TDT4172
Introduksjon til maskinlæring

Forelesning 2: Veiledet læring, data og klassifisering

norwegian
open ai lab

NTNU

Fra sist: Maskinlæring

= programmere datamaskiner sånn de at de kan lære fra data.

“Machine learning in the field of study that gives computers the ability to learn without being explicitly programmed” - Arthur Samuel, 1959

Maskinlæring, mer presist

"A computer program is said to learn from experience ***E*** with respect to some task ***T*** and some performance measure ***P***, if its performance on ***T***, as measured by ***P***, improves with experience ***E***." - Tom Mitchell, 1997

norwegian open ai lab

NTNU

Experience *E*: Data.

Datasett består av datapunkter med egenskaper.

Disse egenskapene omtales som **variabler** innen statistikk, og **features** i maskinlæring

Variabler/features kan være kvantitative eller kvalitative:

	Kvalitative variabler	**Kvantitative variabler**
Representerer	kategorier, og kommer fra diskrete sett	numeriske verdier, og kommer fra kontinuerlige sett
Eksempler	sjanger {action, komedie, ...}, aldersgruppe {<18, 18-30, ...}, type frukt {eple, banan, ...}	inntekt, alder, antall lyttere, areal, nedbørsmengde, temperatur,

Data

Når vi setter sammen datapunkter til et datasett, plasseres de ulike **variablene i kolonner**, og **datainstansene i rader**.

Eksempel: Titanic (https://www.kaggle.com/c/titanic)

PassengerId	Pclass	Name	Sex	Age	Survived
1	3	Braund, Mr. Owen Harris	male	22.0	0
2	1	Cumings, Mrs. John Bradley	female	38.0	1
3	3	Heikkinen, Miss. Laina	female	26.0	1
4	1	Futrelle, Mrs. Jacques Heath	female	35.0	1
5	3	Allen, Mr. William Henry	male	35.0	0
6	3	Moran, Mr. James	male		0
7	1	McCarthy, Mr. Timothy J	male	54.0	0
8	3	Palsson, Master. Gosta Leonard	male	2.0	0
9	2	Montvila, Rev. Juozas	male	27.0	1

Q:

Hvilke variabler er kategoriske?

Hvilke variabler er binære?

Hvilke variabler er kontinuerlige?

Hvilken variabel bør vi ikke bruke?

Task *T*: Vi har lyst til å finne ut noe fra dataene

Avhengig av dataene og problemet kan vi velge mellom **veiledet**, uveiledet og forsterket læring.

Vi er ute etter å finne ut om passasjerer overlevde. *Inneholder dataene denne informasjonen?*

Hvilken læringsform bør vi bruke?

PassengerId	Pclass	Name	Sex	Age	Survived
1	3	Braund, Mr. Owen Harris	male	22.0	0
2	1	Cumings, Mrs. John Bradley	female	38.0	1
3	3	Heikkinen, Miss. Laina	female	26.0	1
4	1	Futrelle, Mrs. Jacques Heath	female	35.0	1
5	3	Allen, Mr. William Henry	male	35.0	0
6	3	Moran, Mr. James	male		0
7	1	McCarthy, Mr. Timothy J	male	54.0	0
8	3	Palsson, Master. Gosta Leonard	male	2.0	0
9	2	Montvila, Rev. Juozas	male	27.0	1

Task *T*: Vi har lyst til å finne ut noe fra dataene

Avhengig av dataene og problemet kan vi velge mellom **veiledet**, uveiledet og forsterket læring.

Vi er ute etter å finne ut om passasjerer overlevde. Dataene inneholder en kolonne med dette svaret.

⇒ Vi kan gjøre veiledet læring

Vi omtaler variabelen med riktig svar som **target**, og **task T** er å estimere denne.

	x_1	x_2	x_3	y
$\mathbf{x}^{(1)}$	3	male	22.0	0
$\mathbf{x}^{(2)}$	1	female	38.0	1
$\mathbf{x}^{(3)}$	3	female	26.0	1
$\mathbf{x}^{(4)}$	1	female	35.0	1
$\mathbf{x}^{(5)}$	3	male	35.0	0
$\mathbf{x}^{(6)}$	3	male		0
$\mathbf{x}^{(7)}$	1	male	54.0	0
$\mathbf{x}^{(8)}$	3	male	2.0	0
$\mathbf{x}^{(9)}$	2	male	27.0	1

Features benevnes x_n
Target benevnes y

Q:
Hva har skjedd med datasettet?
Hvorfor?

Task *T*: Vi har lyst til å finne ut noe fra dataene

Notasjon:

Hver rad i dataene representerer et datapunkt: Indeks i oppe

Hver kolonne representerer en egenskap (feature) og target: Indeks i nede / bold $\mathbf{x}$ for å indikere vektor

	x_1	x_2	x_3	y
$\mathbf{x}^{(1)}$	3	male	22.0	0
$\mathbf{x}^{(2)}$	1	female	38.0	1
$\mathbf{x}^{(3)}$	3	female	26.0	1
$\mathbf{x}^{(4)}$	1	female	35.0	1
$\mathbf{x}^{(5)}$	3	male	35.0	0
$\mathbf{x}^{(6)}$	3	male		0
$\mathbf{x}^{(7)}$	1	male	54.0	0
$\mathbf{x}^{(8)}$	3	male	2.0	0
$\mathbf{x}^{(9)}$	3	female	27.0	1

$$\underbrace{(\text{Pclass, Name, Sex, Age})}_{\text{features } \mathbf{x}^{(i)}=(x_1^{(i)},x_2^{(i)}\ldots)} \rightarrow \underbrace{\text{Survived}}_{\text{target } y^{(i)}}$$

Vi kan gjøre veiledet læring (supervised learning) fordi vi har et target y per datapunkt

norwegian open ai lab

NTNU

Task *T*: Vi har lyst til å finne ut noe fra dataene

Vi har: variabler $\boldsymbol{x}$, targets y.

Vi vil: Estimere y for **nye** $\boldsymbol{x}$, basert på relasjonene representert mellom **kjente** $\boldsymbol{x}$ og y.

Vi kan lage plott for å studere sammenhengene mellom $\boldsymbol{x}$-ene og y.

	x_1	x_2	x_3	y
$\mathbf{x}^{(1)}$	3	male	22.0	0
$\mathbf{x}^{(2)}$	1	female	38.0	1
$\mathbf{x}^{(3)}$	3	female	26.0	1
$\mathbf{x}^{(4)}$	1	female	35.0	1
$\mathbf{x}^{(5)}$	3	male	35.0	0
$\mathbf{x}^{(6)}$	3	male		0
$\mathbf{x}^{(7)}$	1	male	54.0	0
$\mathbf{x}^{(8)}$	3	male	2.0	0
$\mathbf{x}^{(9)}$	3	female	27.0	1

Q:
Hva gjenstår? Hvorfor?

Preprosessere dataene

1) For plotting / maskinlæring trengs numeriske verdier. Vi må encode x_2.
2) Rad 6 mangler verdi for x_3. Noe må gjøres. Options: Fjerne raden, fjerne hele x_3, eller erstatte den manglende verdien med noe (imputering)
3) Størrelsesorden til x_3 er en helt annen enn for x_1 og x_2 etter encoding.

	x_1	x_2	x_3	y
$x^{(1)}$	3	male	22.0	0
$x^{(2)}$	1	female	38.0	1
$x^{(3)}$	3	female	26.0	1
$x^{(4)}$	1	female	35.0	1
$x^{(5)}$	3	male	35.0	0
$x^{(6)}$	3	male		0
$x^{(7)}$	1	male	54.0	0
$x^{(8)}$	3	male	2.0	0
$x^{(9)}$	3	female	27.0	1

Preprosessere dataene

```python
df = pd.read_csv("data/titanic/train.csv")

df = df[["Pclass","Sex","Age","Survived"]]
df.dropna(inplace=True)

features = df.columns

# Column Sex is type string. Encode it to numerical
label_encoder = LabelEncoder()
df["Sex"] = label_encoder.fit_transform(df["Sex"])

min_max_scaler = MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(df)
df = pd.DataFrame(x_scaled, columns=features)

# Check which sex was labelled as what
classes = label_encoder.classes_
label_mapping = {label: idx for idx, label in enumerate(classes)}
print(label_mapping)

df.head(10)
```

```
{'female': 0, 'male': 1}
```

	Pclass	Sex	Age	Survived
0	1.0	1.0	0.271174	0.0
1	0.0	0.0	0.472229	1.0
2	1.0	0.0	0.321438	1.0
3	0.0	0.0	0.434531	1.0
4	1.0	1.0	0.434531	0.0
5	0.0	1.0	0.673285	0.0
6	1.0	1.0	0.019854	0.0
7	1.0	0.0	0.334004	1.0
8	0.5	0.0	0.170646	1.0
9	1.0	0.0	0.044986	1.0

Vekk med NaNs (det finnes mer avanserte løsninger. Mer om det senere)

Encoding av kategoriene i “Sex” til numeriske verdier

Skalering av feature-verdier (skaleres til [0,1] by default)
Obs: dette kan være risky. Se hvorfor når vi snakker om train/test-splitt!

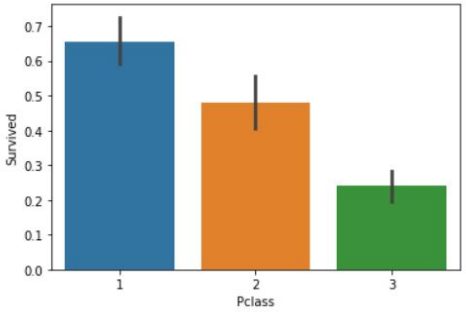
Sjekk hvilke kategorier female og male havnet i

norwegian open ai lab

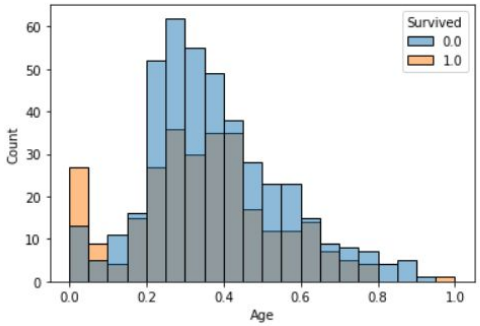
NTNU

Se på dataene

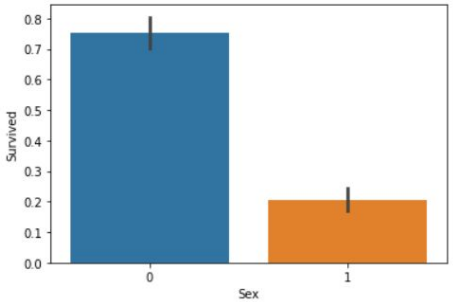
```
sns.barplot(x='Pclass', y='Survived', data=df); plt.show()
```



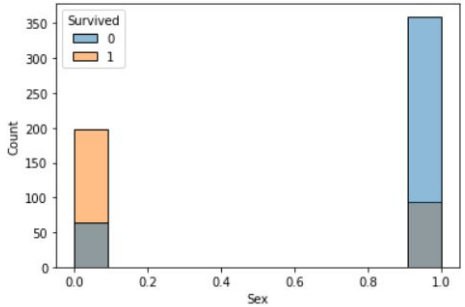

```
sns.histplot(data=df, x="Age", hue="Survived"); plt.show()
```




```
sns.barplot(x='Sex', y='Survived', data=df); plt.show()
```



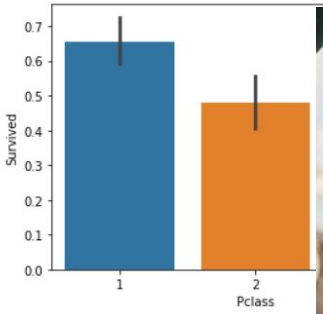

```
sns.histplot(data=df, x="Sex", hue="Survived"); plt.show()
```



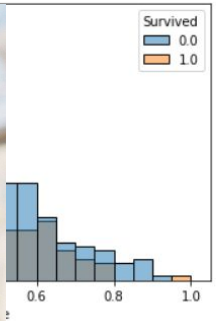
NTNU

Se på dataene

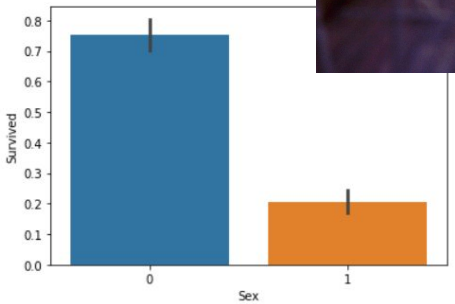
```
sns.barplot(x='Pclass', y='Survived', data=df); plt.show()
```



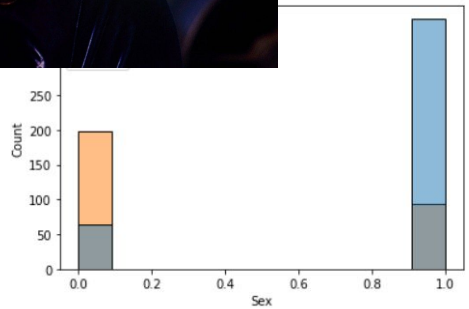

```
sns.histplot(data=df, x="Age", hue="Survived"); plt.show()
```



```
sns.barplot(x='Sex', y='Survived
```



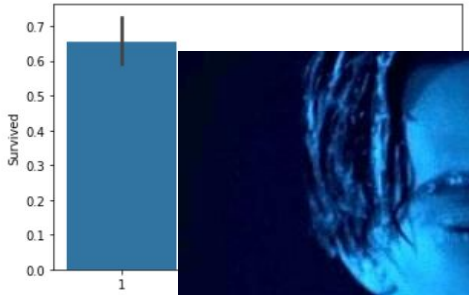

```
x", hue="Survived"); plt.show()
```



NTNU

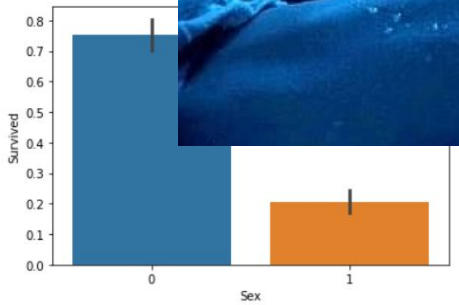
Se på dataene

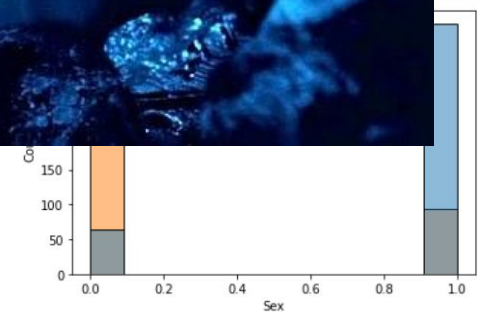
```
sns.barplot(x='Pclass', y='Survived', data=df); plt.show()
```




```
sns.histplot(data=df, x="Age", hue="Survived"); plt.show()
```


```
sns.barplot(x='Sex
                                                    ved"); plt.show()
```





NTNU

Task *T*: Vi har lyst til å finne ut noe fra dataene

Vi har: variabler $\boldsymbol{x}$, targets y.

Vi vil: Estimere y for nye $\boldsymbol{x}$, basert på relasjonene representert mellom kjente $\boldsymbol{x}$ og y.

Vi trenger en **modell** f som estimerer $p(y|\boldsymbol{x})$

En modell som gjør dette, gir oss prediksjoner: $f(\boldsymbol{x})=y_{\text{pred}}$

Modell for $p(y|x)$?

Den enkleste modellen vi kan tenke oss er lineær regresjon: $z = \sum_{i=1}^{n} w_i x_i + b = \mathbf{w}\mathbf{x} + b$

Vektene w_i forteller oss hvor viktig hver feature x_i i $\boldsymbol{x}$ er for utfallet

Spørsmål: Er z et tall eller en vektor?

norwegian open ai lab

NTNU

Modell for $p(y|x)$?

Den enkleste modellen vi kan tenke oss er lineær regresjon: $z = \sum_{i=1}^{n} w_i x_i + b = \mathbf{w}\mathbf{x} + b$

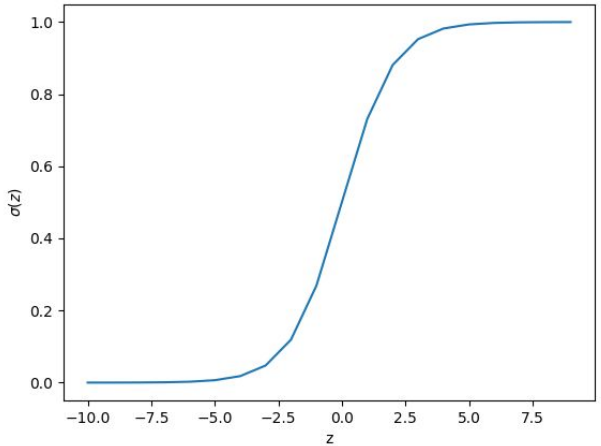
Vektene w_i forteller oss hvor viktig hver feature x_i i $\boldsymbol{x}$ er for utfallet

Problem: verdien til z kan bli hva som helst; vi er ute etter sannsynlighet for klasse.

Løsning: den logistiske funksjonen $\sigma(z)$ (aka sigmoid-funksjonen, fordi den ser ut som en S), mapper reelle tall til domenet (0,1).

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Vi kommer til å møte denne igjen når vi ser på nevrale nettverk, da under navnet aktiveringsfunksjon.



norwegian open ai lab

NTNU

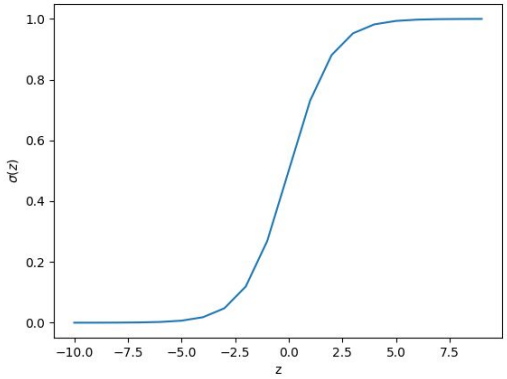
Modell for $p(y|x)$?

Den enkleste modellen vi kan tenke oss er lineær regresjon: $z = \sum_{i=1}^{n} w_i x_i + b = \mathbf{w}\mathbf{x} + b$

Vi mapper z til domenet (0,1) gjennom funksjonen $\sigma(z) = \frac{1}{1+e^{-z}}$

Modellens prediksjoner (estimert y) per datapunkt $\boldsymbol{x}$ er $y_{\text{pred}} = \frac{1}{1+e^{-\sum_i w_i x_i + b}}$

Dette kalles **logistisk regresjon**, pga den logistiske funksjonen σ(z)



norwegian open ai lab

NTNU

Performance measure *P*: hvor godt klarer modellen å estimere *y*?

Læringsprosess: tilpass modellen *f* slik at måloppnåelse øker, basert på data.

I maskinlæring omtales data som brukes til dette som **treningsdata** (training data), og funksjonen som bruker til å beregne måloppnåelse som **tapsfunksjon** (loss function).

Tapsfunksjonen forteller oss for hvert datapunkt (*x*,*y*) hvor nærme prediksjonen er til den riktige outputen:

$$\mathcal{L}(y_{\text{pred}}, y)$$

Hva skal vi velge som tapsfunksjon?

Performance measure *P*: hvor godt klarer modellen å estimere *y*?

Læringsprosess: tilpass modellen *f* slik at måloppnåelse øker, basert på data.

I maskinlæring omtales data som brukes til dette som **treningsdata** (training data), og funksjonen som bruker til å beregne måloppnåelse som **tapsfunksjon** (loss function).

Tapsfunksjonen forteller oss for hvert datapunkt *(x,y)* hvor nærme prediksjonen er til den riktige outputen:

$$\mathcal{L}(y_{\text{pred}}, y)$$

Hva skal vi velge som tapsfunksjon?

Vi ønsker å maksimere likelihood for riktig kategorisering av treningsdataene:

$$p(y|\mathbf{x}) = y_{\text{pred}}^{y}(1 - y_{\text{pred}})^{1-y}$$

(altså når y_{pred}=1 hvis y=1, og y_{pred}=0 hvis y=0)

Tapsfunksjon

Vi tar logaritmen på begge sider. (Logaritmen er en monoton funksjon, så det er lov)

$$\begin{aligned}\ln p(y|\mathbf{x}) &= \ln\left(y_{\text{pred}}^{y}(1-y_{\text{pred}})^{1-y}\right) \\ &= y\ln y_{\text{pred}} + (1-y)\ln(1-y_{\text{pred}})\end{aligned}$$

Log-likelihood skal maksimeres, så for å få en tapsfunksjon, legger vi på et minustegn, og ender opp med såkalt **cross-entropy loss**:

$$\begin{aligned}\mathcal{L}(y_{\text{pred}}, y) &= -\ln p(y|\mathbf{x}) \\ &= -y\ln y_{\text{pred}} - (1-y)\ln(1-y_{\text{pred}}) \\ &= -y\ln\sigma\left(\mathbf{w}\mathbf{x}+b\right) - (1-y)\ln\left(1-\sigma(\mathbf{w}\mathbf{x}+b)\right)\end{aligned}$$

Bra! Nå kan vi finne ut hvilke verdier parametrene må ha for at tapet skal minimeres. Hvordan?

Tapsfunksjon

Modellen vår er definert av $\boldsymbol{w}$ og b. Disse omtales som modellens **parametre**: $\theta = (\boldsymbol{w}, b)$

Vi vil finne parametrene som minimerer tapet, i gjennomsnitt over alle datapunktene. Dette kalles å **trene modellen** på maskinlæringsspråk.

*Mer formelt: Vi vil velge parametrene θ som maksimerer log-likelihood for target-verdiene y for hvert datapunkt $\boldsymbol{x}$. Dette kalles **conditional maximum likelihood estimation***

Vi har altså optimaliseringsproblemet:

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f(\mathbf{x}^{(i)}; \theta), y^{(i)})$$

... og her *(som i dyp læring)* skal vi bruke en metode som heter **gradient descent**.

norwegian open ai lab

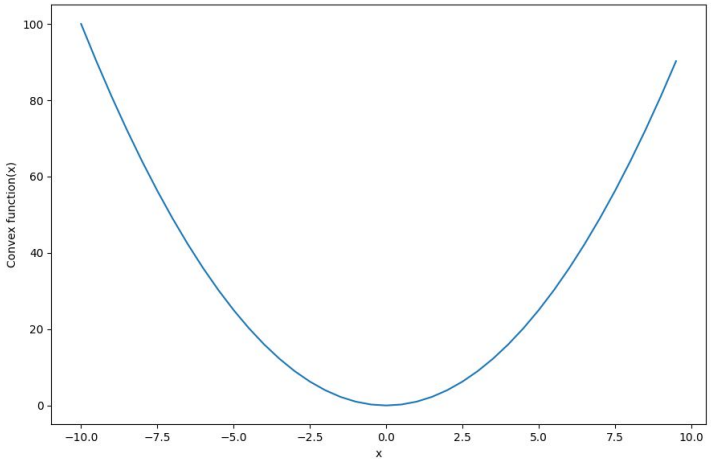
NTNU

Gradient descent

Gradient descent (gradient nedstigning - not gonna stick) er en metode for å finne minimumspunktet i en funksjon som man ikke kjenner formen til.

Strategien er å finne ut i hvilken retning (i rommet til parametrene θ) funksjonen minker brattest.

Intuisjon: Du går rundt i fjellet med bind for øynene og har lyst til å komme deg ned i dalen. Da finner du veien med brattest nedoverbakke, og går dit.

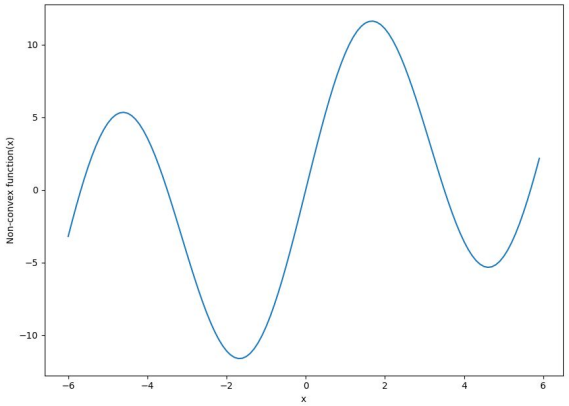


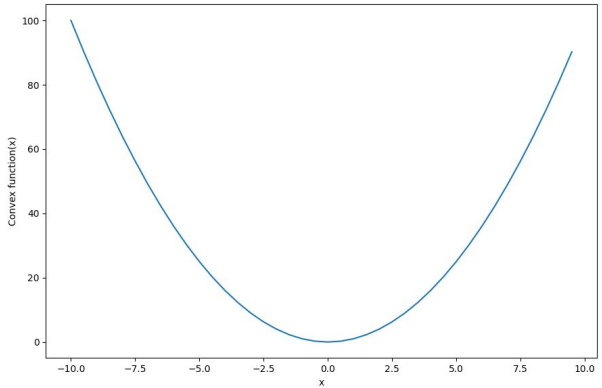
norwegian open ai lab

NTNU

Gradient descent

Potensiell utfordring: Det kan hende at du er i en fjellkjede og går deg vill mellom topper, aka lokale vs globale minima.



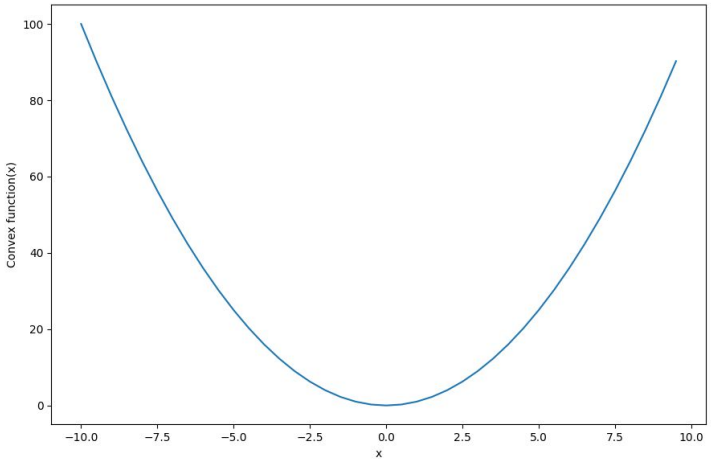


Heldigvis er tapsfunksjonen for logistisk regresjon konveks. En konveks funksjon har kun ett minimum. Derfor har vi en garanti om at gradient descent som finner et minimum har funnet et globalt minimum.

(Det samme skjer ikke for nevrale nettverk, som vi skal se på senere)

Gradient descent

Q: Hvilken retning bør vi bevege oss i? I hvilket rom?



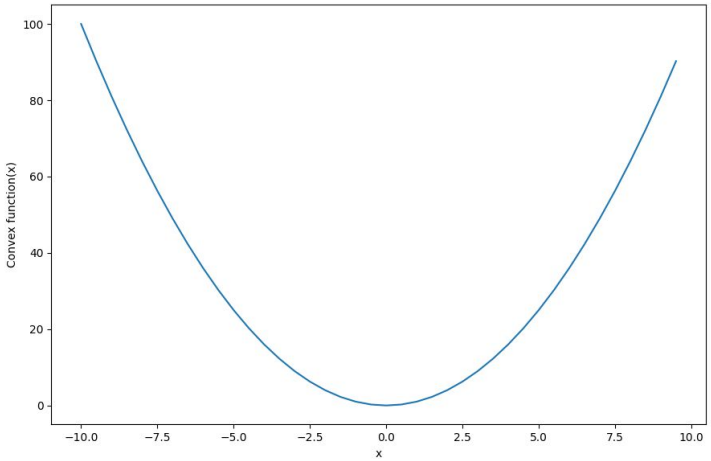
norwegian open ai lab

NTNU

Gradient descent

Q: Hvilken retning bør vi bevege oss i? I hvilket rom?

A: Den retningen i parameterrommet (rommet spent ut av θ-verdiene) der tapet $\mathcal{L}(y_{\text{pred}}, y)$ minker



norwegian open ai lab

NTNU

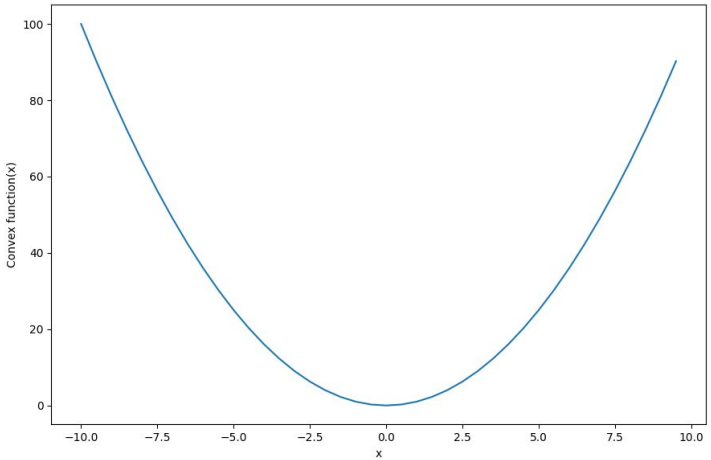
Gradient descent

Q: Hvilken retning bør vi bevege oss i? I hvilket rom?

A: Den retningen i parameterrommet (rommet spent ut av θ-verdiene) der tapet $\mathcal{L}(y_{\text{pred}}, y)$ minker

⇒ vi trenger .. et mål på hvordan $\mathcal{L}$ endrer seg… som funksjon av endring i θ…

Hva er dette?



norwegian open ai lab

NTNU

Gradient descent

Q: Hvilken retning bør vi bevege oss i? I hvilket rom?

A: Den retningen i parameterrommet (rommet spent ut av θ-verdiene) der tapet $\mathcal{L}(y_{\text{pred}}, y)$ minker

⇒ vi trenger **den deriverte av $\mathcal{L}$ med hensyn på θ**.

Her skjønner vi noe livsviktig (innen ML): **Tapsfunksjonen må være deriverbar.** Husk det. For alltid.

Gradient descent

Q: Hvor store steg skal vi ta i denne retningen?

A: Dette må vi bestemme, og størrelsen kalles læringsraten (learning rate) η til modellen.

Høyere læringsrate innebærer større endringer av parametrene θ for hvert steg.

Oppdateringen av parametrene må altså gjøres som:

$$\theta^{t+1} = \theta^t - \eta \frac{\partial}{\partial \theta} \mathcal{L}(f(\mathbf{x}; \theta), y)$$

Mens θ er **parametrene** som tilpasses for at modellen skal passe til dataene, er læringsraten eksempel på en **hyperparameter**. Dette er parametre som definerer læringsprosessen; ikke modellen.

Viktig setning. Mediter litt på den.

Gradient descent: Derivere tapsfunksjonen

$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{\partial}{\partial w_j}\left[-y \ln \sigma\left(\mathbf{wx}+b\right) - (1-y) \ln\left(1-\sigma(\mathbf{wx}+b)\right)\right]$$

(og tilsvarende for b)

Nyttige relasjoner:

$$\frac{\partial}{\partial x} \ln x = \frac{1}{x}$$

$$\frac{\partial}{\partial z}\sigma(z) = \sigma(z)\left(1-\sigma(z)\right)$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial u}\frac{\partial u}{\partial x}$$

Gradient descent: Derivere tapsfunksjonen

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_j} &= \frac{\partial}{\partial w_j}\left[-y \ln \sigma\left(\mathbf{w}\mathbf{x}+b\right)-(1-y) \ln \left(1-\sigma(\mathbf{w}\mathbf{x}+b)\right)\right] \\
&= -y \frac{1}{\sigma(\cdot)} \frac{\partial}{\partial w_j} \sigma(\mathbf{w}\mathbf{x}+b)+(1-y) \frac{1}{1-\sigma(\cdot)} \frac{\partial}{\partial w_j}\left(1-\sigma(\mathbf{w}\mathbf{x}+b)\right) \\
&= \frac{-y\left(1-\sigma(\cdot)\right)+(1-y) \sigma(\cdot)}{\sigma(\cdot)(1-\sigma(\cdot))} \frac{\partial}{\partial w_j} \sigma(\mathbf{w}\mathbf{x}+b) \\
&= \frac{\sigma(\cdot)-y}{\sigma(\cdot)\left(1-\sigma(\cdot)\right)} \sigma(\cdot)\left(1-\sigma(\cdot)\right) \frac{\partial}{\partial w_j}(\mathbf{w}\mathbf{x}+b) \\
&= \left(\sigma(\mathbf{w}\mathbf{x}+b)-y\right) x_j \\
&= \left(y_{\text{pred}}-y\right) x_j \\
\frac{\partial \mathcal{L}}{\partial b} &= \ldots \\
&= \left(\sigma(\mathbf{w}\mathbf{x}+b)-y\right) \frac{\partial}{\partial b}(\mathbf{w}\mathbf{x}+b) \\
&= \left(y_{\text{pred}}-y\right)
\end{aligned}
$$

norwegian open ai lab

NTNU

$$\frac{\partial L}{\partial w_j} = \frac{\partial}{\partial w_j}\Big[\underbrace{-y \ln \sigma(wx+b)}_{1} \underbrace{-(1-y)\ln(1-\sigma(wx+b))}_{2}\Big]$$

1) $f = \ln \sigma(\cdot)$, $u = \sigma(\cdot)$

$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial u}\frac{\partial u}{\partial w} = \frac{1}{\sigma(\cdot)} \cdot \frac{\partial \sigma(\cdot)}{\partial w}$$

2) $u = \sigma(\cdot)$, $f(u) = \ln(1-u)$

$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial u}\frac{\partial u}{\partial w} = \frac{\partial}{\partial u}\ln(1-u) \cdot \frac{\partial \sigma(\cdot)}{\partial w}$$

$v = 1-u$

$$\frac{\partial}{\partial u}\ln(1-u) = \frac{\partial \ln v}{\partial v}\frac{\partial v}{\partial u} = \frac{1}{v}\cdot(-1)$$

$$\Rightarrow \frac{\partial}{\partial w}\ln(1-\sigma(\cdot)) = -\frac{1}{1-\sigma(\cdot)}\frac{\partial \sigma(\cdot)}{\partial w}$$

$$= -y\frac{1}{\sigma(\cdot)}\frac{\partial}{\partial w_j}\sigma(\cdot) + (1-y)\frac{1}{1-\sigma(\cdot)}\frac{\partial \sigma(\cdot)}{\partial w_j}$$

Teilen Bruchstrich $\cdot \frac{1-\sigma}{1-\sigma}$ $\quad \frac{\sigma}{\sigma}$

$$= \frac{-y(1-\sigma(\cdot)) + (1-y)\sigma(\cdot)}{\sigma(\cdot)(1-\sigma(\cdot))}\frac{\partial}{\partial w_j}\sigma(\cdot)$$

$$\frac{\partial}{\partial w}\sigma(w) = \sigma(w)(1-\sigma(w))$$

$$\frac{\partial}{\partial w_j}(wx+b) = x_j$$

$$= -y + \sigma(\cdot)\frac{\partial}{\partial w_j}(wx+b) = (-y+\sigma(\cdot))x_j$$

norwegian open ai lab

NTNU

Læringsalgoritmen

Nødvendigvis funksjonalitet for å gjøre logistisk regresjon:

```
class LogisticRegression:
      def __init__(self, learning_rate=0.1, epochs=1000):
      self.learning_rate = learning_rate
      self.epochs = epochs
      self.weights, self.bias = None, None
      self.losses, self.train_accuracies = [], []

def sigmoid_function(self, x)

def _compute_loss(self, y, y_pred)

def compute_gradients(self, x, y, y_pred)

def update_parameters(self, grad_w, grad_b)
```

norwegian open ai lab

NTNU

Læringsalgoritmen

```
def fit(self, x, y):
	self.weights = np.zeros(x.shape[1]) # x.shape = antall datapunkter, antall features
	self.bias = 0
	# Gradient Descent
	for _ in range(self.epochs): # Epochs: Antall ganger treningsdataene sendes gjennom læringsalgoritmen. Nok en hyperparameter.
		lin_model = np.matmul(self.weights, x.transpose()) + self.bias
		y_pred = self._sigmoid(lin_model)
		grad_w, grad_b = self.compute_gradients(x, y, y_pred)
		self.update_parameters(grad_w, grad_b)

		loss = self._compute_loss(y, y_pred)
		pred_to_class = [1 if _y > 0.5 else 0 for _y in y_pred]
		self.train_accuracies.append(accuracy(y, pred_to_class))
		self.losses.append(loss)

def predict(self, x):
	lin_model = np.matmul(x, self.weights) + self.bias
	y_pred = self._sigmoid(lin_model)
	return [1 if _y > 0.5 else 0 for _y in y_pred]
```

Dette er en **læringsalgoritme**. Det vi får ut er en **modell** med tilpassede parametre.
(samme ordbruk for nevrale nettverk osv)

norwegian open ai lab

NTNU

Treningen

Gullstandarden innen ML (ikke veldig fjong, medisin er feks mye strengere): Dataene som brukes for å trene modellen får ikke brukes til å evaluere den.

Enkel regel: Splitt dataene i to - **train** og **test** - og *aldri* rør test-dataene før modellen er ferdig.

```
# 1: Split data into train and test
X_train, X_test, y_train, y_test = train_test_split(X_data, y_data, test_size=0.2)

# 2: Do fit-stuff only on the train part, and only transform-stuff on the test part
min_max_scaler = MinMaxScaler()
X_train = min_max_scaler.fit_transform(X_train)
X_test = min_max_scaler.transform(X_test)
```

norwegian open ai lab

NTNU

Treningen

Kjør trening, ved hjelp den fine klassen og funksjonene du laget.

```
# Training
train_epochs = 30

# Initialize and train the model
log_reg = LogisticRegression_(learning_rate=0.01, epochs=train_epochs)
log_reg.fit(X_train, y_train)

# Make predictions
predictions = log_reg.predict(X_test)
```

norwegian open ai lab

NTNU

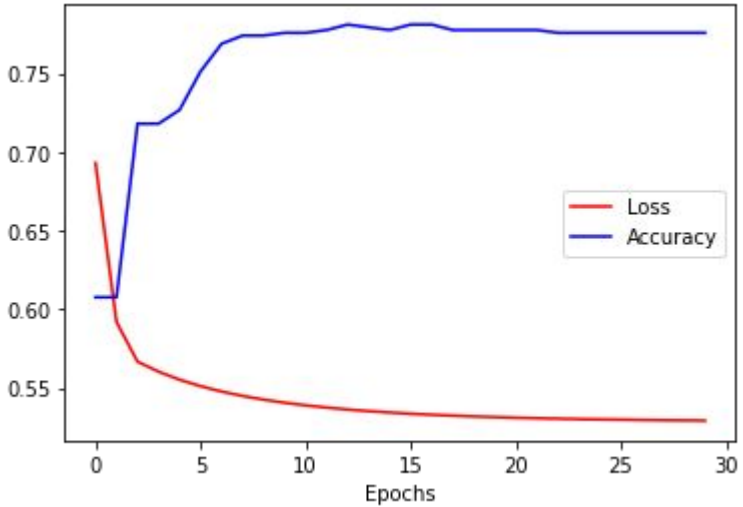
Treningen

Hvordan ser vi om modellen blir bedre under trening? Vi kan se på om tapet (loss) minker, og om treffsikkerhet (accuracy) øker.

```
def accuracy(true_values, predictions):
       return np.mean(true_values == predictions)

print("Accuracy: ", accuracy(y_test, predictions))

epoch_list = np.arange(0, train_epochs,1)
plt.plot(epoch_list, log_reg.losses, c='red', label="Loss")
plt.plot(epoch_list, log_reg.train_accuracies, c='blue', label="Accuracy")
plt.legend()
plt.show()
```



norwegian open ai lab

NTNU

Evaluere modellen

Evalueringsmetrikk og tapsfunksjon **kan ikke være det samme**.

Hvorfor?

Evaluere modellen

Evalueringsmetrikk og tapsfunksjon **kan ikke være det samme**.

Metrikken må gi mening for mennesker, og trenger feks ikke å være deriverbar.

Accuracy = treffsikkerhet er fin, men forteller ikke hele historien.

Hvilke typer feil gjør modellen typisk? (tror den at folk som dør overlever eller motsatt?)

For klassifiseringsmodeller gjelder **confusion matrix**:

		Predicted	
	Total (P+N)	Negative (PN)	Positive (PP)
True	Negative (N)	True negative	False positive
	Positive (P)	False negative	True positive

norwegian open ai lab

NTNU

Confusion matrix

Basert på verdiene i confusion matrix (forvirringsmatrisen, not gonna stick) kan vi regne ut flere metrikker:

True Positive Rate (TPR) = TP / P = TP / (TP + FN) = 0.60 *aka sensitivitet, recall, sannsynlighet for deteksjon,*

True Negative Rate (TNR) = TN / N = TN / (TN + FP) = 0.85 *aka spesifisitet,*

False Positive Rate (FPR) = FP / N = FP / (FP + TN) = 0.15 *aka sannsynligheten for falsk alarm, (1 – spesifisitet),*

Positive Predictive Value (PPV) = TP / (TP + FP) = 0.72 *aka precision*

... osv.

		Predicted	
	Total (P+N)	Negative (PN)	Positive (PP)
True	Negative (N)	73	13
	Positive (P)	23	34

norwegian open ai lab

NTNU

Evaluere modellen

Husk: Prediksjonene er tall i (0,1), og vi brukte en terskel for å bestemme hva som svarte til PP og PN

```
pred_to_class = [1 if _y > 0.5 else 0 for _y in y_pred]
```

Avhengig av denne terskelen skårer klassifiseringsmodeller ulikt på metrikkene

Q: Er det best å være overforsiktig eller konservativ?

A: Det er avhengig av bruksområdet!

Story time!

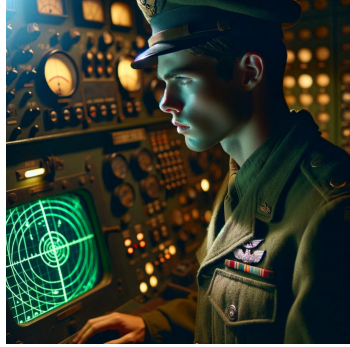
Under andre verdenskrig satt folk og så på radarer, og prøvde å se forskjell på fly (positiv klasse) og ikke-fly, feks gjess (negativ klasse).

Disse folka gjorde **receiver operating**; en klassifiseringsoppgave. Noen av dem var nervøse, noen i overkant kule ⇒ de opererte med ulike **klassifiseringsterskler**.

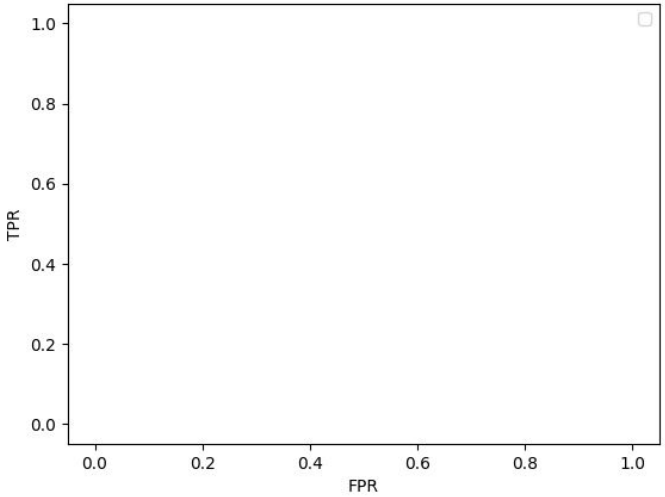
For å analysere **FPR** og **TPR** som funksjon av terskelen, ble **ROC - Receiver Operating Characteristic** - kurven utviklet.

Denne viser FPR og TPR for ulike klassifiseringsterskler.

ROC-kurven



Operatøren er supernervøs (terskel=0)

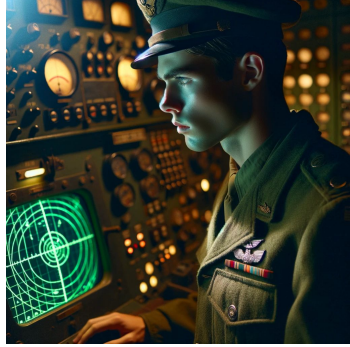


Operatøren sover (terskel>=1)

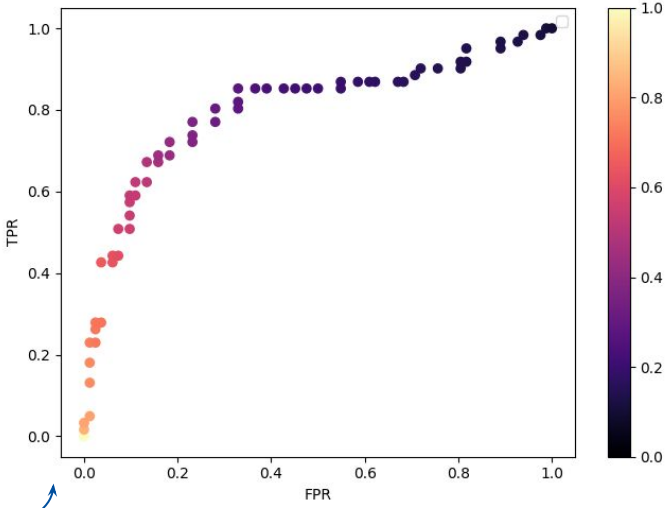
norwegian open ai lab

NTNU

ROC-kurven



Operatøren er supernervøs (terskel=0)



Klassifiseringsterskler

Operatøren sover (terskel>1)

norwegian open ai lab

NTNU

ROC-kurven og arealet under

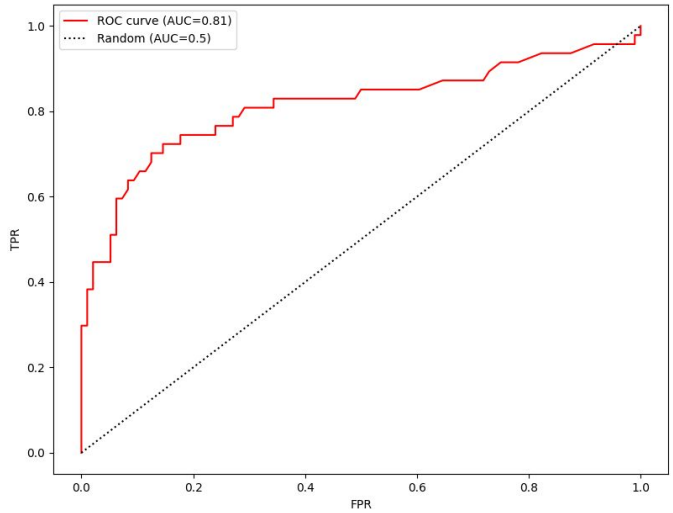
Vi ønsker høy TPR og lav FPR.

Jo høyere TPR per FPR, jo bedre er modellen, for alle terskler.

Myntkast gir diagonalen (0,0)-(1,1).

For å være bedre enn tilfeldig gjetning må kurven

- ligge høyere enn diagonalen, og
- **arealet under kurven (AUC)** være større enn 0.5.



norwegian open ai lab

NTNU

ROC-kurven og arealet under

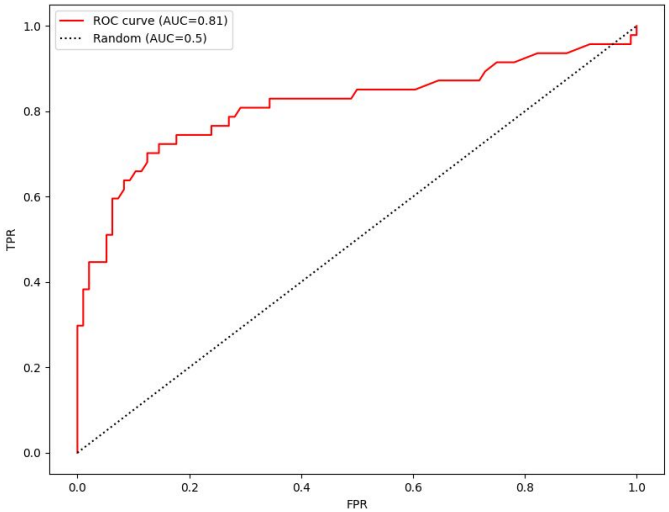
```
def predict_proba(self, x):
        lin_model = np.matmul(x, self.weights) + self.bias
        y_pred = self._sigmoid(lin_model)
        #return [1 if _y > 0.5 else 0 for _y in y_pred] ← ikke den, men
        return y_pred

prob_predictions = log_reg.predict_proba(X_test)

fpr, tpr, thresholds = metrics.roc_curve(y_test, prob_predictions, pos_label=1)

auc = round(metrics.roc_auc_score(y_test, prob_predictions),2)

plt.plot(fpr, tpr, label=f"ROC curve (AUC={auc})", c="red")
plt.plot([0,1], [0,1], c="black", ls="dotted", label="Random (AUC=0.5)")
```



norwegian open ai lab

NTNU

AUC: sannsynlighetstolkning

AUC = **sannsynligheten for at modellen vil gi predikere en høyere verdi** (nærmere 1 enn 0) for et tilfeldig valgt **positivt datapunkt** enn for et tilfeldig valgt negativt datapunkt.

Precision recall

Presisjon = TP / (TP + FP)
= korrekt predikert positive / predikert positive

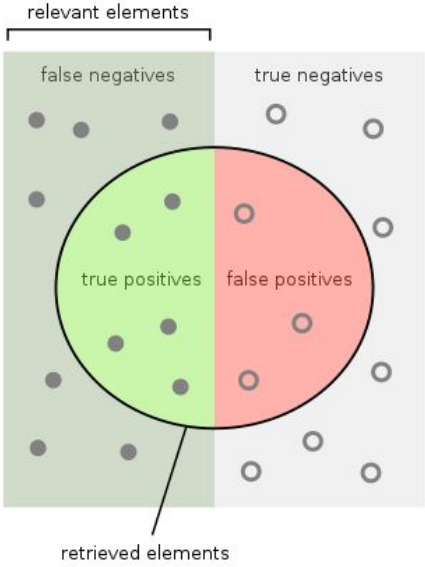
Høy presisjon ⇒ lavt antall falske alarmer

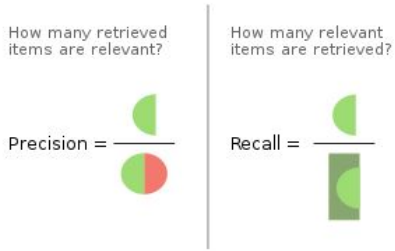
Recall = TP / (TP + FN)
= korrekt predikert positive / alle positive

Høy recall ⇒ lavt antall missed cases

Precision recall tradeoff: når den ene øker, minker den andre. Hva som er en god balanse er avhengig av anvendelsen.

Hvis falsk alarm er dyrt, er høy presisjon viktig (feks utrykning).
Hvis kostnaden med å bomme på en positiv instans er høy, er høy recall viktig (feks diagnostisering av alvorlige sykdommer)





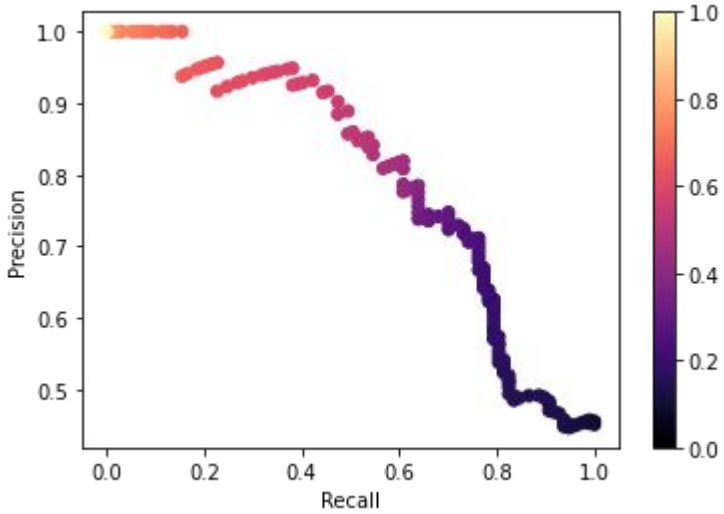
norwegian open ai lab

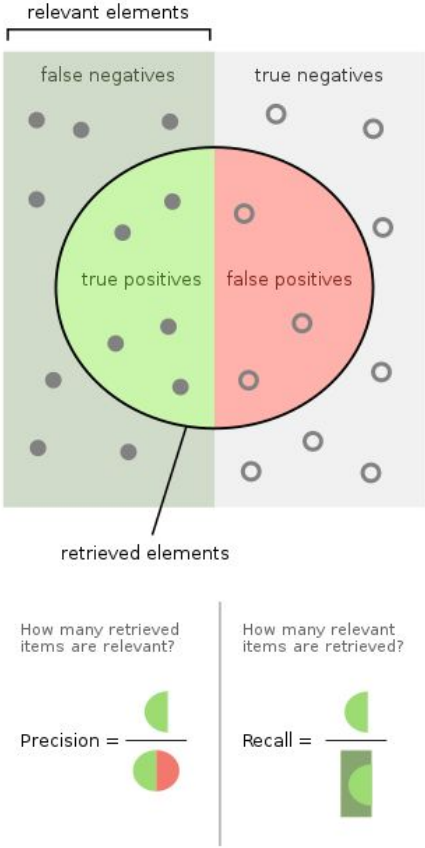
NTNU

Precision recall-kurven

Presisjon = TP / (TP + FP)

Recall = TP / (TP + FN)





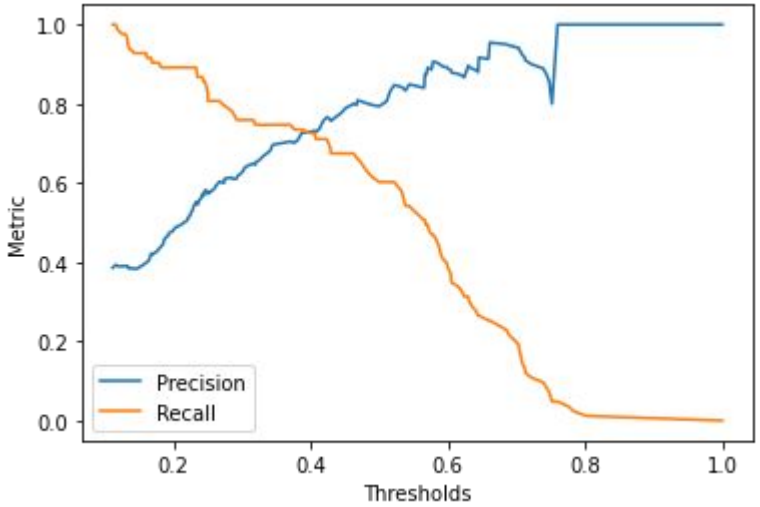
norwegian open ai lab

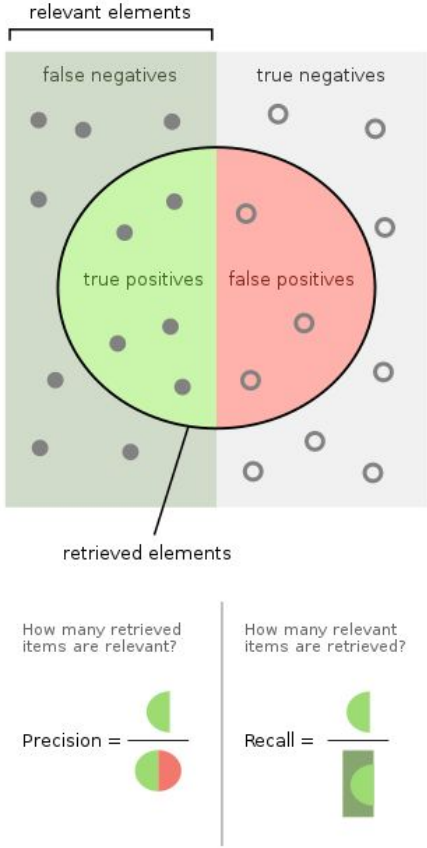
NTNU

Precision recall-kurven

Presisjon = TP / (TP + FP)

Recall = TP / (TP + FN)





norwegian open ai lab

NTNU

Hva har vi så langt?

Machine learning pipeline:

1) Hente, renske og splitte data
2) Tilpasse modell ved bruk av tapsfunksjon og optimaliseringsalgoritme
3) Evaluere modell ved bruk av ulike metrikker.

Det finnes *mange*

- mer avanserte måter å splitte data
- vanskeligere datasett
- flere oppgaver modeller kan løse
- flere tapsfunksjoner
- flere typer modell
- flere evalueringsmetrikker.

Men vi har vært gjennom alle de grunnleggende stegene i en maskinlæringsprosess.

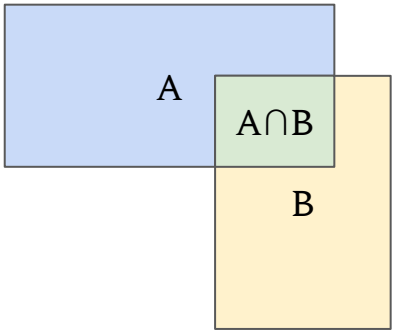
Sannsynlighetstolkning

Vi lagde en modell for å estimere sannsynligheten for klasse, gitt observasjoner: $p(c|x)$

Dette er en **betinget sannsynlighet**: hva er sannsynligheten for c, betinget på x (gitt x)?

For å skjønne betingede sannsynligheter, må vi innom Bayesiansk sannsynlighetstolkning.

Bayes' teorem på 2 sek



$$P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

norwegian open ai lab

NTNU

Bayes' teorem i klassifiseringssammenheng

Vi ønsker å estimere sannsynligheten for en klasse for et datapunkt, altså $p(C_k|x)$. Fra Bayes' teorem har vi

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)}$$

Bayes' teorem i klassifiseringssammenheng

Vi ønsker å estimere sannsynligheten for en klasse for et datapunkt, altså $p(C_k|x)$. Fra Bayes' teorem har vi

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)}$$

$p(C_k)$: sannsynligheten for klassen (hypotesen), uten ytterligere informasjon (*prior probability*)

$p(x)$: sannsynlighetsfordelingen av dataene (*evidence*)

$p(x|C_k)$: sannsynlighet for å observere dataene x gitt en klasse (gitt at hypotesen stemmer) (*likelihood*)

$p(C_k|x)$: sannsynlighet for klasse (hypotesen) i lys av data. Oppdatert prior. (*posterior*)

norwegian open ai lab

NTNU

Bayes' teorem i klassifiseringssammenheng

Dette er NØYAKTIG det vi prøver å estimere med maskinlæring.

Ergo: Hvis vi klarer å beregne alle faktorene i Bayes' teorem trenger vi aldri mer bry oss med maskinlæring.

posterior: sannsynlighet for klasse (hypotesen) i lys av data. Oppdatert prior.

What now?

Start på første øving :)

Tenk over hva dere gjør, **hvorfor** det funker/ikke funker, og hva resultatene forteller dere.

Hvorfor heter det *cross entropy* loss?

Fra informasjonsteori: Entropi (også: Shannon entropy) er et mål på tilfeldigheten (støyet) i en fordeling

$$H(p) = -\sum_i p(x_i) \log p(x_i))$$

Cross-entropy måler forskjellen mellom to fordelinger, feks den sanne fordelingen p og en estimert fordeling q, over de samme datapunktene

$$H(p, q) = -\sum_i p(x_i) \log q(x_i))$$

I supervised learning er målet å minimere forskjellen mellom den predikerte sannsynligheten y_{pred} og den faktiske sannsynligheten y. I binær klassifisering:

$$\mathcal{L}(y_{\text{pred}}, y) = -y \ln y_{\text{pred}} - (1 - y) \ln(1 - y_{\text{pred}})$$

Og generelt for flere klasser i: *(hatt for y_{pred} for å unngå dobbel subscript)*

$$\mathcal{L}(\hat{y}, y) = -\sum_i y_i \log(\hat{y}_i)$$

Tjohei