

Machine learning for begyndere

FOR IDA D. 16/04/2023

Omkring mig

- Anders Bensen Ottsen
- Diplomingeniør i Softwareteknologis
 - Fra Syddansk Universitet
- Læst en kandidat i Computer Science & Engineering
 - Fra Danmarks Tekniske Universitet
 - Skrevet speciale i ML & ansigtsgenkendelse
- Arbejder som Data Scientist
- Har undervist Python & ML kurser for IDA 14+ gange

Omkring Jens

- Jens Kristian Vitus Bering
 - Fra IDA
- BSc. I Software Engineering
 - Fra Syddansk Universitet
- Læser cand.polyt i Software Engineering
 - Ved Syddansk Universitet
- Arbejder som studenterudvikler ved Bankdata
 - Hvor han programmerer finans software

Dagens program

- Machine learning 101
- Data prep i Python miljøet
- Regression
- Klassifikation
- Avancerede emner
- Spørgerunde

Efter i dag kan I:

- Forstå hvad Machine Learning kan og dets koncepter
- Forstå konkrete ML modeller på et abstrakt niveau
 - Både klassifikations + regressions algoritmer
- Implementere mindre ML programmer vha. biblioteker
- Så hvordan når vi der til?
 - Ved mindre ”forelæsninger”
 - Med liveprogrammering
 - Opgaveløsning

Efter i dag kan I:

- Forstå hvad Machine Learning kan og dets koncepter
- Forstå konkrete ML modeller på et abstrakt niveau
 - Både klassifikations + regressions algoritmer
- Implementere mindre ML programmer vha. biblioteker
- Så hvordan når vi der til?
 - Ved mindre ”forelæsninger”
 - Med liveprogrammering
 - Opgaveløsning

Formatet i dag

- Webinar, zoom, breakout rooms
- Mange intro begreber til at starte med
- Herefter en praktisk "forelæsning" i et emne i ~ 20 min
 - Med live coding undervejs
 - Lad vær med at skrives efter mig, fokuser på forståelse
- Efter forelæsning laver i opgaver i ~ 30 minutter
 - Hvor Jens & jeg kommer rundt og hjælper
 - Og sådan fortsætter vi de næste ~ 3 timer
- Links:
 - **ZIP HERE TODO**

Machine learning 101

Hvad er AI

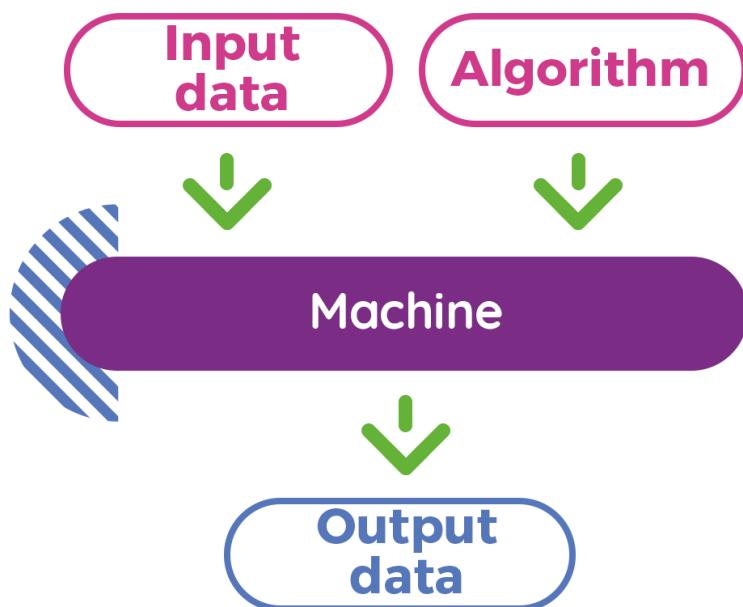
- Artificial intelligence (AI)
 - Et program der opfører sig intelligent
 - Og hvordan definerer man intelligens?
 - Det er utroligt svært
- AI er ikke det samme som Machine Learning (ML)!
 - ML er en ”undergenre” af AI
 - ML er dog den del af AI der har vist mest potentiiale
 - F.eks. – ChatGPT, StableDiffusion, AlphaGO etc.

Machine learning

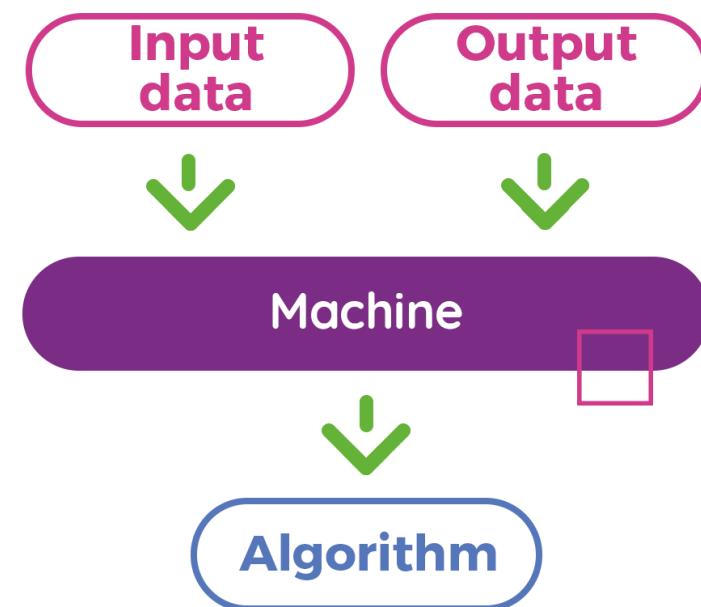
- ”Studiet af algoritmer der automatisk forbedrer sig igennem erfaring og data”
 - Meget formelt
- Vi laver programmer som finder mønstre i data
 - Mere praktisk

Machine learning

TRADITIONAL PROGRAMMING



MACHINE LEARNING



Machine learning algoritmer

- Der er mange forskellige algoritmer (kaldes også modeller når vi bruger dem)
 - Lineær regression
 - Logistisk regression
 - Decision tree
 - Naive bayes
 - K nearest neighbour (KNN)
 - Neurale netværk
 - Osv.
- Nogle er bedre til nogle ting end andre

Machine learning algoritmer

- Der er mange forskellige algoritmer (kaldes også modeller når vi bruger dem)
 - Lineær regression
 - Logistisk regression
 - Decision tree
- Naive bayes
- K nearest neighbour (KNN)
- Neurale netværk
- Osv.
- Nogle er bedre til nogle ting end andre

Dataens rolle i machine learning

- Machine learning tager udgangspunkt i et datasæt
 - Som man vil have at vores ***model*** lærer fra
 - F.eks. – Forudse om en person får en hjertesygdom ud fra sundhedsdata
- Modellen ***træner*** på dataen, for at kunne finde mønstre i den her data
 - Efter den har trænet færdig kan vi bruge modellen til at forudse nye ting
- Vi kalder den data vi bruger til at forudse noget for **X** (en matrice/tabel)
- Det data vi gerne vil forudse kalder vi for **y** (en vektor/liste)

Supervised vs Unsupervised learning

- Supervised learning
 - Vi har allerede svaret i vores datasæt
 - Vi vil gerne have en algoritme til at lære sammenhængen
 - Vores algoritme forbedrer sig ved at træne ud fra X og y
 - Splittes op i Regression / Klassifikation
- Unsupervised learning
 - Vi har ikke svaret i vores datasæt!
 - Men vil gerne have at en algoritme finder en sammenhæng alligevel
 - Vi har kun X , intet y vi gerne vil forudse
- I dag fokuserer vi kun på supervised learning!

Iris datasættet

- Et datasæt ofte bruge i ML
- Lavet af en biolog i ~ 1936
- Består af 3 forskellige slags blomster
 - Hvor der er målt på 50 af hver
- Bruges til at finde mønstre i typerne

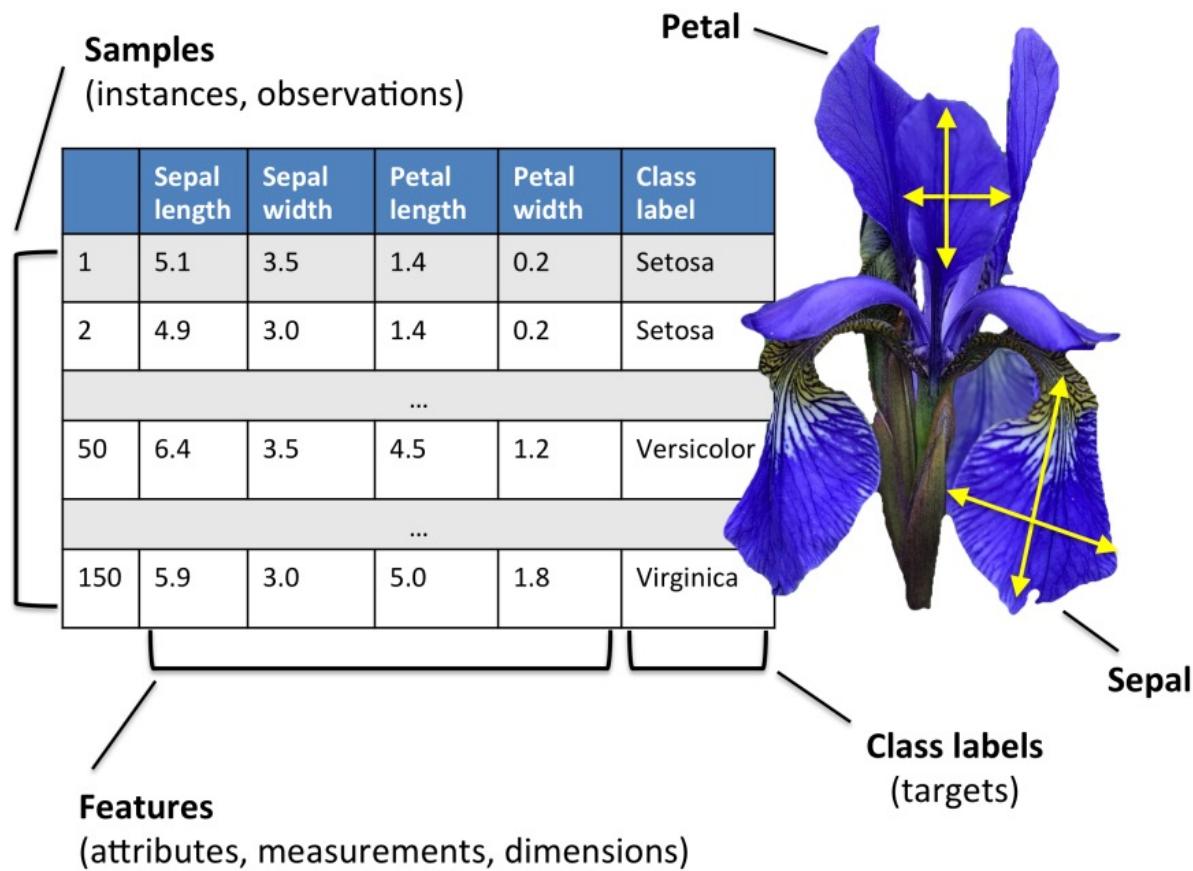


Iris Versicolor

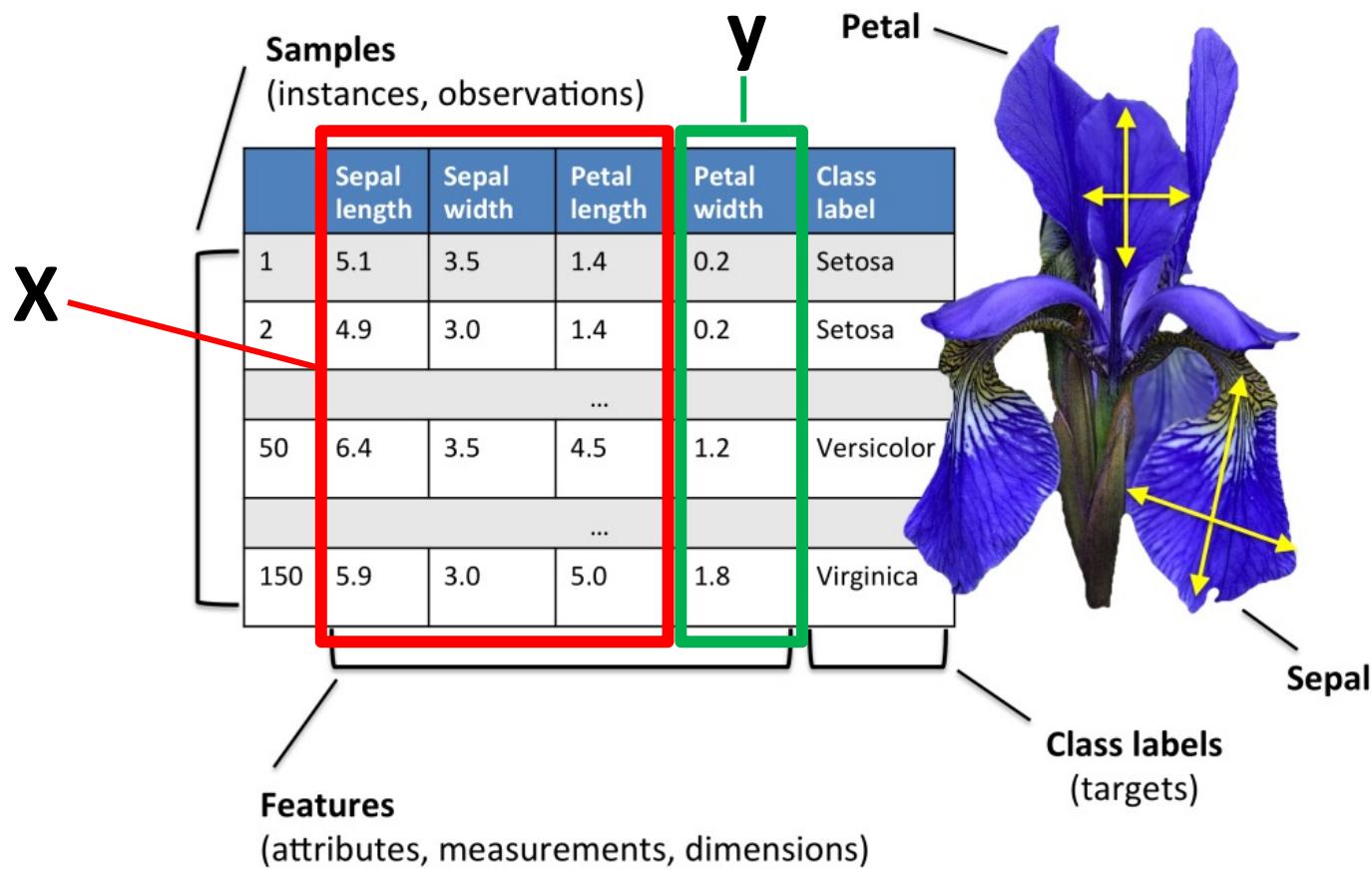
Iris Setosa

Iris Virginica

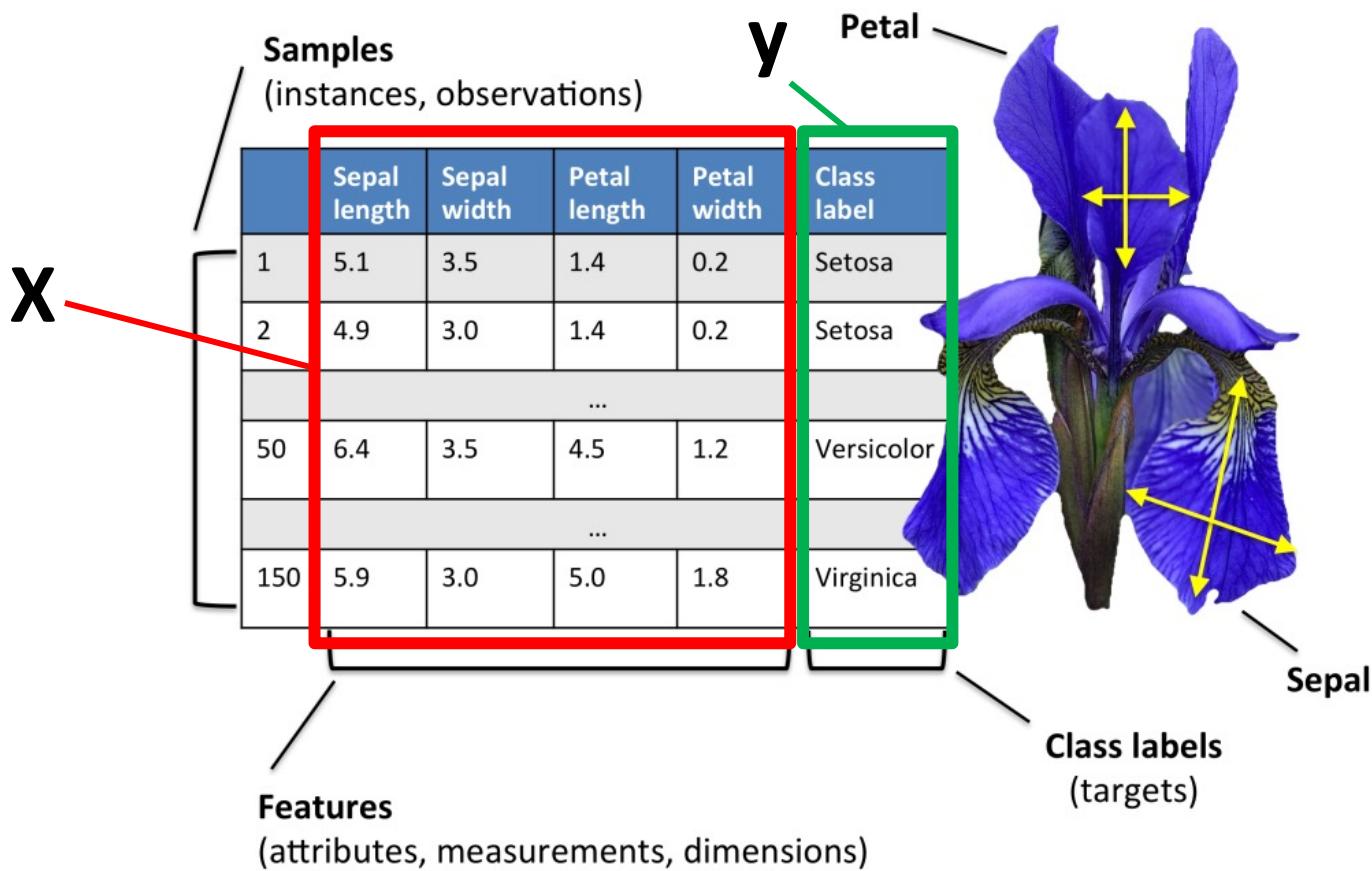
Iris datasættet



Iris datasættet regression



Iris datasættet klassifikation



Prep af data i Python

Pandas

- Et bibliotek lavet til nemt at bearbejde data i Python
- Pandas kan:
 - Nemt hente data ind i Python
 - Modellere data
 - Analysere data
 - Plotte data
- Er næsten blevet de facto standarden til at analysere data i Python

Pandas

- Når man henter data ind igennem Pandas får man en dataframe(df)
 - Hvilket er et slags objekt, der har vildt mange metoder
 - En dataframe minder meget om en tabel i excel
 - Man bearbejder data i den her dataframe før vi bruger ML

Pandas metoder 1

```
1 import pandas as pd
2
3 df = pd.read_csv("iris.data", sep=",")
4
5 len(df)
6
7 df.columns
8
9 df.describe()
10
11 df['type']
12
13 df[['sepal_length', 'sepal_width']]
```

Pandas metoder 1

```
1 import pandas as pd
2
3 df = pd.read_csv("iris.data", sep=",")
4
5 len(df)
6
7 df.columns
8
9 df.describe()
10
11 df['type']
12
13 df[['sepal_length', 'sepal_width']]
```

Importer Pandas

Pandas metoder 1

```
1 import pandas as pd
2
3 df = pd.read_csv("iris.data", sep=",")
4
5 len(df)
6
7 df.columns
8
9 df.describe()
10
11 df['type']
12
13 df[['sepal_length', 'sepal_width']]
```

Importer Pandas

Hent vores data (der er komma separeret!) som en dataframe (df)

Pandas metoder 1

```
1 import pandas as pd
2
3 df = pd.read_csv("iris.data", sep=",")
4
5 len(df)
6
7 df.columns
8
9 df.describe()
10
11 df['type']
12
13 df[['sepal_length', 'sepal_width']]
```

Importer Pandas

Hent vores data (der er komma separeret!) som en dataframe (df)

Hvor mange rækker er der i vores df?

Pandas metoder 1

```
1 import pandas as pd
2
3 df = pd.read_csv("iris.data", sep=",")
4
5 len(df)
6
7 df.columns
8
9 df.describe()
10
11 df['type']
12
13 df[['sepal_length', 'sepal_width']]
```

Importer Pandas

Hent vores data (der er komma separeret!) som en dataframe (df)

Hvor mange rækker er der i vores df?

Hvilke kolonner er der i vores data?

Pandas metoder 1

```
1 import pandas as pd
2
3 df = pd.read_csv("iris.data", sep=",")
4
5 len(df)
6
7 df.columns
8
9 df.describe()
10
11 df['type']
12
13 df[['sepal_length', 'sepal_width']]
```

Importer Pandas

Hent vores data (der er komma separeret!) som en dataframe (df)

Hvor mange rækker er der i vores df?

Hvilke kolonner er der i vores data?

Hvad er den basale statistik for vores data?

Pandas metoder 1

```
1 import pandas as pd  
2  
3 df = pd.read_csv("iris.data", sep=",")  
4  
5 len(df)  
6  
7 df.columns  
8  
9 df.describe()  
10  
11 df['type']  
12  
13 df[['sepal_length', 'sepal_width']]
```

Importer Pandas

Hent vores data (der er komma separeret!) som en dataframe (df)

Hvor mange rækker er der i vores df?

Hvilke kolonner er der i vores data?

Hvad er den basale statistik for vores data?

Returnerer udelukkende 'type' kolonnen som en ny df

Pandas metoder 1

1	import pandas as pd	Importerer Pandas
2		
3	df = pd.read_csv("iris.data", sep=",")	Hent vores data (der er komma separeret!) som en dataframe (df)
4		
5	len(df)	Hvor mange rækker er der i vores df?
6		
7	df.columns	Hvilke kolonner er der i vores data?
8		
9	df.describe()	Hvad er den basale statistik for vores data?
10		
11	df['type']	Returnerer udelukkende 'type' kolonnen som en ny df
12		
13	df[['sepal_length', 'sepal_width']]	Returnerer både 'sepal_length' og 'sepal_width' kolonnerne som en ny df

Pandas metoder 1

1 import pandas as pd	Importerer Pandas
2	
3 df = pd.read_csv("iris.data", sep=",")	Hent vores data (der er komma separeret!) som en dataframe (df)
4	
5 len(df)	Hvor mange rækker er der i vores df?
6	
7 df.columns	Hvilke kolonner er der i vores data?
8	
9 df.describe()	Hvad er den basale statistik for vores data?
10	
11 df['type']	Returnerer udelukkende 'type' kolonnen som en ny df
12	
13 df[['sepal_length', 'sepal_width']]	Returnerer både 'sepal_length' og 'sepal_width' kolonnerne som en ny df

- Live!

Pandas metoder 2

```
1 import pandas as pd
2
3 df = pd.read_csv("iris.data", sep=",")
4
5 df['type'].unique()
6
7 df['type'].value_counts()
8
9 df = df.drop('type', axis=1)
10
11 df['large_sepals_length'] = [1 if x >= 5.7 else 0 for x in df['sepals_length']]
```

Pandas metoder 2

```
1 import pandas as pd
2
3 df = pd.read_csv("iris.data", sep=",")
4
5 df['type'].unique()                                     Hvilke forskellige slags er der i 'type'?
6
7 df['type'].value_counts()
8
9 df = df.drop('type', axis=1)
10
11 df['large_sepallength'] = [1 if x >= 5.7 else 0 for x in df['sepallength']]
```

Pandas metoder 2

```
1 import pandas as pd
2
3 df = pd.read_csv("iris.data", sep=",")
4
5 df['type'].unique()                                Hvilke forskellige slags er der i 'type'?
6
7 df['type'].value_counts()                         Hvor mange er der af hver slags i
8
9 df = df.drop('type', axis=1)
10
11 df['large_sepallength'] = [1 if x >= 5.7 else 0 for x in df['sepallength']]
```

Hvilke forskellige slags er der i 'type'?

Hvor mange er der af hver slags i kolonnen 'type'?

Pandas metoder 2

```
1 import pandas as pd
2
3 df = pd.read_csv("iris.data", sep=",")
4
5 df['type'].unique()                                Hvilke forskellige slags er der i 'type'?
6
7 df['type'].value_counts()                          Hvor mange er der af hver slags i
8
9 df = df.drop('type', axis=1)                      Slet kolonnen (derfor axis=1) 'type'
10
11 df['large_sepal_length'] = [1 if x >= 5.7 else 0 for x in df['sepal_length']]
```

Pandas metoder 2

```
1 import pandas as pd
2
3 df = pd.read_csv("iris.data", sep=",")
4
5 df['type'].unique()                                Hvilke forskellige slags er der i 'type'?
6
7 df['type'].value_counts()                          Hvor mange er der af hver slags i
8
9 df = df.drop('type', axis=1)                      Slet kolonnen (derfor axis=1) 'type'
10
11 df['large_sepal_length'] = [1 if x >= 5.7 else 0 for x in df['sepal_length']]
```

Lav en ny kolonne baseret på en anden kolonne

Pandas metoder 2

```
1 import pandas as pd
2
3 df = pd.read_csv("iris.data", sep=",")
4
5 df['type'].unique()                                Hvilke forskellige slags er der i 'type'?
6
7 df['type'].value_counts()                          Hvor mange er der af hver slags i
8
9 df = df.drop('type', axis=1)                      Slet kolonnen (derfor axis=1) 'type'
10
11 df['large_sepal_length'] = [1 if x >= 5.7 else 0 for x in df['sepal_length']]
```

- Live!

Lav en ny kolonne baseret på en anden kolonne

One-hot-encoding

- Nogle gange er vores data i kategorier
 - F.eks. Kan der være en kolonne med hvilket land folk kommer fra
 - Danmark, Sverige, Polen...
 - Det kan vores ML modeller ikke bruge til noget!
 - Hvordan kunne vi fikse det dilemma?
 - Vi kunne lave enhver kategori om til et tal, f.eks. Danmark er 0, Sverige er 1, Polen er 2 ...
 - Men hvad nu hvis vores model lærer en sammenhæng mellem at 2 er større end 0?
 - Det et problem!
- One-hot-encoding er svaret på alle vores bønner
 - Vi laver en kolonne for hvert land, og giver et 1 hvis personen er fra det land, og 0 ellers

One-hot-encoding af iris

sepal_length	sepal_width	petal_length	petal_width	type
6.5	3.0	5.5	1.8	Iris-virginica
6.3	2.5	4.9	1.5	Iris-versicolor
5.8	2.8	5.1	2.4	Iris-virginica
5.2	3.4	1.4	0.2	Iris-setosa
6.3	2.9	5.6	1.8	Iris-virginica
6.4	3.1	5.5	1.8	Iris-virginica
5.7	2.8	4.5	1.3	Iris-versicolor
4.8	3.0	1.4	0.1	Iris-setosa
6.1	3.0	4.6	1.4	Iris-versicolor
6.4	2.8	5.6	2.2	Iris-virginica
4.5	2.3	1.3	0.3	Iris-setosa
7.0	3.2	4.7	1.4	Iris-versicolor
4.8	3.4	1.9	0.2	Iris-setosa
5.0	2.3	3.3	1.0	Iris-versicolor
4.8	3.4	1.6	0.2	Iris-setosa
6.4	2.8	5.6	2.1	Iris-virginica
4.3	3.0	1.1	0.1	Iris-setosa
6.0	3.0	4.8	1.8	Iris-virginica
6.1	2.6	5.6	1.4	Iris-virginica

```
1 import pandas as pd  
2  
3 df = pd.read_csv("iris.data")
```

One-hot-encoding af iris

sepal_length	sepal_width	petal_length	petal_width	type	Iris-setosa	Iris-versicolor	Iris-virginica
6.5	3.0	5.5	1.8	Iris-virginica	0	0	1
6.3	2.5	4.9	1.5	Iris-versicolor	0	1	0
5.8	2.8	5.1	2.4	Iris-virginica	0	0	1
5.2	3.4	1.4	0.2	Iris-setosa	1	0	0
6.3	2.9	5.6	1.8	Iris-virginica	0	0	1
6.4	3.1	5.5	1.8	Iris-virginica	0	0	1
5.7	2.8	4.5	1.3	Iris-versicolor	0	1	0
4.8	3.0	1.4	0.1	Iris-setosa	1	0	0
6.1	3.0	4.6	1.4	Iris-versicolor	0	1	0
6.4	2.8	5.6	2.2	Iris-virginica	0	0	1
4.5	2.3	1.3	0.3	Iris-setosa	1	0	0
7.0	3.2	4.7	1.4	Iris-versicolor	0	1	0
4.8	3.4	1.9	0.2	Iris-setosa	1	0	0
5.0	2.3	3.3	1.0	Iris-versicolor	0	1	0
4.8	3.4	1.6	0.2	Iris-setosa	1	0	0
6.4	2.8	5.6	2.1	Iris-virginica	0	0	1
4.3	3.0	1.1	0.1	Iris-setosa	1	0	0
6.0	3.0	4.8	1.8	Iris-virginica	0	0	1
6.1	2.6	5.6	1.4	Iris-virginica	0	0	1

```
1 import pandas as pd
2
3 df = pd.read_csv("iris.data")
4
5 dummies = pd.get_dummies(df['type'])
6
7 df = df.join(dummies)
```

One-hot-encoding af iris

sepal_length	sepal_width	petal_length	petal_width	Iris-setosa	Iris-versicolor	Iris-virginica
6.5	3.0	5.5	1.8	0	0	1
6.3	2.5	4.9	1.5	0	1	0
5.8	2.8	5.1	2.4	0	0	1
5.2	3.4	1.4	0.2	1	0	0
6.3	2.9	5.6	1.8	0	0	1
6.4	3.1	5.5	1.8	0	0	1
5.7	2.8	4.5	1.3	0	1	0
4.8	3.0	1.4	0.1	1	0	0
6.1	3.0	4.6	1.4	0	1	0
6.4	2.8	5.6	2.2	0	0	1
4.5	2.3	1.3	0.3	1	0	0
7.0	3.2	4.7	1.4	0	1	0
4.8	3.4	1.9	0.2	1	0	0
5.0	2.3	3.3	1.0	0	1	0
4.8	3.4	1.6	0.2	1	0	0
6.4	2.8	5.6	2.1	0	0	1
4.3	3.0	1.1	0.1	1	0	0
6.0	3.0	4.8	1.8	0	0	1
6.1	2.6	5.6	1.4	0	0	1

```
1 import pandas as pd
2
3 df = pd.read_csv("iris.data")
4
5 dummies = pd.get_dummies(df['type'])
6
7 df = df.join(dummies)
8
9 df = df.drop('type', axis=1)
```

Matplotlib

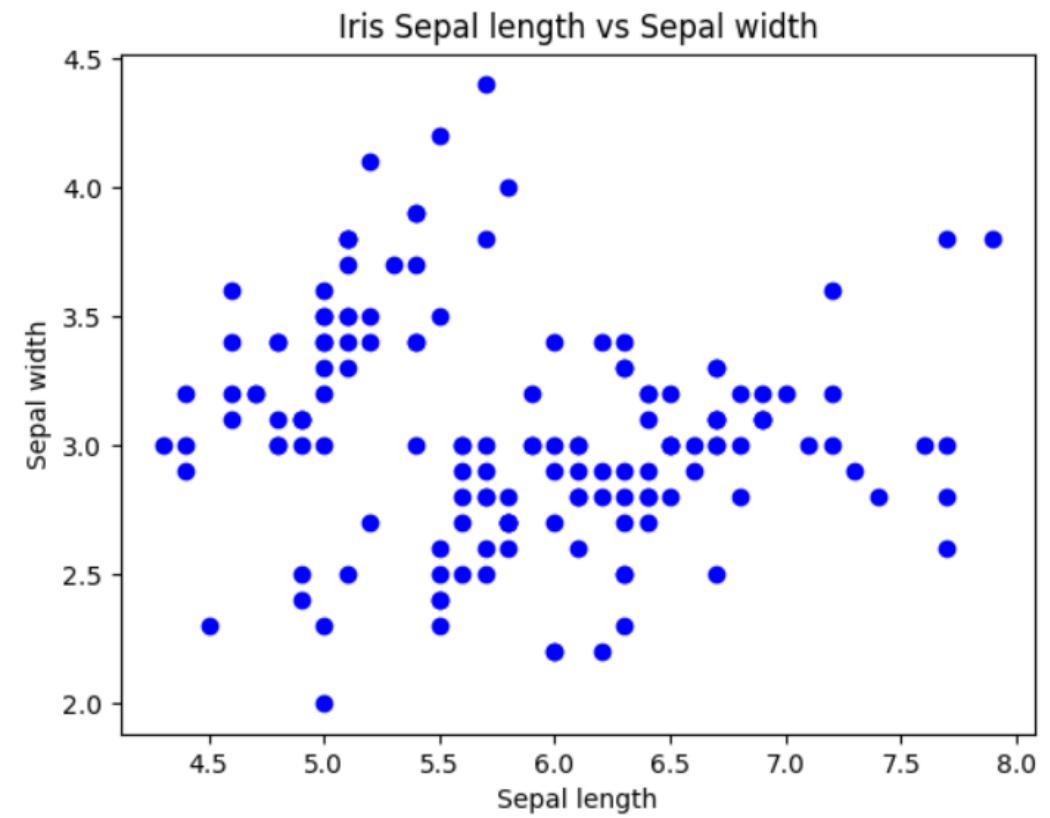
- Et bibliotek der bliver brugt til at plotte sin data
- Indeholder vildt mange forskellige metoder til at visualisere data
 - Histogram, 2d scatter, 3d scatter, scatter matrix, logaritmisk osv...
- Virker sammen med Pandas!

Matplotlib

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 df = pd.read_csv("iris.data", sep=",")
5
6 plt.plot(df['sepal_length'], df['sepal_width'], 'bo')
7 plt.xlabel("Sepal length")
8 plt.ylabel("Sepal width")
9 plt.title("Iris Sepal length vs Sepal width")
10
11 plt.show()
```

Matplotlib

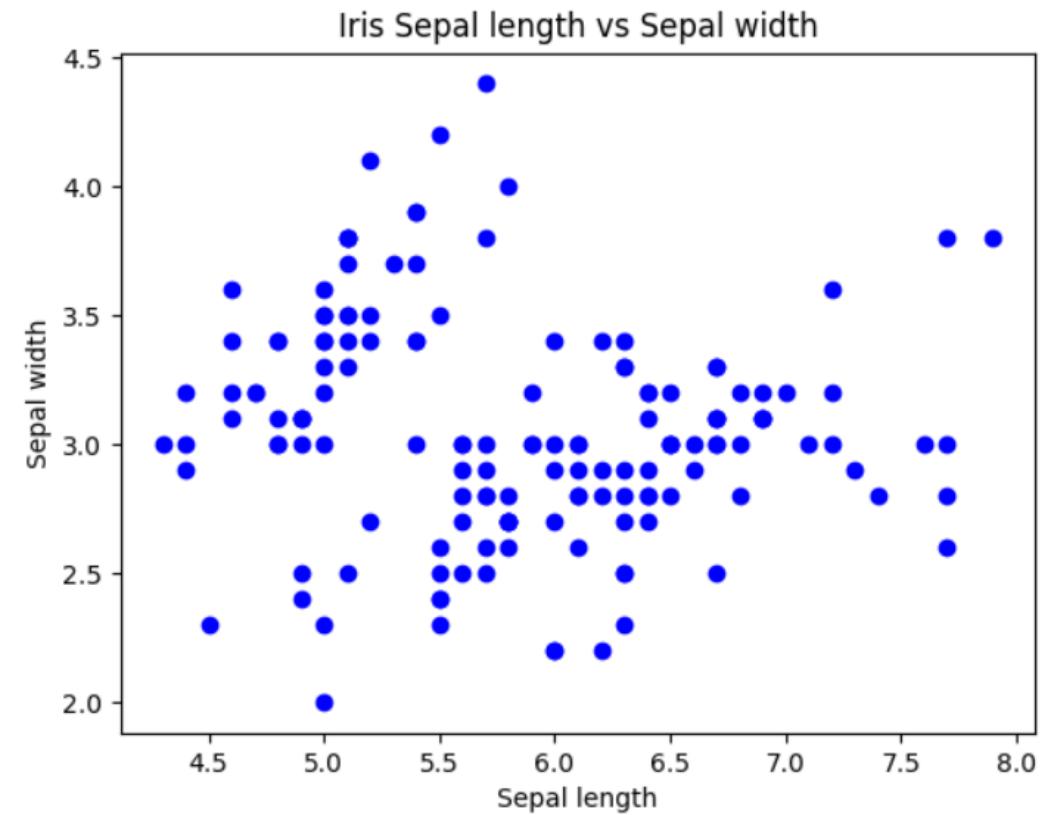
```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 df = pd.read_csv("iris.data", sep=",")
5
6 plt.plot(df['sepal_length'], df['sepal_width'], 'bo')
7 plt.xlabel("Sepal length")
8 plt.ylabel("Sepal width")
9 plt.title("Iris Sepal length vs Sepal width")
10
11 plt.show()
```



Matplotlib

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 df = pd.read_csv("iris.data", sep=",")
5
6 plt.plot(df['sepal_length'], df['sepal_width'], 'bo')
7 plt.xlabel("Sepal length")
8 plt.ylabel("Sepal width")
9 plt.title("Iris Sepal length vs Sepal width")
10
11 plt.show()
```

- Live!

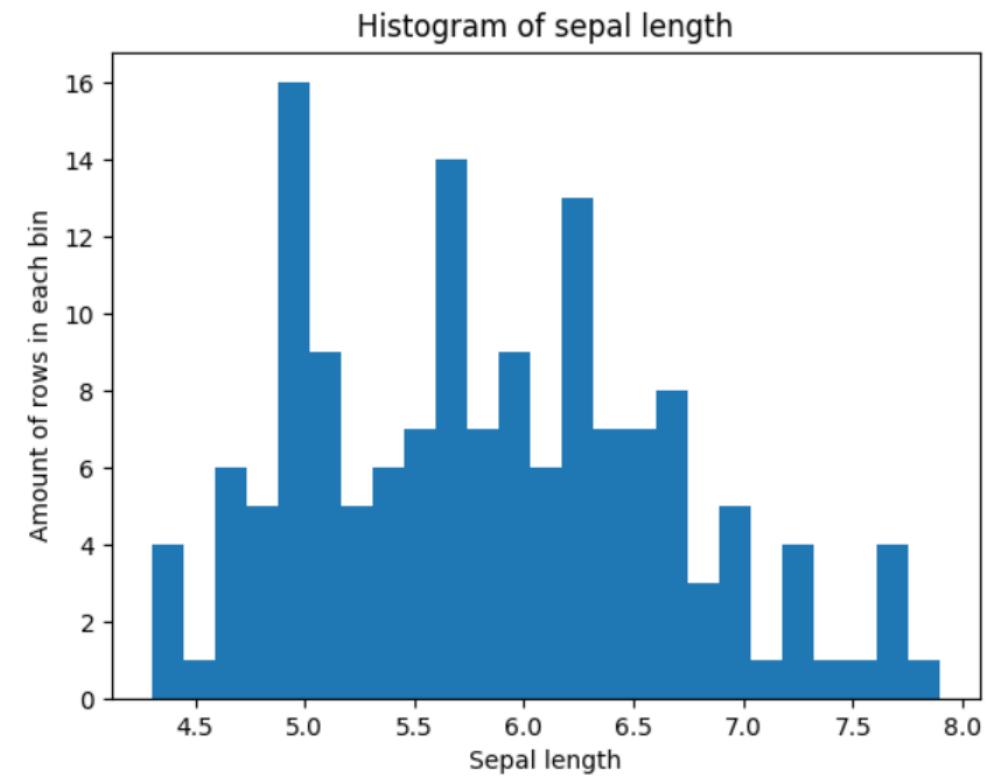


Matplotlib

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 df = pd.read_csv("iris.data", sep=",")
5
6 plt.hist(df['sepal_length'], bins=25)
7 plt.xlabel("Sepal length")
8 plt.ylabel("Amount of rows in each bin")
9 plt.title("Histogram of sepal length")
10
11 plt.show()
```

Matplotlib

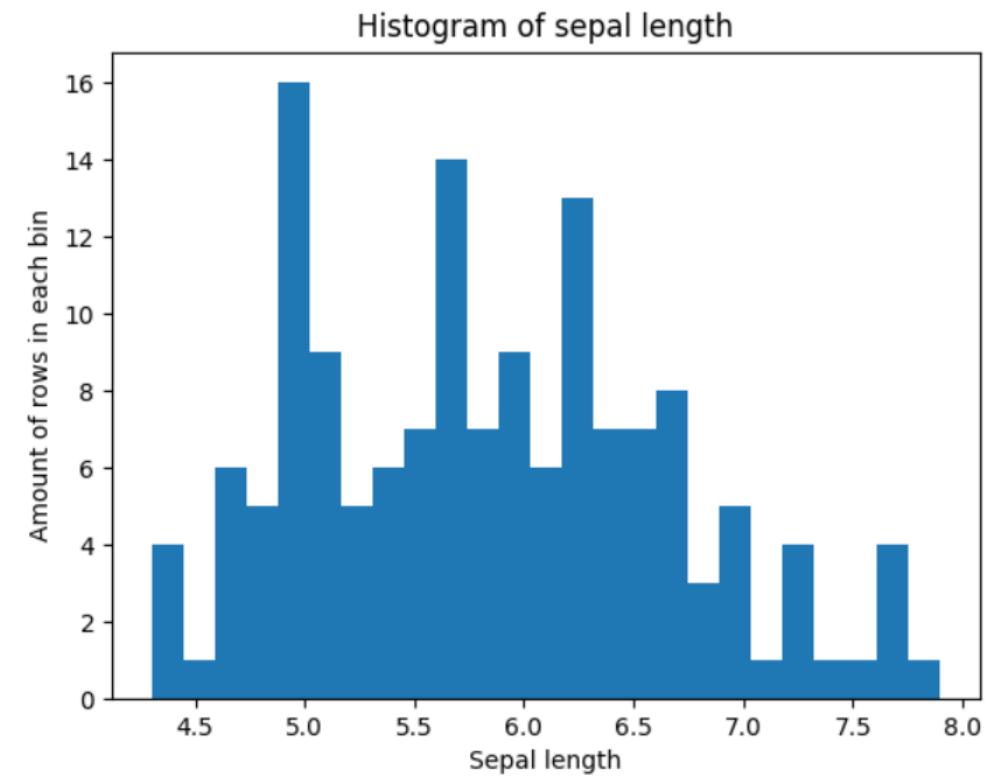
```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 df = pd.read_csv("iris.data", sep=",")
5
6 plt.hist(df['sepal_length'], bins=25)
7 plt.xlabel("Sepal length")
8 plt.ylabel("Amount of rows in each bin")
9 plt.title("Histogram of sepal length")
10
11 plt.show()
```



Matplotlib

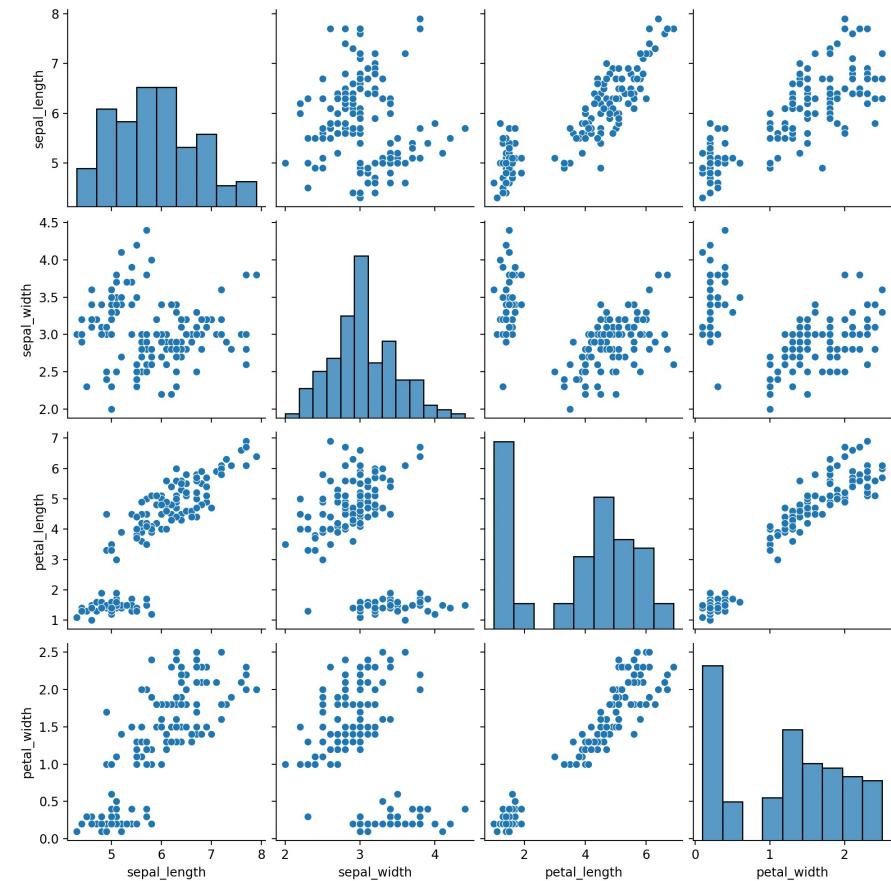
```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 df = pd.read_csv("iris.data", sep=",")
5
6 plt.hist(df['sepal_length'], bins=25)
7 plt.xlabel("Sepal length")
8 plt.ylabel("Amount of rows in each bin")
9 plt.title("Histogram of sepal length")
10
11 plt.show()
```

- Live!



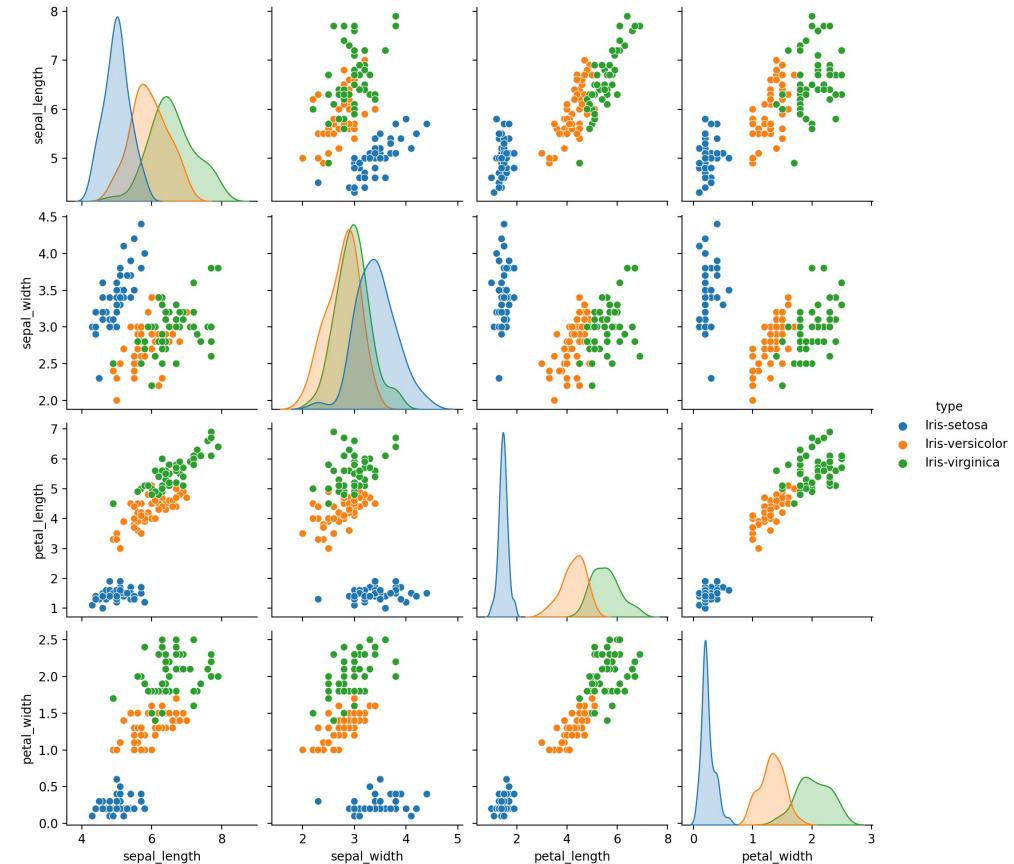
Seaborn (bygger oven på Matplotlib)

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 df = pd.read_csv("iris.data", sep=",")
6
7 sns.pairplot(df)
8
9 plt.show()
```



Seaborn (bygger oven på Matplotlib)

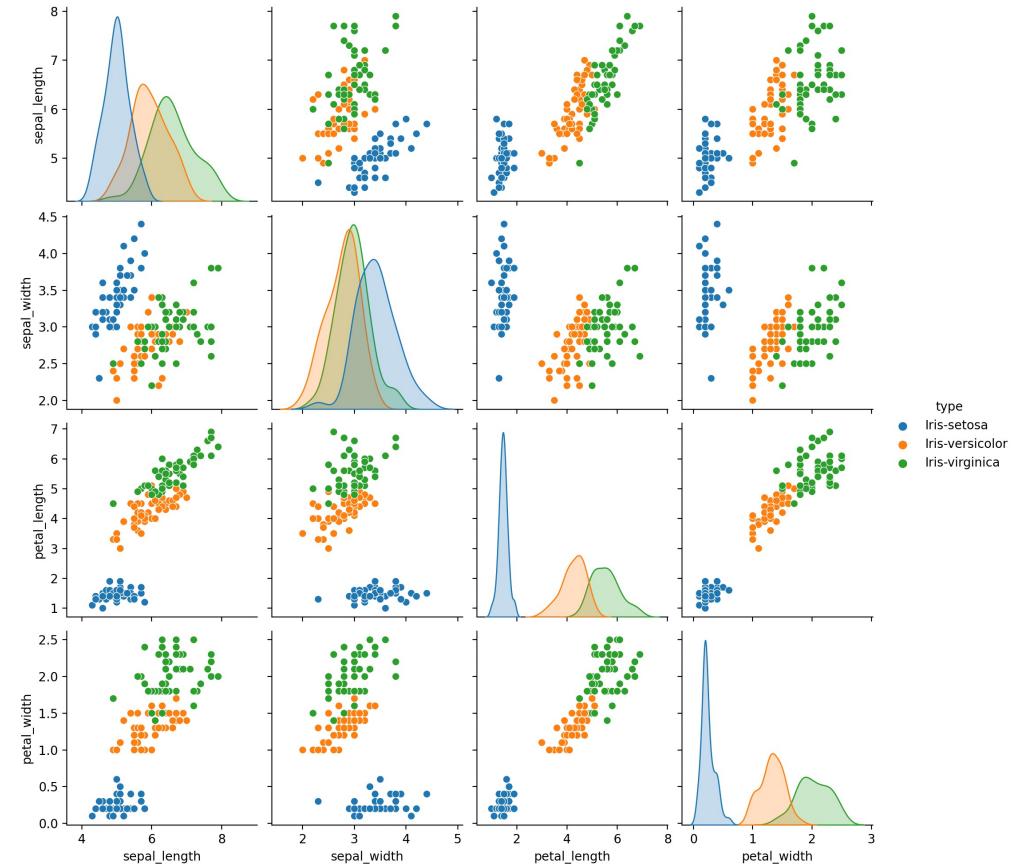
```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 df = pd.read_csv("iris.data", sep=",")
6
7 sns.pairplot(df, hue='type')
8
9 plt.show()
```



Seaborn (bygger oven på Matplotlib)

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 df = pd.read_csv("iris.data", sep=",")
6
7 sns.pairplot(df, hue='type')
8
9 plt.show()
```

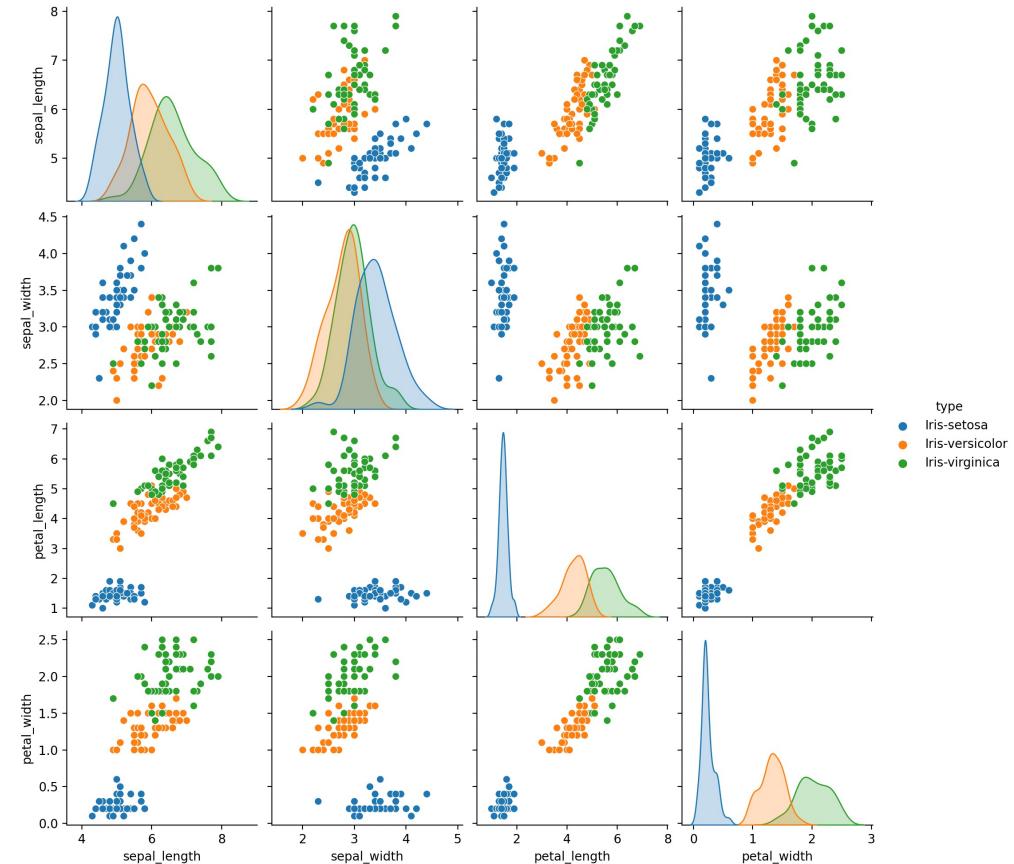
- Så man kan give en "klasse" med
 - Det vi kalder *y* i klassifikation



Seaborn (bygger oven på Matplotlib)

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 df = pd.read_csv("iris.data", sep=",")
6
7 sns.pairplot(df, hue='type')
8
9 plt.show()
```

- Så man kan give en "klasse" med
 - Det vi kalder *y* i klassifikation
- Live!



Opgaver pt. 1 (40 minutter)

1. Start med at downloade alle filer inde i den her mappe: TODO

- Når i har hentet hele mappen skal i bruge pip til at installere alle bibliotekerne i *requirements.txt*
 - "pip install -r requirements.txt"

2. Brug pandas til at hente *winequality-red.csv* dataen ind. Skim den tilhørende *winequality-red.names* fil og forstå hvad dataen er for noget. OBS: dataen er separeret med ";"

- Analyser dataen: antal rækker, kolonner, evt. statistikker osv.
- Visualiser dataen og prøv at se om nogle attributter "hænger sammen"
 - Hint: individuelle plots eller seaborn og pairplot. I kan evt give height=1.1 med som argument til pairplot.
- Først skal i tilføje en ny kolonne til dataframen: *good_quality*, som er lavet ud fra om en vin har en kvalitet på 6 eller over (hvis den har får den tallet 1 (fordi den er god)) og ellers får den et 0 (fordi den er dårlig)
 - Hint: Brug *quality* kolonnen og se hvordan jeg gjorde noget lignende på "Pandas metoder 2" (linje 11)
- Brug nu *.drop* metoden til at fjerne *quality* kolonnen, da vi ikke længere skal bruge den.
 - I kan nu bruge pairplot og give *quality_good* med som "klasse"

3. Brug pandas til at hente *sa_heart.csv* dataen ind. Skim den tilhørende *sa_heart.names* fil og forstå hvad dataen er for noget.

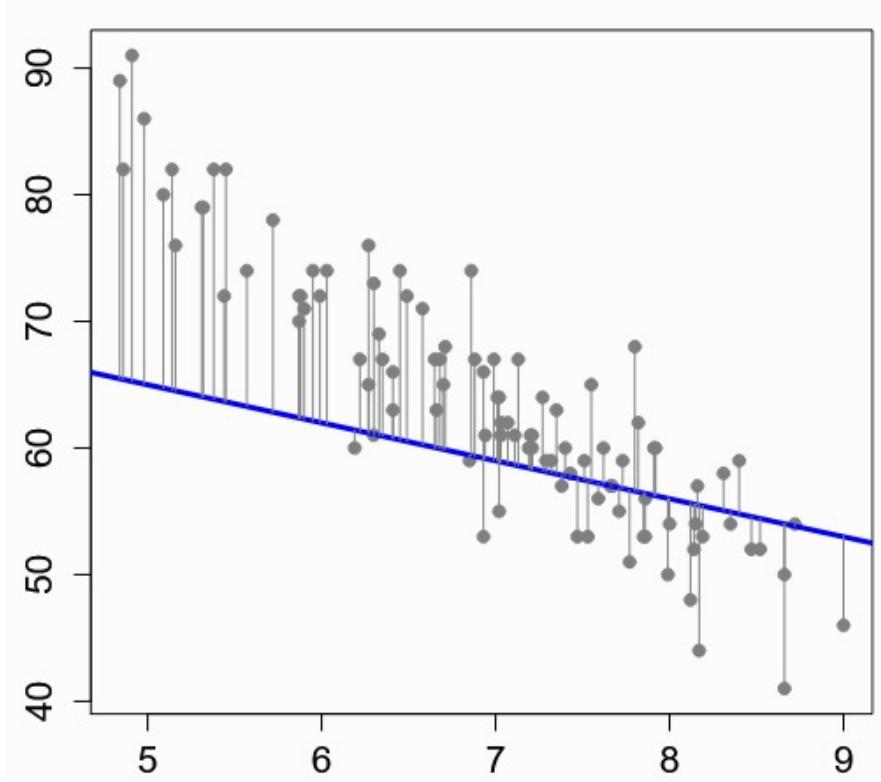
- Analyser dataen: antal rækker, kolonner, evt. statistikker osv.
- Visualiser dataen og prøv at se om nogle attributter "hænger sammen"
 - Hint: : individuelle plots eller seaborn og pairplot
- One-hot-encode *famhist* kolonnen, tilføj de nye kolonner til din dataframe og *.drop famhist*

Regression

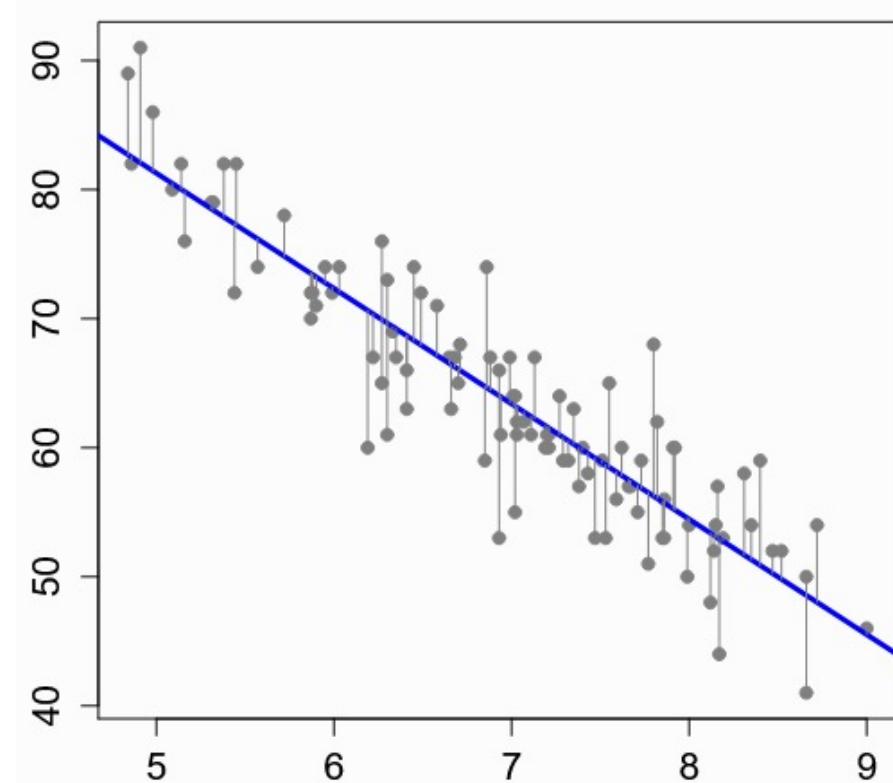
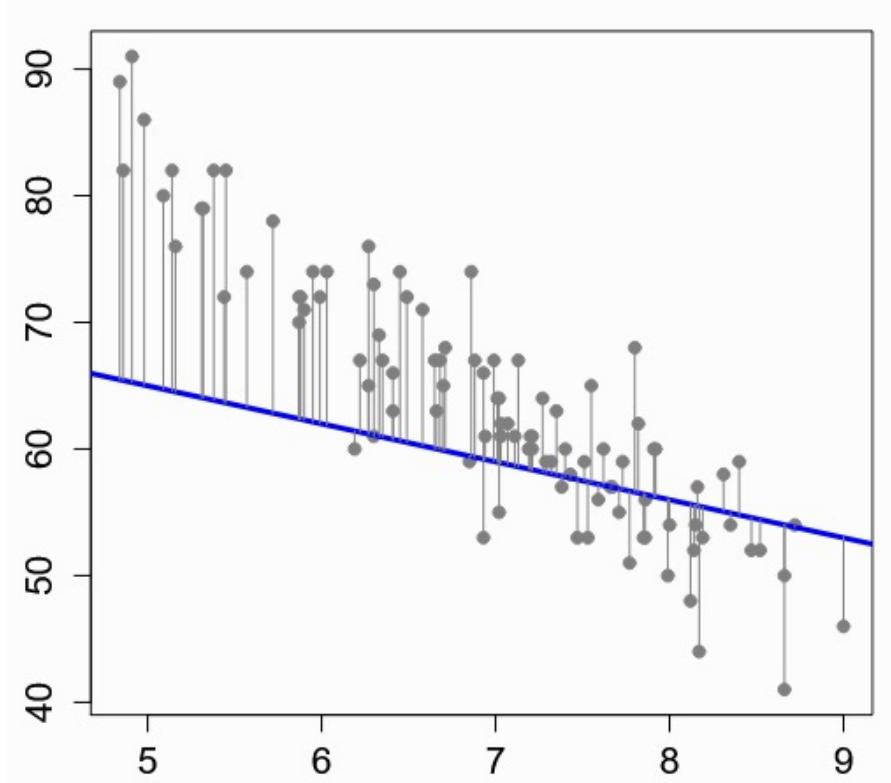
Lineær regression

- En matematisk/statistisk model man bruger til at forudse et tal ud fra data
 - Måske i allerede har stødt på den i et statistik kursus (eller i gymnasiet)
- F.eks. Vi vil gerne forudse *sepal_length* (SL) ud fra *sepal_width* (SW) i Iris sættet
 - Så hvilket tal *a* skal vi multiplicere SW med for at få SL
- Mere formelt; $y=a*x+b$
 - Hvor $y=sepal_length$, $x=sepal_width$
 - Linjen skal være så ”tæt” på dataen som muligt!

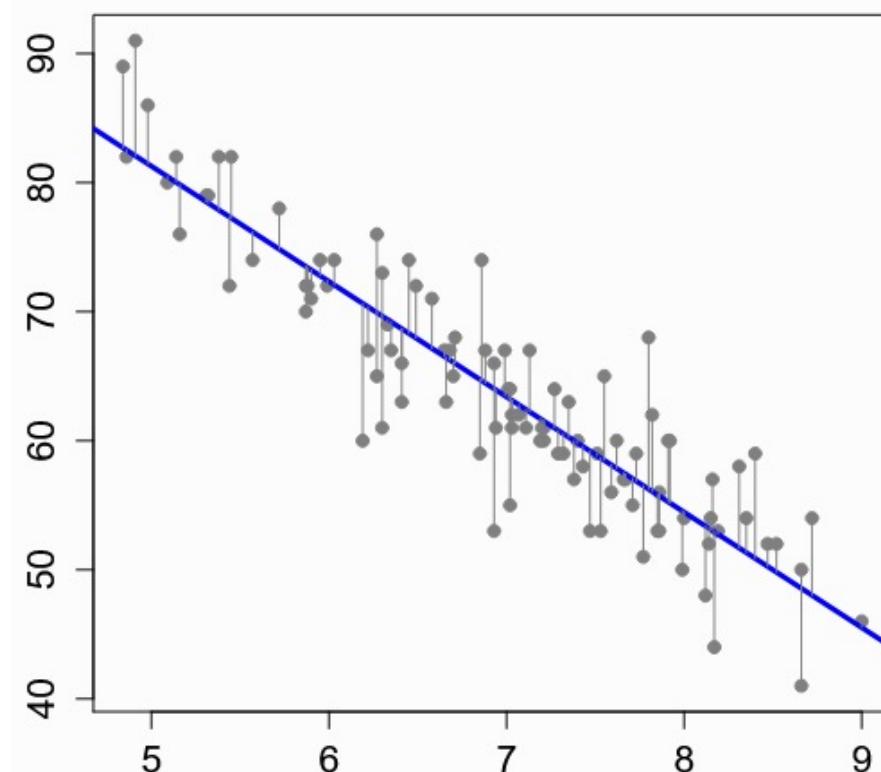
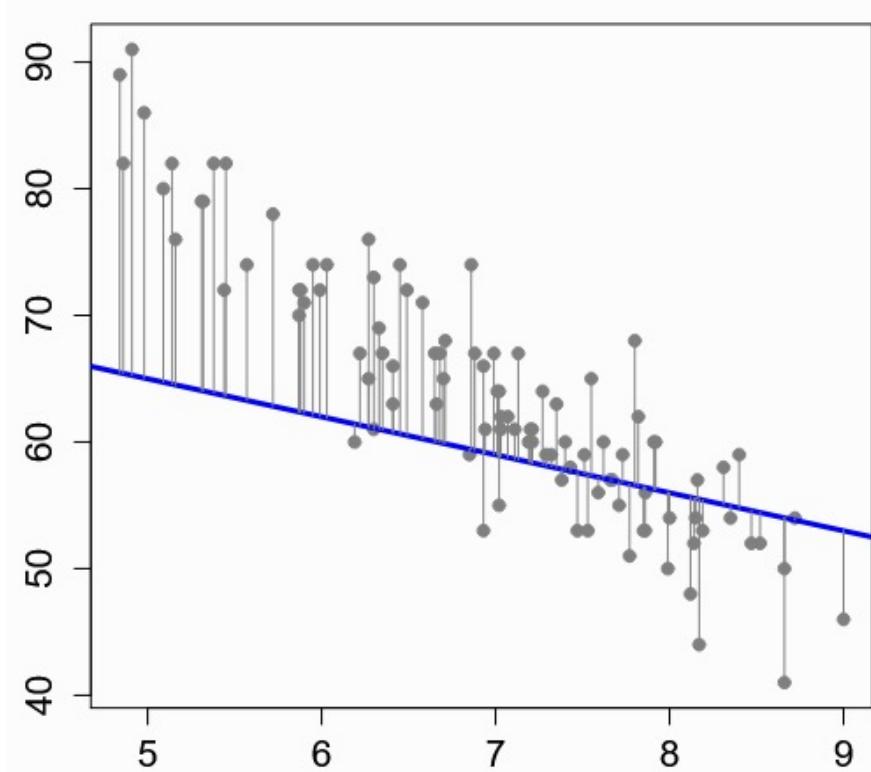
Lineær regression



Lineær regression



Lineær regression



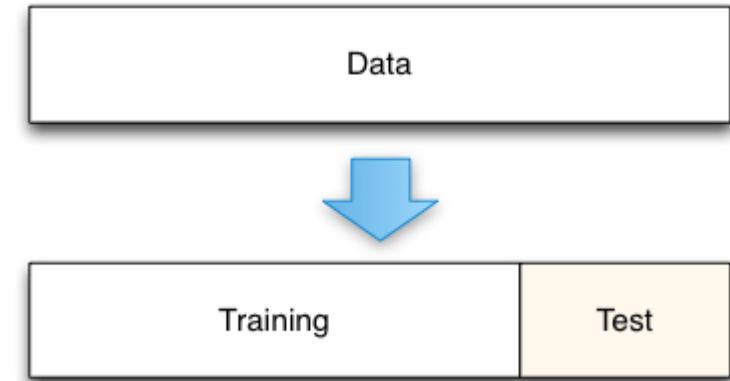
- Processen i at finde den bedste linje i lineær regression kan findes analytisk: $\theta = (X^T X)^{-1} X^T y$
 - https://en.wikipedia.org/wiki/Linear_regression

Træning og test - 80/20 splittet

- Før vi kan finde ud af hvordan vi træner en lineær regressions model i Python
 - Skal vi finde ud af hvordan vi splitter dataen op
- Hvis vi gerne vil teste hvordan vores model performer kan vi ikke give den alt vores data
 - Fordi så har den set dataen før, og så er modellen er blevet farvet!
- Så derfor giver vi modellen 80% af dataen til at træne
 - Og bruger så 20% til at teste hvor god den er til at forudse

80/20 splittet i Python

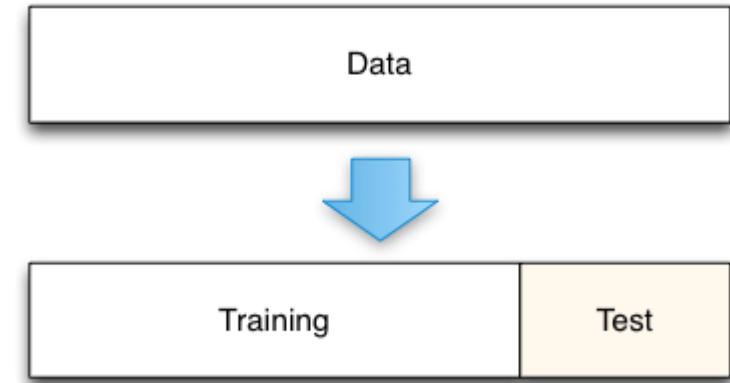
```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3
4 df = pd.read_csv("iris.data")
5 X = df[['sepal_length']]
6 y = df['sepal_width']
7
8 train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=42)
```



80/20 splittet i Python

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3
4 df = pd.read_csv("iris.data")
5 X = df[['sepal_length']]
6 y = df['sepal_width']
7
8 train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=42)
```

Hvor mange % vi tester på

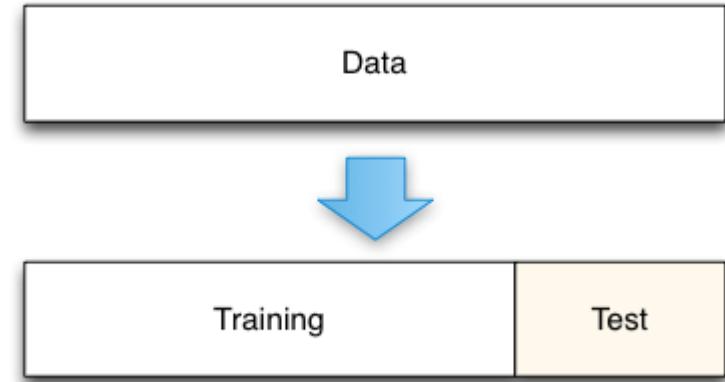


80/20 splittet i Python

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3
4 df = pd.read_csv("iris.data")
5 X = df[['sepal_length']]
6 y = df['sepal_width']
7
8 train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=42)
```

Hvor mange % vi tester på

Et "seed" (tal) man kan sætte for at genskabe splittet.
Et andet tal vil give et andet split.



Lineær regression i Python

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.linear_model import LinearRegression
4
5 df = pd.read_csv("iris.data")
6 X = df[['sepal_length']]
7 X = X.to_numpy()
8 y = df['sepal_width']
9 y = y.to_numpy()
10
11 train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=42)
12
13 model = LinearRegression()
14 model.fit(train_x, train_y)
15
16 a = model.coef_
17 b = model.intercept_
```

Lineær regression i Python

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.linear_model import LinearRegression
4
5 df = pd.read_csv("iris.data")
6 X = df[['sepal_length']]
7 X = X.to_numpy()
8 y = df['sepal_width']
9 y = y.to_numpy()
10
11 train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=42)
12
13 model = LinearRegression()
14 model.fit(train_x, train_y)
15
16 a = model.coef_
17 b = model.intercept_
```

Vi laver X og y om til numpy (matricer/vektorer),
for ellers bliver sklearn ked af det

Lineær regression i Python

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.linear_model import LinearRegression
4
5 df = pd.read_csv("iris.data")
6 X = df[['sepal_length']]
7 X = X.to_numpy()
8 y = df['sepal_width']
9 y = y.to_numpy()
10
11 train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=42)
12
13 model = LinearRegression()
14 model.fit(train_x, train_y)
15
16 a = model.coef_
17 b = model.intercept_
```

Vi laver X og y om til numpy (matricer/vektorer),
for ellers bliver sklearn ked af det

Vi træner vores model med træningsdata

Lineær regression i Python

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.linear_model import LinearRegression
4
5 df = pd.read_csv("iris.data")
6 X = df[['sepal_length']]
7 X = X.to_numpy()
8 y = df['sepal_width']
9 y = y.to_numpy()
10
11 train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=42)
12
13 model = LinearRegression()
14 model.fit(train_x, train_y)
15
16 a = model.coef_
17 b = model.intercept_
```

Vi laver X og y om til numpy (matricer/vektorer),
for ellers bliver sklearn ked af det

Vi træner vores model med træningsdata

Vi kan se den **a** og **b** sklearn har fundet i $y = ax + b$

Lineær regression i Python

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.linear_model import LinearRegression
4
5 df = pd.read_csv("iris.data")
6 X = df[['sepal_length']]
7 X = X.to_numpy()
8 y = df['sepal_width']
9 y = y.to_numpy()
10
11 train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=42)
12
13 model = LinearRegression()
14 model.fit(train_x, train_y)
15
16 a = model.coef_
17 b = model.intercept_
```

Vi laver X og y om til numpy (matricer/vektorer),
for ellers bliver sklearn ked af det

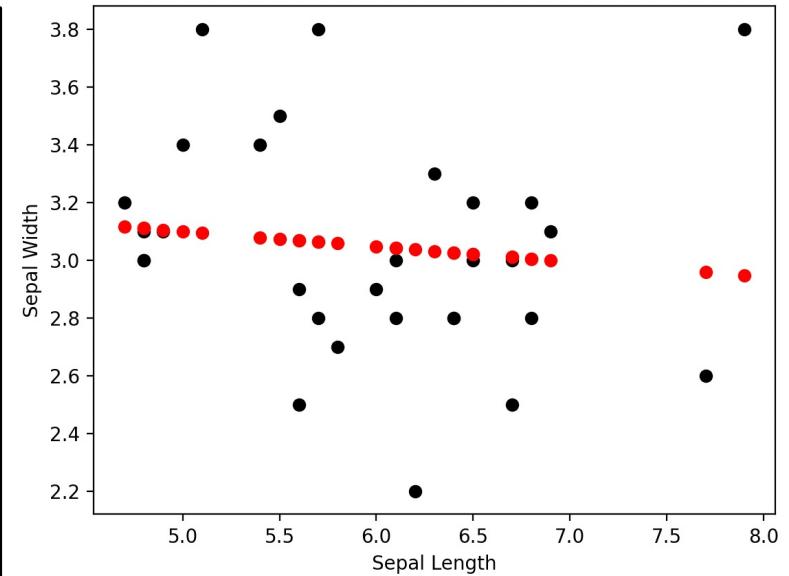
Vi træner vores model med træningsdata

Vi kan se den **a** og **b** sklearn har fundet i $y = ax + b$

- Live!

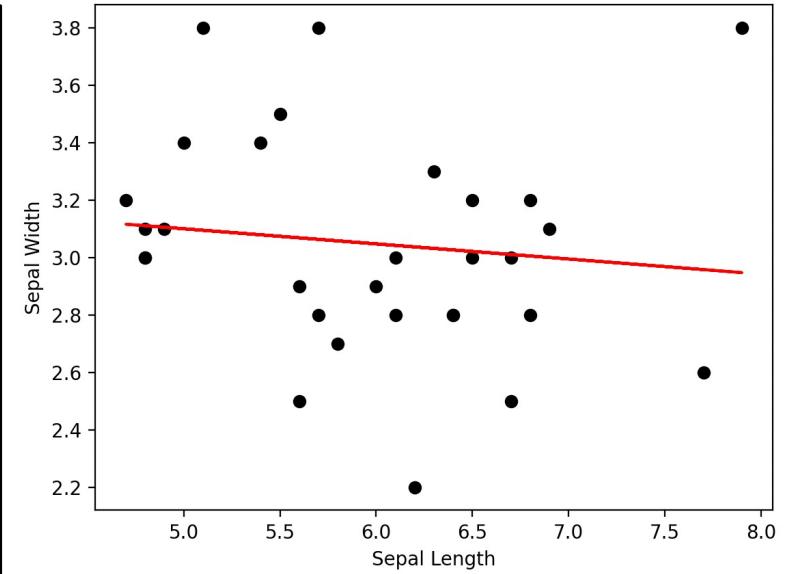
Lineær regression i Python – Predict & Plot

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.linear_model import LinearRegression
4 import matplotlib.pyplot as plt
5
6 df = pd.read_csv("iris.data")
7 X = df[['sepal_length']]
8 X = X.to_numpy()
9 y = df['sepal_width']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=42)
13
14 model = LinearRegression()
15 model.fit(train_x, train_y)
16
17 pred_y = model.predict(test_x) (17)
18
19 plt.scatter(test_x, test_y, color='black')
20 plt.scatter(test_x, pred_y, color='red')
21 plt.xlabel("Sepal Length")
22 plt.ylabel("Sepal Width")
23 plt.show()
```



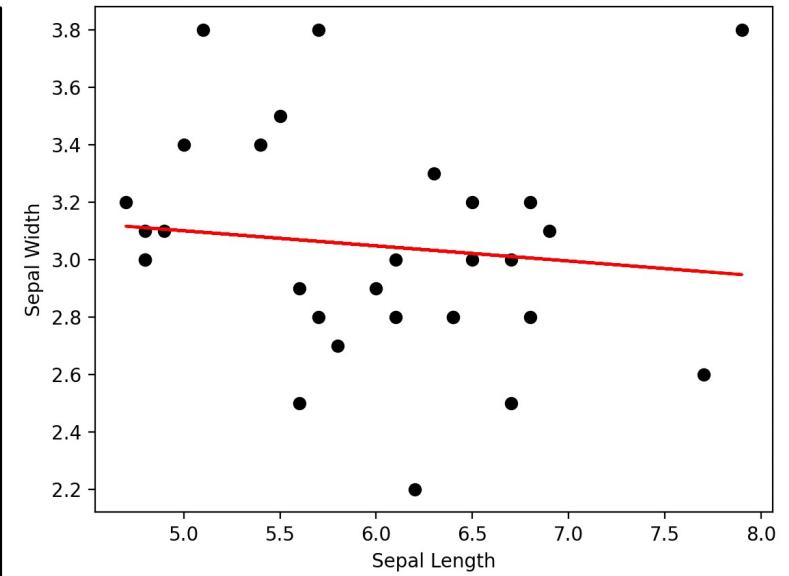
Lineær regression i Python – Predict & Plot

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.linear_model import LinearRegression
4 import matplotlib.pyplot as plt
5
6 df = pd.read_csv("iris.data")
7 X = df[['sepal_length']]
8 X = X.to_numpy()
9 y = df['sepal_width']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=42)
13
14 model = LinearRegression()
15 model.fit(train_x, train_y)
16
17 pred_y = model.predict(test_x)
18
19 plt.scatter(test_x, test_y, color='black')
20 plt.plot(test_x, pred_y, color='red')
21 plt.xlabel("Sepal Length")
22 plt.ylabel("Sepal Width")
23 plt.show()
```



Lineær regression i Python – Predict & Plot

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.linear_model import LinearRegression
4 import matplotlib.pyplot as plt
5
6 df = pd.read_csv("iris.data")
7 X = df[['sepal_length']]
8 X = X.to_numpy()
9 y = df['sepal_width']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=42)
13
14 model = LinearRegression()
15 model.fit(train_x, train_y)
16
17 pred_y = model.predict(test_x)
18
19 plt.scatter(test_x, test_y, color='black')
20 plt.plot(test_x, pred_y, color='red')
21 plt.xlabel("Sepal Length")
22 plt.ylabel("Sepal Width")
23 plt.show()
```

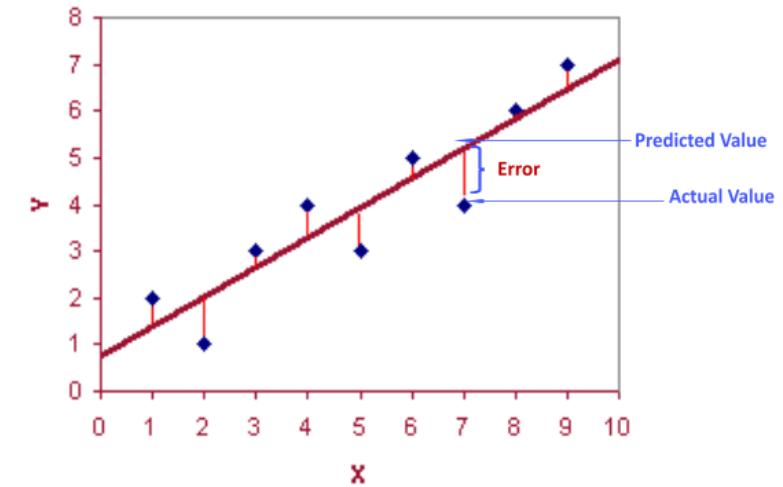


- Live!

Performance af regression

- Så vi kan se at *sklearn* finder en linje i dataen
 - Men helt præcist hvor godt forudser den?
- Det måler vi ved at finde afstanden fra hvert punkt til linjen
 - Og gør så det fra alle datapunkter...
 - ”Hvor langt gennemsnitligt er hvert punkt fra linjen”
- Der er en formel der hedder RMSE til det:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$



RMSE i Python

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.linear_model import LinearRegression
4 from sklearn.metrics import mean_squared_error
5
6 df = pd.read_csv("iris.data")
7 X = df[['sepal_length']]
8 X = X.to_numpy()
9 y = df['sepal_width']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=42)
13
14 model = LinearRegression()
15 model.fit(train_x, train_y)
16
17 pred_y = model.predict(test_x)
18
19 rmse = mean_squared_error(test_y, pred_y, squared=False)
20
21 print(rmse)
```

0.3737

RMSE i Python

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.linear_model import LinearRegression
4 from sklearn.metrics import mean_squared_error
5
6 df = pd.read_csv("iris.data")
7 X = df[['sepal_length']]
8 X = X.to_numpy()
9 y = df['sepal_width']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=42)
13
14 model = LinearRegression()
15 model.fit(train_x, train_y)
16
17 pred_y = model.predict(test_x)
18
19 rmse = mean_squared_error(test_y, pred_y, squared=False)
20
21 print(rmse)
```

0.3737

Regner RMSE, hvis True regner MSE

RMSE i Python

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.linear_model import LinearRegression
4 from sklearn.metrics import mean_squared_error
5
6 df = pd.read_csv("iris.data")
7 X = df[['sepal_length', 'petal_length', 'petal_width']]
8 X = X.to_numpy()
9 y = df['sepal_width']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=42)
13
14 model = LinearRegression()
15 model.fit(train_x, train_y)
16
17 pred_y = model.predict(test_x)
18
19 rmse = mean_squared_error(test_y, pred_y, squared=False)
20
21 print(rmse)
```

0.2945

RMSE i Python

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.linear_model import LinearRegression
4 from sklearn.metrics import mean_squared_error
5
6 df = pd.read_csv("iris.data")
7 X = df[['sepal_length', 'petal_length', 'petal_width']]
8 X = X.to_numpy()
9 y = df['sepal_width']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=42)
13
14 model = LinearRegression()
15 model.fit(train_x, train_y)
16
17 pred_y = model.predict(test_x)
18
19 rmse = mean_squared_error(test_y, pred_y, squared=False)
20
21 print(rmse)
```

0.2945

- Så vores model blev bedre at få mere data i X
 - Mindre RMSE → Tættere på det rigtige svar
 - Det giver mening (med Iris!) da den kan finde mere komplekse forhold så
- Men nu gik vores model lige til fra 2d til 4d
 - Og det kan vi ikke plotte længere...
 - Ikke længere $y = ax + b$
 - Nu: $y = a_1x_1 + a_2x_2 + a_3x_3 + b$

RMSE i Python

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.linear_model import LinearRegression
4 from sklearn.metrics import mean_squared_error
5
6 df = pd.read_csv("iris.data")
7 X = df[['sepal_length', 'petal_length', 'petal_width']]
8 X = X.to_numpy()
9 y = df['sepal_width']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=42)
13
14 model = LinearRegression()
15 model.fit(train_x, train_y)
16
17 pred_y = model.predict(test_x)
18
19 rmse = mean_squared_error(test_y, pred_y, squared=False)
20
21 print(rmse)
```

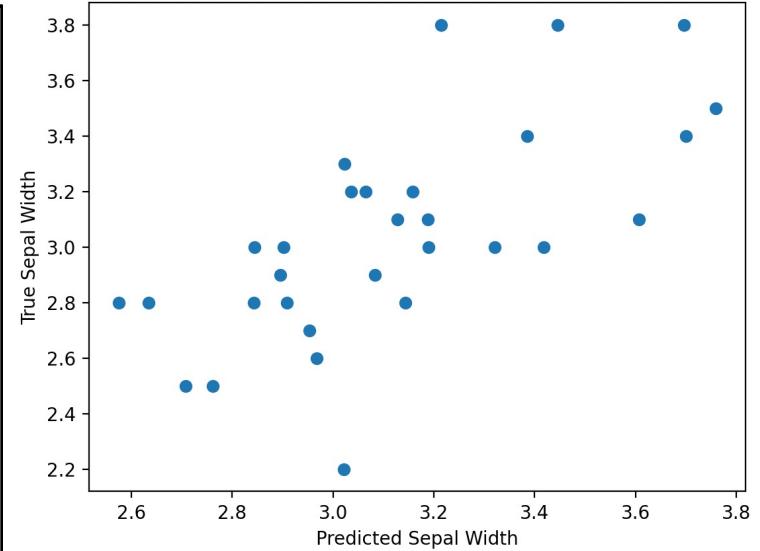
0.2945

• Live!

- Så vores model blev bedre at få mere data i X
 - Mindre RMSE → Tættere på det rigtige svar
 - Det giver mening (med Iris!) da den kan finde mere komplekse forhold så
- Men nu gik vores model lige til fra 2d til 4d
 - Og det kan vi ikke plotte længere...
 - Ikke længere $y = ax + b$
 - Nu: $y = a_1x_1 + a_2x_2 + a_3x_3 + b$

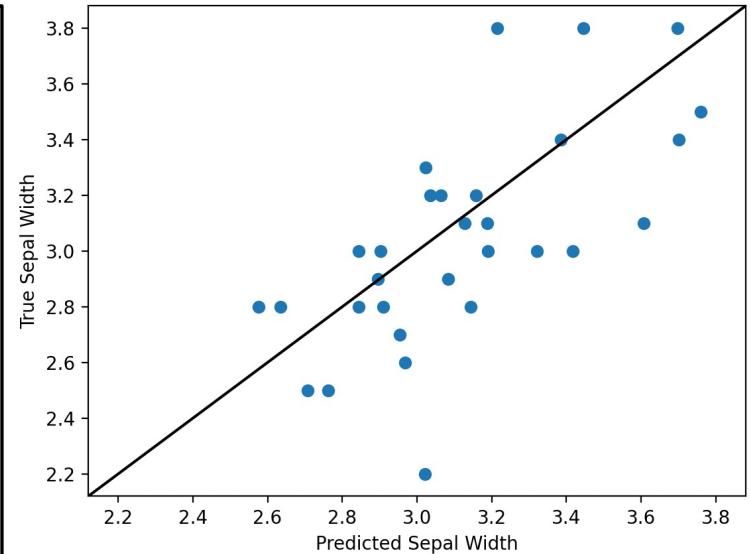
Performance Plot

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.linear_model import LinearRegression
4 import matplotlib.pyplot as plt
5
6 df = pd.read_csv("iris.data")
7 X = df[['sepal_length', 'petal_length', 'petal_width']]
8 X = X.to_numpy()
9 y = df['sepal_width']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=42)
13
14 model = LinearRegression()
15 model.fit(train_x, train_y)
16
17 pred_y = model.predict(test_x)
18
19 plt.scatter(pred_y, test_y)
20 plt.xlabel("Predicted Sepal Width")
21 plt.ylabel("True Sepal Width")
22 plt.show()
```



Performance Plot

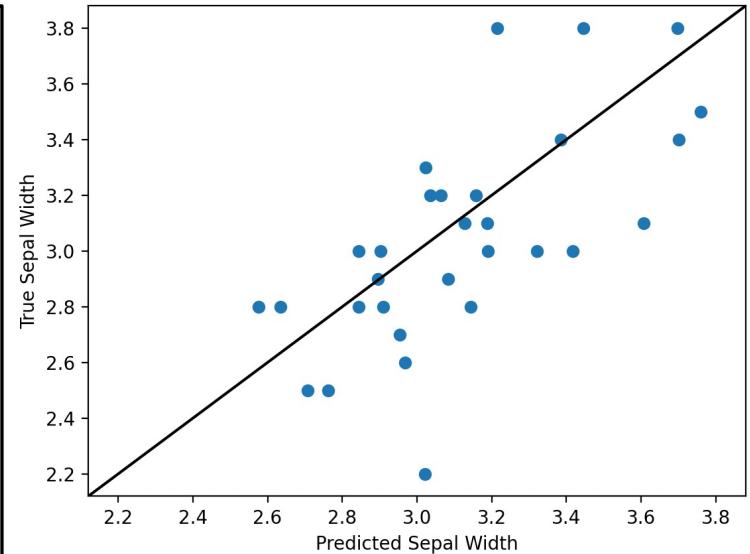
```
 1 import pandas as pd
 2 from sklearn.model_selection import train_test_split
 3 from sklearn.linear_model import LinearRegression
 4 import matplotlib.pyplot as plt
 5
 6 df = pd.read_csv("iris.data")
 7 X = df[['sepal_length', 'petal_length', 'petal_width']]
 8 X = X.to_numpy()
 9 y = df['sepal_width']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=42)
13
14 model = LinearRegression()
15 model.fit(train_x, train_y)
16
17 pred_y = model.predict(test_x)
18
19 plt.scatter(pred_y, test_y)
20 plt.xlabel("Predicted Sepal Width")
21 plt.ylabel("True Sepal Width")
22 min = min([pred_y.min(), test_y.min()])
23 max = max([pred_y.max(), test_y.max()])
24 plt.axline((min, min), (max,max), color='black')
25 plt.show()
```



- Hvis modellen havde været perfekt var alle punkterne på linjen igennem diagonalen

Performance Plot

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.linear_model import LinearRegression
4 import matplotlib.pyplot as plt
5
6 df = pd.read_csv("iris.data")
7 X = df[['sepal_length', 'petal_length', 'petal_width']]
8 X = X.to_numpy()
9 y = df['sepal_width']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=42)
13
14 model = LinearRegression()
15 model.fit(train_x, train_y)
16
17 pred_y = model.predict(test_x)
18
19 plt.scatter(pred_y, test_y)
20 plt.xlabel("Predicted Sepal Width")
21 plt.ylabel("True Sepal Width")
22 min = min([pred_y.min(), test_y.min()])
23 max = max([pred_y.max(), test_y.max()])
24 plt.axline((min, min), (max, max), color='black')
25 plt.show()
```



- Hvis modellen havde været perfekt var alle punkterne på linjen igennem diagonalen
- Live!

Opgaver pt. 2 (30 minutter)

1. Brug lineær regression på vin datasættet til at forudse hvor meget alkohol der er i en vin. Altså hvor $y = alcohol$ og test performance med RSME med et 80/20 split.
 - Hvis ikke du nåede opgave 1.2 så lav den først!
 - Brug random_state=42
 - Skulle gerne give omkring $rmse = 0.595$
 - Plot dine forudsigelser mod det rigtige y , hvordan ser resultaterne ud?
2. Brug lineær regression på på hjerte datasættet til at forudse hvor gammel en person er. Altså hvor $y = age$ og test performance med RSME med et 80/20 split.
 - Hvis ikke du nåede opgave 1.3 så lav den først!
 - Brug random_state=42
 - skulle gerne give omkring $rmse = 9.59$
 - Plot dine forudsigelser mod det rigtige y , hvordan ser resultaterne ud?
3. (Ekstra) Vi vil gerne undersøge om vi kan forøge vores performance på de to datasæt, prøv at undersøg hvilke attributter fra X du kan fjerne for at få bedre performance (hvis nogle?).
 - Hints:
 - Undersøg coefficienterne fra dine modeller i 2.1 og 2.2
 - En meget stor coefficient betyder meget positiv korrelering
 - En meget lav coefficient betyder meget negativ korrelering
 - En coefficient tæt på nul KAN være ligegyldig og dermed dårligere performance

Klassifikation

Klassifikation og performance

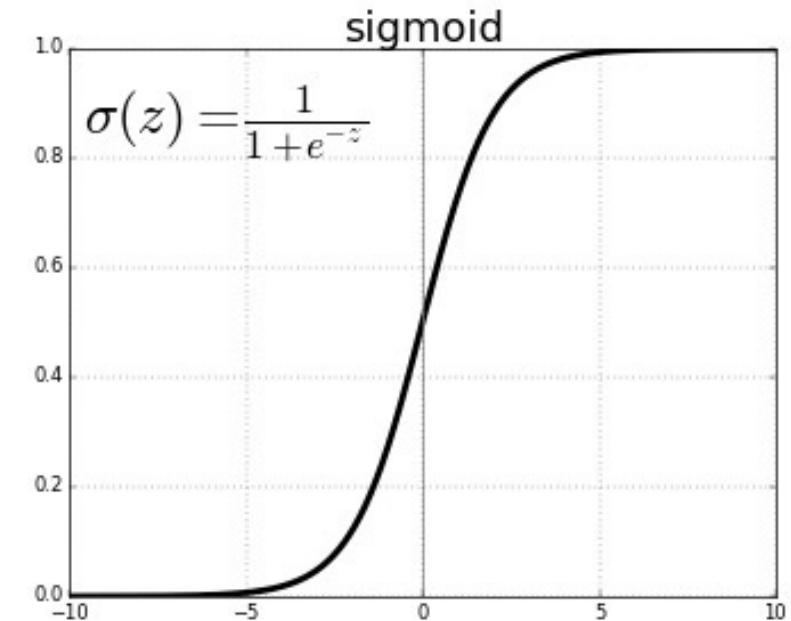
- I klassifikation er vi ikke længe interesseret i at forudse et tal
- I stedet vil vi forudse hvilken ”Klasse” noget data hører til
 - F.eks. Hvilken slags blomst er det her?
- Vi mÅler performance i hvor mange % vores model forudsÅ rigtigt
 - Hedder ”Accuracy”
- Baseline
 - Hvis vi bare forudsÅ alt vores data i Iris sættet til at vÅre ”Iris-virginica” sÅ ville vi have en accuracy på:
$$\frac{\text{rigtige forudset}}{\text{totalt antal}} = \frac{50}{150} = 0,33 = 33\%$$
 - De her 33% kalder vi for baseline, og hvis vores model ikke er bedre end det er det en virkelig dårlig model

Logistisk regression (LR)

- En machine learning algoritme
- Selvom den har regression i navnet, så bruger vi den til at klassificere!
- Dog minder logistisk regression meget om lineær regression
- Så vi bruger stadig vores gængse formel: $y=a*x+b$ hvor vi får et tal ud
 - Men det tal laver vi om til noget vi kan bruge til at klassificere
- Kan ikke løses analytisk, kræver i stedet optimerings algoritmer, f.eks.:
 - https://en.wikipedia.org/wiki/Gradient_descent

Sigmoid

- Så det tal vores logistisk regression model ender med at forudse kalder vi for $\textcolor{blue}{z}$
- $\textcolor{blue}{z}$ smider vi igennem formlen til højre, og så får vi et tal mellem 0 og 1 tilbage
 - Det tal vi får tilbage kan tolkes som hvor mange % vi er sikker på at det data tilhører en given klasse
 - Så som vi kan se på grafen, stort $\textcolor{blue}{z}$ giver en høj sandsynlighed for den ene klasse, lille $\textcolor{blue}{z}$ giver lille sandsynlighed for den samme klasse



Logistisk regression i Python

```
1 import pandas as pd
2 from sklearn.linear_model import LogisticRegression
3 from sklearn.model_selection import train_test_split
4 from sklearn.metrics import accuracy_score
5
6 df = pd.read_csv('iris.data')
7 X = df.drop('type', axis=1)
8 X = X.to_numpy()
9 y = df['type']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X,y, test_size=0.20)
13
14 model = LogisticRegression().fit(train_x, train_y)
15
16 pred_y = model.predict(test_x)
17
18 accuracy = accuracy_score(test_y, pred_y)
19
20 print(accuracy)
```

Logistisk regression i Python

```
1 import pandas as pd
2 from sklearn.linear_model import LogisticRegression
3 from sklearn.model_selection import train_test_split
4 from sklearn.metrics import accuracy_score
5
6 df = pd.read_csv('iris.data')
7 X = df.drop('type', axis=1)
8 X = X.to_numpy()
9 y = df['type']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X,y, test_size=0.20)
13
14 model = LogisticRegression().fit(train_x, train_y)
15
16 pred_y = model.predict(test_x)
17
18 accuracy = accuracy_score(test_y, pred_y)
19
20 print(accuracy)
```

Vi træner vores LR

Logistisk regression i Python

```
1 import pandas as pd
2 from sklearn.linear_model import LogisticRegression
3 from sklearn.model_selection import train_test_split
4 from sklearn.metrics import accuracy_score
5
6 df = pd.read_csv('iris.data')
7 X = df.drop('type', axis=1)
8 X = X.to_numpy()
9 y = df['type']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X,y, test_size=0.20)
13 model = LogisticRegression().fit(train_x, train_y)
14 pred_y = model.predict(test_x)
15
16 accuracy = accuracy_score(test_y, pred_y)
17
18 print(accuracy)
```

Vi træner vores LR

Vi bruger modellen til at forudse typen af blomsten vores test data er. Her har den allerede brugt sigmoid for os!

Logistisk regression i Python

```
1 import pandas as pd
2 from sklearn.linear_model import LogisticRegression
3 from sklearn.model_selection import train_test_split
4 from sklearn.metrics import accuracy_score
5
6 df = pd.read_csv('iris.data')
7 X = df.drop('type', axis=1)
8 X = X.to_numpy()
9 y = df['type']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X,y, test_size=0.20)
13 model = LogisticRegression().fit(train_x, train_y)
14 pred_y = model.predict(test_x)
15
16 accuracy = accuracy_score(test_y, pred_y)
17
18 print(accuracy)
```

Vi træner vores LR

Vi bruger modellen til at forudse typen af blomsten vores test data er. Her har den allerede brugt sigmoid for os!

Vi bruger sklearns accuracy_score metode til at regne hvor mange % den ramte rigtigt

Logistisk regression i Python

```
1 import pandas as pd
2 from sklearn.linear_model import LogisticRegression
3 from sklearn.model_selection import train_test_split
4 from sklearn.metrics import accuracy_score
5
6 df = pd.read_csv('iris.data')
7 X = df.drop('type', axis=1)
8 X = X.to_numpy()
9 y = df['type']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X,y, test_size=0.20)
13 model = LogisticRegression().fit(train_x, train_y)
14 pred_y = model.predict(test_x)
15
16 accuracy = accuracy_score(test_y, pred_y)
17
18 print(accuracy)
```

Som regel i Iris $0.85 - 1 \sim 85 - 100\%$

Vi træner vores LR

Vi bruger modellen til at forudse typen af blomsten vores test data er. Her har den allerede brugt sigmoid for os!

Vi bruger sklearns accuracy_score metode til at regne hvor mange % den ramte rigtigt

Logistisk regression i Python

```
1 import pandas as pd
2 from sklearn.linear_model import LogisticRegression
3 from sklearn.model_selection import train_test_split
4 from sklearn.metrics import accuracy_score
5
6 df = pd.read_csv('iris.data')
7 X = df.drop('type', axis=1)
8 X = X.to_numpy()
9 y = df['type']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X,y, test_size=0.20)
13 model = LogisticRegression().fit(train_x, train_y)
14 pred_y = model.predict(test_x)
15
16 accuracy = accuracy_score(test_y, pred_y)
17
18 print(accuracy)
```

- Live!

Som regel i Iris $0.85 - 1 \sim 85 - 100\%$

Vi træner vores LR

Vi bruger modellen til at forudse typen af blomsten vores test data er. Her har den allerede brugt sigmoid for os!

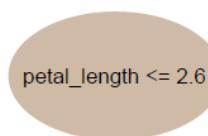
Vi bruger sklearns accuracy_score metode til at regne hvor mange % den ramte rigtigt

Decision tree (DT)

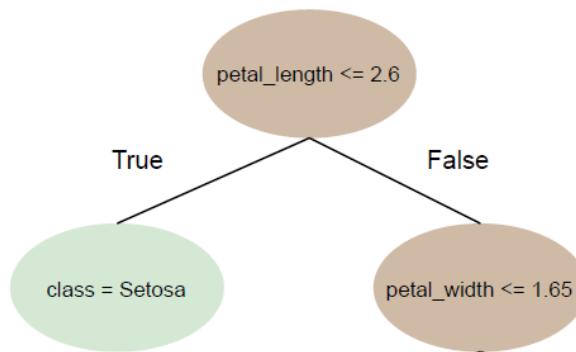
- En anden machine learning algoritme
- Har intet med regression at gøre denne gang!
- Minder om legen ”20 spørgsmål til professoren”
 - En person skal tænke på noget, og så har man 20 spørgsmål til at finde ud af hvad den ting er
 - Personen må kun svare ja/nej
 - Så man starter meget bredt ”er det et dyr” f.eks.
 - Men til sidst er det meget specifikke spørgsmål
- DT stiller også nogle spørgsmål om dataen og klassificere på den måde!

Decision tree (DT) til Iris

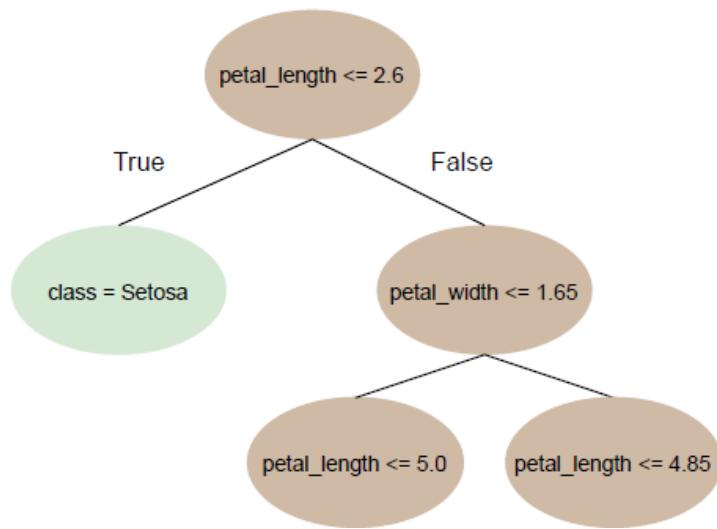
Decision tree (DT) til Iris



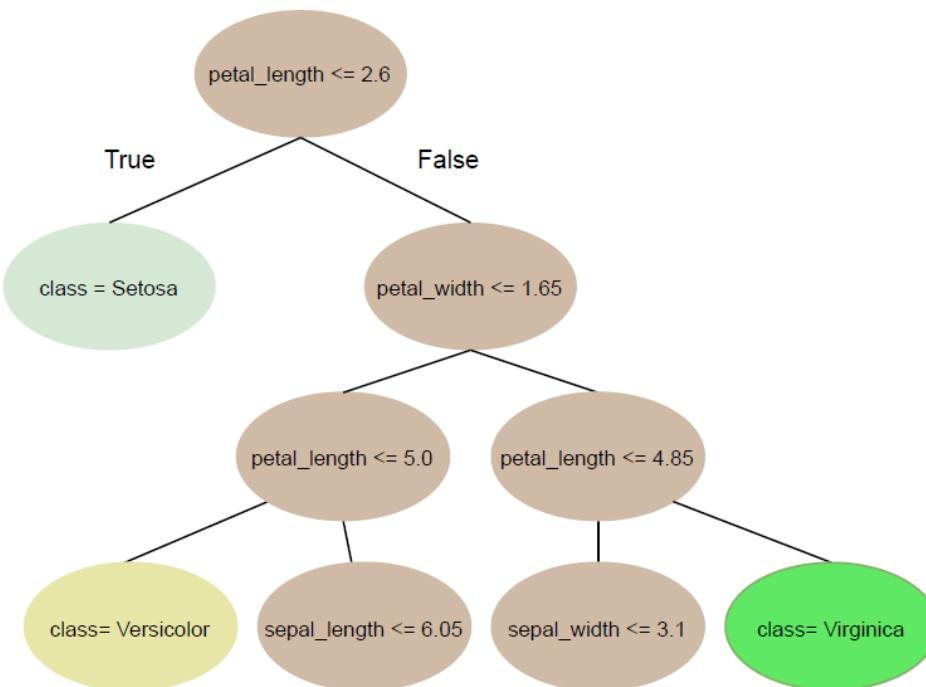
Decision tree (DT) til Iris



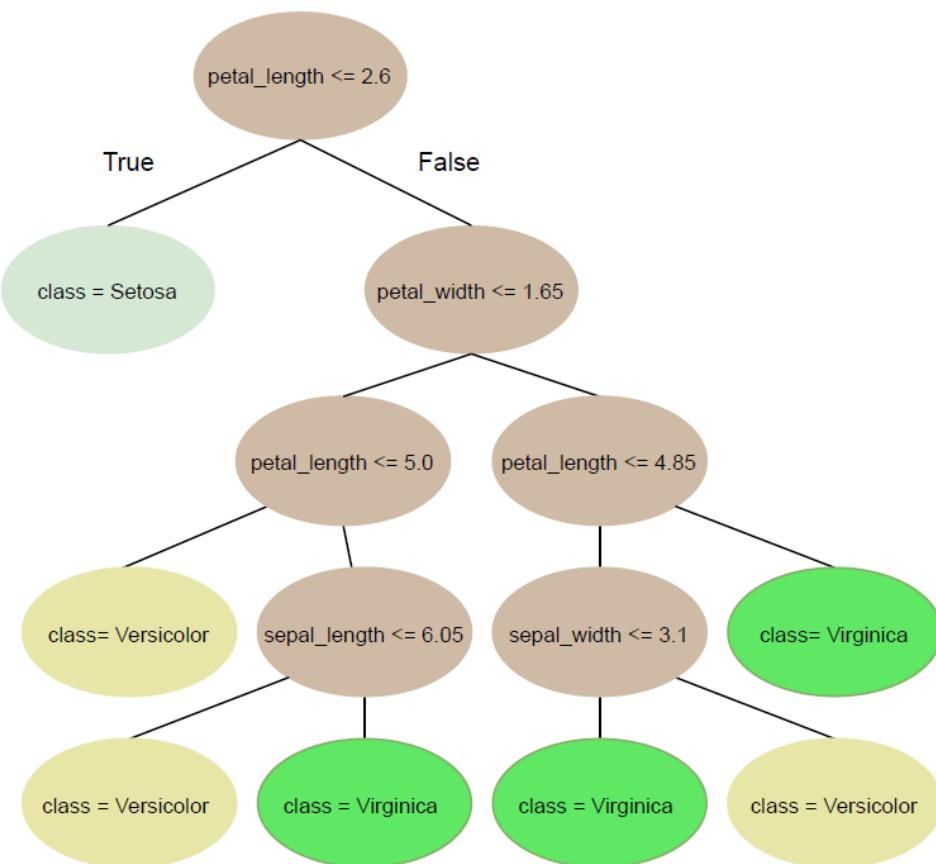
Decision tree (DT) til Iris



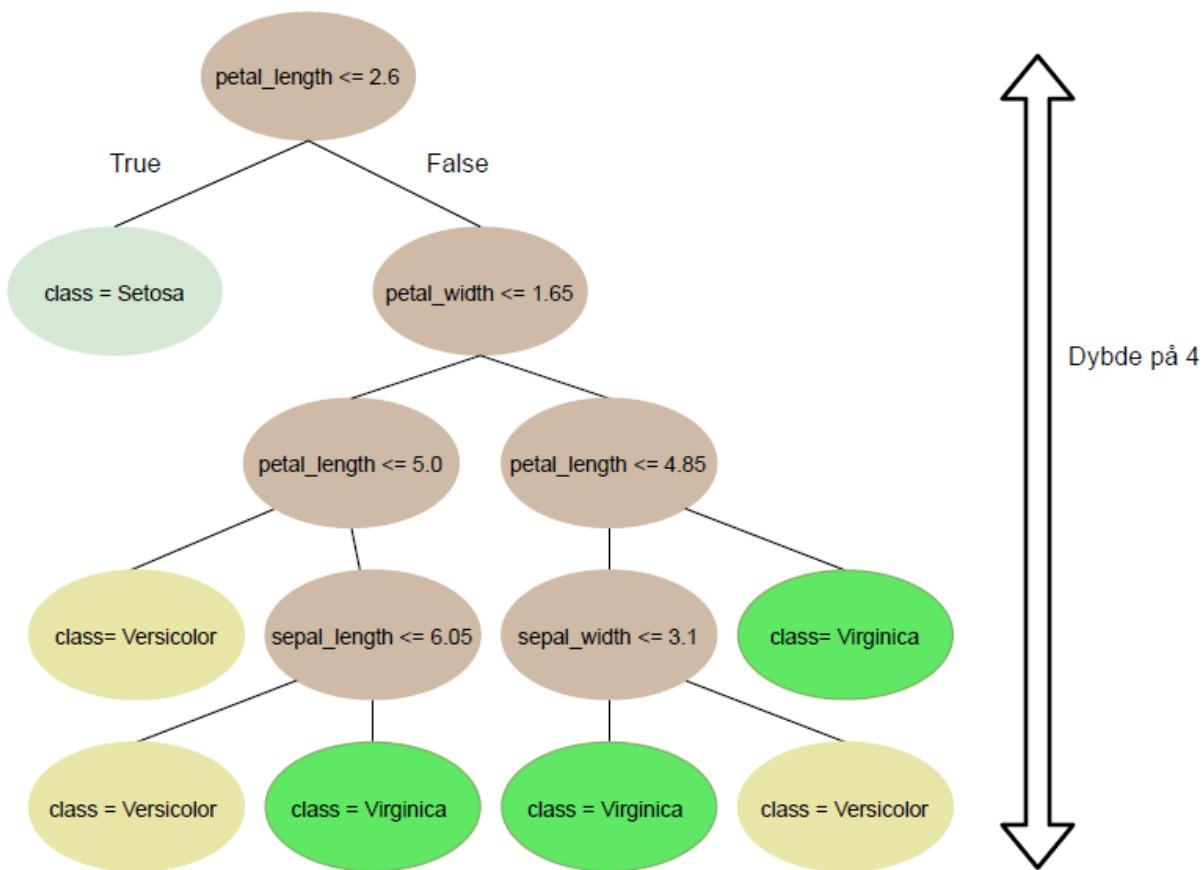
Decision tree (DT) til Iris



Decision tree (DT) til Iris



Decision tree (DT) til Iris



Decision tree i Python

```
1 import pandas as pd
2 from sklearn.tree import DecisionTreeClassifier
3 from sklearn.model_selection import train_test_split
4 from sklearn.metrics import accuracy_score
5
6 df = pd.read_csv('iris.data')
7 X = df.drop('type', axis=1)
8 X = X.to_numpy()
9 y = df['type']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X,y, test_size=0.20)
13
14 model = DecisionTreeClassifier().fit(train_x, train_y)
15
16 pred_y = model.predict(test_x)
17
18 accuracy = accuracy_score(test_y, pred_y)
19
20 print(accuracy)
```

- Måden som *sklearn* finder et godt træ kan regnes matematisk, f.eks. med "Gini coefficienten"

Decision tree i Python

```
1 import pandas as pd
2 from sklearn.tree import DecisionTreeClassifier
3 from sklearn.model_selection import train_test_split
4 from sklearn.metrics import accuracy_score
5
6 df = pd.read_csv('iris.data')
7 X = df.drop('type', axis=1)
8 X = X.to_numpy()
9 y = df['type']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X,y, test_size=0.20)
13
14 model = DecisionTreeClassifier(max_depth=1).fit(train_x, train_y)
15
16 pred_y = model.predict(test_x)
17
18 accuracy = accuracy_score(test_y, pred_y)
19
20 print(accuracy)
```

Decision tree i Python

```
1 import pandas as pd
2 from sklearn.tree import DecisionTreeClassifier
3 from sklearn.model_selection import train_test_split
4 from sklearn.metrics import accuracy_score
5
6 df = pd.read_csv('iris.data')
7 X = df.drop('type', axis=1)
8 X = X.to_numpy()
9 y = df['type']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X,y, test_size=0.20)
13 model = DecisionTreeClassifier(max_depth=1).fit(train_x, train_y)
14 pred_y = model.predict(test_x)
15
16 accuracy = accuracy_score(test_y, pred_y)
17
18 print(accuracy)
```

Man kan sætte en "max dybde" på træet

Decision tree i Python

```
1 import pandas as pd
2 from sklearn.tree import DecisionTreeClassifier
3 from sklearn.model_selection import train_test_split
4 from sklearn.metrics import accuracy_score
5
6 df = pd.read_csv('iris.data')
7 X = df.drop('type', axis=1)
8 X = X.to_numpy()
9 y = df['type']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X,y, test_size=0.20)
13 model = DecisionTreeClassifier(max_depth=1).fit(train_x, train_y)
14 pred_y = model.predict(test_x)
15
16 accuracy = accuracy_score(test_y, pred_y)
17
18 print(accuracy)
```

Man kan sætte en "max dybde" på træet

- Live!

Decision tree i Python plottet

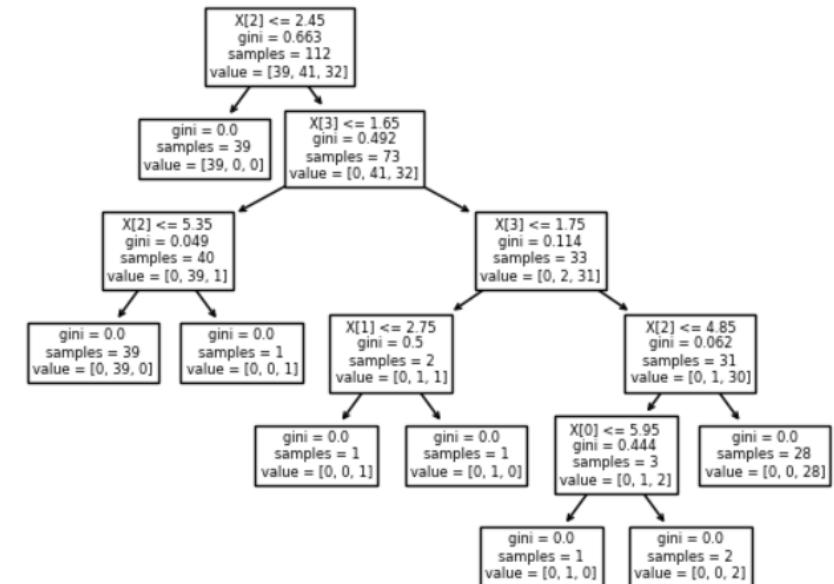
```
1 import pandas as pd
2 from sklearn.tree import DecisionTreeClassifier, plot_tree
3 from sklearn.model_selection import train_test_split
4 import matplotlib.pyplot as plt
5
6 df = pd.read_csv("iris.data", sep=",")
7 X = df[['sepal_length', 'sepal_width','petal_length', 'petal_width']]
8 X = X.to_numpy()
9 y = df['type']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X,y)
13
14 model = DecisionTreeClassifier().fit(train_x, train_y)
15 pred_y = model.predict(test_x)
16
17 plot_tree(model)
18 plt.show()
```

Decision tree i Python plottet

```
1 import pandas as pd
2 from sklearn.tree import DecisionTreeClassifier, plot_tree
3 from sklearn.model_selection import train_test_split
4 import matplotlib.pyplot as plt
5
6 df = pd.read_csv("iris.data", sep=",")
7 X = df[['sepal_length', 'sepal_width','petal_length', 'petal_width']]
8 X = X.to_numpy()
9 y = df['type']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X,y)
13
14 model = DecisionTreeClassifier().fit(train_x, train_y)
15 pred_y = model.predict(test_x)
16
17 plot_tree(model)
18 plt.show()
```

Decision tree i Python plottet

```
1 import pandas as pd
2 from sklearn.tree import DecisionTreeClassifier, plot_tree
3 from sklearn.model_selection import train_test_split
4 import matplotlib.pyplot as plt
5
6 df = pd.read_csv("iris.data", sep=",")
7 X = df[['sepal_length', 'sepal_width','petal_length', 'petal_width']]
8 X = X.to_numpy()
9 y = df['type']
10 y = y.to_numpy()
11
12 train_x, test_x, train_y, test_y = train_test_split(X,y)
13
14 model = DecisionTreeClassifier().fit(train_x, train_y)
15 pred_y = model.predict(test_x)
16
17 plot_tree(model)
18 plt.show()
```



Opgaver pt. 3 (30 minutter)

1. **Vi vil gerne forudse om en person får en hjertesygdom eller ej, altså $y=chd$. Udregn først en baseline. Kan du forudse bedre end baselinen med Logistisk Regression?**
 - Hint til baseline:
 - Husk vi kan kalde `.value_counts()` på vores chd kolonne for at få antallet i hver klasse
 - Skulle gerne give omkring 65% hvis vi forudsætter ingen ($chd=0$) til at have en hjertesygdom
 - Test med et 80/20 split og `random_state=42`
 - Skulle gerne give omkring 74%
2. **Vi vil gerne forudse om en vin er god eller ej, altså $y=good_quality$. Udregn først en baseline. Brug et Decision Tree til at forudse vin kvalitet (og se om den er bedre end baseline).**
 - Test med 80/20 split og `random_state=42`
 - Skulle gerne give omkring 73%
 - Prøv at giv træet en relativ lav max dybde (måske 3-4) og plot det, hvordan ser det ud?
3. **(Ekstra) Vi vil gerne finde den bedste træ dybde til at forudse om en person får en hjertesygdom, altså $y=chd$. Det kræver følgende skridt:**
 - Start først med at lave et 80/20 split og `random_state=42`.
 - Definer herefter en liste til at gemme dine accuracies i.
 - Kør et loop fra 1 til 20 hvor du inde i loopet træner et træ med den dybde du er nået til i loopet.
 - Husk at gemme accuracien fra det her træ og tilføj det til listen
 - Hints: for i in `range(1,20)`, `max_depth=i`
 - Plot til sidst listen med `matplotlib`, hvilken dybde giver den bedste accuracy?

Avanceret

Klassificering af kriminalitet i San Francisco

- De fleste af de data eksempler vi har set i dag har været næsten "lige til"
 - Har ikke krævet så meget bearbejdelse
 - Og er nogle klassiske skole eksempler
 - Og er i virkeligheden nogle ret små datasæt!
- Datasæt i virkeligheden kan være ekstremt rodet
 - Og ekstremt store
- Så lad os prøve at kigge på et rodet stort datasæt!
 - Kriminalitet i San Francisco

San Francisco crime datasæt

- Har brugt et ældre et fra 2003-2018
 - Kan hentes her:
 - <https://data.sfgov.org/Public-Safety/Police-Department-Incident-Reports-Historical-2003/tmnf-yvry>
- Består af 2 millioner + rækker
 - Altså 2 millioner kriminaliteter!
- Består af 36 kolonner
 - Hvilken type kriminalitet
 - Hvor skete det
 - Hvornår
 - Osv...
- Mål:
 - Forudse om det er svindel eller biltyveri der er sket ud fra placering og tidspunkt i SF

Afrunding & anbefalinger

- Machine learning er meget populært lige nu!
 - Og med gode grunde, det har vist virkelig gode resultater
 - Specielt den forgrenning af Machine Learning der hedder "Deep Learning" (Neurale Netværk)
 - Hvilket er det jeg skrev speciale i
 - Hvis i synes det er spændende, så fortsæt med at prøve nogle ting ud i det.
- Gode ressourcer:
 - <https://scikit-learn.org/stable/tutorial/basic/tutorial.html>
 - https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html
- Og artikler hvor ML bliver brugt:
 - <https://towardsdatascience.com/>
- Datasæt og ML konkurrencer:
 - <https://www.kaggle.com/>
- Løsningerne til i dag, både exercises og advanced:
 - https://github.com/AndersBensen/ml_beginner/raw/main/course_folder/solutions.zip

Spørgsmål?

- anders_bensen@hotmail.com