

<b>2a.1)</b>	Hvorfor ønsker vi å dele dataene inn i trening-, validering- og test-sett?
<b>Svar</b>	<p>Vi deler dataene inn i trening-, validering- og testsett for å utvikle, velge og til slutt evaluere en klassifiseringsmodell på en måte som sikrer at modellen generaliserer godt til ny, usett data. Dette gjøres for å unngå overtilpasning og for å kunne vurdere modellens sanne prediktive evne.</p> <ul style="list-style-type: none"> <li>□ <b>Treningssettet</b> brukes for å lære eller estimere modellens parametere.</li> <li>□ <b>Valideringssettet</b> benyttes for å velge og finjustere modellens hyperparametere og for å gjøre modellvalg uten å påvirke testsettet.</li> <li>□ <b>Testsettet</b> gir en objektiv vurdering av modellens ytelse på data den ikke har "sett" før, og dermed en indikasjon på hvordan den vil prestere i praksis.</li> </ul> <p>Ved å dele opp dataene på denne måten sikrer vi at vi kan utvikle modeller som er både tilpasset de dataene vi har, og som kan fungere godt i virkelige situasjoner, noe som er det ultimate målet med prediktiv modellering.</p>

<b>2a.2)</b>	Hvor stor andel av dataene er nå i hver av de tre settene? Ser de tre datasettene ut til å ha lik fordeling for de tre forklaringsvariablene og responsen?
<b>Svar</b>	<p>Først blir det opprinnelige datasettet delt i to deler, 80% for trening og validering og 20% for testing. Deretter blir de 80% delt for trening og validering delt opp igjen inn i 2 sett der hvor 75% av 80% blir treningssett og 25% av 80% blir valideringssett.</p> <p>Totalt sett så blir andelene med observasjonene:</p> <p>Treningssett 60% med 1221 av 2036 observasjoner  Valideringssett: 20% med 407 av 2036 observasjoner  Testsett 20% med 408 av 2036 observasjoner</p> <p>Ut fra disse observasjonene kan vi også si at de tre datasettene ser ut til å ha en lik fordeling for både forklaringsvariablene og responsen.</p>

<b>2a.3)</b>	La oss si at vi hadde valgt League 1 og 2 som treningssett, Championship som valideringssett, og Premier League som testsett. Hvorfor hadde dette vært dumt?
<b>Svar</b>	<p>De fire divisjonene i datasettet representerer ulike nivåer av konkurransedyktighet. Med Premier League som den høyeste og League 2 som den laveste divisjonen. Dette betyr at kvaliteten på kampene kan variere betydelig mellom de forskjellige divisjonene. Dette betyr at lagene i de lavere divisjonene kan spille en annen type fotball enn de i de høyere divisjonene. Dette kan påvirke antall skudd på mål, cornere og forseelser i kamper. En modell trent på data fra lavere divisjoner vil da ikke nødvendigvis kunne fange opp disse forskjellene. Hvis da treningssettet er basert på de lavere divisjonene og tester på Premier League, kan man ende opp med en modell som ikke klarer å generalisere godt på grunn av disse statistiske forskjellene.</p>

<b>2a.4)</b>	Kommenter kort på hva du ser i plottene og utskriften (maks 5 setninger).
<b>Svar</b>	Det vi ser i plottene er at forklaringsvariabelen <code>skudd_paa_maal_diff</code> ser ut til å ha høyest korrelasjon med at hjemmelaget vinner både i plottene og i utskriften. Hvis <code>skudd_paa_maal_diff</code> er positiv så er det veldig høy sjanse for hjemmeseier. De andre to forklaringsvariablene <code>corner_diff</code> og <code>forseelse_diff</code> ser ut til å ha liten relasjon til Y om hjemmelaget vinner. Litt vanskelig å tyde om det er noe reell relasjon mellom Y og <code>corner_diff</code> og <code>forseelse_diff</code> bare ved å visuelt se på plottene og utskriften, men <code>skudd_paa_maal_diff</code> ser ut til å være bedre til å predikere om det blir hjemmeseier.

<b>2a.5)</b>	Hvilke(n) av de tre variablene tror du vil være god(e) til å bruke til å predikere om det blir hjemmeseier? Begrunn svaret kort (maks 3 setninger).
<b>Svar</b>	Forklaringsvariabelen <code>skudd_paa_maal_diff</code> ser ut til å være den beste predikatoren for om det blir hjemmeseier. Dette kan vi se ut ifra korrelasjonskoeffisient i tetthetsplottet og vi ser at den har høyest positiv verdi med Y at det blir hjemmeseier.

<b>2b.1)</b>	I en kamp der <code>skudd_paa_maal_diff</code> er 2, <code>corner_diff</code> er -2 og <code>forseelse_diff</code> er 6, hva er ifølge modellen sannsynligheten for at hjemmelaget vinner? Vis utregninger og/eller kode, og oppgi svaret med tre desimaler.
<b>Svar</b>	<pre># Definere koeffisientene fra modellen intercept = -0.591661 skudd_paa_maal_diff_coef = 0.382565 corner_diff_coef = -0.100377 forseelse_diff_coef = 0.012009  # Definere differanse verdiene skudd_paa_maal_diff_value = 2 corner_diff_value = -2 forseelse_diff_value = 6  # Kalkulere log-oddsen for at hjemmelaget vinner log_odds = (intercept +             (skudd_paa_maal_diff_coef * skudd_paa_maal_diff_value) +             (corner_diff_coef * corner_diff_value) +             (forseelse_diff_coef * forseelse_diff_value))  # Konvertere log-oddsen til sannsynlighet odds = np.exp(log_odds) probability = odds / (1 + odds) print(probability)</pre> <p>Sannsynligheten for at hjemmelaget vinner med disse verdiene er 0.609 eller 61%.</p>

<b>2b.2)</b>	Hvordan kan du tolke verdien av $e^{\beta_{\text{skudd-paa-maal-diff}}}$ ?
<b>Svar</b>	$e^{\beta_{\text{skudd-paa-maal-diff}}} = 1.466040163871275$  Siden verdien er over 1 betyr det at det er en positiv sammenheng mellom <code>skudd_paa_maal_diff</code> og at hjemmelaget vinner. Vi kan tolke selve verdien av $e^{\beta}$ med at oddsen for at hjemmelaget vinner øker med 46% for hvert mål hjemmelaget har over bortelaget.

<b>2b.3)</b>	Hva angir feilraten til modellen? Hvilket datasett er feilraten regnet ut fra? Er du fornøyd med verdien til feilraten?
<b>Svar</b>	Modellen er nå trent på treningsdataen og bruker dette til å først predikere om hjemmelaget vinner basert på om den beregnede sannsynligheten for at hjemmelaget vinner er større enn 0.5. Etter denne prediksjonen sammenlignes det predikerte utfallet med det faktiske utfallet av kampen for å bestemme om prediksjonen var korrekt. Dette er hvordan modellens nøyaktighet og feilrate blir angitt og evaluert.  Feilraten til modellen er 0.285, men om vi kan anse oss fornøyd med denne verdien er avhengig av kritikaliteten rundt denne feilraten. Skal resultatet f.eks brukes til gambling så kan denne feilraten regnes som veldig gunstig, både grunnet fotballs uforutsigbare natur og hvor ofte du faktisk ville vunnet veddemålet (nesten 72% av tiden.)

<b>2b.4)</b>	Diskuter kort hvordan koeffisientene ( $\beta$ – <b>ene</b> ) og feilraten endrer seg når <code>forseelse_diff</code> tas ut av modellen (maks 3 setninger).
<b>Svar</b>	Koeffisientene kan endre seg fordi modellen nå har færre variabler å forklare variansen i dataen på, nå uten informasjonen fra <code>forseelse_diff</code> . Feilraten kan også endre seg avhengig av hvor viktig <code>forseelse_diff</code> variabelen var for modellen.  Koeffisientene endrer seg ikke engang bittelitt Men feilraten går ned (som er positivt) <b>TODO</b>

<b>2b.5)</b>	Med den nye modellen: I en kamp der <code>skudd_paa_maal_diff</code> = 2, <code>corner_diff</code> = -2 og <code>forseelse_diff</code> = 6, hva er sannsynligheten for at hjemmelaget vinner ifølge den nye modellen? Oppgi svaret med tre desimaler.
<b>Svar</b>	Med den nye modellen i en kamp med disse verdiene er sannsynligheten for at hjemmelaget vinner: 0.590 eller 59%.

<b>2b.6)</b>	Hvis du skal finne en så god som mulig klassifikasjonsmodell med logistisk regresjon, vil du velge modellen med eller uten <code>forseelse_diff</code> som kovariat? Begrunn kort svaret (maks 3 setninger).
<b>Svar</b>	Forskjellen i feilraten mellom modellen med <code>forseelse_diff</code> og uten er bare 0.003, eller 0.3%. Denne forskjellen i feilrate er lite markant, men med tanke på at også modellens kompleksitet går ned samtidig som feilraten så lyder dette godt i favør av å velge modellen uten <code>forseelse_diff</code> .

<b>2c.1)</b>	Påstand: kNN kan bare brukes når vi har maksimalt to forklaringsvariabler. Fleip eller fakta?
<b>Svar</b>	Dette er fleip. Det er dog litt vanskeligere å jobbe med høydimensjonale data for å opprettholde modellens nøyaktighet, i tillegg til den økte beregningskostnaden.

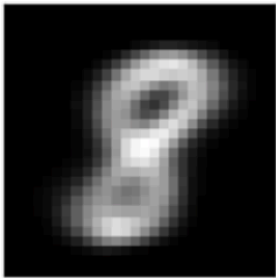

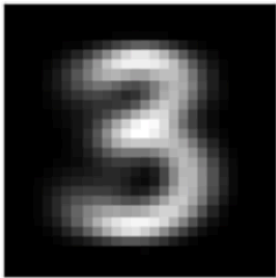
<b>2c.2)</b>	Hvilken verdi av <b>k</b> vil du velge?
<b>Svar</b>	<p>NTNU-Jupyter: 115 (Feilrate=0.2825)  Windows: 139 (Feilrate= 0.2776)  MacOS: 91 (Feilrate= 0.2800)</p> <p>Disse er de tre forskjellige k-verdiene vi fikk for «Minimum feilrate». Kort sagt blir den gunstigste k-verdien den verdien av k med den laveste feilraten på din spesifikke maskin og programvare-versjon. Valg av k-verdi er viktig for at modellen skal virke best mulig.</p>

<b>2d.1)</b>	Gjør logistisk regresjon eller <b>k</b> -nærmeste-nabo-klassifikasjon det best på fotballkampdataene?
<b>Svar</b>	<p>Med k-verdi lik 115 på NTNU's Jupyter-hub får både logistisk regresjon og kNN den nøyaktige samme feilraten.  Samme skjedde med både k-verdi lik 91 og 115 på Macbooken.</p> <p>Windows-maskinen vi prøvde, som fikk k-verdi 139, fikk kNN lavest feilrate med 0.31862 (logistisk regresjon med 0.31862), så her kan det derfor sies at kNN er bedre i dette tilfellet over logistisk regresjon, men tallene er relativt like, så det er liten forskjell på dem dog kNN har lavest feilrate.</p> <p>Alt i alt er det vanskelig å si hva som er best grunnet de blandede resultatene, men likevel er det 1-0 til kNN.</p> <pre>[21]: # beste resultat for logistisk regresjon bestelogist = resultat2 # hva er navnet på resultatobjektet fra den logistiske regresjon du valgte? va test_pred = bestelogist.predict(exog = df_test) y_testpred = np.where(test_pred &gt; cutoff, 1, 0) y_testobs = df_test['y'] print("Feilrate logistisk regresjon:", 1 - accuracy_score(y_true = y_testobs, y_pred = y_testpred))  # beste resultat for kNN bestek = 115 # hva er din beste k? knn = KNeighborsClassifier(n_neighbors = bestek, p = 2) knn.fit(X_tren, df_tren['y']) X_test=df_test[['skudd_paa_maal_diff', 'corner_diff']] print("Feilrate kNN:", 1 - knn.score(X_test, df_test['y']))  Feilrate logistisk regresjon: 0.31862745098039214 Feilrate kNN: 0.31862745098039214</pre>

<b>2d.2)</b>	Drøft klassegrensene (plottet under) for de to beste modellene (én logistisk regresjon og én kNN). Hva forteller klassegrensene deg om problemet? Skriv maksimalt 3 setninger.
<b>Svar</b>	Hvis corner-differansen og skudd på mål-differansen er positiv så er det høyere sjanse for at hjemmelaget vinner i begge modellene. Modellen for logistisk regresjon viser også en trend for hvis corner-differansen og skudd på mål-differansen er negativ at bortelaget vinner, men det viser seg ikke å være tilfellet med kNN modellen. Den mer komplekse grensen for kNN modellen viser at forholdet mellom modellene kanskje ikke er så lineært avhengige som de ser ut i førstegang og at det kan være høyere kompleksitet i dataene som kNN prøver å fange opp i modellen.

<b>3a.1)</b>	Hvilke 3 siffer har vi i datasettet? Hvor mange bilder har vi totalt i datasettet?
<b>Svar</b>	Datasettet inneholder totalt 6000 bilder og består av sifrene 3, 8 og 9.

<b>3a.2)</b>	Hvilket siffer ligner det 500. bildet i datasettet vårt på? Lag et bilde som viser dette sifferet. (Husk at Python begynner nummereringen med 0, og derfor refereres det 500. bildet til [499])
<b>Svar</b>	Ligner på sifferet '9'

<b>3b.1)</b>	Tegn sentroidene av de 3 klyngene fra $K$ -gjennomsnitt modellen. Tilpass koden over for å plote. Her kan du ta skjermbilde av sentroidene og lime inn i svararket. Hint: Sentroidene har samme format som dataene (de er 384-dimensjonale), og hvis de er representative vil de se ut som tall.
<b>Svar</b>	<div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="text-align: center;"> <p>Centroid 1</p>  </div> <div style="text-align: center;"> <p>Centroid 2</p>  </div> <div style="text-align: center;"> <p>Centroid 3</p>  </div> </div>

<b>3b.2)</b>	Synes du at grupperingen i klynger er relevant og nyttig? Forklar. Maks 3 setninger.
<b>Svar</b>	Grupperingen av de håndskrevne tallene i klynger ved bruk av $K$ -gjennomsnitt modellen ser ut til å bli både relevant og nyttig. Sentroidene som representerer hver klynge viser ganske tydelig og gjenkjennelig sifrene '8', '9' og '3', selvsagt med en viss uskarphet. Dette indikerer at modellen har klart å fange opp de underliggende mønstrene og variasjonene til hvert av sifrene.

**3b.3)** Vi har valgt  $K = 3$  for dette eksempelet fordi vi vil finne klynger som representerer de 3 sifrene. Men generelt er  $K$  vilkårlig. Kom opp med et forslag for hvordan man (generelt, ikke nødvendigvis her) best kan velge  $K$ . Beskriv i egne ord med maks 3 setninger.

**Svar** For å finne beste  $K$ -verdi benyttes gjerne «The Elbow Method» eller «The Silhouette Method», hvorav disse er begge egne tester/utregninger som kan og bør gjøres i Python, eller lignende språk, i forkant av selve klyngeanalysen. «The Silhouette Method» vil være mer presis, men den er også mer krevende/tregere og foretrekkes derfor ikke for større datasett.

**3b.4)** Kjør analysen igjen med  $K = 2$  og  $K = 4$ . Synes du de nye grupperingene er relevante?

**Svar** Nei, det er veldig åpenbart ut ifra sentroidene at disse nye grupperingene ikke er relevante. Med  $K = 3$  ser vi et tydelig skille mellom de tre sifrene, hvorav med lavere eller høyere  $K$ -verdier så prøver algoritmen å kombinere eller skille ut dataene i datasettet til færre eller flere sifre selv om datasettet bare inneholder tre forskjellige.

Med  $K = 2$

Centroid 1 Centroid 2



Med  $K = 4$

Centroid 1 Centroid 2 Centroid 3 Centroid 4



**3c.1)** Vurder dendrogrammet nedenfor. Synes du at den hierarkiske grupperingsalgoritmen har laget gode/meningfulle grupper av bildene? (Maks 3 setninger).

**Svar** Nei. Grupperingsalgoritmen har blandet mye mellom hvilke sifre som hører sammen og ikke. Den treffer ofte godt på tallet '8', men har lett for å blande spesielt sifrene '3' og '9'.

**3c.2)** I koden under har vi brukt gjennomsnittskobling (`method = 'average'`). Hvordan fungerer gjennomsnittskobling? (Maks 3 setninger).

**Svar** "Average" metoden i `scipy.cluster.hierarchy` for dendrogrammer fungerer ved å først beregne gjennomsnittlig avstand mellom alle par av datapunkter i separate klynger. Under klyngeprosessen kombineres de to klyngene med minst gjennomsnittlig avstand til en enkelt klynge. Denne prosessen repeteres til alle datapunkter er gruppert i én klynge, og resultatet visualiseres som et dendrogram.

<b>3c.3)</b>	Velg en annen metode enn 'average' til å koble klyngene sammen (vi har lært om dette i undervisningen, her heter de <code>single</code> , <code>complete</code> og <code>centriod</code> ) og lag et nytt dendrogram ved å tilpasse koden nedenfor. Ser det bedre/verre ut? (Maks 3 setninger).
<b>Svar</b>	Endte med å bruke metode 'complete' da denne så merkbart mer konsistent og riktig ut enn noen av de andre metodene. Nå er klyngene mye mer representative for hva som faktisk finnes i datasettet og er for det meste plassert sammen med samme siffer som seg selv. Grupperingsalgoritmen blander fortsatt spesielt sifferet '3' med '8' og '9', dog er den likevel nå mer presis enn før.
<b>3d.1)</b>	Hvis vi skulle brukt en metode for å predikere/klassifisere hvilket siffer et håndskrevet tall er, og ikke bare samle dem i klynge, hva ville du brukt?
<b>Svar</b>	For å klassifisere hvilket siffer et håndskrevet tall representerer, basert på pensumet, er K-nærmeste-nabo (KNN) metoden et godt valg. Denne metoden sammenligner det nye håndskrevne tallet med kjente eksempler i et treningssett for å identifisere det mest sannsynlige sifferet. Dette gjøres ved å se på likheten med de 'k' nærmeste tallene i treningssettet. Valget av 'k' er kritisk og bestemmes ved å velge en verdi som gir lavest feilrate på et valideringssett. (Her vil k igjen være 3, da datasettet bare inneholder 3 sifre.