Detta dokument innehåller övningsuppgifter för boken "Lär dig AI från grunden - Tillämpad maskininlärning med Python". Varje kapitel har tre typer av frågor enligt nedan.

Faktafrågor - Här finns frågor som kan besvaras utifrån fakta från boken. Syftet är att befästa kunskaper som vi läst om.

Resonemangfrågor - Här finns frågor där vi i högre utsträckning behöver resonera.

Koduppgifter - Här finns praktiska koduppgifter.

Fler dataset
I koduppgifterna kommer olika dataset användas. Den intresserade läsaren kan hitta många fler dataset att öva på genom att besöka Kaggle:

https://www.kaggle.com/datasets

Kapitel 1 - Introduktion till maskininlärning

Faktafrågor

1. Hur hänger AI, ML och DL ihop?

Artificiell Intelligens (AL) handlar om förmågan hos datorprogram och robotar att efterlikna oss människors eller djurs intelligens.

Machine Learning (ML) ger datorer förmåga att lära sig själva med hjälp av data.

Deep Learning (DL) är en modellarkitektur som ursprungligen var inspirerade av bland annat hur människohjärnan fungerar så kallade neurala nätverk.

2. Vilka är de fyra problemkategorierna inom ML?

Regression, Klassificering, Dimensionsreducering och Klustring.

Regression, Klassificering är vägledande lärande. X variabler vägleder hur man ska prediktera y.
Dimensionsreducering och Klustring är icke vägledande lärande.

3. Förklara följande:

a) Vad är syftet med att dela upp data i träningsdata, valideringsdata och testdata?

Syftet med att dela upp datat i tre kategorier är när vi tränar ML-modellerna.

På träningsdatan tränar vi olika modeller.
På valideringsdatan utvärderar vi de olika modellerna som tränats på träningsdatan för att se vilken modell som presterar bäst. För att veta vilken modell som presterar bäst används ett utvärderingsmått. För regressionsproblem används ofta RMSE.
När vi har den bäst presterade modellen från valideringsdatan ska vi, givet att vi är nöjda med resultatet, träna om den på träningsdatan och valideringsdatan ihopslagen innan modellen slutligen testas på testdatan.

b) Vad är k-delad korsvalidering för något?

Strukturera upp data i k-delar och sedan rotera träning över dessa k-delar.
Om man tar tennis som exempel, man är en ny spelare och sätter en serv som blir väldigt bra. Tränaren ger då öveningen att göra tio servar för att se om man är bra och inte bara hade tur att man satte en bra serv.

Först delar man in träningsdata och testdata i 80-20 och 70-30 uppdelning. Träningsdatan delas därefter upp i k lika stora delar där modellen tränas k gånger. Om vi använder en 5-delad korsvalidering så kommer vi få totalt fem olika siffror som vi tar medelvärdet av vilket gör att vi får en slutgiltig siffra som används för att mäta hur bra vår modell presterar.

c) Vad är root mean squared error (RMSE) för något?

RMSE är ett utvärderingsmått för regressionsproblem.

d) Vad är en hyperparameter för något? Vad är en parameter för något?

En hyperparameter svarar på frågan "hur vi lär oss" och en parameter svarar på frågan "vad vi lärt oss".

En hyperparameter är där vi användare ställer in hur programmet ska tränas.
En parameter är där programmet räknar ut själv och ställer in sig självt.

Om man tar tennis som exempel så kan man välja själv hur man vill träna. Träna inomhus eller träna utomhus. Kan träna på att slå på olika sätt. Parametern är då vad man har lärt sig av träningen.

I Pythons sklarn så kallas hyperparametern för parameter och parameter för attribut.

e) Vad är grid search och hur hänger namnet "grid" och "search" ihop med processen som genomförs? Läser vi dokumentationen: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV så ser vi att hyperparametern `refit` har standardvärdet `True`, vad innebär det?

Hyperparametern refit är en estimerare som hittar de bästa parametrarna i hela datasätet. När vi har genomfört vår grid search, det vill säga använt .fit() metoden, så tränas modellen om med de bästa hyperparametrarna på hela datasetet och sparas så vi kan använda den, exempelvis för att genomföra prediktioner.

Grid search är ett sätt att utvärdera olika analysmodeller och utvärdera olika hyperparameter. Det gör man genom att man utvärderar alla modeller i en slags kordinatsystem för att se vilken som är den bästa.

f) Vad är kategorisk data och hur hanteras det? I ditt svar, använd begreppen nominal data, ordinal data, one-hot-encoding, dummy-variable-encoding och ordinal encoding.

När vi arbetar med ML-modeller, exempelvis den linjära regressions-modellen, så behöver modellerna numerisk indata. Trots detta så vet vi från verkligheten att kategorisk data som antar värden i olika kategororier såsom färg och kön kan vara viktiga. Vi omvandlar den kategoriska datan till numerisk data och kan därefter använda den i våra modeller. När vi arbetar med kategorisk data kan vi skilja mellan nominal data och ordinal data.

Nominal data är t.ex färgerna: ['röd', 'grön', 'blå', 'svart', 'röd'], datat har inte en inbördes ordning.

Ordnial data finns det en inbördes rangordning.

För att göra vår one-hot-encoding så använder vi Pandas funktionen get_dummies(). Funktionen heter så eftersom variabler/kolumner som endast kan anta värdet 0 eller 1 kallas för dummyvariabel.

Skillnaden mellan one-hot-encoding och dummy-variable-encoding? Rent informationsmässigt ger metoderna samma information men när vi ska använda nominal data i den linjära regressions-

modellen så ska vi använda dummy-variable-encoding med vi för övriga modeller använder one-hot-encoding. Varför det är så går att matematiskt motivera.

g) Vad är feature engineering för något?

Feature engineering handlar om vilka variabler som ska väljas (variabelselektion) i en modell, skapandet av nya variabler och transformering av variabler. Generellt sett åsyftas de oberoende variablerna (features) men även exempelvis transformering av den beroende variabeln kan göras.

Variabelselektion är generellt sett mycket viktigt eftersom oavsett hur komplicerade modeller vi använder oss av så kommer de inte att prestera bra om fel variabler används. "Shit in, shit out" är ett uttryck för detta, det vill säga stoppar vi in dåliga variabler i modellera får vi dåliga predektioner. Den viktigaste metodiken för att välja variabler är sitt omdöme och eventuellt teori.

h) Vad menas med principle of parsimony?

En generell princip inom modellering benämns principle of parsimony och den säger att givet en viss prestationsnivå så försöker man ha en så enkel modell som möjligt. Rent generellt, inom vetenskapliga sammanhang, benämns det Ockhams rakkniv. Att göra saker så enkelt som möjligt.

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

Exempel en modell som visar att med mer inkomst ökas lyckan. Det är en förenkling av verkligheten eftersom vi vet att det finns fler faktorer.
Målet är inte att skapa en modell som ger "exakt bild av verkligheten" utan en modell som hjälper oss att uppnå det syfte vi har.

5. Vad menas med att en modell är "överanpassad" eller overfitted på engelska?

Om vi tar exmplet med lycka och inkomst i boken. Följer man prickarna genom att dra ett streck från punkt till punkt så blir den "överanpassad". I "överanpassad" kan en person med lägre inkomst ha större lycka än en person med hög inkomst. Det är såklart för lite att gå på för att det ska stämma i verkligheten, man tar inte hänsyn till miljö, stress och hur mycket tid man behöver arbeta på och utanför arbetsplatsen.

Lagomt är bäst i ML.

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

Scikit-learn är en allmän princip att "högre värden är bättre" i samband med score beräknas.

MSE = neg_mean_squared_error

Antag att vi har två modeller och att modell 1 har en MSE på 100 och modell 2 har en MSE på 150. Då är modell 1 bättre eftersom felet (MSE) är lägre. Men i scikit-learn hade modell 2 betraktas som bättre eftersom scikit-learn följer den allmäna principen att "högre värden är bättre" i samband med scoring vilket det är uppenbart inte är i detta fallet. För att hantera detta tar vi de negativa värdena (negative mean squared error) och får -100 och -150. Nu är -100 större än -150 eftersom det är nämre noll. Scikit-learn hade därmed betraktat modell 1 som bättre. Nu blir det alltså korrekt rangordning av modellerna!

7. Vad är tvärsnittsdata, tidsseriedata och paneldata? Exemplifiera när respektive datakategori kan uppstå.

Data kan kategoriseras på olika sätt. Bra kategorier som är bra att känna till är tvärsnittsdata, tidsseriedata och paneldata.

Tvärsnittsdata är sådan data där observationer för olika individer (det kan exempelvis vara människor, företag eller länder) samlas in vid en tidpunkt

Tidseriedata är sådan data där observationer över tid samlas in för en individ (det kan exempelvis vara människor, företag eller länder).

Paneldata är sådan data där observationer för individer (det kan exempelvis vara människor, företag eller länder) samlas in över tid. Notera alltså att tvärsnittsdata och tidseriedata kan betraktas som specialfall av paneldata. Tvärsnittsdata är en tidpunkt från paneldatan och tidseriedatan är en individ från paneldatan.

Resonemangfrågor

8. Ge några exempel på verkliga tillämpningsområden inom ML. Sök gärna på nätet för att besvara frågan.

Bildigenkänning är en annan maskininlärningsteknik som förekommer i vardagen.

Rekommendationsmotorer är en av de mest populära tillämpningarna av maskininlärning.

Prediktiv analys kan hjälpa till att avgöra om en kreditkortstransaktion är bedräglig eller legitim.

9. Generellt sett gäller det att högre är bättre i scikit-learn scoring, därför används exempelvis `scoring='neg_mean_squared_error'`. Förklara logiken bakom detta, det vill säga att vi använder "negative" mean squared error.

Scikit-learn anser att "högre värden är bättre än lägre värden". I det här fallet visar en liten MSE att förutsägelser ligger nära data, så det följer den motsatta regeln. Det är därför sklearn betraktar den negativa (faktiskt motsatta) MSE som poäng. Således är en stor neg_mean_squared_error bättre än en låg. Det är också konsistent med en graf med extrema värden, för extrema värden i allmänhet försämrar en modell.

Koduppgifter

10. Gå igenom samtliga kodexempel i kapitlet och skriv gärna av koden manuellt.

Det är även bra att experimentera genom att ändra vissa delar av koden och läsa dokumentationen. Se dokumentet kod_exempel.

11. I koden nedan använder vi först `test_size=0.2` och sedan `test_size=0.25`, förklara varför det ger oss proportionerna 60-20-20.

```
X_train_full, X_test, y_train_full, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

X_train, X_val, y_train, y_val = train_test_split(X_train_full, y_train_full,
test_size=0.25, random_state=42)
```

Vi får proportionerna 60-20-20 eftersom:
Först reserveras 20% för test.
Sedan delas de återstående 80% upp i 75% träning (som blir 60%) och 25% validering (som blir 20%)

Kapitel 2 - Ett ML projekt från början till slut

Faktafrågor

1. I kapitlet beskrivs en checklista med sju steg. Beskriv de sju stegen översiktligt. I verkligheten, följs dessa steg i en rak progression eller arbetar man generellt sett mer iterativt?

1) **Definera problemet och skapa en helhetsbild**
   Vad är målet med projektet?

2) **Få tillgång till datan**
   Vilken data behöver vi för projektet och hur får vi tillgång till den?

3) **Utforska datan, gör en exploratory data analysis (EDA)**
   Finns det mönster som syns i datan?
   Finns det några visualiseringar som kan vara särskilt användbara?

4) **Bearbeta datan**
   Det kan finnas data/variabler som inte är användbara för vårt ändamål, dessa kan vi ta bort.

5) **ML-modellering**
   Prova olika modeller med standardvärden.

6) **Presentera din lösning för intressenter**
   Ofta ska vi presentera en lösning för olika intressenter. Det kan exempelvis vara kollegor inom en team.

7) **Produktionssättning av modellen och övervakning av implementering**

Man arbetar iterativt där man går "fram-och-tilbaka" och hoppar mellan stegen.

2. Vad menas med att en modell produktionssätts?

Då har vi skapat en modell som fungerar bra är det en god idé att bevaka modellens prestanda. Ofta tränas modeller regelbundet om på ny data, exempelvis månadsvis eller veckovis. Modellen kan bli sämre över tid och behöver därför bevakas. Det får att skapa olika enhetstester för att säkerhetställa att funktionaliteten är som den ska vara.

3. Vad är scikit-learn för något? Biblioteket följer några centrala designprinciper. Vilka är dessa? Vad är estimators, predictors och transformers?

Scikit-learn är ett populärt, open-source Python-bibliotek för maskininlärning (machine learning). Scikit-learn är byggt ovanpå NumPy, SciPy och matplotlib och är mycket använt både i forskning och industri tack vare sin enkelhet, effektivitet och goda dokumentation.

Biblioteket följer följande designprinciper som gör biblioteket användarvänligt och kraftfullt:

- Konsistens
- Inspektion
- Begränsning av klasser
- Rimliga standardvärden för hyperparameterarna

Det centrala objektet i scikit-learn benämns estimators och de lär sig (estimerar) från data genom .fit() metoden. Estimators som kan göra prediktioner genom .predict() metoden kallas för predictors. Predictors är alltså en delmängd av estimators.

LinearRegression() är både en estimator och en predictor.

Estimators som transformerar data genom .transform() metoden kallas för transformers. Transformers är alltså en delmängd av estimators.

StandardScalar() är både en estimator och en transformer.

4. Vad är TensorFlow och Keras?

TensorFlow och Keras är båda verktyg (ramverk) som används inom maskininlärning och särskilt inom djupinlärning (deep learning). Här är en enkel förklaring av båda:

TensorFlow är utvecklat av Google Brain team som är ett open-source ramverk för maskininlärning. Det används till att bygga, träna och köra neurala nätverk och andra typer av maskininlärningsmodeller. Har stöd för olika språk, men används mest med Python.

Keras är utvecklat ursprungligen av François Chollet, nu en del av TensorFlow.
Det är ett högnivå-API (gränssnitt) för att snabbt och enkelt bygga neurala nätverk.
Det används till att definiera och träna modeller på ett mer intuitivt och lättläst sätt.
Perfekt för prototyper och undervisning.

Resonemangfrågor

5. Kalle och Stina diskuterar maskininlärning över en lunch. Kalle säger "om jag tränat en modell och den inte presterar bra nog på testdatan så justerar jag den tills den gör det." Stina säger "det är ett stort fel att göra så, det enda du då åstadkommer är att du överanpassar testdatan. Hela syftet med testdatan försvinner då". Vad säger du om deras dialog?

Stina invänder och säger att det är ett stort fel att använda testdatan på det sättet, och hon har helt rätt. Om du börjar justera modellen baserat på testresultat så är det inte längre testdata i egentlig mening, eftersom modellen då anpassas till det.

6. Många AI/ML projekt uppnår inte de ursprungligen satta målen eller att ens passera någon form av prototyp-stadie Vad tror du detta beror på och hur ska vi förhålla oss till det?

Vilka orsaker kan det bero på?

- Orealistiska förväntningar.
- Dålig eller otillräcklig data.
- Brister i problemanalys och måldefinition.
- Saknad av ML-kompetens i organisationen.
- Ingen koppling till affärsnytta eller användning.

Hur ska vi förhålla oss till det?

- Starta smått och hypotesdrivet.
- Involvera domänexperter och användare tidigt.
- Fokusera på datakvalitet först.
- Tänk i ML-produktens livscykel (MLOps).
- Acceptera att vissa projekt ska skrotas.

Koduppgifter

7. Gå igenom samtliga kodexempel i kapitlet och skriv gärna av koden manuellt. Det är även bra att experimentera genom att ändra vissa delar av koden och läsa dokumentationen.

8. Förklara vad koden nedan gör. Varför är det viktigt att kunna spara en modell?

```
from sklearn.datasets import make_regression
from sklearn.linear_model import LinearRegression
from joblib import dump, load

X, y = make_regression(n_samples=20000, n_features=3, noise=0.1)

model = LinearRegression().fit(X, y)
dump(model, "linear_model.joblib")
model_loaded = load("linear_model.joblib")

print(model_loaded.predict(X[:5]))
```

Koden ovan gör följande

- Skapar ett dataset.
- Tränar en linjär regressionsmodell.
- Sparar modellen till en fil linear_model.joblib. enna filen kan sedan laddas igen utan att träna modellen.
- Laddar in modellen igen, och använder den för att göra prediktioner.

Att kunna spara modeller är centralt för att bygga skalbara, reproducerbara och användbara AI/ML-system.

9. Denna uppgift består av flera steg enligt nedan.

a) Läs in datasetet "data_01.csv" med funktionen `read_csv()` från Pandas. Funktionen returnerar en `DataFrame`.

```
import pandas as pd

df = pd.read_csv("data_01.csv")
```

b) Dela upp datasetet i `X` och `y`.

```
X = df.iloc[:, :-1]  # Alla kolumner utom sista
y = df.iloc[:, -1]   # Sista kolumnen som mål
```

c) Dela upp datan ytterligare, i ett träning-, ett validering-, och ett testset med `train_test_split()`. Låt 20% av datan vara testdata och 15% av den återstående datan vara valideringsdata.

```
from sklearn.model_selection import train_test_split

# Först dela ut 20% till testdata
X_train_val, X_test, y_train_val, y_test = train_test_split(
    X, y, test_size=0.20, random_state=42)

# Av de återstående 80%, ta ut 15% som valideringsdata
# 15% av 80% = 0.15 / 0.8 ≈ 0.1875
X_train, X_val, y_train, y_val = train_test_split(
    X_train_val, y_train_val, test_size=0.1875, random_state=42)
```

Nu har jag:
X_train, y_train – träning (ca 65%)
X_val, y_val – validering (ca 15%)
X_test, y_test – test (20%)

d) Träna två valfria regressionsmodeller (exempelvis `LinearRegression` och `DecisionTreeRegressor`) på träningsdatan. Notera, i kapitel 3 kommer vi lära oss mer om `DecisionTreeRegressor`.

```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor

# Modell 1: Linjär regression
linreg = LinearRegression()
linreg.fit(X_train, y_train)

# Modell 2: Beslutsträd
tree = DecisionTreeRegressor(random_state=42)
tree.fit(X_train, y_train)
```

e) Utvärdera modellerna på valideringsdatan.

```
from sklearn.metrics import mean_squared_error
```

```
# Linjär regression
y_val_pred_lin = linreg.predict(X_val)
mse_lin = mean_squared_error(y_val, y_val_pred_lin)

# Beslutsträd
y_val_pred_tree = tree.predict(X_val)
mse_tree = mean_squared_error(y_val, y_val_pred_tree)

print(f"Validerings-MSE (Linjär): {mse_lin:.2f}")
print(f"Validerings-MSE (Träd): {mse_tree:.2f}")
```

f) Träna om den bäst presterande modellen på både tränings- och valideringsdatan.

```
# Kombinera X_train och X_val
X_train_val = pd.concat([X_train, X_val])
y_train_val = pd.concat([y_train, y_val])

# Träna om den bästa modellen
best_model = DecisionTreeRegressor(random_state=42)
best_model.fit(X_train_val, y_train_val)
```

g) Utvärdera modellen på testdatan.

```
y_test_pred = best_model.predict(X_test)
mse_test = mean_squared_error(y_test, y_test_pred)

print(f"Test-MSE: {mse_test:.2f}")
```

h) Träna om modellen på hela datasetet.

```
final_model = DecisionTreeRegressor(random_state=42)
final_model.fit(X, y)
```

10. Datasetet "salary_dataset.csv" Innehåller 29 observationer av personer. `YearsExperience` är hur många år de arbetat, och `Salary` är deras lön.

a) Läs in datasetet "salary_dataset.csv" med pandas `read_csv()`-funktion och dela upp i `X` och `y`. Avgör själv vilken variabel som ska vara den beroende variabeln `y`.

b) Dela upp datasetet i träning- och testset. (Inget validerings-set alltså!)

c) Träna två regressionsmodeller med k-delad korsvalidering med `cross_validate()`-funktionen från scikit-learn. Använd `neg_root_mean_squared_error` som `scoring`. Välj själv hur många iterationer den ska göra genom hyperparametern `cv`.

d) Utvärdera modellen som presterar bäst på testsetet.

11. I denna uppgift kommer vi arbeta med kategorisk data. Nominaldata är kategorisk data där kategorierna inte har någon inbördes rangordning.

Datasetet `mpg` som följer med biblioteket seaborn är ett dataset med 398 observationer av bilar. Den beroende variabeln `mpg` står för miles per gallon och beskriver bilarnas bränsleeffektivitet.

Om vi vill träna en regressionsmodell på `mpg`-datasetet behöver vi hantera de två kategoriska variablerna `origin` och `name`. Variabeln `origin` är en kategorisk variabel med tre olika värden: `europe`, `japan` och `usa`. Den lämpar sig bra för one hot encoding. Variabeln `name` har 305 unika värden. Skulle vi utföra one hot encoding på den skulle vårt dataset få 305 nya dimensioner vilket inte är så lämpligt i detta fall. Dessutom - tror du att namnet på bilen har något att göra med hur långt den kör på en gallon bensin? Vi ska därför droppa `name`-kolumnen innan vi börjar träna en modell på datan.

```
import seaborn as sns

df = sns.load_dataset("mpg")
print(df.head())
```

a) Läs in datasetet `mpg` med seaborns `load_dataset()`-funktion (det gjordes i koden ovan).

b) Droppa rader med saknade värden med `dropna()`-metoden.

c) Droppa kolumnen `name`.

d) Utför en dummy-variable-encoding på `origin`-kolumnen med pandas `get_dummies()`-funktion. Ange `drop_first=True` så att vi får 2 nya kolumner istället för 3.

e) Dela upp datasetet i `X` och `y`, med `mpg` som den beroende variabeln `y`.

f) Dela upp datasetet i träning- och testset.

g) Träna en linjär regressionsmodell på träningsdatan och utvärdera den på testdatan.

12. I avsnitt 2.2 "Ett kodexempel från början till slut - Huspriser i Kalifornien" så gås ett komplett kodexempel igenom. På valideringsdatan fick vi `RMSE Random Forest Regression: 52277.96578719621`. Försök få ett bättre resultat genom att exempelvis justera hyperparametrar eller genomföra variabelselektion.

Har ej utfört koduppgifterna 11 och 12.

Kapitel 3 - Regression

Faktafrågor

1. Vad kännetecknar regressionsproblem? Ge några exempel på tillämpningsområden.

Ett regressionsproblem handlar om att förutsäga ett kontinuerligt värde baserat på en eller flera oberoende variabler (features). Målet är att hitta en funktion eller modell som kan förutsäga ett numeriskt värde för en given uppsättning av indata.

Det kan användas till bilvärdering, fastighetsvärdering, jordbruksproduktion och försäkringar.

2. Förklara utvärderingsmåtten RMSE, MSE och MAE.

MSE är ett mått på medelvärdet av de kvadrerade skillnaderna mellan de verkliga värdena och de förutsagda värdena. Det ger ett kvantitativt mått på hur mycket modellens förutsägelser skiljer sig från de faktiska värdena.

RMSE är roten ur MSE. Det innebär att RMSE omvandlar den kvadrerade enheten (som vi får i MSE) tillbaka till samma enhet som de ursprungliga datavärdena. Det är måttet som används mest i samband med att regressionsmodellerna utvärderas.

MAE beräknar det genomsnittliga absolutbeloppet av skillnaderna mellan de verkliga och de förutsagda värdena. Den skiljer sig från både MSE och RMSE genom att den inte kvadrerar skillnaderna, utan bara tar det absoluta värdet.

För alla tre utvärderingsmåtten så gäller lägre värde indikerar bättre modell.

3. Om vi ska rangordna olika modeller, spelar det någon roll om RMSE eller MSE används? Varför?

Är vi intresserade av rangordning av modeller är det dock generellt sett mer effektivt att använda MSE eftersom rangordningen av modellerna blir detsamma men det är beräkningsmässigt billigare att beräkna MSE eftersom inga beräkningar av kvadratroten behöver göras vilket kan spara tid.

4. Förklara mycket översiktligt vad gradient descent är.

Inom maskininlärning tränar vi modeller där målet med träningen är att hitta optimala parametrar för modellerna. Den används för att hitta en funktions minimum och maximum.

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

Ett viktigt teoretiskt resultat från statistik och maskininlärning är det som benämns the bias variance trade-off. Det går matematiskt bevisa resultatet men det är överkurs och inget som beskrivs i boken.

The bias variance trade-off säger att felet en modell gör när den predikterar ny, osedd data kan delas in i tre delar, Bias, Variance och Irreducerbart.

När vi ökar en modells komplexitet, exempelvis går från vanlig enkel linjär regression till polynomregression, ökar vi generellt sett dess varians men minskar dess bias och vice versa om vi minskar en modells komplexitet, då minskar vi generellt sett dess varians men ökar dess bias. Det är därför inte alltid bättre

6. Några vanligt förekommande modeller för regressionsproblem är enligt nedan. Förklara översiktligt hur respektive modell fungerar. Läs även igenom respektive modells dokumentation, notera att du inte behöver förstå alla detaljer från dokumentationen men det är bra att ha läst igenom den.

a) Linjär regression

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

Denna modell är enkel och är väldigt central. Den används ofta i verkligheten och är en bra modell för rent allmänt förstå modellering.

b) Ridge regression

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ridge_regression.html

Ridge regression är en regulariserad variant av den linjära regressionsmodellen där överanpassning motverkas genom att minska värdet på vikterna. Genom att minska värdet på vikterna och därmed begränsa modellen så blir den mindre flexibel, något som ökar bias men minskar variansen. Förhoppningen är alltså då att minskningen i varians blir större än vad ökningen i bias blir.

c) Lasso regression

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html

Lasso är en regulariserad variant av linjär regression. Lasso regression tenderar att sätta sina vikter till 0 som kan jämföras med ridge regression där vikterna generellt sett krymtes så de närmade sig 0.

Vissa variabler som inte är tillräckligt användbara då är det bättre att sätta till noll för att spara budgeten till de variabler som i högre utsträckning bidrar till att vi kan genomföra bra prediktioner.

d) Elastic net

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html

Elastic net är en kombination av lasso och ridge regularisering. Den balanserar båda teknikerna med hjälp av mix ratio r.

Om r är satt till 0 är kostnadsfunktionen för elastic net samma som ridge regression.
Om r är satt till 1 är kostnadsfunktionen istället samma som för lasso regression.

Denna kombination för det möjligt för Elastic Net att välja viktiga features (som lasso) samtidigt som den kan ha kvar några små vikter som inte är 0 (som ridge).

e) Support vector machines

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html
Support vector machines är en modell som kan användas både för regressionsproblem och för klassificeringsproblem.
För regressionsproblem försöker SVM modellen skapa en väg, med en given bredd, som innehåller så många observationer som möjligt. Mitten-linjen i vägen är det som blir modellens predikterade värden.

f) Beslutsträd

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html

Modellen kan användas för både regressionsproblem och klassificeringsproblem.

g) Ensemble learning

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingRegressor.html

Ensemble learning handlar om att kombinera flera modeller med förhoppningen att de gemensamt kan prestera bättre än vad varje enskild modell kan göra.

h) Random forest

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

Random forest är ett ensemble av beslutsträd som oftast skapas genom bagging även pasting också förekommer. Precis som för beslutsträd är det ofta bra att regularisera random forest.

7. Vad menas med white box modeller och black box modeller?

White box-modeller är modeller där vi kan förstå och förklara exakt hur beslut fattas, och vi kan se sambanden mellan indata och resultat.

Black box-modeller är modeller där det är mycket svårt, om inte omöjligt, att förstå exakt hur beslut fattas. Man vet inte riktigt vad som händer "inuti" modellen, men den ger förutsägelser baserade på indata.
Exempelvis kan vi med ett beslutsträd manuellt stega igenom och se vilken prediktion som görs.
Det kan vi generellt sett inte göra med Random Forest och neutrala nätverk.

8. Vad är skillnaden mellan bagging och pasting?

Från ett givet dataset kan vi skapa nya dataset genom antingen begging eller pasting.

Bagging gör vi ett slumpmässigt urval med återläggning för att skapa ett nytt dataset.

Pasting gör vi ett slumpmässigt urval utan återläggning för att skapa ett nytt dataset.

Resonemangfrågor

9. Förklara hur man kan tolka figur 3.1 på sidan 113.

Det är en linjär regressions modell med lön som y och ålder som x. Sanna värdet y och predikterande ŷ. Den blåa linjen är det som modellen har lärt sig. De svarta punkterna är ursprunglig data. I resultatet går det att tolka som att åldern påverkar inkomst.

10. Förklara hur man kan tolka figur 3.13 på sidan 140. Hur hänger den ihop med figur 3.14 på sidan 141?

Figuren visualisera hur beslutsträd fungerar. I sista rutan i visualiseringen visas vilket värde som predikteras av en observation x1 och x2.

I nästa figur så visualiseras ett bestluträd grafiskt. Den visar vad tidigare figur visade i sina rutor i ett kordinatsystem med x1 och x2 och en färgskala för predikterad y.

Koduppgifter

11. Gå igenom samtliga kodexempel i kapitlet och skriv gärna av koden manuellt. Det är även bra att experimentera genom att ändra vissa delar av koden och läsa dokumentationen.

12. Datasetet "hr_employee_data.xlsx" innehåller data kopplat till ett företags anställda. Gör en EDA och tillhörande analys, föreställ dig att ledningsgruppen på ett företag ska ta del av analysen. I denna uppgift ska ingen ML-modellering göras (det kommer göras i nästa kapitel). Syftet är dels att träna på att genomföra analyser dels att demonstrera hur mycket man kan göra även utan ML-modellering.

13. Nedan ser vi en början till modellering av ett dataset kopplat till diabetes. Läs på vad datasetet innebär och genomför ett komplett ML-flöde. Börja med att göra ett grundläggande flöde som fungerar, därefter kan du om du önskar försöka förbättra resultatet.

```
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split

# This code is merely executed to see the description of the data in a smooth
way
data = load_diabetes()

print(data.DESCR)

# Storing/loading the data the way it will be used
X, y = load_diabetes(return_X_y=True, as_frame=True)
print(X.info())
print(y.info())

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

14. I denna uppgift ska vi arbeta med datasetet "car_price_dataset.csv".

a) Gör ett komplett ML flöde där den beroende variabeln, y, är `price`. Vi vill alltså skapa en modell som kan prediktera en bils pris. Ett tips är att först göra ett enkelt flöde som fungerar, har du därefter tid kan du försöka förbättra resultaten.

b) Gör en applikation med hjälp av Streamlit där användaren kan mata in olika uppgifter om en bil och därefter få ett predikterat pris. Du kan se följande video för att lära dig mer om hur Streamlit fungerar: https://www.youtube.com/watch?v=ggDa-RzPP7A&list=PLgzaMbMPEHEx9Als3F3sKKXexWnyEKH45&index=11&t=158s

c) Det du gjort i denna uppgift, hade det kunnat användas i verkligheten? Till vad i sådana fall?

15. I denna uppgift ska du göra ett komplett ML-flöde där du modellerar diamantpriser som finns tillgängliga i datasetet "diamonds.csv". Du kan läsa mer om datasetet här: https://www.kaggle.com/datasets/shivam2503/diamonds

Jag har provat göra koduppgifterna 12 och 14 för att utföra ML.

Kapitel 4 - Klassificering

Faktafrågor

1. Vad kännetecknar klassificeringsproblem? Ge några exempel på tillämpningsområden.

- Kundanalys
- Sjukvård
- Bildigenkänning
- Finans
- Kreditrisk

2. Förklara hur OvR- och OvO-algoritmerna fungerar.

Binära klassificeringsmodeller kan med hjälp av två algoritmer som heter One-vs-Rest (OvR) och One-vs-One (OvO).

OvR innebär att för varje klass i ett multiklass problem tränas en modell för att kunna särskilja om en datapunkt tillhör den klassen eller inte. När en ny observation ska klassificeras så körs alla modeller där den slutgiltiga prediktionen blir det som den modell med högst score (såsom sannolikhet) predikterar.

När vi använder oss av OvO tränar vi istället en modell för varje par av klasser. Det innebär att om det finns N stycken klasser kommer (N x (N – 1)) / 2 stycken modeller behöva tränas. Varje modell jämför klasspar och predikterar ett av de två klasserna.

3. Förklara följande utvärderingsmått:

a) Confusion matrix

En tabell som visar hur en klassificeringsmodell presterar genom att jämföra *faktiska klasser* med *modellens prediktioner.*
För en binär klassificering ser den ut så här:

	Predikterad Positiv	**Predikterad Negativ**
Faktisk Positiv	True Positive (TP)	False Negative (FN)
Faktisk Negativ	False Positive (FP)	True Negative (TN)

b) Accuracy

Andel korrekta prediktioner totalt. Ett högt värde accuracy innebär stor andel observationer har klassificerats korrekt.

Accuracy = TP + TN / TP + TN + FP + FN

Kan vara missvisande vid obalanserade klasser (t.ex. när en klass är mycket vanligare än den andra).

c) Precision

Hur många av de predikterade positiva som faktiskt är positiva.

Precision = TP / TP + FP

Hög precision ger få falska positiva.

d) Recall

Hur många av de faktiska positiva som modellen lyckas hitta.

Recall = TP / TP + FN

Hög recall ger få falska negativa.

e) F1-score

Ett balanserat mått som kombinerar precision och recall (harmoniskt medelvärde).
Bra när man vill ha balans mellan precision och recall, speciellt vid obalanserade klasser.

f) ROC-kurvan

Istället för att visualisera sambandet mellan precision och recall, visualiseras sambandet TPR (True Positive Rate) och FPR (False Positive Rate). TPR är alltså en annan benämning för recall.

Precis som för precision och recall kan vi ändra threshold för att minska och öka TPR respektive FPR beroende på vad vi vill att modellen ska prioritera.

ROC-kurvan liknar väldigt mycket precision and recall kurvan.

4. Vad är precision-recall tradeoff för något?

Precision och recall hör ihop genom det samband som benämns precision-recall tradeoff.
Precision mäter andelen av de positiva prediktionerna som faktiskt är korrekta.
Teoretiskt sett, om ett högt värde för precision är önskevärt är det möjligt att bygga en modell som endast klassificerar en observation som positiv om den är väldigt säker. Detta kommer dock, generellt sätt, ofta leda till att många värden som faktiskt är positiva felaktigt predikteras som negativa. Det lder då till att recall blir lågt.

Precision = TP / TP + FP

Recall mäter andelen av den positiva klassen som predikteras korrekt.
Teoretiskt sätt, om vi bygger en modell som alltid klassificerar en observation som positiv kommer vi att få en 100% recall (under antagandet att det finns minst en positiv observation i datasetet). Det ledet till att precision minskar eftersom andelen av positiva prediktioner som faktiskt är korrekta minskar.

Recall = TP / TP + FN

Det är detta, att högre recall generellt sett leder till en lägre precision och vice versa som benämns som precision-recall tradeoff

5. Några vanligt förekommande modeller för klassificeringsproblem är enligt nedan. Förklara översiktligt hur respektive modell fungerar. Läs även igenom respektive modells dokumentation, notera att du inte behöver förstå alla detaljer från dokumentationen men det är bra att ha läst igenom den.

a) Logistisk regression

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Logistisk regressionsmodell som klassificerar observationer i en av två klasser baserat på varablerna x1 och x2.

b) Support vector machines

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

För klassificeringsproblem (tidigare förklarades regressionsproblem) vill man istället separera datapunkter som tillhör olika klasser och målet blir därför istället att hitta så bred väg som möjligt mellan datapunkter från de två olika klasserna.

c) Beslutsträd

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Bestlutsträd kan, precis som SVM, hantera både regressions- och klassificeringsproblem. Ett beslutsträd har samma generella struktur för både regressions- och klassificeringsproblem, där trädet består av noder. Den första noden benämns rotnod och de sista noderna benämns lövnoder. För att genomföra klassificering med beslutsträd startar processen vid rotnoden och går därefter via inre noder nedåt för att slutligen komma till en lövnod, som representerar en predikterad klass.

d) Ensemble learning

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html och
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html

Ensemble Learning är en metod där flera modeller kombineras med målet att skapa en modell med bättre prediktions- generaliseringsförmåga än varje enskild modell.

e) Random forest

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Random forest är en ensemble av bestlutsträd som oftast skapas genom bagging även om pasting också förekommer. En random forrest är när flera bestlutsträd är kombinerade.

f) Extra trees

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html

Om vi för varje förgrening/split, förutom en slumpmässig grupp av features även väljer ett slumpmässig värde på tröskelvärdet får vi den modell som benämns extra tree. Eftersom optimalt tröskelvärde inte böver hittas går denna modellen generellt sett snabbare att träna.

6. Vad innebär det att vi kan kolla på feature importance med hjälp av trädmodeller såsom beslutsträd eller random forest?

Vad som är intressant med random forrest är att den kan hjälpa oss att uppskatta vilka variabler vi ska inkludera (feature_importances_). Detta hjälper oss att identifiera vilka variabler som driver modellens beslut, vilket kan användas för feature selection, modellförbättring och för att få djupare förståelse för vårt problem.

För att få fram feature importance i scikit-learn:

```
model = RandomForestClassifier()
model.fit(X_train, y_train)
importance = model.feature_importance
```

7. Stina säger till Kalle på lunchsamtalet "jag vill ha högsta möjliga precision för vår klassificieringsmodell". Kalle funderar ett tag och säger "men vad händer då med recall"? Vad hade du svarat? I vilka fall kan man tänka sig vilja ha en så hög precision som möjligt? I vilka fall kan det vara dåligt? Om vi tänker oss rättsväsendet där en slutgiltig dom kan leda till fängelse, vad kan vi då säga om precision-recall tradeoff?

Om man optimerar för hög precision innebär det att man nästan aldrig vill säga ”positiv” om man inte är väldigt säker. Färre falska positiva gör att nästan alla positiva prediktioner är korrekta. Det kan vara bra i rättsväsendet, då ”hellre fria än fälla” betyder att precision prioriteras framför recall.

8. Förklara hur man kan tolka figur 4.8 på sidan 175.

I figur 4.8 ser vi hur beslutsgränserna som modellen skapat ser ut.
Logistisk regression är en linjär modell, och således är beslutgränserna också linjära vilket vi ser i figur 4.8. Man kan välja vilken beslutsgräns som ska användas för att styra önskad nivå på precision och recall. Default om inget anges så ligger den på 50%.

9. På sidan 209 står det "på träningsdatan använder vi `.fit_transform()`, på valideringsdatan och testdatan använder vi endast `.transform()`." Förklara logiken bakom detta.

.fit_transform() metoden används på träningsdata.
transform() används på valideringsdatan och testdatan.

Det beror på hur man ska hantera skalning eller transformation. .fit_transform() gör både beräkningarna och transformationen på samma dataset. Syftet är att man vill att modellen ska lära sig av träningsdatan.

.transform() används på valideringss- och testdatan för att säkerställa att dessa datasätt behandlas på samma sätt som träningsdatan, utan att lära om statistiken på datat (dvs utan en .fit()).

Koduppgifter

10. Gå igenom samtliga kodexempel i kapitlet och skriv gärna av koden manuellt. Det är även bra att experimentera genom att ändra vissa delar av koden och läsa dokumentationen.

11. Förklara vad nedanstående kod gör och tolka resultatet från koden.

```
from sklearn.metrics import classification_report
y_true = [0, 1, 2, 2, 2]
y_pred = [0, 0, 2, 2, 1]
target_names = ['class 0', 'class 1', 'class 2']
print(classification_report(y_true, y_pred, target_names=target_names))
```

Klassificerings rapport som jämför true labels med predicted labels.

Klassificeringsraporten kommer att visa precision, recall, f1-score och support för varje class, samt overall accuracy och macro/weighted medelvärde.

12. Läs dokumentationen och visualisera ett beslutsträd: https://scikit-learn.org/stable/modules/generated/sklearn.tree.plot_tree.html

Finns ett exempel på decision tree i kod_exempel.

13. I föregående kapitel gjordes en EDA på datasetet "hr_employee_data.xlsx". Gör nu ett komplett ML-flöde där den beroende variabeln, y, är `left`. Om den variabeln är 1 så betyder det att den anställde har lämnat företaget och om den är 0 så betyder det att den anställde inte har lämnat företaget, det vill säga jobbar kvar.

14. Nedan ser vi en början till modellering av ett mycket berömt dataset kopplat till blommor. Läs på vad datasetet innebär och genomför ett komplett ML-flöde. Börja med att göra ett grundläggande flöde som fungerar, därefter kan du om du önskar försöka förbättra resultatet.

```
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

# This code is merely executed to see the description and target names in a
smooth way
iris = load_iris()
print(iris.DESCR)
print(iris.target_names)

X, y = load_iris(return_X_y=True, as_frame=True)
# Only choose two variables for the modelling to keep it simple
X = X[['sepal length (cm)', 'sepal width (cm)']]

X_train_full, X_test, y_train_full, y_test = train_test_split(X, y,
test_size=0.2, random_state=40)
```

```
X_train, X_val, y_train, y_val = train_test_split(X_train_full, y_train_full,
test_size=0.3, random_state=36)

classes = ['setosa', 'versicolor', 'virginica']
scatter = plt.scatter(X_train['sepal length (cm)'], X_train['sepal width (cm)'],
c=y_train)
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.title('Scatter Plot of Sepal Length vs. Sepal Width (Train Data)')
plt.legend(handles=scatter.legend_elements()[0], labels=classes)
```

15. I avsnitt 4.4.1 "Kodexempel 1 - Klassificering av MNIST" modelleras MNIST. Skriv kod så att du kan ta egna bilder (via exempelvis mobilen) på handskrivna siffror och prediktera dessa. Om du känner dig inspirerad kan du också skapa en Streamlit applikation. Notera, det huvudsakliga arbetet blir alltså att preprocessa, det vill säga bearbeta bilderna som tas via mobilen så modellen kan prediktera dem. I verkligheten är det ofta långt ifrån trivialt att genomföra preprocessing.