

Övningsuppgifter

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?
2. Vilka är de fyra problemkategorierna inom ML?

3. Förklara följande:

- a) Vad är syftet med att dela upp data i träningsdata, valideringsdata och testdata?
 - b) Vad är k-delad korsvalidering för något?
 - c) Vad är *root mean squared error (RMSE)* för något?
 - d) Vad är en hyperparameter för något? Vad är en parameter för något?
 - 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?
 - 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*.
 - g) Vad är *feature engineering* för något?
 - h) Vad menas med *principle of parsimony*?
4. Vad menas med att “en modell är en förenkling av verkligheten”?
5. Vad menas med att en modell är “överanpassad” eller *overfitted* på engelska?
6. Högre är bättre i *scikit-learn scoring*, vad innebär det?
7. Vad är tvärsnittsdata, tidsseriedata och paneldata? Exemplifiera när respektive datakategori kan uppstå.

Resonemangfrågor

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

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. 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)
```

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?

2. Vad menas med att en modell produktionssätts?
3. Vad är *scikit-learn* för något? Biblioteket följer några centrala designprinciper. Vilka är dessa? Vad är *estimators*, *predictors* och *transformers*?
4. Vad är *TensorFlow* och *Keras*?

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?
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?

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]))

```

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*.
 - b) Dela upp datasetet i *X* och *y*.
 - c) Dela upp datan ytterligare, i ett tränings-, ett validering-, och ett testset med `train_test_split()`. Låt 20% av datan vara testdata och 15% av den återstående datan vara valideringsdata.
 - d) Träna två valfria regressionsmodeller (exempelvis *LinearRegression* och *DecisionTreeRegressor*) på träningsdatan. Notera, i kapitel 3 kommer vi lära oss mer om *DecisionTreeRegressor*.
 - e) Utvärdera modellerna på valideringsdatan.
 - f) Träna om den bäst presterande modellen på både tränings- och valideringsdatan.
 - g) Utvärdera modellen på testdatan.
 - h) Träna om modellen på hela datasetet.
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: `europa`, `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 dropa `name`-kolumnen innan vi börjar träna en modell på datan.

```
import seaborn as sns

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

	mpg	cylinders	displacement	horsepower	weight	acceleration \
0	18.0	8	307.0	130.0	3504	12.0
1	15.0	8	350.0	165.0	3693	11.5
2	18.0	8	318.0	150.0	3436	11.0
3	16.0	8	304.0	150.0	3433	12.0
4	17.0	8	302.0	140.0	3449	10.5

	model_year	origin	name
0	70	usa	chevrolet chevelle malibu
1	70	usa	buick skylark 320
2	70	usa	plymouth satellite
3	70	usa	amc rebel sst
4	70	usa	ford torino

- Läs in datasetet mpg med seaborns `load_dataset()`-funktion (det gjordes i koden ovan).
- Dropa rader med saknade värden med `dropna()`-metoden.
- Dropa kolumnen `name`.
- 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.
- Dela upp datasetet i `x` och `y`, med mpg som den beroende variabeln `y`.
- Dela upp datasetet i träning- och testset.
- 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.

Kapitel 3 - Regression

Faktafrågor

1. Vad kännetecknar regressionsproblem? Ge några exempel på tillämpningsområden.
2. Förklara utvärderingsmått $RMSE$, MSE och MAE .
3. Om vi ska rangordna olika modeller, spelar det någon roll om $RMSE$ eller MSE används? Varför?
4. Förklara mycket översiktligt vad *gradient descent* är.
5. Vad är *the bias variance trade-off*? Varför är mer komplexa modeller 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
 - b) Ridge regression - https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ridge_regression.html
 - c) Lasso regression - https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html
 - d) Elastic net - https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html
 - e) Support vector machines - <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html>
 - f) Beslutsträd - <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html>

- g) *Ensemble learning* - <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingRegressor.html> och <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html>
- h) *Random forest* - <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>
7. Vad menas med *white box* modeller och *black box* modeller?
8. Vad är skillnaden mellan *bagging* och *pasting*?

Resonemangfrågor

9. Förklara hur man kan tolka figur 3.1 på sidan 113.
10. Förklara hur man kan tolka figur 3.13 på sidan 140. Hur hänger den ihop med figur 3.14 på sidan 141?
11. Vi har lärt oss utvärderingsmått $RMSE$, MSE och MAE . Ett annat utvärderingsmått är det som benämns för determinationskoefficienten eller R^2 . Förklara vad det är för mått. Du kan läsa dokumentationen om det på nedanstående två länkar.
- https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html
 - https://scikit-learn.org/stable/modules/model_evaluation.html#r2-score

Koduppgifter

12. 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.

13. 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.
14. 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)
```

15. 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?
16. 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>

Kapitel 4 - Klassificering

Faktafrågor

1. Vad kännetecknar klassificeringsproblem? Ge några exempel på tillämpningsområden.
2. Förklara hur *OvR*- och *OvO*-algoritmerna fungerar.
3. Förklara följande utvärderingsmått:
 - a) *Confusion matrix*
 - b) *Accuracy*
 - c) *Precision*
 - d) *Recall*
 - e) *F1-score*
 - f) *ROC*-kurvan
4. Vad är *precision-recall tradeoff* för något?

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
 - b) *Support vector machines* - <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>
 - c) Beslutsträd - <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>
 - 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>
 - e) *Random forest* - <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
 - f) *Extra trees* - <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html>
6. Vad innebär det att vi kan kolla på *feature importance* med hjälp av trädmodeller såsom beslutsträd eller *random forest*?

Resonemangfrågor

7. Stina säger till Kalle på lunchsamtalet “jag vill ha högsta möjliga *precision* för vår klassificeringsmodell”. 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*?
8. Förklara hur man kan tolka figur 4.8 på sidan 175.

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

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))
```

12. Läs dokumentationen och visualisera ett beslutsträd: https://scikit-learn.org/stable/modules/generated/sklearn.tree.plot_tree.html
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 anställda har lämnat företaget och om den är 0 så betyder det att den anställda 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, där efter 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*. Exempel på hur denna typ av projekt kan se ut kan du se här: https://www.linkedin.com/posts/antonioirgomet_toppresultat-fr%C3%A5n-deltagarna-p%C3%A5-kursen-pythonprogrammering-activity-7292893718028718081-oAD0?utm_source=share&utm_medium=member_desktop&rcm=ACoAADH7YOkBJcbaNUfHDmlA4-6mf94mMwPgnRk

Kapitel 5 - Dimensionsreducering

Faktafrågor

1. Vad menas med *curse of dimensionality*.
2. Vad är dimensionsreducering och varför görs det?
3. Förklara översiktligt hur *PCA* fungerar. Använd figur 5.4 på sidan 224 i din förklaring.
4. Hur kan *kernel PCA* utvärderas?

Resonemangfrågor

5. Stina påstår att man i maskininlärning alltid vill ha modeller som genomför så bra prediktioner som möjligt. Kalle påstår att det inte riktigt stämmer eftersom tid också är en viktig aspekt. Både för själva modellträningen och för själva prediktionerna. Vad säger du?
6. Efter att vi genomfört en *PCA*, vad händer med tolkningen av variablerna?

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 nedanstående kod gör.

```
import numpy as np
from sklearn.decomposition import PCA

# Creating a dataset with 3 features/columns
X = np.random.rand(1000, 3)
print(X[0:5])
# Reducing the data to 2 dimensions
pca = PCA(n_components=2)
X2D = pca.fit_transform(X)
print(X2D[0:5])

# "Recreating" the data to 3 dimensions
X3D_inv = pca.inverse_transform(X2D)
# Not exactly equal since some information was lost in the
  ↪ transformation
print(np.allclose(X3D_inv, X))
```

9. Genomför en *PCA* på “car_price_dataset.csv” från kapitel 3 innan du modellerar det med ML. Hur påverkas resultatet?

Kapitel 6 - Klustering

Faktafrågor

1. Vad är klustering för något? Ge några exempel på tillämpningsområden.

2. Förklara översiktligt hur *K-means* fungerar. Använd figur 6.3 (sidan 238) och figur 6.4 (sidan 239) i din förklaring.

Resonemangfrågor

3. Hur kan man välja vilket antal kluster som ska användas för en K-means modell? Använd *inertia*, *silhouette score* och *silhouette diagram* i ditt svar.
4. Om du kollar på figur 6.10 på sidan 247, hur många kluster hade du valt och varför? Är det en “exakt vetenskap” att välja antalet kluster?
5. Hur tolkar man figur 6.13 på sidan 251?

Koduppgifter

6. 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.
7. I koden nedan använder vi datasetet “housing.csv” som användes i kapitel 2.

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.cluster import KMeans

df = pd.read_csv("housing.csv")
X = df.loc[:, ["median_income", "latitude", "longitude"]]
print(X.head())

kmeans = KMeans(n_clusters=6)
X["Cluster"] = kmeans.fit_predict(X)
X["Cluster"] = X["Cluster"].astype("category")
```

```
print(X.head())

sns.relplot(
    x="longitude", y="latitude", hue="Cluster", data=X,
    ↪ height=6,
)
```

- a) Förklara vad koden gör.
 - b) I koden har vi sex kluster (0-5). I verkligheten hade man kunnat undersöka respektive kluster för att se vilka preferenser de har, t.ex. hur de föredrar att bli kontaktade, vilka hustyper de gillar och dylikt. Om du hade lett en undersökning, vilken information hade du velat ha om respektive kluster och hur hade du designat marknadsundersökningen?
 - c) Det som tagits fram, hur kan det vara användbart i verkligheten för ett företag exempelvis?
 - d) Vi har gjort ett enkelt kodexempel, lek gärna runt och fortsätt utveckla det. Exempelvis kan du undersöka hur många kluster som vore bra att använda.
8. Genomför en klusteranalys på datasetet "hr_employee_data.xlsx". Notera, i kapitel 4, uppgift 11, så använde vi `left` som den beroende variabeln y . Eftersom vi nu genomför en klusteranalys så modellerar vi inte med avseende på y utan nu betraktar vi alla variabler som x . Fundera på vilka variabler du vill använda och hur det du skapar hade kunnat användas i verkligheten.
 9. Genomför en klustringanalys på datasetet som du hittar här: <https://www.kaggle.com/datasets/thebumpkin/1980s-classic-hits-with-spotify-data>

Kapitel 7 - Artificiella neurala nätverk (ANN)

Faktafrågor

1. Vad har ANN modellerna inspirerats av?
2. Vad refererar “djup” till i begreppet “djupinlärning”?
3. Förklara vad som händer i figur 7.2, 7.3, figur 7.4 och figur 7.5. Varför används aktiveringsfunktioner?
4. Har neurala nätverk få eller många parametrar?
5. Förklara intuitivt hur *dropout*-regularisering fungerar.

Resonemangfrågor

6. Din kollega ber dig förklara tabell 7.1 och tabell 7.2. Gör det!
7. Experimentera med neurala nätverk på följande länk: <https://playground.tensorflow.org/>
8. Förklara översiktligt hur *backpropagation* fungerar. Använd figur 7.7 till din hjälp.

Koduppgifter

9. 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.
10. I denna uppgift ska vi använda MNIST-datan som vi arbetat med tidigare. Detta för att ha ett bekant dataset att arbeta med och experimentera med ANN-modeller.
 - a) Träna en ANN-modell på MNIST-datan. Vad får du för resultat?

- b) Prova justera hyperparametrarna med *KerasTuner*. Notera, du har som vanligt dokumentationen till din hjälp. Får du bättre resultat?

Kapitel 8 - *Convolutional neural network* (CNN)

Faktafrågor

1. Vad krävs det för att ett neuralt nätverk ska klassas som CNN?
2. Inom vilket tillämpningsområde är CNN generellt sett en väldigt kraftfull modell?
3. Vad menas med RGB?
4. Vad är *data augmentation*?

Resonemangfrågor

5. Förklara översiktligt hur CNN fungerar. Använd begreppen *convolution layer* och *pooling layer* i ditt svar.
6. Din kollega ber dig förklara figur 8.4. Gör det!

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. I denna uppgift arbetar vi med CIFAR-100 datasetet som gicks igenom i kodexempel 1.
 - a) Skapa en CNN-modell för att prediktera datasetet.
 - b) Om du justerar hyperparametrar med *KerasTuner*, får du bättre resultat?

- c) Prova använd *transfer learning* för att genomföra prediktioner, får du bättre resultat?
9. I denna uppgift utgår vi ifrån kodexempel 2 i detta kapitel.
- a) Ta egna bilder som du predikterar med en förtränad modell.
 - b) Bygg en applikation (med exempelvis *Streamlit*) som använder en förtränad modell för att prediktera bilder. Hur du designar applikationen och vilken funktionalitet du inkluderar väljer du själv.
10. Använd dig utav CNN för att skapa en modell som kan prediktera teckenspråk. Datan finns tillgänglig här: <https://www.kaggle.com/datasets/datamunge/sign-language-mnist>

Kapitel 9 - *Recurrent neural network* (RNN)

Faktafrågor

- 1. Rent generellt, vad är RNN-modeller designade för att användas till?
- 2. Vad heter två populära modellarkitekturer inom RNN?
- 3. Vad står NLP för? Vad är *embeddings*?

Resonemangfrågor

- 4. Din kollega ber dig förklara figur 9.2. Gör det!

Koduppgifter

5. 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.
6. Vill du fördjupa dig inom RNN, gå till Kaggle, <https://www.kaggle.com/datasets>, och hitta ett lämpligt dataset att arbeta med.

Kapitel 10 - Chattbottar

Faktafrågor

1. Vad är *prompt engineering*?
2. Vad är *RAG*?
3. Vad är *chunking* och *embeddings*?

Resonemangfrågor

4. Förklara översiktligt hur man kan evaluera en chattbot.
5. Din kollega frågar dig vad ELIZA och Turingtestet är. Läs nedanstående två länkar innan du besvarar frågan.
 - <https://en.wikipedia.org/wiki/ELIZA>
 - <https://sv.wikipedia.org/wiki/Turingtestet>
6. Ge konkreta exempel på hur olika företag och organisationer kan ta nytta av chattbottar samt vilka risker som finns.

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. Skapa en chattbot som svarar på frågor utifrån något dokument som du själv väljer.