Kunskapskontroll – Frågor & resonemang

Det här dokumentet innehåller endast fakta- och resonemangsfrågor. Koduppgifter finns i `kunskapskontroll_kod.ipynb`. Streamlit-applikationen finns i `chapter_8_assignment_9`.

Kapitel 1 – Introduktion till maskininlärning

Faktafrågor

1. Hur hänger AI, ML och DL ihop?

AI är det bredaste begreppet: system som uppvisar "intelligent" beteende. ML är en delmängd av AI där man tränar modeller från data istället för att hårdkoda regler. DL är en delmängd av ML där modellen är ett neuralt nätverk med många lager.

2. Vilka är de fyra problemkategorierna inom ML?

En vanlig indelning är:

1. **Supervised learning**: vi har input X och facit/target y (t.ex. regression, klassificering).
2. **Unsupervised learning**: vi har bara X och söker struktur (t.ex. klustring, dimensionsreducering).
3. **Semi-supervised learning**: lite data har labels, mycket saknar labels.
4. **Reinforcement learning**: en agent lär sig via belöning/straff genom interaktion med en miljö.

3. Förklara följande

a) Syftet med att dela upp data i träningsdata, valideringsdata och testdata

- **Träning :** används för att anpassa modellens parametrar.
- **Validering :** används för att jämföra modeller och välja hyperparametrar utan att "titta" på testet.
- **Test :** används sist för en så opartisk uppskattning som möjligt av generalisering.

b) Vad är k-delad korsvalidering? K-fold CV delar datan i k delar och tränar k gånger där varje del används en gång som validering och resten som träning. Resultaten kan medelvärdesbildas för en stabilare skattning.

c) Vad är RMSE? RMSE (root mean squared error) är roten ur medelvärdet av kvadrerade fel för regression. Den mäts i samma enhet som y.

d) Hyperparameter vs parameter

- **Parameter:** lärs från datan under träning (t.ex. vikter).
- **Hyperparameter:** sätts innan/utanför träningen och styr modellen/träningen (t.ex. `max_depth`, `alpha`).

e) Vad är grid search? Varför "grid" och "search"? Vad betyder `refit=True`?
Grid search testar systematiskt kombinationer i ett hyperparameter-"rutnät" och "söker" den kombination som ger bäst score (ofta via CV). `refit=True` betyder att modellen tränas om på hela träningsunderlaget med bästa hyperparametrarna efter sökningen.

f) Kategorisk data och hantering (nominal, ordinal, one-hot, dummy, ordinal encoding)

- **Nominal data:** utan ordning → one-hot encoding eller dummy-variable encoding.
- **Ordinal data:** med ordning → ordinal encoding med heltal som bevarar ordningen.

g) Vad är feature engineering? Skapa/transformera/välja features för att hjälpa modellen (t.ex. skalning, log-transform, datumfeatures, aggregeringar).

h) Principle of parsimony Om två modeller presterar liknande: välj den enklare (Ockhams razor).

4. Vad menas med att "en modell är en förenkling av verkligheten"?

En modell är en approximation med antaganden och begränsningar. Verkligheten har fler faktorer, brus och förändring över tid, så modellen fångar bara delar av mönstret.

5. Vad menas med att en modell är överanpassad (overfitted)?

Modellen har lärt sig detaljer/brus i träningsdatan som inte generaliserar: bra träning, sämre validering/test.

6. Högre är bättre i scikit-learn scoring, vad innebär det?

scikit-learn vill maximera score. Därför används ibland negativa varianter av felmått (t.ex. `neg_mean_squared_error`) så att "högre" fortfarande betyder "bättre".

7. Tvärsnittsdata, tidsseriedata och paneldata

- **Tvärsnittsdata:** många enheter vid en tidpunkt (t.ex. löner för 500 personer januari 2026).
- **Tidsseriedata:** en enhet över tid (t.ex. daglig temperatur 10 år).
- **Paneldata:** många enheter över tid (t.ex. månadsvis försäljning per butik i 5 år).

Resonemangfrågor

8. Ge några exempel på verkliga tillämpningsområden inom ML

Här är några vanliga och konkreta tillämpningar (med typ av ML-problem i parentes):

- `Rekommendationssystem` (klassificering/ranking): t.ex. film-/musikrekommendationer eller "kunder som köpte X köpte också Y".
- `Bedrägeri- och anomalidetektion` (klassificering/unsupervised): upptäcka avvikande transaktioner i bankdata.
- `Efterfrågeprognoser` (tidsserie/regression): förutsäga försäljning per vecka för lagerplanering och inköp.

Gemensamt: man har data (historik) och vill göra prediktioner/beslut som skalar bättre än manuell hantering.

9. Förklara logiken bakom scoring='neg_mean_squared_error'

I scikit-learn är principen att **högre score ska betyda bättre modell** (man maximerar score). Men felmått som MSE/RMSE är tvärtom: **lägre är bättre**.

För att ändå kunna använda en maximeringslogik returnerar scikit-learn därför den **negativa** varianten:

- MSE är alltid ≥ 0 och vi vill minimera den.
- `neg_mean_squared_error` returnerar $-\mathrm{MSE}$, som alltid är ≤ 0.
- Om modell A har MSE 10 och modell B har MSE 20 så får vi:
  - A: -10
  - B: -20 Då är -10 **större** än -20, vilket gör att "bäst modell" fortfarande fångas av "störst score".

Kapitel 2 – Ett ML-projekt från början till slut

Faktafrågor

1. Checklistan med sju steg

1. Formulera problemet (mål, ML-typ, metrik, constraints).
2. Samla in data (källor, etik, åtkomst).
3. EDA och datakvalitet.
4. Förbehandling + features (encoding, skalning, pipeline).
5. Välj och träna modeller (baseline → bättre).
6. Tuning + utvärdering (validering/CV).
7. Sluttest + dokumentera + produktionssätt (deploy, monitorering).

I praktiken är detta oftast **iterativt** (man går fram och tillbaka).

2. Vad menas med att en modell produktionssätts?

Att deploya modellen i ett riktigt system (API/app/batch), inklusive versionshantering, reproducerbar pipeline, monitorering och rutiner för uppdatering/omträning.

3. Vad är scikit-learn? Designprinciper? Estimators/predictors/transformers

scikit-learn är ett ML-bibliotek i Python. Centralt: enhetligt API (`fit` , `predict` , `transform`), pipelines/komposition och tydliga objekt.

- **Estimator:** kan `fit` .
- **Predictor:** estimator som kan `predict` efter `fit` .
- **Transformer:** estimator som kan `transform` efter `fit` .

4. Vad är TensorFlow och Keras?

TensorFlow är ett DL-ramverk. Keras är ett högre nivå-API för att bygga och träna neurala nätverk.

Resonemangfrågor

5. Kalle vs Stina om att justera tills test blir bra

Stina har rätt i sak. Om Kalle gång på gång tittar på testresultatet och justerar modellen för att få upp testprestandan, så börjar testdatan fungera som en **valideringsmängd**. Då "läcker" information från testet in i modellvalet och testet slutar vara ett opartiskt mått på hur modellen presterar på helt ny data.

En bra process är istället:

- Gör modellval + hyperparametertuning på **train/validation** (eller k-fold CV).
- Använd **testsetet en gång på slutet**.
- Om du behöver iterera efter test: skapa ett nytt, orört testset (eller använd nested CV i mer avancerade upplägg).

Det betyder inte att man aldrig får förbättra en modell efter test, men då måste man vara tydlig med att man i praktiken har "förbrukat" testet som oberoende utvärdering.

6. Varför når många AI/ML-projekt inte målen? Hur ska vi förhålla oss?

Vanliga orsaker är ofta mer "produkt/organisation/data" än algoritm:

- **Otydligt problem och fel metrik:** man vet inte exakt vad som ska optimeras eller vad som räknas som "bra nog".
- **Dataproblem:** saknade värden, bias, fel etiketter, för lite data, eller data som inte representerar produktionen.
- **Förväntningsgap:** man hoppas på "AI-magi" där ett enklare BI/regel-system hade räckt (eller där data inte stödjer uppgiften).
- **Svårt att produktionssätta:** integration i system, drift, monitorering, prestanda/latens, säkerhet och underhåll (MLOps).
- **Datadrift och ändrade processer:** verkligheten förändras → modellen tappar kvalitet över tid.
- **Juridik/etik/åtkomst:** data får inte användas som tänkt, eller kräver tillstånd/anonymisering.

Hur vi bör förhålla oss:

- Börja med en **baseline** och en tydlig "definition of done" (mätbar metrik, acceptanskriterier).
- Gör tidigt en **data-audit** (kvalitet, representativitet, leakage-risker).
- Arbeta **iterativt**: PoC → MVP → produktion, och bygg in monitorering/uppföljning.
- Var realistisk: vissa problem är olämpliga för ML, eller kräver mer data/ändrad datainsamling först.

Kapitel 3 – Regression

Faktafrågor

1. Vad kännetecknar regressionsproblem? Ge några exempel.

Regression innebär att vi predikterar en **kontinuerlig numerisk variabel** y (t.ex. pris, temperatur, tid, vikt). Exempel: huspriser, försäljning, energiförbrukning, lön.

2. Förklara RMSE, MSE och MAE.

- **MAE** (mean absolute error): medelvärdet av absoluta fel $|y - \hat{y}|$. Robustare mot outliers än MSE/RMSE.
- **MSE** (mean squared error): medelvärdet av kvadrerade fel $(y - \hat{y})^2$. Straffar stora fel hårt.
- **RMSE**: $\sqrt{\text{MSE}}$ (samma enhet som y), lättare att tolka än MSE i många fall.

3. Spelar det någon roll om RMSE eller MSE används för att rangordna modeller?

Om man utvärderar på **samma dataset** och jämför två modeller så ger MSE och RMSE i praktiken **samma rangordning**, eftersom RMSE är en strikt växande transformation av MSE: $\text{RMSE} = \sqrt{\text{MSE}}$. Däremot ändras **skalan** (och hur skillnader "känns") vilket kan påverka tolkning/rapportering.

4. Förklara mycket översiktligt vad gradient descent är.

Gradient descent är en optimeringsmetod för att minimera en förlustfunktion. Man uppdaterar parametrar i riktning mot negativ gradient:

$$\theta \leftarrow \theta - \eta \nabla_\theta L(\theta)$$

där η är lärhastigheten (learning rate).

5. Vad är bias–variance trade-off? Varför är mer komplexa modeller inte alltid bättre?

- **Bias**: systematiskt fel p.g.a. för enkel modell (underfitting).

- **Variance**: modellen är för känslig för träningsdata (overfitting). När komplexiteten ökar brukar bias minska men variance öka. För hög komplexitet kan ge sämre generalisering trots perfekt träning.

6. Förklara översiktligt hur respektive modell fungerar.

a) Linjär regression: predikterar $\hat{y} = w^T x + b$ genom att minimera MSE. Antas linjära samband i features. **b) Ridge regression:** linjär regression + L2-regularisering (straff på stora vikter). Minskar överanpassning, bra vid multikollinearitet. **c) Lasso regression:** linjär regression + L1-regularisering. Kan ge vikter exakt 0 → fungerar som variabelselektion. **d) Elastic Net:** kombinerar L1 och L2 (bra kompromiss när features är korrelerade). **e) SVR (Support Vector Regression):** försöker hitta en funktion som ligger inom en "epsilon-tub" och straffar avvikelser; kan använda kernel för icke-linjäritet. **f) Beslutsträd (DecisionTreeRegressor):** delar upp feature-rymden med regler (splits) för att minimera t.ex. MSE i varje blad. **g) Ensemble learning:** kombinerar flera modeller. VotingRegressor medelvärdes-/viktar prediktioner. BaggingRegressor tränar många modeller på bootstrap-samples och aggregerar. **h) Random forest:** många beslutsträd + bagging + slumpmässigt urval av features per split. Ofta stark baseline med bra generalisering.

7. White box vs black box modeller

- **White box:** lätt att tolka (t.ex. linjär regression, små beslutsträd).
- **Black box:** svårare att tolka (t.ex. stora ensemble-modeller, neurala nät).

8. Skillnaden mellan bagging och pasting

- **Bagging:** tränar modeller på **bootstrap-samples**.
- **Pasting:** tränar modeller på **subsamples utan återläggning**. Båda syftar till att minska variance genom att aggregera många modeller.

Resonemangfrågor

9. Tolkning av figur 3.1 (generellt)

Figuren visar **enkel linjär regression**. Man har en massa datapunkter (svarta punkter) och försöker dra en **rak linje** som beskriver sambandet mellan x (här: ålder) och y (här: inkomst).

- Den **blå linjen** är modellens gissning/prediktion för varje x.
- **Interceptet** är där linjen skär y-axeln (vad modellen gissar när $x = 0$).
- **Lutningen** är hur mycket y förväntas ändras när x ökar med 1.
- De **röda strecken** är felen (residualer): avståndet mellan en punkt och linjen i y-led. När man tränar modellen försöker man hitta den linje som gör felen så små som möjligt (oftast genom att minimera MSE).

10. Tolkning av figur 3.13 och koppling till figur 3.14 (generellt)

Figur 3.13 visar ett **beslutsträd för regression**. Trädet ställer frågor om variablerna (t.ex. om $X2$ är mindre än ett visst värde) och delar upp datan steg för steg.

- Om villkoret stämmer går man åt ena hållet (True), annars åt andra (False).
- `samples` visar hur många datapunkter som hamnar i noden.
- `squared_error` är ett mått på hur "spridda" y-värdena är i noden (lägre = mer lika).
- `value` är modellens gissning i noden (ofta medelvärdet av y där). I en slutnod (leaf) gör trädet ingen mer uppdelning → alla punkter i den "rutan" får samma prediktion.

Figur 3.14 visar samma sak fast som en bild i planet $(X1, X2)$. Varje gång trädet splittrar så blir det en rak gräns (horisontell om man splittrar på $X2$, vertikal om man splittrar på $X1$). Färgerna visar vilket y trädet gissar i varje område, och därför blir det lite "blockigt" (bitvis konstant).

11. Vad är R^2 (determinationskoefficienten)?

R^2 hur mycket bättre modellen är än att bara gissa medelvärdet:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

- $R^2 = 1$ är perfekt.
- $R^2 = 0$ betyder "lika bra som att alltid gissa medelvärdet".
- Kan bli negativt om modellen är sämre än medelvärdesprediktion (på ett givet testset).

Här ska jag erkänna att jag använder AI för hjälp med LaTeX-formatet på ekvationerna. Innehållet och resonemanget är dock mitt eget. Hoppas det är okej!

Kapitel 4 – Klassificering

Faktafrågor

1. Vad kännetecknar klassificeringsproblem? Ge exempel.

Klassificering innebär att vi predikterar en **diskret klass** (t.ex. 0/1 eller flera klasser). Exempel: spam/inte spam, sjuk/frisk, bedrägeri/inte bedrägeri, handskriven siffra 0–9, churn ja/nej.

2. Hur fungerar OvR och OvO?

- **OvR (One-vs-Rest):** för K klasser tränas K binära modeller; klass k mot "alla andra". Prediktion tas som högst score/sannolikhet.
- **OvO (One-vs-One):** tränar en modell för varje par av klasser. Vid prediktion röstar modellerna fram vinnande klass.

3. Förklara utvärderingsmått

a) Confusion matrix: tabell med verkliga vs predikterade klasser. **b) Accuracy:** andel korrekta prediktioner. **c) Precision:** andel av positiva prediktioner som är rätt. **d) Recall:** andel av verkliga positiva som hittas. **e) F1-score:** harmoniskt medel av precision och recall. **f) ROC-kurva:** visar trade-off mellan TPR (=recall) och FPR när man varierar tröskel. AUC summerar kurvan.

4. Vad är precision–recall tradeoff?

När man ändrar klassificeringströskeln brukar precision och recall påverkas åt olika håll: lägre tröskel → fler positiva → högre recall men ofta lägre precision. Högre tröskel → färre positiva → ofta högre precision men lägre recall.

5. Förklara modellerna översiktligt

a) Logistisk regression: linjär modell som modellerar sannolikhet via sigmoid/softmax; optimeras med log-loss. **b) SVC (SVM):** hittar max-marginal-separerande gräns; kernel kan ge icke-linjära beslutgränser. **c) Beslutsträd:** regelbaserade splits som maximerar t.ex. information gain/minimerar impurity. **d) Ensemble:** VotingClassifier kombinerar flera modeller (hard/soft voting). BaggingClassifier tränar många modeller på bootstrap-samples och aggregerar. **e) Random forest:** många träd + bagging + feature-subset per split → stabilt och kraftfullt. **f) ExtraTrees:** liknar random forest men mer randomiserade splits; kan minska variance ytterligare.

6. Feature importance i trädmodeller

Träd/skogar kan ge uppskattning av feature importance, ofta baserat på hur mycket en feature minskar impurity över alla splits (eller via permutation importance). Det ger en indikation på vilka variabler som påverkar modellen mest.

Resonemangfrågor

7. Precision vs recall (Stina och Kalle)

Om Stina vill maximera precision så blir det ofta på bekostnad av recall: modellen säger "positiv" mer sällan, men när den väl gör det är den oftare rätt.

- Bra när falska positiva är dyra. I rättsväsendet vill man typiskt undvika att oskyldiga döms → man vill ha **extremt låg FP-rate** (hög precision), men man behöver också processer utanför modellen (mänsklig granskning, beviskrav). Det centrala är att välja trade-off utifrån konsekvenser och gärna använda olika trösklar/arbetsflöden (t.ex. "flagga för granskning" istället för automatisk dom).

8. Tolkning av figur 4.8 (generellt)

Figuren visar hur **logistisk regression** delar upp ett 2D-område i två klasser.

- Punkterna är datapunkter från de två klasserna.
- Bakgrundsfärgen visar hur säker modellen är: mörkare/lila betyder låg chans för klass 1 och gulare betyder hög chans för klass 1.
- Där färgen skiftar som mest (ungefär mitt emellan) ligger **beslutsgränsen**. Med en vanlig tröskel på 0.5 hamnar man på varsin sida av den. Man ser också att modellen är mest osäker nära gränsen, och mer säker längre bort.

9. Varför `.fit_transform()` på träning men bara `.transform()` på val/test?

För att undvika **data leakage**. På träning får transformern (t.ex. scaler/encoder) lära sina parametrar (medelvärde, std, kategorier, etc.) via `fit`, och sedan appliceras transformationen. På val/test ska man **inte** lära om transformationen (då skulle info från val/test påverka modellen). Därför kör man bara `transform()` med samma parametrar som lärdes på träningsdata.

Kapitel 7 – Artificiella neurala nätverk (ANN)

Faktafrågor

1. Vad har ANN-modellerna inspirerats av?

De är inspirerade av **biologiska nervsystem**, framförallt neuroner och hur signaler kombineras och skickas vidare.

2. Vad refererar "djup" till i "djupinlärning"?

Antalet **lager** i ett neuralt nätverk, särskilt antalet "dolda" lager mellan input och output. Fler lager → djupare modell.

3. Förklara vad som händer i figur 7.2–7.5. Varför aktiveringsfunktioner?

Figur 7.2 och 7.3 visar att neurala nätverk utan aktiveringsfunktioner bara är linjära modeller, figur 7.4 introducerar icke-linjära aktiveringsfunktioner, och figur 7.5 visar hur dessa används i en neuron för att göra nätverket uttrycksfullt och kapabelt att modellera komplexa samband.

Aktiveringsfunktioner (t.ex ReLu och sigmoid) används för att annars är hela nätverket bara en linjär modell, oavsett antal lager.

4. Har neurala nätverk få eller många parametrar?

Ofta **många** parametrar (vikter och bias i varje lager). Antalet beror på arkitektur (antal lager, antal neuroner per lager).

5. Förklara intuitivt hur dropout-regularisering fungerar.

Dropout innebär att man under träning slumpmässigt "stänger av" (sätter till 0) en andel av neuroner/aktiveringar i ett lager per batch. Det tvingar nätet att inte förlita sig på enstaka vägar/neuron (minskar co-adaptation) och fungerar som regularisering → minskar överanpassning.

Resonemangfrågor

6. Förklara tabell 7.1 och tabell 7.2

Tabell 7.1 visar hur man bygger ett MLP för att förutsäga numeriska värden, medan tabell 7.2 visar hur output-lager, aktiveringsfunktion och loss-funktion anpassas beroende på om klassificeringen är binär, multilabel eller multiklass.

7. Experimentera på TensorFlow Playground – vad kan man lära sig?

Playground visar hur val av nätarkitektur (antal lager/neuron), aktivering, regularisering och learning rate påverkar:

- hur beslutgränser blir mer/ mindre komplexa,
- hur snabbt/långsamt modellen "convergar",
- under/överanpassning (train vs test-loss).

8. Översikt: hur backpropagation fungerar (med figur 7.7)

Backpropagation är metoden för att beräkna gradienter effektivt i ett nätverk:

1. **Forward pass:** beräkna prediktion och loss.
2. **Backward pass:** använd kedjeregeln för att sprida lossens gradient bakåt genom lagren och få $\partial L / \partial w$ för alla vikter.
3. **Uppdatering:** optimizer (t.ex. Adam) uppdaterar vikter i riktning som minskar loss.

Så det består alltså av forward pass, backward pass och till sist en optimizer.

Kapitel 8 – Convolutional neural network (CNN)

Faktafrågor

1. Vad krävs för att ett nät ska klassas som CNN?

Att nätet innehåller **konvolutionslager (convolution layers)** som utnyttjar lokala receptiva fält och vikt-delning (weight sharing). Ofta kombinerat med pooling och/eller normalisering, men kärnan är konvolutionslagren.

2. Inom vilket område är CNN generellt sett kraftfullt?

Framförallt **bild- och visuella data** (computer vision): klassificering, objektigenkänning, segmentering. Även andra "grid-liknande" data.

3. Vad menas med RGB?

RGB står för **Red, Green, Blue** och är ett vanligt sätt att representera färgbilder med tre kanaler.

4. Vad är data augmentation?

Tekniker för att syntetiskt öka träningsdata genom rimliga transformationer (t.ex. rotation, beskärning, spegling, ljus/kontrast). Syftet är att förbättra generalisering och minska överanpassning.

Resonemangfrågor

5. Översiktligt: hur CNN fungerar (convolution layer, pooling layer)

- **Convolution layer:** filtrerar bilden med små kernels (t.ex. 3×3) och lär filter som upptäcker lokala mönster (kanter → former → objekt). Vikt-delning gör att samma filter används över hela bilden.
- **Pooling layer:** minskar upplösning (t.ex. max pooling) och gör representationen mer robust mot små förskjutningar; minskar även beräkningskostnad. Totalt bygger CNN en hierarki av features: tidiga lager → enkla mönster, senare lager → mer abstrakta mönster.

6. Förklara figur 8.4

Figuren visar hur ett **CNN** ofta är uppbyggt.

- Först kommer flera **convolution-lager** som lär sig hitta enkla mönster i bilden (kanter, former osv).
- Mellan dem ligger ofta **max pooling** som gör bilden "mindre" (lägre upplösning) och minskar beräkningskostnad.
- Efter några sådana steg gör man **flatten** så att allt blir en lång vektor.
- Till sist kommer ett eller flera **dense-lager** som använder de lärda bild-features för att bestämma klass.
- Output-lagret har lika många noder som klasser (i figuren 100), ofta med softmax. Kort sagt: conv + pooling plockar ut bra features, och dense-delen gör själva klassificeringen.

Kapitel 10 – Chattbottar

Faktafrågor

1. Vad är prompt engineering?

Att medvetet utforma instruktioner (prompter) till en språkmodell för att styra beteendet: t.ex. roll, formatkrav, exempel, avgränsningar och kontrollfrågor.

2. Vad är RAG?

RAG (Retrieval-Augmented Generation) kombinerar **informationshämtning** och **generering**: man hämtar relevanta dokument-/textbitar från en kunskapskälla och ger dem som kontext till modellen som sedan formulerar ett svar. Syftet är ofta bättre faktastöd och uppdaterad kunskap.

3. Vad är chunking och embeddings?

- **Chunking:** att dela upp längre text i mindre bitar (chunks) som går att söka och mata in i modellen.
- **Embeddings:** vektorrepresentationer av text som fångar semantisk likhet. Används för att hitta relevanta chunks via t.ex. cosine similarity.

Resonemangfrågor

4. Hur kan man evaluera en chattbot?

Man kan evaluera både offline och online:

- **Kvalitet i svar:** korrekthet, relevans, täckning, tydlighet, formatkrav uppfyllda.
- **RAG-specifikt:** retrieval precision/recall (hämtar den rätt källor?), citationer/grounding, hallucinationsgrad.
- **Robusthet:** hur den hanterar otydliga frågor, edge cases, motstridiga instruktioner.
- **Säkerhet/Policy:** prompt injection, dataläckage, olämpligt innehåll.
- **UX-mått:** svarstid/latens, användarnöjdhet, task completion rate, fallback-rate. Ofta gör man en testsvit med representativa frågor + mänsklig granskning och/eller automatiska checks.

5. Vad är ELIZA och Turingtestet?

ELIZA var en tidig chattbot (1960-talet) som använde regelbaserade mönster och enkla transformationer. Den "förstår" inte språk i modern mening men kan upplevas övertygande i vissa dialoger. **Turingtestet** är ett tankeexperiment: om en människa i en textbaserad konversation inte kan avgöra om motparten är människa eller maskin, så sägs maskinen uppvisa intelligens enligt testets kriterium. Testet mäter imitation av mänsklig dialog, men kritiseras bl.a. för att inte nödvändigtvis mäta verklig förståelse.

6. Exempel på nytta och risker med chattbottar

Nytta: kundtjänst, intern kunskapsassistent, hjälp i vård, utbildning/handledning, rapport-/textutkast. **Risker:** felaktiga svar (hallucinationer), bias/diskriminering, sekretess/dataläckage, säkerhetsproblem, överförtroende hos användare, otydligt ansvar vid beslut. Ett bra förhållningssätt är tydliga begränsningar,

loggning/monitorering, människa-i-loopen där det krävs, samt att RAG-svar förankras i källor.