# Självutvärdering

**1.** Har något varit utmanande i kursen/kunskapskontrollerna? Om ja, hur har du hanterat det?
**Svar**: Svårt i perioder att förstå materialet i boken. Många nya begrepp och koncept.

**2**. Vilket betyg anser du att du ska ha och varför?
**Svar**: Jag anser att jag ska få VG för att jag har försökt att förstå alla nya koncept och testat mig fram. Det har jag gjort genom bland annat ett ambitiöst testupplägg och gjort en Streamlit app.

**3.** Något du vill lyfta fram till Terese?
**Svar**: -

# Teoretiska frågor

Svara gärna kort och koncist.

## **1.** All data i Python representeras av objekt. I Python består varje objekt av tre grundläggande delar. Vilka är dessa delar? Beskriv varje del kort.

**Identitet:** Det unika "id" som varje objekt har. Samma under objektets livstid.
**Datatyp:** Avgör möjliga operationer som går att utföra på objektet. Styr vilka attribut och metoder ett objekt har.
**Värde:** Datainnehållet i objektet. Kan vara mutabla och immutabla

## **2.** Förklara skillnaden mellan mutabla och immutabla objekt i Python och ge exempel på varje.

**Mutabla**: Är objekt som kan ändras efter att de har skapats. Exempelvis list, dict och set.
**Immutabla**: Objekt som inte kan ändra sitt värde efter att de skapas. Exempel tuple och frozenset. I praktiken är även int, float, bool och str immutabla för att om jag ändrar dem skapas ett nytt objekt.

## **3.** Vad är ett set i Python? Vilka egenskaper särskiljer det från listor? Ge ett praktiskt exempel där det är mer effektivt att använda ett set än en lista.

Set är en oordnad mängd med unika objekt. Det går att lägga till och ta bort objekt men det går inte att kalla på ett objekt genom att ange ett indexnummer. Däremot går att det loopa genom ett set för att hitta om det innehåller ett visst element eller ej.
Det som skiljer set från en lista är att set inte har dubbletter, är hashbaserad och är oordnade.
Set är snabbare på att loopa igenom en lista (x in my_set) för att den använder hash istället för index. Det gör att set blir snabbare på en lång samling av registerdata. Till exempel för att kolla om en person är medlem i en fotbollsklubb eller ej.

## **4.** Förklara vad en loop är och ge exempel på när den används.

En loop är en kod som går igenom en samling. Den kan antingen gå igenom hela samlingen eller tills ett villkor uppfylls.
For loop kör igenom en samling en omgång. Går att lägga till villkor/intervaller för körningen men samlingen körs bara igenom en omgång. Kan vara bra när en lista innehåller nettopriser (utan moms) och det finns olika momssatser för olika branscher som behöver plussas på.
While loop kör tills villkoret är sant. Det innebär att den kan fortsätta köra i all oändlighet (evighetsloop). Det är bra när det inte går att säga hur många omgångar som behövs.

## **5.** Vad är en klass och hur skiljer sig en instans från själva klassen?

Klass är samma som datatyp ovan. Klassen sätter ramar ("mall") för vilka värden och tillstånd ett objekt kan ha. Klassens attribut är gemensamma för alla instanser av klassen.
En instans är ett specifikt objekt som skapas när klassen anropas. Klassen kan initiera instansens starttillstånd via __init__. Instansens attribut är specifika för varje instans.

## **6.** Vad är en funktion och varför använder programmerare dem? Förklara skillnaden mellan en funktion med returvärde och en som inte returnerar något.

Funktionen är ett kodblock som kan återanvändas. Det skapas för att utföra specifika uppgifter. Det skapas genom att skriva "def" före namnet på funktionen som skrivs med snakecase och understreck mellan orden.
Programmerare använder funktioner för att slippa skriva om kod, strukturera koden, underlätta uppdatering av koden och för att bryta ner stora problem i små delar.
För att funktionen ska skicka ett resultat till kod som anropar funktionen behöver ordet "return" användas. Då kan värdet användas eller sparas i en variabel.
Om funktionen inte returnerar något explicit så kan det antingen vara en print utskrift med standardutmatning eller returvärde "None" eftersom den alltid returnerar något.

## 7. Förklara begreppet parameter jämfört med argument i Python

Parametrar är varibler som listas i funktionens definition. De funkar som platshållare och bestämmer vilken typ av argument som funktionen kan ta emot. I def say_hello(name): är name en parameter. Argument är det värdet som skickas till funktionen när den anropas (exekeveras). I say_hello('Kalle') är Kalle argumentet.
Det finns olika typer av argument och parameterar. För argumenten finns positionsargument, nyckelordsargument, standardargument. Parametrar i sin tur har även specialparametrarna *arg och **kwarg.

## 8. Titta på följande video om R: R Programming. Vad är skillnaden mellan R och Python?

Båda är programmeringsspråk som liknar varandra i syntaxen. R används mer inom statistik medan Python är betydligt bredare språk. Python används främst till AI, maskininlärning, dataanalys. Det går även att använda till att t.ex utveckla spel. Python är ett objektorienterat språk.

R har sex grundläggande datastrukturer: vektorer, arrays, matriser, faktorer, lists och data fames. Till det tillkommer tredjeparts bibliotek som Tidyverse. Python använder främst lists, tuples, dictionaries och sets. För matrisberäkningar och tabulär data används tredjepartsbibliotek som NumPy, Scikit-learn och Pandas.
Python är långsamt för tunga beräkningar. Därför används NumPy (skrivet i C) för att beräkningarna ska gå snabbare. R har däremot mycket av sin statistik direkt i språket. Det är lättare att göra snygga grafer i python matplotlib och seaborn. R använder ggplot2 där det också går att göra relativt snygga grafer men inte riktigt så bra som Python alternativen.

## 9. Kalle ska bygga en ML-modell och delar upp sin data i "Träning", "Validering" och "Test", vad används respektive del för?

Den vanligaste uppledningen är 60-20-20 eller 70-15-15. MNIST datasetet däremot har en rekomendation på 10 000 (14 %) för test och resterade för träning/validering.
**Träningsdata** är den största delen. Det används för att modellen ska lära sig mönster. Det sker genom att anpassa parametrar.
**Valideringsdatan** används under utvecklingsfasen för att jämföra olika modeller och ställa in hyperparametrar. Det görs för att utvärdera modellen utan att behöva tjuvtitta på testdatan. Ungefär som att öva på gamla prov.
**Testdata** används som ett avslutande steg för att få en rättvis bild av hur modellen presterar på ny osedd data. Det blir den riktiga skarpa tentemen för den tränade modellen.
Du får aldrig göra en fit (modellträning) på valideringsdatan eller testdatan. Det innebär att det blir dataläkage.

## 10. 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"?

Eftersom Julia inte har skapat ett separat valideringsset bör hon använda k-delad korsvalidering (K-fold cross validation) på träningsdatat för att få den bästa modellen. Det gör hon genom att dela upp datan i folds (normalt 5 eller 10)
Modellen tränas sedan så många gånger den en del används som validering och resterande som träning i varje iteration (körning).
För varje modell hon har (linjär regression, Lasso och Random Forest) beräknar hon ett medelvärde för felet från de olika körningarna. Sedan bör hon använda ett utvärderingsmått som RMSE (Root Mean Squared Error) eller i andra hand MSE för att mäta hur mycket modellernas prediktioner avviker från de faktiska värderna.
Den modell som ger bäst genomsnittlig CV-score (lägsta genomsnittliga fel) är den hon bör fortsätta använda. För att få fram CV-score är ett bra metod GridSearchCV. GridSearch testar automatiskt olika hyperparametrar, genomför korsvalidering för varje kombination och hittar den bästa modellen baserat på resultatet.
När Julia har valt sin bästa modell tränar hon modellen på hela träningsdatan och testar det på testdatan för att få en rättvis bild på osedd data.

## 11. Vad är "regressionsproblem? Kan du ge några exempel på modeller som används och potentiella tillämpningsområden?

Ett regressionsproblem är när du vill förutsäga en numerisk och kontinuelig variabel. Den tillhör kategorin vägglett lärande (supervised learning). Det innebär att modellen tränas på data där facit redan finns.
Inom regression används termen Y för en beroende variabeln (målvariabel, target) och X för de oberoende variablerna (features). Målet är att hitta optimala värden på modellens inre parametrar. På så sätt hittas sambandet mellan variablerna.
Exempel på regressionsmodeller är Linjär regression (med undergrupperna Lasso, Ridge och Elasticnet), Support Vector Regressor (SVR), Beslutsträd, Random Forest, k-Nearest Neighbors (kNN), Polynomregression.
Potensiella tillämpningsområden är att göra förutsägelser i verkliga scenarier. Det kan vara värdering av bilar (Y = pris, X = mil, årsmodell, märke, hästkrafter), ekonomisk analys (Y = inkomst, X = ålder, utbildningsnivå), hälsa (Y = hälsa, X = motion, kostvanor, socialt nätverk)

## 12. Hur kan du tolka RMSE och vad används det till.

RMSE (Root Mean Squared Error) är det vanligaste utvärderingsmåttet vid regressionsproblem. Det används för att mäta hur mycket en modell förutsägelser i genomsnitt avviker från det sanna värdet.
En stor fördel med RMSE är att det mäter felet i samma enhet som målvariablern (y). Om jag till exempel vill förutse bilpriser så anges RMSE i kronor. RMSE beräknas som roten ur medelvärdet av det kvadrerade felet. Det gör att stora fel bestraffas hårdare än små fel. Ett lågt RMSE borde innebära en bättre modell med mindre fel men det är inte riktigt så enkelt alla gånger. För att avgöra om RMSE är bra eller dåligt kan man sätta det i relation till datans medelvärde. Exempelvis om RMSE är 54 men medelvärdet är 145. Då har modellen 37 procent felaktiga prediktioner, inte jättebra med andra ord.
RMSE används främst för att rangordna olika modeller mot varandra. Vid användning av GridSearchCV kan en variant av RMSE, Mean Squared Error (MSE) används. Den används som mått för att avgöra vilka inställningar som är bäst.

## 13. Vad är "klassificieringsproblem? Kan du ge några exempel på modeller som används och potentiella tillämpningsområden?

Ett klassificeringsproblem innebär att förutsäga vilken kategori eller klass en viss datapunkt tillhör. Det är en typ av väglett lärande. Väglett lärande innebär att modellen tränar på data där facit redan finns. Klassificering ger diskret output genom att modellen uppskattar sannolikheten för att en observation tillhör en viss klass.
Exempel på modeller är:

- **Logistisk regression** som används för binär klassificering (sannolikhet att något tillhör en klass eller ej). Exempel på det är ja eller nej.
- **Beslutsträd (Decision Trees)** som delar datan i steg genom en serie ja/nej frågor kombinerat med cut-off värden tills den når en slutgiltig kategori.
- **Random Forest** som kombinerar resultatet från många olika beslutsträd för att göra stabilare prediktioner

- **Support Vector Classifier (SVC)** som försöker hitta en så bred "väg" som möjligt för att separera olika klasser
- **k-Nearest Neighbors (kNN)** mäter avståndet mellan datapunkter i ett n-dimensionellt rum. När modellen ska göra en prediktion för en ny punkt letar den upp de punkter i träningsdatan som ligger närmast (grannar).

Potentiella användningsområden är e-posthantering (spam/icke-spam), ställa diagnoser inom sjukvården, upptäcka bedrägerier vid transaktioner, förutsäga chrun, identifiera objekt i bilder.

## 14. Vad är en "Confusion Matrix"?

Confusion matrix används för att utvärdera prestandan hos en klassificeringsmodell. Dess styrka är att den kan ge både en visuell och numerisk sammanställning av hur väl modellens prediktioner stämmer överens med det faktiska "facit".
Matrisen delar in reslutatet i:

- True Positive (TP): Modellen gissade "Ja" och det var sant
- True Negative (TN): "Nej" och det var sant
- False Positive (FP): "Ja" men det var falskt (kallas även typ I-fel)
- False Negative (FN): "Nej" men det var falskt (kallas även typ II-fel)

Diagonalen i matrisen representerar alla korrekta gissningar (TP och TN) och de utanför är alla felaktiga gissningar. Matrisen går även att skala upp när svaret kan vara flera olika värden, t.ex. MNIST 0-9.
Att bara titta på modellens träffsäkerhet (accuracy) kan vara helt fel. Det gäller speciellt på obalanserad data. Om 95 % av all data är negativ. Om modellen gissar nej på allt kommer den ändå få 95 % accuracy. Den kan däremot inte prediktera de positiva.

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

K-means är den vanligaste algoritmen för klustring. Klustring på sin sida är en form av icke-väglett lärande. Modellen K-means används för att dela in datapunkter i olika grupper, så kallade kluster, så att punkter som liknar varandra hamnar i samma grupp. Det ser iterativ genom stegen:

- Initialisering (centroider slumpmässiga mittpunkter placeras ut)
- Tilldelning: varje punkt tilldelas närmaste centroid
- Uppdatering: centroider flyttas till medelvärdet av punkterna i klustret
- Upprepning tills det stabiliseras

Exempel är segmentera kunder i en webshop, t.ex. “storköpare”, “fönstershoppare”, “kampanjköpare”, baserat på köpdata. Ändra tillämpningar är anomalidetektion och bildbehandling (genom att hittta naturliga grupperingar av handskrivna av siffror.

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

**Ordinal encoding** innebär att varje kategori får ett heltal som speglar en naturlig inbördes ordning. Poängen är att modellen ska kunna tolka att ett värde är “större” eller “mindre” än ett annat. Ett klassiskt exempel är klädstorlekar där “Liten” kan mappas till 1, “Medium” till 2 och “Stor” till 3. Då representerar siffrorna rangordningen och inte bara en etikett.

**One-hot encoding** skapar i stället en ny kolumn för varje unik kategori. För varje rad sätts värdet till 1 i kolumnen som motsvarar kategorin och 0 i alla andra. Exempelvis för färgerna röd, blå och grön får en röd bil värdet 1 i kolumnen “Röd” och 0 i “Blå” och “Grön”. På så sätt behandlas kategorierna som fristående utan att någon ordning antyds.
**Dummy variable encoding** är en variant av one-hot encoding där man följer samma idé men tar bort en av kolumnerna. Den borttagna kategorin blir en slags referens: om alla kvarvarande kolumner är 0 innebär det att observationen tillhör den borttagna kategorin. Ett exempel är kön (Man, Kvinna) där man kan skapa bara en kolumn “Kvinna”. Om värdet är 1 betyder det kvinna och om värdet är 0 betyder det man (eftersom “Man” togs bort). Detta görs ofta för att undvika statistiska problem som multikollinearitet.

## 17. 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?

Göran har rätt i den grundläggande teorin. Data brukar delas upp i nominal (ingen naturlig ordning, t.ex. färger) och ordinal (tydlig ordning, t.ex. liten–medium–stor). Men Julia har mest rätt i praktiken för att källorna betonar att det beror på situationen hur en variabel ska tolkas. Färger är normalt nominala eftersom “röd” inte är mer eller mindre än “blå”, men om färger i ett specifikt sammanhang representerar en rangordning. Som i Julias exempel där röd = “bäst”. Då har man skapat en artificiell ordning. Därför bör man behandla det som ordinal data så att modellen kan få med den informationen. Skillnaden spelar roll i preprocessing. Om man ser färgerna som nominala använder man one-hot/dummy encoding men om de faktiskt har en rangordning i just problemet kan ordinal encoding vara mer korrekt för att bevara den ordningen.

## 18. Vad är skillnaden mellan parametrar och hyperparametrar i en maskininlärningsmodell? Ge ett exempel på varje och förklara varför de inte kan optimeras på samma sätt.

**Parametrar** är värden som modellen lär sig från träningsdatan. De är modellens “kunskap” efter träning. Exempelvis i linjär regression är koefficienterna och intercept parametrar. De optimeras automatiskt under träningen. Ofta genom en optimeringsmetod som gradient descent eller en sluten formel.
**Hyperparametrar** är inställningar som du väljer innan träningen. De styr hur modellen ska lära sig och hur komplex den blir. Exempelvis k i kNN, max_depth i beslutsträd, C och gamma i SVM.De optimeras inte av träningsalgoritmen. Du provar flera alternativ och jämför resultat.
Parametrar kan optimeras direkt på träningsdatan. Målet är att minimera en förlustfunktion. Hyperparametrar måste utvärderas på data som modellen inte tränat på. Annars riskerar du att välja inställningar som passar träningsdatan för bra och ger sämre generalisering. Därför används ofta korsvalidering med GridSearchCV eller RandomizedSearchCV.

## 19. Förklara skillnaden mellan overfitting och underfitting i en maskininlärningsmodell. Beskriv även hur man kan upptäcka respektive åtgärda dem.

**Underfitting** betyder att modellen är för enkel. Den fångar inte mönster i datan. Det hänger ofta ihop med hög bias. Du upptäcker det genom att modellen får låg prestanda både på träningsdata och valideringsdata. I regression kan man också se det i en residualplot. Då får felen ofta ett tydligt mönster. Du kan åtgärda det genom att använda en mer komplex modell. Du kan också lägga till bättre och fler relevanta features. En annan åtgärd är att minska regularisering om du har strypt modellen för hårt.
**Overfitting** betyder att modellen blir för komplex. Den börjar lära sig brus och detaljer i träningsdatan. Det hänger ofta ihop med hög varians. Du upptäcker det genom att modellen är väldigt bra på träningsdata men märkbart sämre på valideringsdata och testdata. Ett stort gap mellan train och val är en tydlig signal.
Det går att åt genom bland annat regularisering som Ridge eller Lasso. Du kan också förenkla modellen. Mer träningsdata hjälper ofta. Feature selection kan minska risken om du har irrelevanta variabler. Ensemblemetoder som Random Forest kan också minska överanpassning.
Målet är att hitta en balans mellan bias och varians. Korsvalidering hjälper dig att se om du är på väg mot underfitting eller overfitting innan du låser din slutmodell.