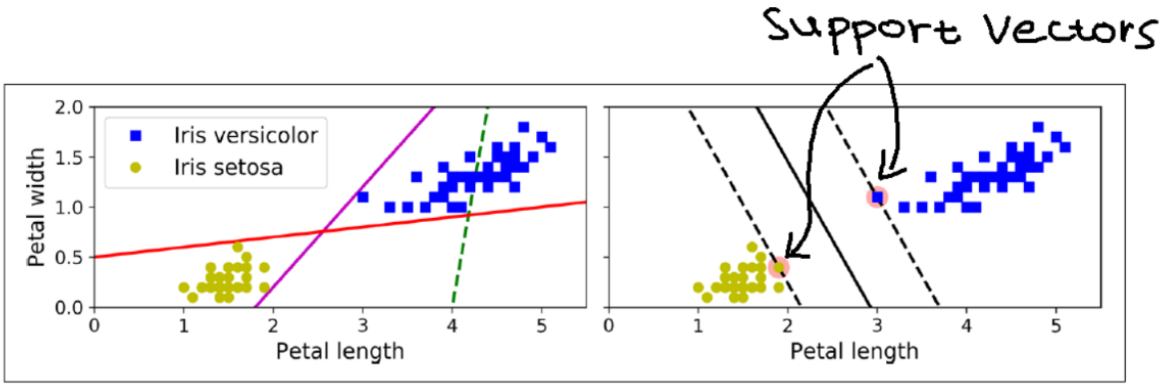


Figur 3. Exempel på en Confusion Matrix för ett multiklass problem. (Prgomet, 02_klassificering.pptx)

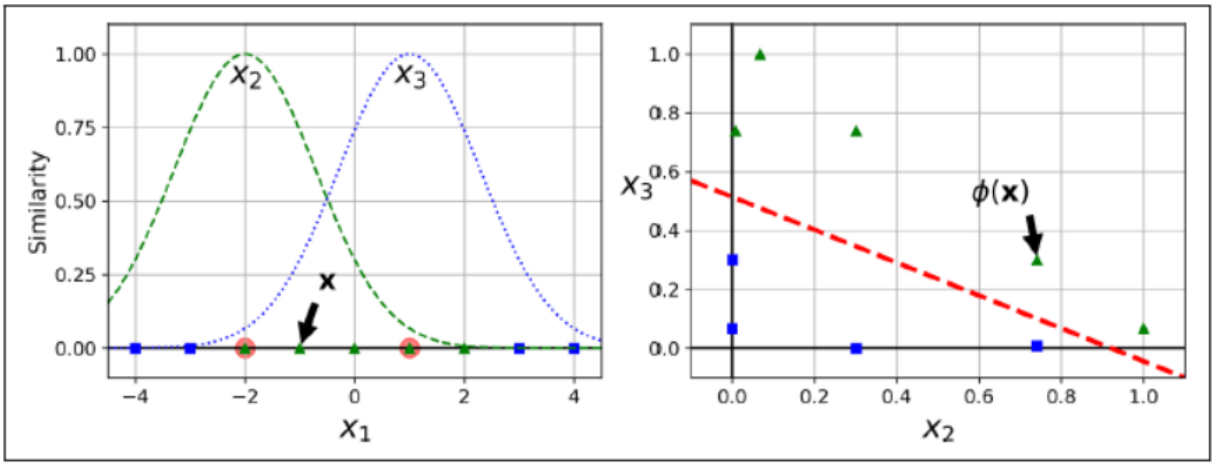
2.3 SVC (Support Vector Machine)

Är en kraftfull supervised learning model för klassifikations problem. Den strävar för att hitta den optimala "vägen" som separerar klasserna så mycket som möjligt. Punkterna som ligger närmast vägen kallas för support vectors och avgör vart gränsen går. (Se Figur 4)

För icke-linjära problem använder modellen "kärntricket," som transformerar data till en högre dimension där de olika klasserna kan separeras linjärt. (Se Figur 5)



Figur 4. (Prgomet, 07_svm.pptx)



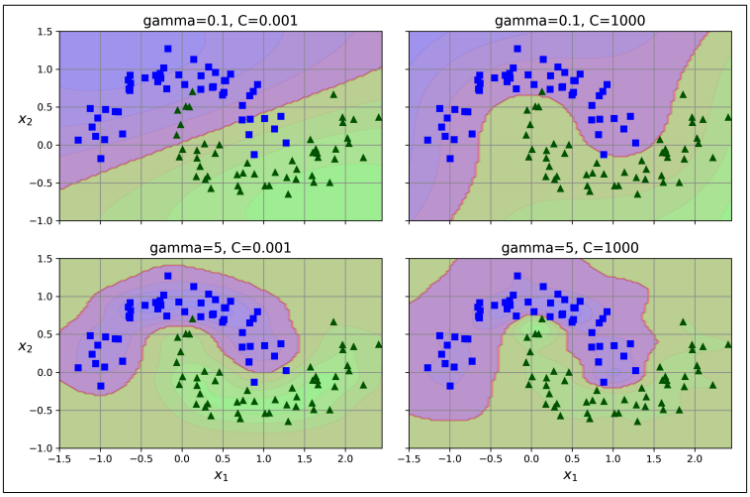
Figur 5. (Prgomet – 07_svm.pptx)

2.3.1 Hyperparameter C

För att skapa robusta modeller är regularisering avgörande. Genom att justera värdet på C kan vi förbättra modellens generaliseringsförmåga. Högt C tvingar modellen att klassificera datan så korrekt som möjligt vilket riskerar att vi skapar en modell som är överanpassad på träningsdatan. Lågt C tillåter fler fel på träningsdatan men oftast ger oss en bättre modell med bättre generalisering förmåga. (Se Figur 6)

2.3.2 Hyperparameter gamma

Med ett högt gamma-värde påverkar varje datapunkt endast ett litet område runt sig själv, vilket ökar risken för att modellen blir överanpassad till träningsdata. Å andra sidan, med ett lågt gamma-värde har varje datapunkt ett bredare inflytande, vilket resulterar i en enklare och mer generaliserad beslutsgräns. (Se Figur 6)



Figur 6. (Géron, Sidan 210)

2.4 Extra Trees Classifier (Beslutsträd)

Extra Trees Classifier är en modell som använder en trådliknande struktur för att fatta beslut. Varje nod i trädet representerar ett test (som ett if-statement) för en feature i datan. Varje gren representerar antingen sant (true) eller falskt (false), och varje lövnod representerar ett beslut (utdata). Extra Trees Classifier är en så kallad white-box modell där modellens beslut kan visualiseras på ett enkelt sätt. (Se Figur 7)

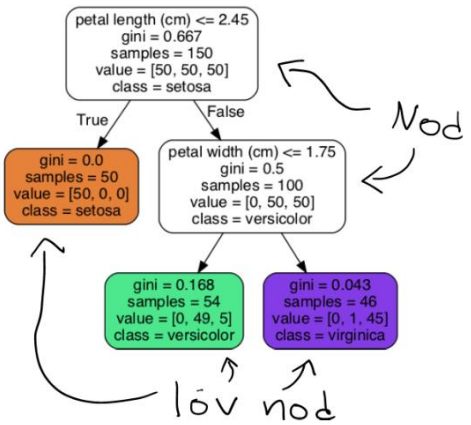
2.4.1 Regularisering i Extra Trees Classifier

Genom att finjustera olika hyperparametrar kan vi förhindra att modellen överanpassar sig till träningsdatan.

Parametern max_depth begränsar trädets djup. Genom att sätta ett maxdjup minskar vi risken för överanpassning till träningsdatan.

En annan viktig hyperparameter är min_samples_split, som anger den minsta antalet exempel som krävs för att en nod ska delas.

Det finns flera andra hyperparametrar att överväga, men jag nämner dessa två som exempel.



Figur 7. (Prgomet, 08_beslutsträd.pptx)

2.5 MLP Classifier (Neural Nätverk)

MLP Classifier modellen är en implementering av multi-layer perceptron. Det är ett neural nätverk som består av flera lager av neuroner där varje lager bearbetar datan och skickar vidare resultatet nästa lager. Det sista lagret ger utdata i form av klassificering.

2.5.1 Regularisering i MLP Classifier

Hyperparametern hidden_layer_sizes specificerar antalet dolda neuroner i varje lager.

Activation sätter aktiveringsfunktionen, som bestämmer om neuroner ska vidarebefordra data till nästa lager eller inte.

Det finns flera andra hyperparametrar att överväga, men jag nämner dessa två som exempel.

3 Metod

3.1 Data

Jag använde Scikit-learns fetch_openml-funktion för att hämta MNIST-datasättet, som innehåller 70 000 handskrivna siffror från 0 till 9. Datasettet delades upp i tre delar: träningsdata, valideringsdata och testdata. Detta gjordes för att säkerställa att modellerna kunde tränas och utvärderas på separata datamängder, vilket hjälper till att förhindra överanpassning och ger en mer realistisk uppfattning om modellernas prestanda.

3.2 Modelval

Extra Trees Classifier: En ensemble-metod som använder flera beslutsträd för att förbättra prestanda och robusthet.

Pipeline med PCA och SVC: En pipeline som först använder PCA (Principal Component Analysis) för att reducera dimensionaliteten i datan, följt av en SVC (Support Vector Classifier) för klassificering.

MLP Classifier: En multi-layer perceptron-klassificerare, som är en neuralt nätverk, anpassat för klassificeringsuppgifter.

3.3 Validering och Test

Varje modell tränades på träningsdatan och validerades på valideringsdatan. Detta hjälper till att utvärdera modellernas prestanda och avgöra vilken modell som fungerar bäst på okända data.

Bästa modellen tränades om på den kombinerade tränings- och valideringsdatan och testades sedan på testdatan för att få en slutlig uppskattning av dess prestanda.

3.4 Implementera bildbehandling

Tog foton av 10 handskrivna siffror (0-9) och utvecklade en funktion med OpenCV-paketet för att förbereda dessa bilder. Bilderna processades för att matcha formatet på MNIST-datasättet och användes sedan för att göra prediktioner med den tränade modellen.(Se Figur 8)

3.5 Driftsätta med Streamlit

Påbörjade utvecklingen av en Streamlit-applikation för att driftsätta sparade modellen i molnet.

4 Resultat

Under valideringsfasen presterade modellen PCA- SVC bäst, med en accuracy på hela 98.29%. (Se tabell 1)

Accuracy för olika modeller på validerings datan	
Extra Trees Classifier	97.25%
PCA – SVC	98.29%
MLP Classifier	97.67 %

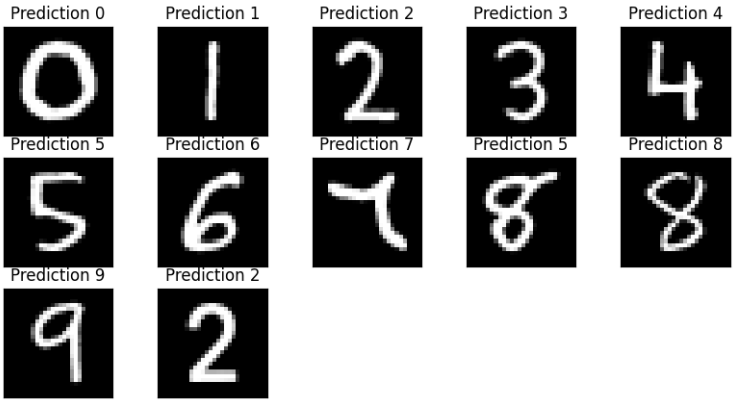
Tabell 1. Accuracy score för de 3 valda modellerna. (github, mnist_model_eval notebook)

När modellen testades på testdatan sjönk accuracy något jämfört med valideringsdatan. Detta är dock önskvärt, eftersom det visar att modellen generaliserar bra till ny osedd data. (Se tabell 2)

Modell PCA – SVC accuracy score på test datan	
PCA – SVC	98.04%

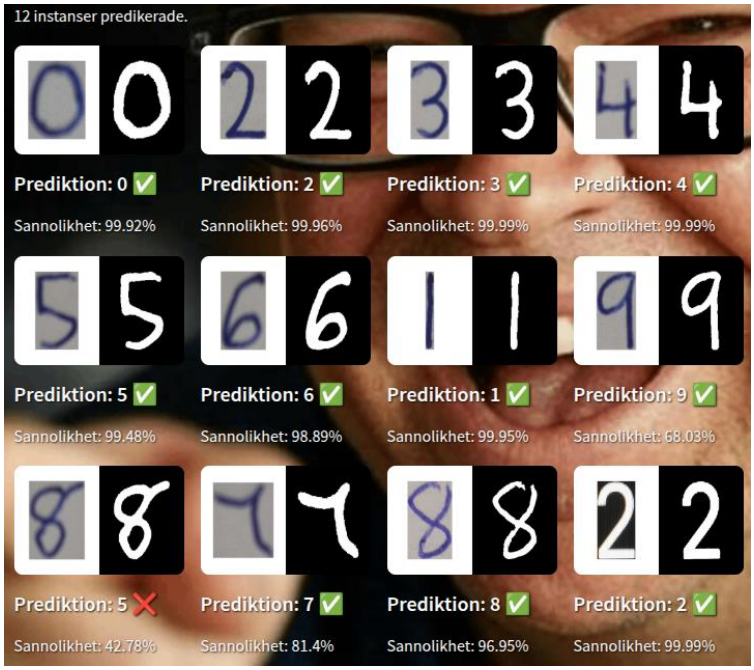
Tabell 2. Accuracy score på testdatan med PCA-SVC. (github, mnist_model_eval notebook)

Modellen uppnådde en accuracy på 92% när den testades på mina handskrivna siffror. Detta visar att min funktion funkar och att modellen generaliserar bra på osedd data. (Se figur 8)



Figur 8. PCA-SVC utvärderad på mina handskrivna bilder. (github, mnist_model_eval notebook)

Bilduppladdningsfunktionen i Streamlit-applikationen testades, och modellen uppnådde återigen en accuracy på 92%. Här kan användaren ladda upp egna bilder och få dem klassificerad. (Se figur 9)

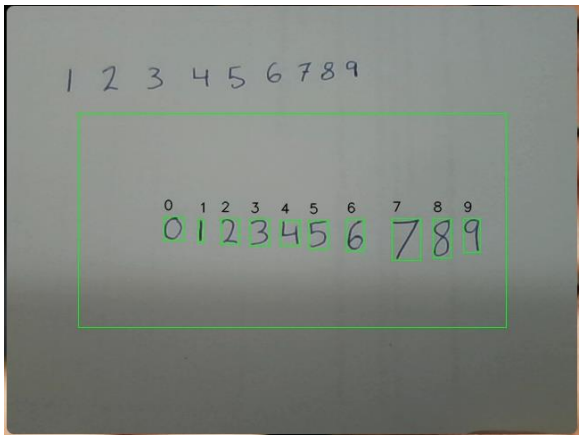


Figur 9. Mina handskrivna siffror klassificerades återigen med 92% accuracy. (github, pictures.py)

Ritfunktionen validerades. Här kan användaren på ett mer direkt sätt testa modellens prestanda. (Se figur 10)

Funktionen att prediktera i realtid via webkameran testades och validerades. (Se Figur 11)

Figur 10. Användaren ritade en tvåa som modellen predikterade korrekt. (github, draw.py)



Figur 11. Användaren riktade en A4-papper med siffror från 0 till 9 mot webkameran och siffrorna predikterades korrekt i realtid. (github, webcam.py)

5 Slutsatser

Att uppnå en accuracy över 80% på osedda data med modeller tränade på MNIST-datasettet krävde flera pre-processing steg med hjälp av OpenCV-biblioteket.

Processen involverade steg som:

1. Omvandla bilden från färg till gråskala.
2. Binarisera bilden så att bakgrunden blev svart och siffran vit med threshold-funktionen. Detta hjälper till att tydligt skilja siffrorna från bakgrunden.
3. Identifiera siffrans konturer i bilden med hjälp av findCountours-funktionen, vilket möjliggjorde beskärning av bilden så att endast siffran behölls.
4. En ny ram som matchar MNIST-datasettet skapades med copyMakeBorder-funktionen.
5. Slutligen skalades bilden ner till 28x28 pixlar med resize-funktionen för att passa den storlek som modellen är tränad på.

Dessa steg var avgörande för att modellen skulle kunna generalisera effektivt och belyser hur viktigt databehandling är. Oavsett vilken modell som används har databehandlingen en direkt och avgörande effekt på modellens prestanda.

En annan utmaning var att lära sig Streamlit. Det krävde en förståelse för hur hela flödet fungerade, särskilt hur Streamlit-appen körs från topp till botten vid varje användarinteraktion. Tack vare tydlig dokumentation och flera inspirerande exempel kunde jag dock snabbt komma igång och driftsätta min app i molnet.

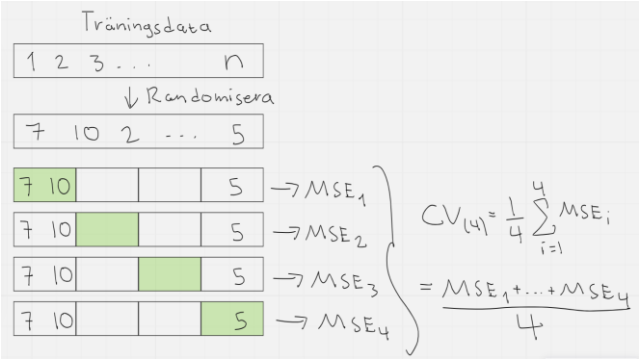
6 Teoretiska frågor

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

   Träning: Används för att träna modellen.
   Validering: Efter träning, validerar vi modellens prestandan med osedd data och justerar olika hyperparametrar om så behövs.
   Test: Används för att mäta den slutliga prestandan hos den tränade modellen.

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 ”valideringsdataset”?

   Man kan använda cross-validation metoden för att dela upp träningsdatan i flera “folds”. Detta innebär att modellen tränas och valideras upprepade gånger, varje gång med olika “folds” av datan. Speciellt användbart när träningsdatan är liten. (Se Figur 12)



Figur 12. (Prgomet, 01_introduktion_till_maskininlärning.pptx)

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

   Regressionsproblem är en typ av supervised-learning problem där målet är att prediktera ett kontinuerligt värde. Olika modeller, såsom linjär regression, beslutsträd eller neurala nätverk, kan användas beroende på problemet man försöker lösa. Som ett exempel om det finns en linjär relation mellan husets olika features kan man använda linjär regression. Om förhållandet är komplext/icke-linjärt kanske beslutsträd funkar bättre.

4. Hur kan du tolka RMSE och vad används det till?

   RMSE (root mean squared error) mäter det genomsnittliga felet mellan de predikterade och sanna värdena. Genom att kvadrera felen straffas större avvikelser hårdare och genom att ta roten ur det genomsnittliga kvadratiska felet får vi ett mått i samma enhet som den beroende variabeln.

Måttet används i regressionsproblem.

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

   Här är målet att förutsäga ett kategoriskt värde. Ett exempel är MNIST-datasättet, där vi baserat på observationernas features försöker prediktera vilken siffra(klass) som visas. Modeller som SVC och beslutsträdsklassificerare kan användas beroende på vilket problem som ska lösas. Som exempel kan vi använda dessa modeller för att prediktera om en kund kommer att avsluta sitt abonnemang (churn) eller om mailet är spam eller inte.

6. Vad är en ”Confusion Matrix”?

   Confusion matrix är ett kraftfullt sätt att visualisera prestandan på en modell. Prediktioner som hamnar längs diagonalen representerar de korrekta klassifikationerna i en confusion matrix.

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

   En unsupervised-learning model för dataset utan labels. K-means modellen används för att dela in observationerna i k-antal icke överlappande kluster. Som exempel används K-means till kundsegmentering eller bildsegmentering.

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

   Ordinal encoding:
   Används för att omvandla kategoriska värden med en inbördes ordning till heltal. Till exempel stor, medium, liten kan mappas till 2, 1, 0.

   One-hot-encoding:
   Används för att omvandla kategoriska värden som inte har en inbördes ordning till kolumner med binära värden. Till exempel värdena röd, grön, blå kan mappas till kolumner med värden [1,0,0], [0,1,0], [0,0,1].

   Dummy variable encoding:
   En variant av one-hot-encoding där en av kolumnerna tas bort då det kan skapa problem för modeller med en intercept-term. Till exempel här tar vi bort röd och mappar värdena till kolumner med värden [0,0], [1,0], [0,1].

9. 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?

   Om man ser färgerna som kategorier utan given ordning är datan nominal. Men om man som Julia, tolkar att röd är vackrast, blir datan ordinal och kan rangordnas.

Kort sagt beror det på hur man tolkar datan. Båda har rätt.

10. Vad är Streamlit för något och vad kan det användas till?

    Streamlit är ett ramverk för att skapa data applikationer i python för maskininlärning och data science.

7 Självutvärdering

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

   Se slutsatser.

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

   Jag anser att jag förtjänar ett VG, eftersom jag har uppfyllt alla kriterier för godkänt och dessutom driftsatt min applikation i molnet.

3. Något du vill lyfta fram till Antonio?

   Det var ett väldigt roligt projekt där vi tränade och testade olika modeller. Sedan driftsatte vi det i molnet med Streamlit. Rapportskrivandet var kanske inte lika roligt, men jag lärde mer när jag dokumenterade processen.

Appendix A

Projektets repository (2025). github.com/dodo-ds/ml-app

Källförteckning

Clustering. Scikit-learn. (2025). https://scikit-learn.org/stable/modules/clustering.html#k-means

Decision Trees. Scikit-learn. (2025). https://scikit-learn.org/stable/modules/tree.html

MLPClassifier. Scikit-learn. (2025). https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

Brownlee, J. One-Hot Encoding for Categorical Data. Machine Learning Mastery. (2020). https://machinelearningmastery.com/one-hot-encoding-for-categorical-data/

Géron, A. (2022). Hands-On Machine Learning Third Edition. O'Reilly Media.

Prgomet, A. (2025). Support Vector Machines (SVM), 07_svm.pptx.

Prgomet, A. (2025). Beslutsträd, 08_beslutsträd.pptx.

Prgomet, A. (2025). Klassificering, 02_klassificering.pptx.

Prgomet, A. (2025). Introduktion till Maskininlärning, 01_introduktion_till_maskininlärning.pptx.