

Tellende prosjekt i ISTx1003 - 2021

I prosjektdelen av ISTx1003 Statistikk, Statistisk læring og data science, har vi fokus på tre hovedtemaer: regresjon, klassifikasjon og klyngeanalyse.

Dette er oppgaveteksten til den tellende prosjektoppgaven, der besvarelsen teller 30% av karakteren i emnet.

Veiledning av prosjektoppgaven annonseres på Blackboard og på den eksterne kurssiden: <https://wiki.math.ntnu.no/istx1003/2021h/start>.

Det må være 4-6 studenter på hver gruppe, og alle studenter må være oppmeldt i den samme emnekoden (det vil si: en av ISTA1003, ISTG1003, ISTT1003). **Det er opprettet gruppesett under "Prosjektgrupper" på Blackboard**, der dere melder dere på innen 8. November 2021.

Informasjon om Prosjektmodulen finnes i Blackboard, og alt kursmateriale er også på den eksterne kurssiden: <https://wiki.math.ntnu.no/istx1003/2021h/start>.

Oppgaven skal utføres i Python, ved hjelp av Jupyter-notatbok-versjonen av denne filen som du nå leser. Notatboken er lastet opp på Jupyterhubben (todo), men kan også finnes på den eksterne kurssiden: <https://wiki.math.ntnu.no/istx1003/2021h/start>

Det er meningen at dere skal kjøre notatboken på Jupyterhubben vår <https://s.ntnu.no/isthub>, eller på deres egen installasjon.

Antall spørsmålspunkter per oppgave er angitt ved hver oppgave (total 40 poeng). Karakteren settes med prosentvurderingsmetoden hvor poeng blir konvertert i en prosentandel (ikke-heltall prosent blir avrundet): <https://innsida.ntnu.no/wiki/-/wiki/Norsk/Prosentvurderingsmetoden>

Dere kan svare på alle spørsmålene direkte her i notatboken. **!!Viktig!!**: Skriv alltid svar i feltet etter ordet "**Løsning**", ellers blir det vanskelig for oss å rette prosjektoppgaven.

Prosjektet leveres i Inspira, se informasjon om dette kommer på Blackboard.

Følgende skal leveres inn (lastes opp til Inspira):

- Denne Jupyter-notatboken, der de endringene dere har gjort i koden er med.
- Jupyter-notatboken eksportert som pdf. Dette er bare en backup, vi vil først og fremst sjekke notatboken.

Viktig: For å lage en pdf skal du først generere en html via: File - Download as - HTML, og så åpner du html i en nettleser og eksporterer den som pdf.

Frist for innlevering av prosjektet til Inspira er mandag 29.november kl 09.00.

Kontakt eksamenskontoret ved problemer. Det kan ikke gis utsettelse på innleveringsfristen.

Oppgave 1: Regresjon (16 poeng)

Som dataingeniør vil du antakeligvis jobbe med mange forskjellige type problemstillinger og data, som for eksempel i forskning, i finanssektoren, teknologigrupper. Dette er spennende.

I dette dataeksempelet ser vi på kostnader i amerikanske sykehus mellom 1989 og 1991. Spørsmålet vi vil stille er hvilke faktorene som er avgjørende for å forutsi hvor mye sykehusoppholdet til en pasient koster.

Oppgaven inneholder følgende elementer:

- Laste inn og utforske et datasett

- Tilpasse en enkel lineær regresjonsmodell og diskutere denne
- Tilpasse en multipel lineær regresjonsmodell og diskutere denne

Totale kostnader for pasienter inlagt i amerikanske sykehus (1989-1991)

- `totcst` : totale kostnader
- `age` : alder
- `dzgroup` : sykdomsgruppe (type sykdom)
- `num.co` : antall komorbiditeter (hvor mange sykdommer noen har)
- `edu` : antall år med utdanning
- `income` : inntekt (kategorisert)
- `scoma` : et mål for Glasgow Coma indeks (grad av våkenhet)
- `meanbp` : gjennomsnittlig blodtrykk
- `hrt` : puls
- `resp` : respirasjonsfrekvens
- `temp` : kroppstemperatur
- `race` : etnisitet
- `pafi` : forhold av PaO2/FiO2 (blod/gass blanding)

```
# importere pakker og funksjoner vi trenger i oppgave 1

# generelt - numerikk og nyttige funksjoner
import numpy as np
import pandas as pd

# plotting
import matplotlib.pyplot as plt
import seaborn as sns

# Fordelinger, modeller for regresjon, qq-plott
from scipy import stats
import statsmodels.formula.api as smf
import statsmodels.api as sm

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
InteractiveShell.ast_node_interactivity = "last_expr"
```

Datasettet

Vi leser inn data og skriver ut de første og siste observasjonene i datasettet, og så sjekker vi datatyper. Vi konverterer noen typer til type *category* fordi det er kategoriske variabler.

```
# Lese inn datasettet ved funksjon fra pandas (df=data frame - vanlig navn å gi et datasett)

df = pd.read_csv("https://www.math.ntnu.no/emner/IST100x/ISTx1003/support.csv", sep = ',')

# Endre navn for å forhindre syntax problemer
df = df.rename(columns={'num.co': 'numco'})

# Skriv ut de første og siste radene
print(df)
```

	age	dzgroup	numco	edu	income	scoma	totcst	\
0	43.54	ARF/MOSF w/Sepsis	1	NaN	NaN	26	390460.50	
1	63.66	ARF/MOSF w/Sepsis	0	22.0	\$25-\$50k	26	156674.13	
2	31.84	Cirrhosis	2	16.0	under \$11k	0	17528.44	
3	48.70	Lung Cancer	0	16.0	NaN	0	33002.50	

```

4      49.61  ARF/MOSF w/Sepsis      1  12.0      $25-$50k      0  288592.25
...      ...      ...      ...      ...      ...      ...
8137  68.62      COPD      2  12.0  under $11k      0      1847.38
8138  66.07  ARF/MOSF w/Sepsis      1   8.0      NaN      0  34329.31
8139  55.15      Coma      1  11.0      NaN      41  23558.50
8140  70.38  ARF/MOSF w/Sepsis      1  NaN      NaN      0  31409.02
8141  81.54  ARF/MOSF w/Sepsis      1   8.0    $11-$25k      0  10605.76

```

```

      race  meanbp    hrt    resp    temp    pafi
0    white     67  172.0    20  38.80  113.33
1    white     69  108.0    22  36.70  155.53
2    white     83  100.0    24  37.40     NaN
3   other     66  125.0    30  37.00  170.00
4    white     67  120.0    48  38.90  200.00
...      ...      ...      ...      ...      ...
8137 white     71  110.0    10  36.20  135.00
8138 white    109  104.0    22  35.70  280.00
8139 white     43   0.0     8  38.59  218.50
8140 white    111  83.0    24  36.70  180.00
8141 white     75  69.0    24  36.20  230.41

```

[8142 rows x 13 columns]

```
# Konverter dzgroup, race og income til "category"
```

```

df=df.astype({'dzgroup':'category','race':'category','income':'category'})
print(df["dzgroup"].value_counts())
print(df["race"].value_counts())
print(df["income"].value_counts())

```

```

ARF/MOSF w/Sepsis      3076
CHF                    1290
COPD                   895
Lung Cancer            825
MOSF w/Malig           613
Coma                   528
Cirrhosis              458
Colon Cancer           457
Name: dzgroup, dtype: int64
white                  6499
black                  1178
hispanic               263
other                  94
asian                  71
Name: race, dtype: int64
under $11k             2571
$11-$25k               1360
$25-$50k               942
>$50k                  605
Name: income, dtype: int64

```

```
# Få oversikt over datasettet
```

```
df.describe()
```

	age	numco	edu	scoma	totcst	meanbp	hrt	resp	temp	p
count	8142.000000	8142.000000	6680.000000	8142.000000	8142.000000	8142.000000	8142.000000	8142.000000	8142.000000	6
mean	63.020722	1.901498	11.756587	11.648858	30865.642767	84.775608	97.709862	23.525669	37.100981	2
std	15.537342	1.352183	3.429399	24.258079	45717.962369	26.725561	30.638716	9.464641	1.244434	1
min	18.120000	0.000000	0.000000	0.000000	420.310000	22.000000	0.000000	0.000000	31.700000	1
25%	53.250000	1.000000	10.000000	0.000000	5958.347500	63.000000	72.000000	18.000000	36.200000	1
50%	65.145000	2.000000	12.000000	0.000000	14484.290000	77.000000	100.000000	24.000000	36.700000	2
75%	74.400000	3.000000	14.000000	9.000000	36146.585000	107.000000	120.000000	28.000000	38.090000	3
max	101.850000	9.000000	31.000000	100.000000	633212.000000	195.000000	300.000000	90.000000	41.700000	8

```
# Sjekk datatyper for alle variabler
```

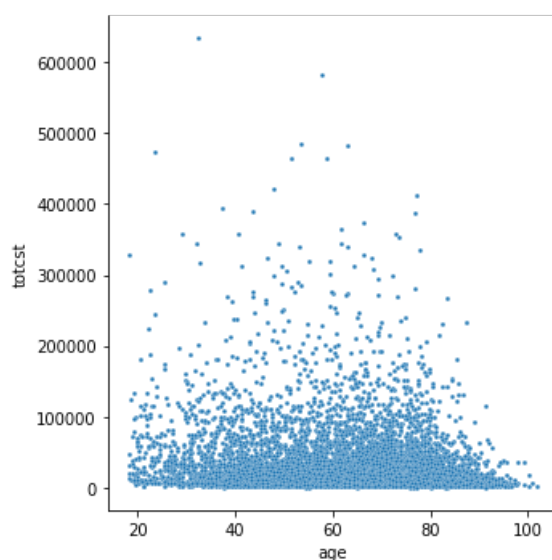
```
df.dtypes
```

Her ser vi at de fleste variablene er kontinuert eller heltall (int), mens dzgroup, income og race er kategoriske. Disse variablene må vi kode om med såkalt dummy-variabelkoding for at de skal bli behandlet riktig i regresjonsanalysen, men først ser vi på en enkel linear regresjon:

Enkel lineær regresjon

Vi skal starte med å se på de totale kostnader `totcst` som responsen vi ønsker å undersøke/forklare, og vi velger variabelen `age` (alder) som eneste forklaringsvariabel. Følgende er et kryssplott av `totcst` mot `age`:

```
sns.relplot(x = 'age', y = 'totcst', kind = 'scatter', s=8, data = df)
plt.show()
```



Problem 1a) (1P)

Vil du konkludere med at en lineær regressjonsmodell passer bra? Hvorfor (ikke)? Maks 2 setninger.

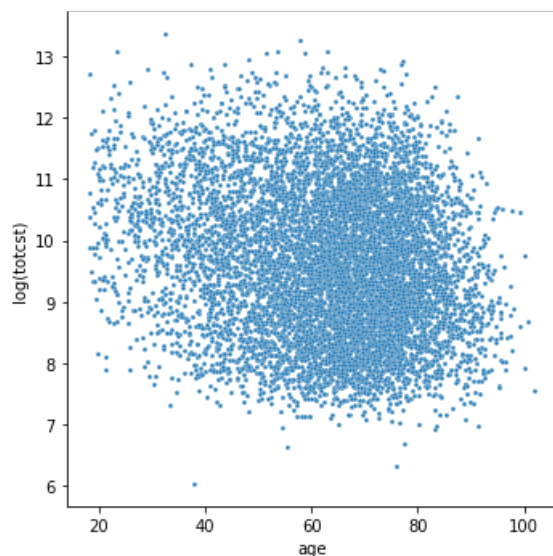
Løsning:

Nei, det passer ikke bra fordi det er for stor spredning i dataene. Det er ikke mulig å se en lineær sammenheng

mellom kostnad og alder - en lineær regressjonsmodell passer derfor ikke for disse dataene.

I dette tilfelle er det lurt å transformere én eller flere variabler. Som statistiker vet vi at en logaritmisk transformasjon ofte hjelper. Vi tar log av responsen og ser på kryssplottet igjen:

```
# Vi lager en ny variabel som er log(totcst):  
df['logtotcst'] = np.log(df['totcst'])  
sns.relplot(x = 'age', y = 'logtotcst', kind = 'scatter', s=8, data = df)  
plt.ylabel("log(totcst)")  
plt.xlabel("age")  
  
plt.show()
```



Problem 1b) (1P)

Sammenlign den nye grafen (med log(totcost) som respons) med den gamle grafen lengre opp. Passer en lineær regresjonsmodell bedre nå? Maks 2 setninger.

Løsning:

En lineær modell passer også dårlig her da det ikke kommer frem en tydelig lineær sammenheng i dataene. Vi vil allikevel si at en lineær regresjonsmodell passer **bedre** for denne grafen enn den lengre opp, da man her kan se en neglisjerbar negativ korrelasjon.

Uavhengig av hva du svarte under b) skal vi nå tilpasse en enkel lineær modell med `logtotcst` som respons og `age` som forklaringsvariabel. For å oppsummere det vi har snakket om i undervisningen, så består en (enkel og multipl) lineær regresjonsanalyse av følgende steg:

- Steg 1: Bli kjent med dataene ved å se på oppsummeringsmål og ulike typer plott
- Steg 2: Spesifiser en matematisk modell (med modellformel)
- Steg 3: Initialiser og tilpass modellen
- Steg 4: Presenter resultater fra den tilpassede modellen
- Steg 5: Evaluer om modellen passer til dataene

Vi har nå gjort Steg 1, og under finner du kode for å gjøre steg 2-4. Studer og kjør koden.

```
# kodechunk Steg2-4

# Steg 2: spesifiser matematisk modell
formel='logtotcst ~ age'

# Steg 3: Initialiser og tilpass en enkel lineær regresjonsmodell
# først initialisere
modell = smf.ols(formel,data=df)
# deretter tilpasse
resultat = modell.fit()

# Steg 4: Presenter resultater fra den tilpassede regresjonsmodellen
print(resultat.summary())
```

```

=====
                        OLS Regression Results
=====
Dep. Variable:          logtotcst      R-squared:                0.033
Model:                  OLS           Adj. R-squared:           0.032
Method:                 Least Squares  F-statistic:              273.6
Date:                   Wed, 24 Nov 2021  Prob (F-statistic):      1.83e-60
Time:                   12:27:44       Log-Likelihood:           -12832.
No. Observations:      8142          AIC:                     2.567e+04
Df Residuals:          8140          BIC:                     2.568e+04
Df Model:               1
Covariance Type:       nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
Intercept	10.4962	0.054	193.713	0.000	10.390	10.602
age	-0.0138	0.001	-16.540	0.000	-0.015	-0.012

```

=====
Omnibus:                 302.470    Durbin-Watson:           1.783
Prob(Omnibus):            0.000    Jarque-Bera (JB):        166.802
Skew:                     0.185    Prob(JB):                 6.02e-37
Kurtosis:                 2.404    Cond. No.                 271.
=====

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Nå skal vi studere resultatene fra `resultat.summary()`, og vi refererer til *øvre panel* som linjene mellom første og andre doble strek `====` (dette er delen som starter med `Dep.Variable`), *midtre panel*, og *nedre panel* (som starter med `Omnibus`).

Problem 1c) (2P)

(i) Skriv ned ligningen for den estimerte regresjonsmodellen (se midtre panel).

(ii) Se på det øvre panelet og rapporter R_{adj}^2 og gi en tolking av verdien (er den stor/liten, hva betyr det her?).

Løsning:

(i)

Med utgangspunkt i formel $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$, kan vi lese av konstantleddet $\hat{\beta}_0$ fra raden "Intercept" og koeffisienten $\hat{\beta}_1$ til forklaringsvariabelen x fra raden "age". Her ser vi at `Intercept = 10.4962` og `age = -0.0138`, som gir oss ligningen:

$$\hat{y} = 10.4962 - 0.0138x$$

(ii)

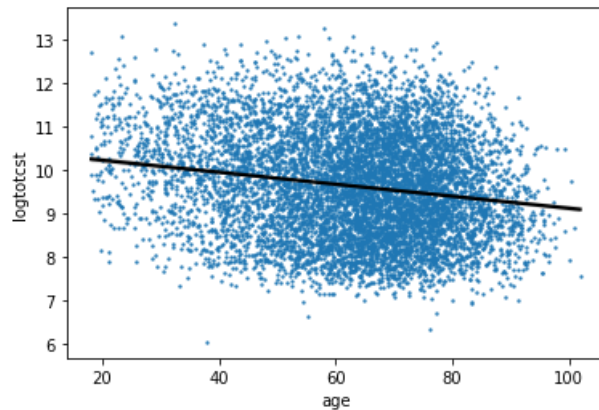
$$R_{adj}^2 = 0.032.$$

R_{adj}^2 er et mål på hvor godt en lineær regresjonsmodell beskriver datasettet der $R_{adj}^2 = 1$ vil si at en lineær regresjonsmodell beskriver datasettet perfekt. Her ser vi at R_{adj}^2 er svært lav, noe som gjør at vi kan konkludere med at en lineær regresjonsmodell beskriver dette datasettet svært dårlig, ettersom $R_{adj}^2 = 0.032$ indikerer svært lite korrelasjon.

Vi kan nå også plote den estimerte regresjonslinjen i kryssplott.

```
sns.regplot(x=df['age'], y=df['logtotcst'], line_kws={"color": "black"}, scatter_kws={"s": 2})
```

```
<AxesSubplot: xlabel='age', ylabel='logtotcst'>
```



Problem 1d) (3P)

(i) Vi ser at for `age` er `coef` lik -0.0138 (dette er $\hat{\beta}_1$). Forklar dette tallet til en sykehusansatt som ikke har hørt om enkel lineær regresjon. Er kostnadene høyere eller lavere for eldre pasienter? Maks 2 setninger.

(ii) Oppgi 95% konfidensintervall for $\hat{\beta}_1$. Forklar hva det betyr til din kollega som jobber på sykehuset med maks 2 setninger.

(iii) Hva er p -verdien for $\hat{\beta}_1$? Forklar om vi kan være sikker på at alderen har linear sammenheng med `logtotcst`. Hvorfor (ikke)?

Løsning:

(i)

$\hat{\beta}_1 = -0.0138$ betyr at kostnadene vil synke fra år til år. Dette tilsier at kostnadene blir lavere jo eldre pasienten er.

(ii)

$C = 0.95$ (altså $\alpha = 0.05$) gir konfidensintervallet $[-0.015, -0.012]$ for $\hat{\beta}_1$.

Dette vil si at det er 95% sjanse for at intervallet inneholder den faktiske verdien av $\hat{\beta}_1$.

(iii)

p -verdien til $\hat{\beta}_1$ er såpass lav at Python representerer den som p -verdi = 0.000.

Selv om dette ikke er riktig (p -verdi kan aldri være 0), kan vi med sikkerhet si at p -verdi $< \alpha$, og at vi dermed *kan* forkaste null-hypotesen, $H_0 : \hat{\beta}_1 = 0$, til fordel for alternativhypotesen $H_1 : \hat{\beta}_1 \neq 0$.

Mer praktisk kan vi si at nullhypotesen er at linjen er flat, mens alternativhypotesen er at linjen stiger/synker. Her konkluderer hypotesetesten vår med at linjen stiger/synker - altså at vi har en lineær sammenheng.

Allikevel har vi fra forrige oppgave at R_{adj}^2 er svært liten, som indikerer at modellen forklarer datasettet dårlig. Vi kan dermed ikke være helt sikre på om det er en lineær sammenheng mellom `logtotcst` og `age`.

En viktig steg når vi tilpasser en modell er å sjekke om modellen faktisk passer, ellers kan vi ende med feil konklusjoner!

For å sjekke modellantakelser ser vi på følgende plot:

```
# kodechunk Steg5
```

```
# Steg 5: Evaluer om modellen passer til dataene
```

```
# Plotte predikert verdi mot residual
```

```
sns.scatterplot(x=resultat.fittedvalues, y=resultat.resid,s=8)
```

```
plt.ylabel("Residual")
```

```
plt.xlabel("Predikert verdi")
```

```
plt.axhline(y=0, color='black', linestyle='--')
```

```
plt.title("TA plot")
```

```
plt.show()
```

```
# Lage kvantil-kvantil-plott for residualene
```

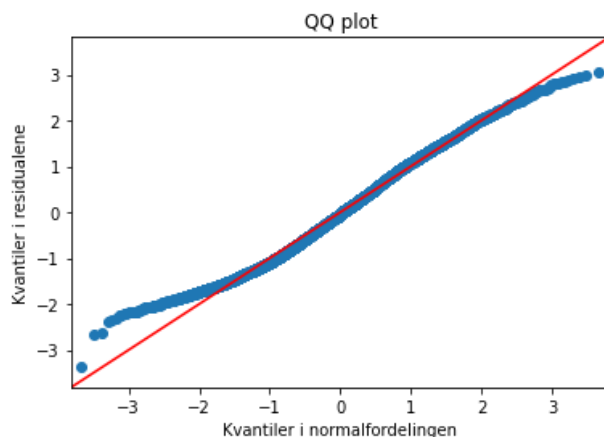
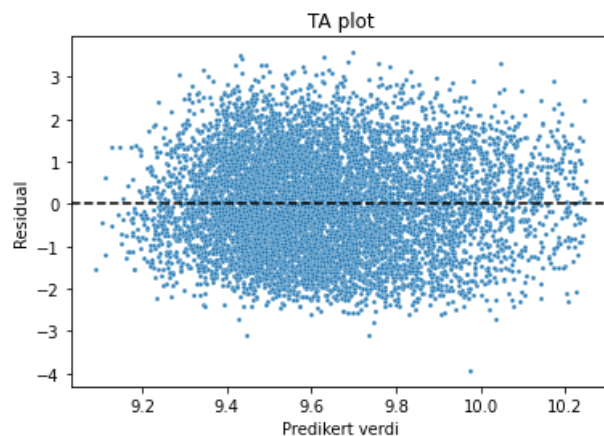
```
sm.qqplot(resultat.resid,line='45',fit=True)
```

```
plt.ylabel("Kvantiler i residualene")
```

```
plt.xlabel("Kvantiler i normalfordelingen")
```

```
plt.title("QQ plot")
```

```
plt.show()
```



Problem 1e) (3P)

(i) (1P) Studer plottet av predikert verdi mot residual (Tukey-Anscombe (TA) plot) og QQ-plottet. Vurderer du at modellantagelsene er oppfylt? Gi en kort begrunnelse hvor to tolker begge plottene.

(ii) (2P) Generer de samme to plottene når vi bruker `totcst` uten log-transformasjon `formel='totcst ~ age'` i en enkel linear regresjon (kopier python koden fra steg 2-5 og se på TA- og QQ-plott). Er modellantakelsene oppfylt for denne regresjonen? Forklar.

Løsning:

(i)

Ref. begrunnelse i forrige oppgave ser det ikke ut til å være lineær sammenheng, på tross av at Python har konkludert ved hypotesetest at $\hat{\beta}_1 = -0.0138$. Den første modellantakelsen er dermed *ikke* oppfylt.

For å kunne vurdere om den andre antagelsen, at $\hat{e}_i \sim N(0, \sigma)$, er oppfylt kan vi studere TA- og QQ-plottet. I TA-plottet ser vi ingen struktur og en form for sentrering rundt 0: Altså har residualene samme standardavvik uavhengig av den predikerte verdien. I QQ-plottet ser vi at residualene ligger opp mot den lineære linjen: altså kan vi konkludere med at residualene $\hat{e}_i \sim N(0, \sigma)$.

Kun én av to modellantakelser er dermed oppfylt.

(ii)

Se kodecelle under for TA- og QQ-plottet.

Her er modellantakelsen ikke oppfylt. Vi ser en form for struktur i TA-plottet noe som tilsier at stadardavviket er delvis avhengig av den predikerte verdien. I tillegg viser QQ-plottet at residualene avviker veldig fra linjen, altså er de ikke normalfordelt.

```
formel='totcst ~ age'

modell = smf.ols(formel,data=df)

resultat = modell.fit()

print(resultat.summary())

sns.scatterplot(x=resultat.fittedvalues, y=resultat.resid,s=8)
plt.ylabel("Residual")
plt.xlabel("Predikert verdi")
plt.axhline(y=0, color='black', linestyle='--')
plt.title("TA plot")
plt.show()

sm.qqplot(resultat.resid,line='45',fit=True)
plt.ylabel("Kvantiler i residualene")
plt.xlabel("Kvantiler i normalfordelingen")
plt.title("QQ plot")
plt.show()
```

OLS Regression Results						
=====						
Dep. Variable:	totcst		R-squared:	0.025		
Model:	OLS		Adj. R-squared:	0.024		
Method:	Least Squares		F-statistic:	205.3		
Date:	Wed, 24 Nov 2021		Prob (F-statistic):	5.37e-46		
Time:	12:27:45		Log-Likelihood:	-98817.		
No. Observations:	8142		AIC:	1.976e+05		
Df Residuals:	8140		BIC:	1.977e+05		
Df Model:	1					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]

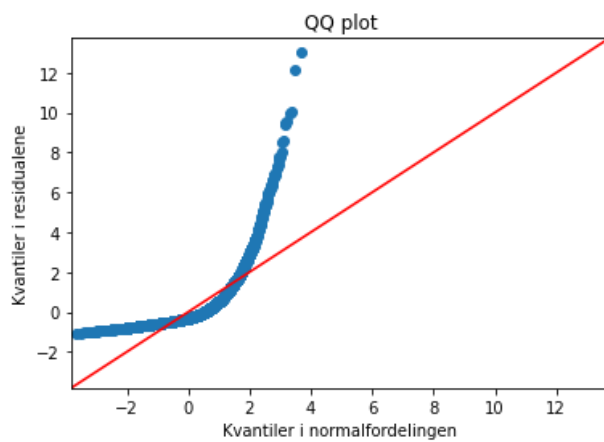
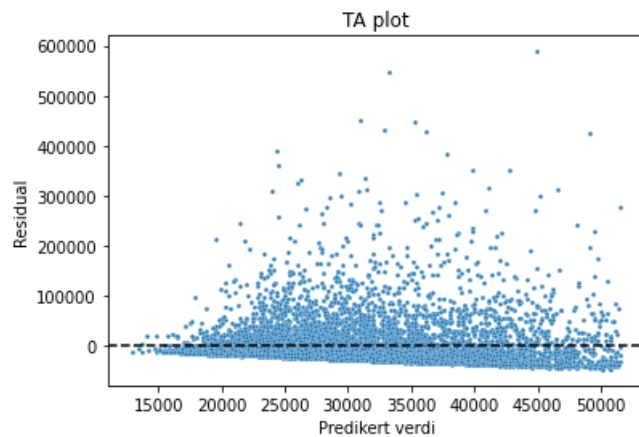
Intercept	5.995e+04	2090.670	28.674	0.000	5.58e+04	6.4e+04
age	-461.4678	32.210	-14.327	0.000	-524.608	-398.328
=====						

Omnibus:	6829.846	Durbin-Watson:	1.853
Prob(Omnibus):	0.000	Jarque-Bera (JB):	215828.226
Skew:	3.921	Prob(JB):	0.00
Kurtosis:	26.973	Cond. No.	271.

=====

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.



Multippel lineær regresjon

For å bedre predikere de totale kostnadene for en pasjent er det lurt å ta med flere forklaringsvariabler inn i regresjonsanalysen.

En rekke plott vises nå. Vi ser kryssplott, tetthetsplott (som er en glattet versjon av histogram) og boksplott. For tetthetsplottene og boksplottene deler vi data inn i de forskjellige sykdomsgruppene for å se om sykdomstype påvirker effekten som alder og numco har på kostnadene (logtotcst).

```
# Kryssplott av logtotcst mot age, numco, meanbp og temp
# På diagonalen er glattede histogrammer (tetthetsplott) av logtotcst, age, numco, meanbp og temp
sns.pairplot(data=df, vars = ['logtotcst', 'age', 'numco', 'meanbp', 'temp'],
              diag_kind = 'kde',
              plot_kws=dict(alpha=0.4))

plt.show()

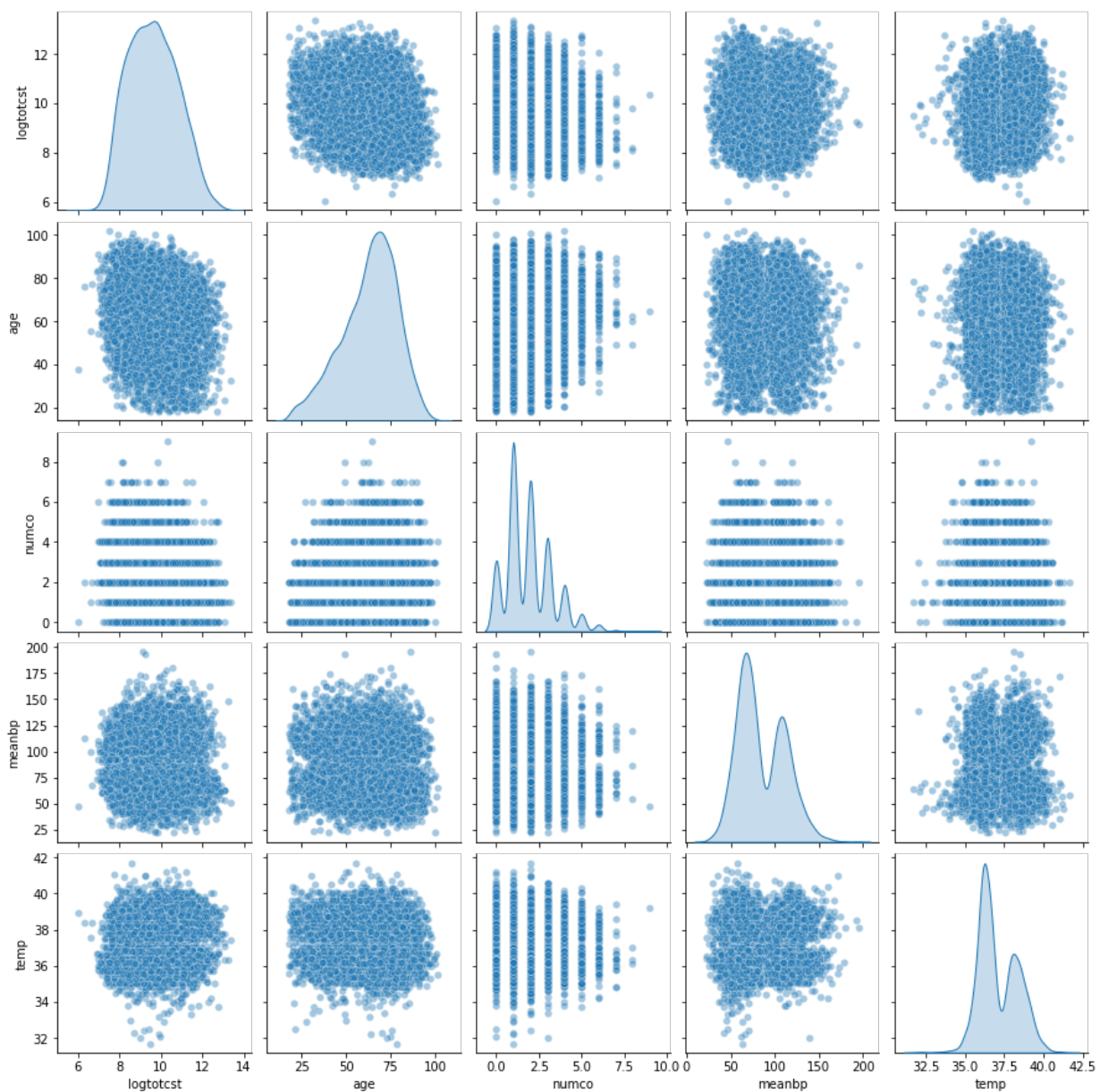
# Boksplott av Blodceller for hvert Kjoenn og for hver Sport

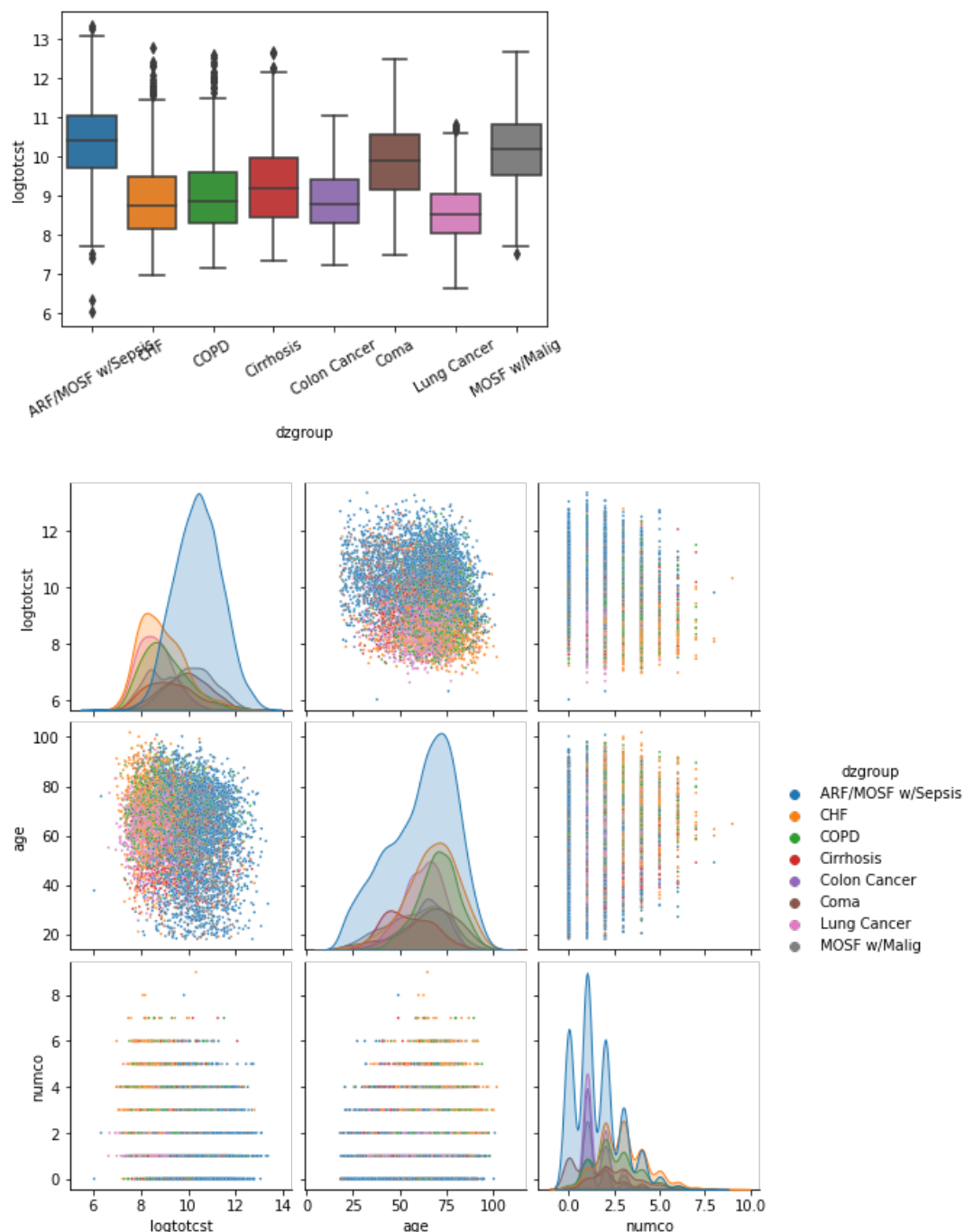
ax = sns.boxplot(x="dzgroup", y="logtotcst", data=df)
ax.set_xticklabels(ax.get_xticklabels(), rotation=30)

plt.show()

sns.pairplot(df, vars = ['logtotcst', 'age', 'numco'],
              hue = 'dzgroup',
              diag_kind = 'kde',
              plot_kws={"s":3})

plt.show()
```





Vi skal nå tilpasse en multipl lineær regresjon med (igjen) `logtotcst` som respons. Vi tar med forklaringsvariablene `age`, `numco`, `meanbp`, `temp` og `dzgroup`.

Problem 1f) (4P)

(i) Utfør regresjonen på nytt med den nye modellformelen (som er gitt under) ved å kopiere inn akkurat samme kode for steg 3, 4 og 5.

(ii) Hvor mange regresjonsparametere er estimert?

(iii) Hvis vi sammenligner en person som er innlagt med Lung Cancer (lungekreft) med en person med COPD (lunnesykdommen kols), som begge er like gamle, har samme antall comorbiditeter (`numco`), samme gjennomsnittlig blodtrykk (`meanbp`) og samme kroppstemperatur, hvilken pasient har de høyeste forventede kostnadene? Forklar.

(iv) Hva er de predikerte kostnadene på den originale skalaen (altså `totcst`, ikke `logtotcst`!) for en pasient på 50 år med Colon Cancer, en comorbiditet (`numco`=1), `meanbp`=130 og `temp`=36.2? Regn for hånd ved å sette inn tall fra

resultat.summary() og vis beregning.

Løsning:

(i)

Se kodecelle under

```
# Tilhører oppgave (i)

# Steg 3: definere formel og tilpasse regresjonsmodell
formel='logtotcst ~ age + dzgroup + numco + meanbp + temp'
modell = smf.ols(formel,data=df)

resultat = modell.fit()

# Steg 4: skrive ut resultater
print(resultat.summary())

# Steg 5: Lage TA og QQ plots
sns.scatterplot(x=resultat.fittedvalues, y=resultat.resid,s=8)
plt.ylabel("Residual")
plt.xlabel("Predikert verdi")
plt.axhline(y=0, color='black', linestyle='--')
plt.title("TA plot")
plt.show()

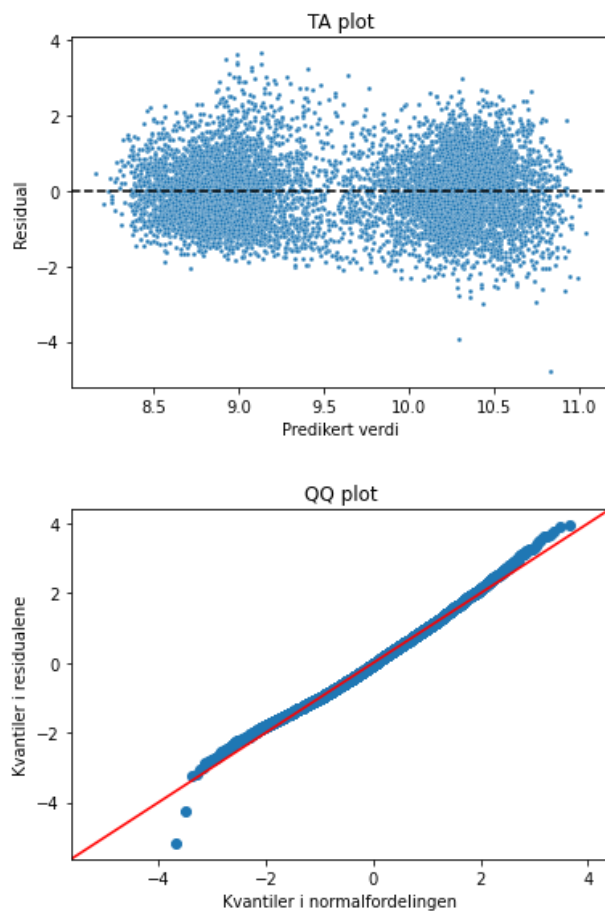
sm.qqplot(resultat.resid,line='45',fit=True)
plt.ylabel("Kvantiler i residualene")
plt.xlabel("Kvantiler i normalfordelingen")
plt.title("QQ plot")
plt.show()
```

OLS Regression Results						
=====						
Dep. Variable:	logtotcst	R-squared:	0.394			
Model:	OLS	Adj. R-squared:	0.393			
Method:	Least Squares	F-statistic:	480.0			
Date:	Wed, 24 Nov 2021	Prob (F-statistic):	0.00			
Time:	12:27:57	Log-Likelihood:	-10930.			
No. Observations:	8142	AIC:	2.188e+04			
Df Residuals:	8130	BIC:	2.197e+04			
Df Model:	11					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]

Intercept	8.6679	0.326	26.598	0.000	8.029	9.307
dzgroup[T.CHF]	-1.3486	0.034	-40.221	0.000	-1.414	-1.283
dzgroup[T.COPD]	-1.2107	0.037	-32.958	0.000	-1.283	-1.139
dzgroup[T.Cirrhosis]	-1.0448	0.048	-21.781	0.000	-1.139	-0.951
dzgroup[T.Colon Cancer]	-1.5160	0.047	-32.489	0.000	-1.608	-1.425
dzgroup[T.Coma]	-0.5218	0.044	-11.874	0.000	-0.608	-0.436
dzgroup[T.Lung Cancer]	-1.7661	0.037	-48.215	0.000	-1.838	-1.694
dzgroup[T.MOSF w/Malig]	-0.2397	0.041	-5.834	0.000	-0.320	-0.159
age	-0.0084	0.001	-12.127	0.000	-0.010	-0.007
numco	-0.0546	0.009	-6.368	0.000	-0.071	-0.038
meanbp	-0.0013	0.000	-3.233	0.001	-0.002	-0.000
temp	0.0651	0.008	7.664	0.000	0.048	0.082
=====						
Omnibus:	90.497	Durbin-Watson:	1.801			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	93.566			
Skew:	0.254	Prob(JB):	4.81e-21			
Kurtosis:	3.136	Cond. No.	3.62e+03			
=====						

Notes:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
 [2] The condition number is large, 3.62e+03. This might indicate that there are strong multicollinearity or other numerical problems.



(ii)

Her har vi estimert 12 regresjonsparametre: konstantleddet og forklaringsvariablene `age`, `dzgroup` (består av 7 kategorier), `numco`, `meanbp` og `temp`.

(iii)

Vi kan direkte sammenlikne koeffisientene til de to kategoriene fra regresjonsanalysen.

Her ser vi at `coef COPD > coef Lung Cancer`. Det vil i praksis si at en person med COPD (kols) vil få en høyere forventet kostnad, da den estimerte koeffisienten til COPD trekker i fra mindre på kostnaden enn den estimerte koeffisienten til lungekreft (begge er negative).

(iv)

Vi bytter ut `logtotcst` med `totcst` (se kodecelle under) og leser av verdiene på de ulike regresjonsparamterene for å regne ut den predikerte kostnaden av denne pasienten. Vi kaller kostnaden \hat{y} .

`age = 50`

`coloncancer = 1`

`numco = 1`

`meanbp = 130`

`temp = 36.2`

`chf = copd = cirrhosis = mosf = coma = 0`

$$\begin{aligned} \hat{y} = & (-2658.5312) + (-3.345 * 10^4 * chf) + (-3.095 * 10^4 * copd) + (-2.903 * 10^4 * cirrhosis) \\ & + (-4.175 * 10^4 * coloncancer) + (-2.18 * 10^4 * coma) + (-4.33 * 10^4 * lungcancer) \\ & + (-1.321 * 10^4 * mosf) + (-328.2267 * age) + (-1655.5698 * numco) + (-41.5186 * meanbp) \\ & + (2165.7620 * temp) = 10527.73 \end{aligned}$$

```
# Tilhører oppgave (iv)
```

```
# Steg 3: definere formel og tilpasse regresjonsmodell
```

```
formel='totcst ~ age + dzgroup + numco + meanbp + temp'
modell = smf.ols(formel,data=df)
```

```
resultat = modell.fit()
```

```
# Steg 4: skrive ut resultater
```

```
print(resultat.summary())
```

```
# Steg 5: Lage TA og QQ plots
```

```
sns.scatterplot(x=resultat.fittedvalues, y=resultat.resid,s=8)
```

```
plt.ylabel("Residual")
```

```
plt.xlabel("Predikert verdi")
```

```
plt.axhline(y=0, color='black', linestyle='--')
```

```
plt.title("TA plot")
```

```
plt.show()
```

```
sm.qqplot(resultat.resid,line='45',fit=True)
```

```
plt.ylabel("Kvantiler i residualene")
```

```
plt.xlabel("Kvantiler i normalfordelingen")
```

```
plt.title("QQ plot")
```

```
plt.show()
```

OLS Regression Results

Dep. Variable:	totcst	R-squared:	0.181			
Model:	OLS	Adj. R-squared:	0.180			
Method:	Least Squares	F-statistic:	163.5			
Date:	Wed, 24 Nov 2021	Prob (F-statistic):	0.00			
Time:	12:27:58	Log-Likelihood:	-98104.			
No. Observations:	8142	AIC:	1.962e+05			
Df Residuals:	8130	BIC:	1.963e+05			
Df Model:	11					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]

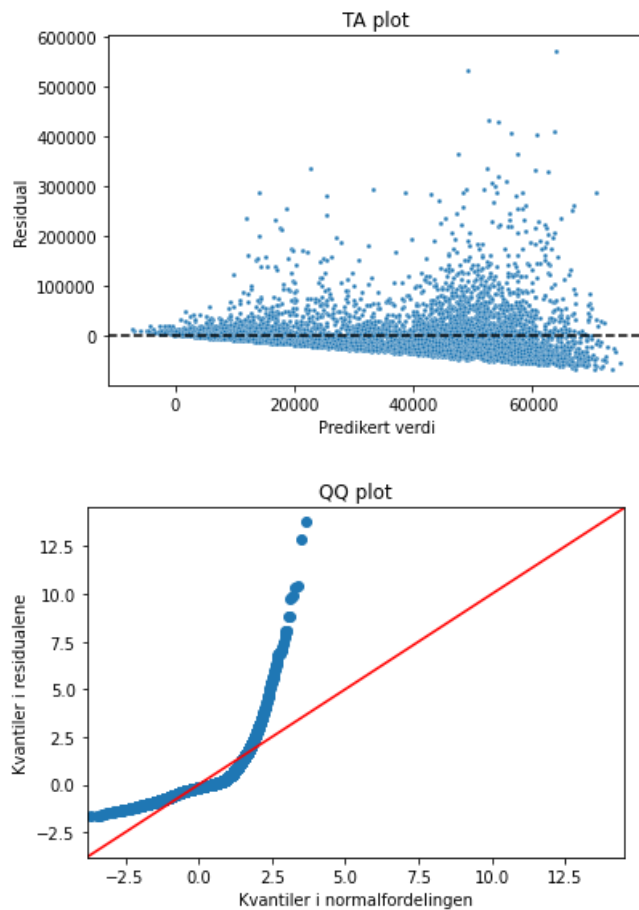
Intercept	-2658.5312	1.46e+04	-0.183	0.855	-3.12e+04	2.59e+04
dzgroup[T.CHF]	-3.345e+04	1497.403	-22.340	0.000	-3.64e+04	-3.05e+04
dzgroup[T.COPD]	-3.095e+04	1640.558	-18.866	0.000	-3.42e+04	-2.77e+04
dzgroup[T.Cirrhosis]	-2.903e+04	2142.172	-13.551	0.000	-3.32e+04	-2.48e+04
dzgroup[T.Colon Cancer]	-4.175e+04	2083.920	-20.037	0.000	-4.58e+04	-3.77e+04
dzgroup[T.Coma]	-2.18e+04	1962.560	-11.109	0.000	-2.56e+04	-1.8e+04
dzgroup[T.Lung Cancer]	-4.33e+04	1635.817	-26.471	0.000	-4.65e+04	-4.01e+04
dzgroup[T.MOSF w/Malig]	-1.321e+04	1834.962	-7.198	0.000	-1.68e+04	-9611.881
age	-328.2267	30.762	-10.670	0.000	-388.529	-267.925
numco	-1655.5698	382.624	-4.327	0.000	-2405.610	-905.530
meanbp	-41.5186	17.303	-2.399	0.016	-75.438	-7.600
temp	2165.7620	379.484	5.707	0.000	1421.876	2909.648
=====						
Omnibus:	7119.496	Durbin-Watson:	1.864			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	286190.558			
Skew:	4.085	Prob(JB):	0.00			
Kurtosis:	30.872	Cond. No.	3.62e+03			

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 3.62e+03. This might indicate that there are

strong multicollinearity or other numerical problems.



Problem 1g) (2P)

(i) Forklaringsvariabelen `dzgroup` er kategorisk og vi har brukt en såkalt dummy-variabelkoding, der 'ARF/MOSF w/Sepsis' er referansekategorien. Er effekten av de andre sykdomgruppene på `logtotcst` forskjellig fra effekten for referansekategorien? Forklar.

(ii) Hva er andel forklart variasjon? Sammenlign med verdien du fant i Problem 1c(ii). Var det en god idé å bruke multipl linear regresjon, eller var enkel linear regresjon godt nok?

Løsning:

(i)

Referansekategorien er det vi bruker som utgangspunkt for de andre kategoriene. Vi setter den som 0 og ser på tetthets- og boksplottet at alle andre kategorier er mindre enn referansen i datasettene, noe som reflekteres i alle de andre kategoriens koeffisienter. Fordi referansesykdommens kostnad er den høyeste, må disse koeffisientene være negative.

Ettersom koeffisientene til de andre variablene for sykdom er $\neq 0$, noe som bekreftes av summary og hypotesetesten, har de andre sykdommene en annen effekt enn referansekategorien.

(ii)

Her er $R_{adj}^2 = 0.393$, kontra i 1c(ii) der $R_{adj}^2 = 0.032$. Vi ser dermed at multipl regersjon er bedre for dette datasettet.

Oppgave 2: Klassifikasjon (14 poeng)

Del 1: Logistisk regresjon

Et populært eksempel i klassifikasjon er implementasjonen av et spam filter. Det er noe vi alle er kjent med, men har du noen gang spurt deg selv hvordan det egentlig fungerer?

Her bruker vi en datasett med 5575 SMS meldinger som allerede har blitt klassifisert som "spam" eller "ham". Dataene er tatt fra:

<https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection>

og den originale studien finner du her: <https://dl.acm.org/doi/10.1145/2034691.2034742>

Dette er til informasjon. Du trenger ikke laste noe ned eller lese dette, du får alt du trenger her i oppgaveteksten.

I denne oppgaven skal vi tilpasse en logistisk regresjonsmodell for å lage et spam filter.

```
# Vi begynner igjen med å importere pakker og funksjoner vi trenger i oppgaven
```

```
import pandas as pd
import numpy as np
import random as rnd
```

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

```
from sklearn.model_selection import train_test_split, cross_val_score
```

```
# plotting
import matplotlib.pyplot as plt
```

Spamfilter datasett

Først leser vi inn datasettet. Vi gir nye navn til kolonnene (som skal hete 'y' og 'tekst'), koder om 'spam' til '1' og 'ham' til '0', og ser på innholdet av datasettet.

```
# Lese inn datasettet ved funksjon fra pandas (df=data frame - vanlig navn å gi et datasett)
df = pd.read_csv('https://www.math.ntnu.no/emner/IST100x/ISTx1003/SMSSpamCollection.txt', delimiter=';
```

```
# Gi nye navner til den første kolonnen (slik at den heter 'y') og den andre kolonnen (som skal hete
# y er den binære responsen som koder y=1 for "spam" og y = 0 for "ham" (=ikke spam)
# Kolonnen 'text' inneholder SMS teksten
```

```
df = df.rename(columns={0: 'y', 1: 'text'})
```

```
# Vi bytter at spam er 1 og ham er 0
df.replace(['spam', 'ham'], (1, 0), inplace=True)
```

```
print(df)
```

```
   y      text
0  0  Go until jurong point, crazy.. Available only ...
1  0              Ok lar... Joking wif u oni...
2  1  Free entry in 2 a wkly comp to win FA Cup fina...
3  0  U dun say so early hor... U c already then say...
4  0  Nah I don't think he goes to usf, he lives aro...
... ..
5567 1  This is the 2nd time we have tried 2 contact u...
5568 0              Will ü b going to esplanade fr home?
5569 0  Pity, * was in mood for that. So...any other s...
```

```
5570 0 The guy did some bitching but I acted like i'd...
5571 0                                     Rofl. Its true to its name
```

```
[5572 rows x 2 columns]
```

Problem 2a) (2P)

(i) Hvor mange av SMS meldingene er spam og hvor mange er ham? Tips: Bruk en funksjon du har sett i oppgave 1 som heter `value_counts()`.

(ii) Lag et histogram for responsen y ved bruk av `plt.hist()` funksjonen.

Løsning:

(Svarene er basert på kodelinen under denne markdowncellen)

(i)

Som beskrevet i oppgaveteksten er `y=1` koding for "spam" og `y=0` koding for ham.

Ut i fra outputen for kodelinen nedenfor, ser vi at datasettet inneholder 747 tilfeller av spam og 4825 tilfeller av ham.

(ii)

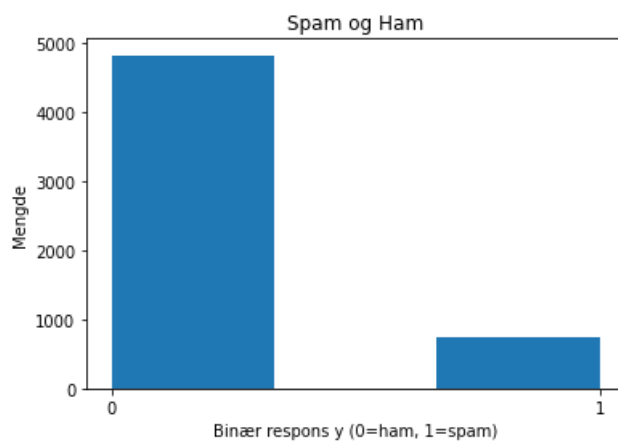
Se kodelinen nedenfor

```
# Tilhører oppgave (i)

# Spam og ham
df.value_counts('y')
```

```
# Tilhører oppgave (ii)

# Histogram
plt.title("Spam og Ham")
plt.xlabel("Binær respons y (0=ham, 1=spam)")
plt.ylabel("Mengde")
plt.hist(df['y'], bins=3, range=(0,1), histtype="bar")
plt.xticks((0,1))
plt.show()
```



Trening, validerings og testsett

Før vi fortsetter med modellering skal vi dele datasettet i tre deler: et trenings-, validerings- og et testsett. Her bestemmer vi oss for at 70% av data er i treningssettet, og 15% er i validerings og 15% i testsettet.

```
# Først del dataene i trenings og testsett (70-30%)
X_train_raw, X_test_raw, y_train, y_test = train_test_split(df['text'],df['y'],test_size=0.3,random_s

# Og så del testsettet igjen i et test- og et valideringssett slik at begge inneholder 15% (50% av de
X_test_raw, X_val_raw, y_test, y_val = train_test_split(X_test_raw,y_test,test_size=0.5,random_state=
```

Husk hvorfor vi må dele datasettet i disse ulike delene. Sjekk forelesningsmaterial hvis du har glemt det.

Logistisk regresjon

I undervisningen har vi lært at logistisk regresjon trenger forklaringsvariabler. Med to forklaringsvariabler x_1 og x_2 , for eksempel, kan vi modellere sannsynligheten for en "suksess" (e.g., at en SMS melding er spam) som

$$p_i = \frac{\exp(\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i})}{1 + \exp(\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i})}.$$

Problem 2b) (1P)

I dette eksempelet har vi en datasett med 'text' som mulige forklaringsvariable. Er dette et problem? Forklar med maks 2 setninger.

Løsning:

Å bruke tekster som en forklaringsvariabel er nødvendig for å klassifisere tekstene, men det *kan* fortsatt være problematisk å bruke uprosessert tekst som en forklaringsvariabel. Det er fordi det er veldig mye støy i tekster som ikke nødvendigvis hjelper klassifiseringen, som vil si at vi ideelt sett burde preprossesert teksten slik at vi får trekt ut den dataen fra teksten vi vet kan være interessant, og bruke det som forklaringsvariabler i stedet.

For å hjelpe Python med å jobbe med tekstmeldingene, må vi *vektorisere* dataene. Hva betyr det? Vel, det vil si at vi samler hvert ord og frekvensen til ordet i hver SMS. Vi oppretter en matrise hvor hver kolonne representerer et ord, og så skriver vi "1" i matrisen hvis ordet finnes i SMS meldingen som står i den respektive raden (eller "2" hvis ordet finnes dobbelt osv), ellers skriver vi "0". Vektoriseringen vil derfor produsere en matrise med gaaaanske mange 0-er (det er mange forskjellig ord, og de aller fleste ikke forekommer i en gitt SMS tekst), og derfor kan vi komprimere matrisen ganske mye.

Alt det kan vi gjøre med bruk av følgende kode:

```
# Vektorisering av SMS tekstene
vect = CountVectorizer()
vect.fit(X_train_raw)
x_train = vect.transform(X_train_raw)
x_test = vect.transform(X_test_raw)
x_val = vect.transform(X_val_raw)
```

```
# Hva betyr dette? x_train, x_test og x_val er nå komprimerte matriser,
# hvor hver rad er en SMS og hver kolonne er et ord som finnes i de ulike SMS meldingene.

# Dette er komprimerte matriser, og derfor er det litt vanskelig å se på dem.
# Men vi kan prøve å se litt på en dekomprimert versjon av x_train her:
# (Obs! Men vi skal fortsette med de komprimerte versjonene x_train, x_test, x_val etterpå)
type(x_train)
print(pd.DataFrame(x_train.toarray()))
```

```
      0      1      2      3      4      5      6      7      8      9      ...  7176  \
0      0      0      0      0      0      0      0      0      0      0      ...    0
1      0      0      0      0      0      0      0      0      0      0      ...    0
2      0      0      0      0      0      0      0      0      0      0      ...    0
3      0      0      0      0      0      0      0      0      0      0      ...    0
4      0      0      0      0      0      0      0      0      0      0      ...    0
...    ...    ...    ...    ...    ...    ...    ...    ...    ...    ...    ...
3895    0      0      0      0      0      0      0      0      0      0      ...    0
3896    0      0      0      0      0      0      0      0      0      0      ...    0
3897    0      0      0      0      0      0      0      0      0      0      ...    0
3898    0      0      0      0      0      0      0      0      0      0      ...    0
3899    0      0      0      0      0      0      0      0      0      0      ...    0

      7177  7178  7179  7180  7181  7182  7183  7184  7185
0      0      0      0      0      0      0      0      0      0
1      0      0      0      0      0      0      0      0      0
2      0      0      0      0      0      0      0      0      0
3      0      0      0      0      0      0      0      0      0
4      0      0      0      0      0      0      0      0      0
...    ...    ...    ...    ...    ...    ...    ...    ...    ...
3895    0      0      0      0      0      0      0      0      0
3896    0      0      0      0      0      0      0      0      0
3897    0      0      0      0      0      0      0      0      0
3898    0      0      0      0      0      0      0      0      0
3899    0      0      0      0      0      0      0      0      0
```

[3900 rows x 7186 columns]

Problem 2c) (1P)

Hvor mange SMS-meldinger er det i treningsdatasettet? Hvor mange forskjellige ord finnes i disse meldingene?

Løsning:

I matrisen over ser vi vektorisering av treningsdatasettet, der hver rad tilsvarer én SMS og hver kolonne ett ord. I matrisen ser vi at det er 3900 rader og 7186 kolonner, altså er det 3900 SMS-meldinger og 7186 ord i treningsdatasettet.

Okei... nå har vi egentlig et problem, fordi vi har flere forklaringsvariabler (flere forskjellige ord!) enn observasjoner. Vi har ikke snakket om hvordan vi må bruke logistisk regresjon i en slikt tilfelle, så dere trenger ikke å forstå hvordan det fungerer og hva som akkurat skjer når vi tilpasser en logistisk regresjonsmodell (vi bruker noe som kalles *regularisering*, men som sagt er det utenfor pensum).

Det som er viktig er at vi gjør en logistisk regresjon med binær respons $y = 0$ for "ham" og $y=1$ for "spam". Det gjør vi nå for treningsdatasettet:

```
# Tilpass en logistisk regression med treningsdata:

lr = LogisticRegression(penalty='l2')
lr.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
## Og nå kan vi regne ut feilrate på treningssettet:

lrscore = lr.score(x_train,y_train)
print('Logistic regression feilrate: ',round(1-lrscore,5))
```

```
Logistic regression feilrate: 0.00231
```

Problem 2d) (2P)

(i) Hvor stor er andelen av SMS-meldingene som er riktig klassifisert i treningssettet?

(ii) Er feilraten i valideringssettet større eller mindre enn i treningssettet, og hvorfor? For å finne dette ut, kopier koden hvor vi har regnet ut feilrate på treningssettet, og erstatt trenings- med valideringssettet. Viktig: Bruk modellen som du har tilpasset til treningssettet!

Løsning:

(i)

Feilraten er definert som *andel feilklassifiserte observasjoner*. Vi kan derfor finne andel riktig klassifiserte observasjoner ved komplementregelen:

$$P(\text{riktig klassifisert}) = 1 - P(\text{feil klassifisert}) = 1 - 0.00231 = 0.99769 = 99.77\%.$$

Dette kan også leses av ved å bruke `LogisticRegression.score(...)`.

(ii)

Ut i fra kodelinen under kan vi se at valideringssettet hadde en feilrate på `0.02153`, som er mye høyere enn feilraten til treningssettet (som forventet). Dette er fordi modellen har blitt trent på treningssettet.

```
# Tilhører oppgave (ii)

lrscore = lr.score(x_val,y_val)
print('Logistic regression feilrate: ',round(1-lrscore,5))
```

```
Logistic regression feilrate: 0.02153
```

Vi har lært i undervisningen at logistisk regresjon predikerer sannsynligheten p_i for at $y_i = 1$. Klassifiseringsregelen som brukes er å forutsi $\hat{y}_i = 1$ hvis $p_i > 0.5$ og ellers $y_i = 0$. Det er akkurat det som scorefunksjonen som brukes ovenfor (`lr.score`) faktisk gjør. Valget av 0.5 er imidlertid vilkårlig, og det kan være at en annen cutoff ville gi bedre prediksjoner.

Problem 2e) (3P)

(i) Bruk igjen modellen over som du har tilpasset til treningsdatasettet, men bruk alle cutoff-verdier 0.1, 0.2, ... 0.9 for å klassifisere observasjonene i valideringssettet (se og kjør kode under). Hvilken cutoff fungerer best og hva er den tilsvarende feilraten på valideringssettet?

(ii) Nå kan du bruke den beste cutoff-verdien og regne ut feilraten på testsettet. Tilpass koden.

(iii) Hvorfor er det ikke rart at vi på testsettet har en litt høyere feilrate enn på både trenings og valideringssettet? Maks 2 setninger.

Løsning:

(i)

$cutoff = 0.2$ er den verdien som fungerer best. Ved denne verdien får vi den laveste feilraten, nemlig 0.01675.

(ii)

Se nedre halvdel av kodecelle under (øvre halvdel er fra oppgavebeskrivelsen).

Her får vi $feilrate = 0.02392$

(iii)

Først og fremst er modellen basert på treningssettet, som vil si at feilraten der alltid vil være minst. I tillegg til dette vil feilraten på valideringssettet være mindre fordi valideringssettet inneholder mer data, som fører til at utslaget på eventuelle uteliggere negeres.

```
# For å få sannsynlighet P(y=1) for at en SMS var spam
val_prob = lr.predict_proba(X=x_val)[:,[1]]
test_prob = lr.predict_proba(X=x_test)[:,[1]]
train_prob = lr.predict_proba(X=x_train)[:,[1]]
results = []

# Og nå klassifiser med forskjellige cutoff verdier mellom 0.1 og 0.9:
for x in np.arange(0.1, 1.0, 0.1):
    cutoff = x
    # Prediker spam eller ham avhengig av P(y=1)
    y_valpred = np.where(val_prob > cutoff, 1, 0)
    y_testpred = np.where(test_prob > cutoff, 1, 0)
    y_trainpred = np.where(train_prob > cutoff, 1, 0)

    # Finn andel korrekte klassifikasjoner
    print("cutoff:", round(x,1), "Accuracy:", round(accuracy_score(y_true=y_val, y_pred=y_valpred),5),
          "Feilrate:", round(1-accuracy_score(y_true=y_val, y_pred=y_valpred),5))
    results.append({
        "cutoff": cutoff,
        "val_error": round(1-accuracy_score(y_true=y_val, y_pred=y_valpred),5),
        "test_error": round(1-accuracy_score(y_true=y_test, y_pred=y_testpred),5),
        "train_error": round(1-accuracy_score(y_true=y_train, y_pred=y_trainpred),5)
    })

# Tilhører oppgave (ii)
min_val_result = min(results, key = lambda x: x.get("val_error"))
cutoff = min_val_result.get("cutoff")
val_error = min_val_result.get("val_error")
test_error = min_val_result.get("test_error")
train_error = min_val_result.get("train_error")

print("")
print("Output for oppgave (ii):")
print(f"cutoff = {cutoff}")
print(f"    Feilrate på valideringssett: {val_error}")
print(f"    Feilrate på testsett          : {test_error}")
print(f"    Feilrate på treningssett      : {train_error}")
```

```
cutoff: 0.1 Accuracy: 0.97727 Feilrate: 0.02273
cutoff: 0.2 Accuracy: 0.98325 Feilrate: 0.01675
cutoff: 0.3 Accuracy: 0.98206 Feilrate: 0.01794
cutoff: 0.4 Accuracy: 0.97847 Feilrate: 0.02153
cutoff: 0.5 Accuracy: 0.97847 Feilrate: 0.02153
cutoff: 0.6 Accuracy: 0.97727 Feilrate: 0.02273
cutoff: 0.7 Accuracy: 0.97727 Feilrate: 0.02273
cutoff: 0.8 Accuracy: 0.9701 Feilrate: 0.0299
cutoff: 0.9 Accuracy: 0.96053 Feilrate: 0.03947
```

```
Output for oppgave (ii):
cutoff = 0.2
  Feilrate på valideringssett: 0.01675
  Feilrate på testsett       : 0.02392
  Feilrate på treningssett   : 0.00179
```

Del 2: k -nærmeste-nabo-klassifikasjon (KNN)

Spamfiltereksempelet dessverre ikke brukes for KNN klassifikasjon. Problemet er noe som heter "curse of dimensionality" og det vises når det er for mange forklaringsvariabler (se her: https://en.wikipedia.org/wiki/Curse_of_dimensionality).

Vi skal derfor bruke et annet dataset hvor vi vil predikere om en person kjøper en campingvognforsikring eller ikke, avhengig av 85 forklaringsvariabler. Datasettet med 5822 observasjoner er en del av ISLR R-pakken, og mer informasjon finner du her: <https://liacs.leidenuniv.nl/~puttenpwhvander/library/cc2000/data.html>.

Forsikringsselskapet vil bruke prediksjonsmodellen til å velge ut noen potensielle kunder som de kan ringe, slik at at får solgt mulige forsikringer uten å ringe for mange personer.

```
# Vi begynner igjen med å importere pakker og funksjoner vi trenger i oppgaven

import numpy as np
import pandas as pd

# plotting
import matplotlib.pyplot as plt
import seaborn as sns

# Trening og testsett, evaluering av klassifikasjonsmetoder
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
```

Caravan datasettet

```
df = pd.read_csv('https://www.math.ntnu.no/emner/IST100x/ISTx1003/Caravan.csv', delimiter=',')

df.shape
df.describe
```

```
<bound method NDFrame.describe of
0    0.680848 -0.272557  0.406662 -1.216859  0.779338 -0.694251  0.217425
1    0.992212 -0.272557 -0.859426 -1.216859  0.779338  0.302526 -0.365379
2    0.992212 -0.272557 -0.859426 -1.216859  0.779338 -0.694251 -0.365379
3   -1.187335 -0.272557  0.406662  0.010754 -0.970896  1.299302 -0.948183
4    1.225735 -0.272557  1.672750 -1.216859  1.479432  0.302526 -0.365379
...
5817  0.914371 -0.272557 -2.125514 -1.216859  0.779338 -0.694251  0.800229
5818  0.836530 -0.272557  1.672750  1.238367  0.779338  0.302526 -0.365379
5819  0.680848 -0.272557  0.406662  1.238367  0.779338 -0.694251  0.800229
5820  0.758689 -0.272557  0.406662 -1.216859  0.779338 -0.694251  1.383033
5821  0.680848 -0.272557  0.406662  0.010754  0.779338 -0.694251  0.800229

      MGODOV  MGODGE  MRELGE  ...  APERSONG  AGEZONG  AWAOREG  \
0   -0.068705 -0.161802  0.427633  ... -0.073159 -0.081048 -0.059915
1   -0.068705  0.464119 -0.096069  ... -0.073159 -0.081048 -0.059915
2    0.914094  0.464119 -1.667175  ... -0.073159 -0.081048 -0.059915
3    0.914094  0.464119 -0.619771  ... -0.073159 -0.081048 -0.059915
```

```

4      -0.068705  0.464119  0.427633  ... -0.073159 -0.081048 -0.059915
...      ...      ...      ...      ...      ...      ...
5817 -0.068705 -0.787723 -2.714580  ... -0.073159 -0.081048 -0.059915
5818 -0.068705  0.464119 -0.096069  ... -0.073159 -0.081048 -0.059915
5819 -1.051503 -0.161802 -0.619771  ... -0.073159 -0.081048 -0.059915
5820 -1.051503 -0.787723  0.427633  ... -0.073159 -0.081048 -0.059915
5821 -0.068705 -0.787723  0.427633  ... -0.073159 -0.081048 -0.059915

      ABRAND  AZEILPL  APLEZIER  AFIETS  AINBOED  ABYSTAND  Purchase
0      0.764905 -0.022704 -0.073644 -0.150608 -0.08734 -0.118806      No
1      0.764905 -0.022704 -0.073644 -0.150608 -0.08734 -0.118806      No
2      0.764905 -0.022704 -0.073644 -0.150608 -0.08734 -0.118806      No
3      0.764905 -0.022704 -0.073644 -0.150608 -0.08734 -0.118806      No
4      0.764905 -0.022704 -0.073644 -0.150608 -0.08734 -0.118806      No
...      ...      ...      ...      ...      ...      ...
5817  0.764905 -0.022704 -0.073644 -0.150608 -0.08734 -0.118806      No
5818  0.764905 -0.022704 -0.073644 -0.150608 -0.08734 -0.118806      No
5819  0.764905 -0.022704 -0.073644 -0.150608 -0.08734 -0.118806      Yes
5820 -1.014271 -0.022704 -0.073644 -0.150608 -0.08734 -0.118806      No
5821 -1.014271 -0.022704 -0.073644 -0.150608 -0.08734 -0.118806      No

[5822 rows x 86 columns]>

```

```

# Vi erstatter Yes/No variablen med 1/0
df.replace({'Yes', 'No'}, (1, 0), inplace=True)

# Se på antall kjøpere i datasettet.
# Vi ser at bare en liten andel av mulige kunder faktisk kjøper en forsikring:
print(df["Purchase"].value_counts())

0      5474
1       348
Name: Purchase, dtype: int64

```

Problem 2f) (1P)

Hva er *andelen* av kunder som faktisk kjøper en forsikring?

Løsning:

Vi ser at 348 av $(348 + 5474) = 5822$ kunder kjøpte forsikring.

$$\frac{348}{5822} \approx 0.05977$$

Dermed er det omtrent 6% av kunder som faktisk kjøper forsikring.

Vi fortsetter med å dele datasettet inn i et trenings- og valideringssett (vi lager ikke testsett for å ikke gjøre oppgaven for vanskelig).

```

# Del dataene i et trenings- og et valideringssett (60-40%)
df_tren, df_val = train_test_split(df, test_size = 0.4, random_state=1, stratify=df['Purchase'])

X_tren = df_tren.drop('Purchase', axis=1)
X_val = df_val.drop('Purchase', axis=1)

```


Og så tilpasser vi en k -nærmeste-nabo (KNN) klassifikasjon med $k=1$. Deretter ser vi på feilraten og forvirringsmatrisen for valideringssettet:

```
# Start med k=1
k=1
knn = KNeighborsClassifier(n_neighbors=k,p=2)
knn.fit(X_tren, df_tren['Purchase'])

# Feilrate på testsettet (1-score):
1-knn.score(X_val, df_val['Purchase'])

# Forvirringsmatrise:
pd.DataFrame(
    confusion_matrix(y_true=df_val['Purchase'], y_pred=knn.predict(X_val)),
    index=['true:no', 'true:yes'],
    columns=['pred:no', 'pred:yes']
)
```

	pred:no	pred:yes
true:no	2087	103
true:yes	121	18

```
/opt/python/envs/default/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not have
warnings.warn(
/opt/python/envs/default/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not have
warnings.warn(
```

```
# Se på antall kjøpere og ikke-kjøpere i valideringssettet:

print(df_val["Purchase"].value_counts())
```

```
0    2190
1     139
Name: Purchase, dtype: int64
```

Problem 2g) (2P)

(i) Vi antar at alle potensielle kunder som predikeres at de skal kjøpe en forsikring i valideringssettet (**pred:yes**) blir kontaktet av en selger. Hva er nå andelen av kunder som ble kontaktet som faktisk kjøper en forsikring? Se på forvirringsmatrisen over, hvor vi har brukt $k = 1$, og vis din beregning.

(ii) Sammenlign andelen fra (i) med andelen kjøpere i valideringssttet. Har suksesraten blitt bedre, sammenlignet med et tilfeldig utvalg av kunder fra valideringssettet? Kommentér.

Løsning:

(i)

Vi kan se at $(103 + 18) = 121$ kunder blir kontaktet. Av disse 121 kundene kjøpte 18 stk. forsikring.

$$\frac{18}{121} \approx 0.14876$$

Omtrent 15% av kundene som ble kontaktet kjøpte forsikring.

(ii)

Vi ser fra kodecellen over at det var 139 kjøpere av $(139 + 2190) = 2329$ kunder totalt i valideringssettet.

$$\frac{139}{2329} \approx 0.059682$$

Dermed kjøpte omtrent 6% av kundene i valideringssettet forsikring.

Suksessraten på 15% fra (i) er dermed bedre.

Til slutt tilpasser vi KNN med $k = 1, 3, \dots, 11$, og deretter regnes feilrate ut på valideringssettet og plottes.

```
knaboer = np.arange(1,12,step=2)
val_feilrate = np.empty(len(knaboer))

for i,k in enumerate(knaboer):

    #Initialiser kNN med k neighbors
    knn = KNeighborsClassifier(n_neighbors=k,p=2) # p=2 gir euklidisk avstand

    # Tilpass modellen med treningssettet
    knn.fit(X_tren, df_tren['Purchase'])

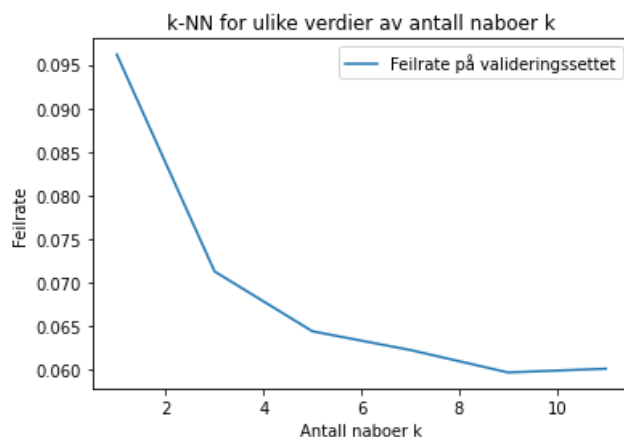
    # Beregn feilrate på valideringssett
    # Score er accuracy= "andel korrekt"
    val_feilrate[i] = 1-knn.score(X_val, df_val['Purchase'])

    # Lage plott
plt.title('k-NN for ulike verdier av antall naboer k')
plt.plot(knaboer, val_feilrate, label='Feilrate på valideringssettet')
plt.legend()
plt.xlabel('Antall naboer k')
plt.ylabel('Feilrate')
plt.show()

valres=np.vstack((knaboer, val_feilrate))
print("Valideringsfeilrate:")
print(valres.T)
```

Valideringsfeilrate:

```
[[ 1.          0.09617862]
 [ 3.          0.07127523]
 [ 5.          0.06440532]
 [ 7.          0.06225848]
 [ 9.          0.05968227]
 [11.          0.06011164]]
```



/opt/python/envs/default/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not have warnings.warn(

```

/opt/python/envs/default/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not hav
warnings.warn(
/opt/python/envs/default/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not hav
warnings.warn(
/opt/python/envs/default/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not hav
warnings.warn(
/opt/python/envs/default/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not hav
warnings.warn(
/opt/python/envs/default/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not hav
warnings.warn(

```

Problem 2h) (2P)

(i) Velg k med den minste feilraten og beregn forvirringsmatrise for valideringssettet (bruke samme code some for $k = 1$ oppover).

(ii) Er det lurt å bruke denne k -verdien? Hvorfor (ikke)?

Løsning:

(i)

Vi ser i tabellen og grafen for de ulike feilratene at $k = 9$ gir lavest feilrate.

Se kodecelle nedenfor for forvirringsmatrise.

(ii)

Ved valg av k -verdi velger man den verdien som gir lavest feilrate og er høyest mulig (dette gir mindre fleksibilitet). Denne k -verdien har lavest feilrate, og er nest høyest av alternativene, og sånn sett en god kandidat til k -verdi.

Til tross for dette, ser vi at vi har *vel*dig mange falske negative, som er bekymringsverdig i dette tilfellet.

Dog ser vi igjen at alle k -verdiene gir mange falske negative, så vi konkluderer at det er lurt å bruke $k = 9$.

```

# Tilhører oppgave (i)

k=9
knn = KNeighborsClassifier(n_neighbors=k,p=2)
knn.fit(X_tren, df_tren['Purchase'])

# Feilrate på testsettet (1-score):
1-knn.score(X_val, df_val['Purchase'])

# Forvirringsmatrise:
pd.DataFrame(
    confusion_matrix(y_true=df_val['Purchase'], y_pred=knn.predict(X_val)),
    index=['true:no', 'true:yes'],
    columns=['pred:no', 'pred:yes']
)

```

	pred:no	pred:yes
true:no	2188	2
true:yes	137	2

```

/opt/python/envs/default/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not hav
warnings.warn(
/opt/python/envs/default/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not hav
warnings.warn(

```

Oppgave 3: Klyngeanalyse (10 poeng)

Oppgaven inneholder følgende elementer:

- Last inn et datasett og undersøk innholdet.
- Bruk K -gjennomsnitt-klyngeanalyse for å finne klynger i bilder av håndskrevne tall (0-9).
- Diskuter resultatene av klyngeanalysen..
- Diskutere K -gjennomsnitt-klyngeanalyse vs. hierarkisk klyngeanalyse
- Kommenter resultatene fra en hierarkisk klyngeanalyse.

Finn klynger (struktur) i bildene av håndskrevne tall

Et stort felt inom maskinlæring er bildegjenkjenning (image recognition) og klassifisering. Dette betyr at vi vil laste inn et bilde til en statistisk algoritme, og får tilbake en etikett som beskriver hva bildet representerer, for eksempel en person, et tall, en sykdom,...

I dette eksempelet skal vi jobbe med data fra *mnist* (Modified National Institute of Standards and Technology) databasen, som for eksempel finnes her: <https://www.kaggle.com/c/digit-recognizer>. Datasettet inneholder digitaliserte bilder av håndskrevne siffer (0-9), og dette pleier å bli brukt for å trene maskinlæringalgoritmer for klassifisering (det betyr at du vil forutsi hvilket tall et bestemt bilde inneholder, slik at datamaskinen kan lese tall, for eksempel postnummer på et brev).

Her bruker vi dette datasettet til noe annet: Vi vil undersøke om vi finner struktur i bildene ved bruk av en klyngeanalyse. Vi vil jo kanskje håpe at de samme sifrene kommer til å bli gruppert i en klynge fordi de inneholder lignende mønstre. Her skal vi se om det er sant.

Vi har lært om to populære algoritmer som brukes til å lage klynger av objekter:

- K -gjennomsnitt (K -means) algoritmen: Målet med denne algoritmen er å plassere de individuelle observasjonene i K grupper (kalt klynger) basert på observasjonenes avstand til grupper av andre observasjoner. Algoritmen kjøres iterativt, og stoppes når at alle observasjoner som ligner hverandre er plassert i en gruppe (klynge) sammen.
- Hierarkisk klyngeanalyse (se nedenfor).

```
# importere pakker og funksjoner vi trenger i oppgave 3
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans # k-gjennomsnitt klyngeanalyse

from scipy.cluster.hierarchy import dendrogram, linkage

from matplotlib.offsetbox import OffsetImage, AnnotationBbox
```

Les inn datasettet

Vi begynner med å lese inn datasettet som inneholder informasjon om digitaliserte håndskrevne siffer. Hvert siffer finnes i en firkant med $28 \cdot 28 = 784$ pixler, og dermed er hvert bilde representert som en vektor med lengde 784.

```
## Les inn datasettet og se på de første 5 eksempler
```

```
images = pd.read_csv('https://www.math.ntnu.no/emner/IST100x/ISTx1003/images.csv', sep = ",", index_c  
images.head()
```

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V775	V776	V777	V778	V779	V780	V781	V782	V783
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0

5 rows × 784 columns

Å skrive ut headeren gir ingenting interessant: pikslene for disse bildene er gitt som et tall mellom 0 og 255 (gråskala), hvor 0 representerer helt svart og 255 representerer helt hvitt. Pikslene representert ovenfor er det øverste venstre hjørnet av bildet, og det nedre høyre hjørnet av bildet, som vi forventer ville være tomt (svart, og derfor 0).

Vi ønsker å standardisere disse tallene slik at de er mellom 0.0 og 1.0 før analysen utføres.

```
## Standardisering av pikselne
```

```
images = images/255
```

Utskrift av tilleggsinformasjon om strukturen og objektene til datasettet gir:

```
# hvilken type er bildet vårt  
print("Bildet har type", type(images))  
  
# bildet er en numpytabell. Hva er formatet?  
print("Formatet til tabellen er", images.shape)  
  
#Average colour in image 50  
print("Gjennomsnittsfarge i bilde er", images.iloc[49].mean())  
  
print('Dataformatet til en piksel er', type(images.iloc[1,1]))
```

```
Bildet har type <class 'pandas.core.frame.DataFrame'>  
Formatet til tabellen er (6000, 784)  
Gjennomsnittsfarge i bilde er 0.16487595038015201  
Dataformatet til en piksel er <class 'numpy.float64'>
```

Og så skal vi se hvordan de første 10 bildene i datasettet ser ut:

```

features = np.array(images)
features = features.reshape(features.shape[0], 28,28)

fig = plt.figure(figsize=(10,10))

for i in range(10):
    fig.add_subplot(1, 10, i+1)

    plt.imshow(features[i], cmap = 'gray')

    plt.xticks([])
    plt.yticks([])
    plt.tight_layout()

```



Det er ganske enkelt for oss mennesker å gjenkjenne disse individuelle sifferene -- selv om håndskrift kan være ganske uleselig. Nå vil vi se hvor bra K-gjennomsnittsalgoritmen finner struktur i disse dataene.

Problem 3a) (2P)

(i) Hvor mange bilder har vi i datasettet?

(ii) Hvilket siffer ligner det 50. bildet i datasettet vårt på? Lag et plott som viser dette sifferet. (Husk at Python begynner nummereringen med 0, og derfor refereres det 50. bildet til [49])

Løsning:

(i)

Vi ser i kodecellene over at tabellstørrelsen er 6000×784 , hvor hver rad er en vektor-representasjon av et bilde.

Vi har dermed 6000 bilder i datasettet, hvor hvert bilde er på 784 piksler.

(ii)

Se kodecelle under for å se løsningen.

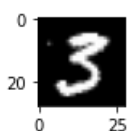
Her ser vi at det 50. bildet i datasettet ligner på sifferet 3.

```

fig = plt.figure(figsize=(1,1))
plt.imshow(features[49], cmap = 'gray')

```

<matplotlib.image.AxesImage at 0x7f63a9a67550>



Klyngeanalyse med K -gjennomsnitt

La oss nå endelig utføre K -gjennomsnittsalgoritme. Vi må gå gjennom følgende steg:

1. Angi antall klynger du ønsker
2. Initialiser K -gjennomsnitt-modellen

3. Tilpass K -gjennomsnitt-modellen

Vi bruker K -gjennomsnittsfunksjonen med $K = 10$ (se kode under), fordi vi håper jo å finne igjen de 10 sifferene (0-9) i klyngene.

I tillegg kan vi få tilgang til sentroidene i klyngene våre i tabellen `means.cluster_centers_`, og plotte dem for å se hvordan algoritmen mener de typiske bildene i de 10 gruppene ser ut.

Hint: Her kan du lese litt mer om KMeans-funksjonen: <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans>

```
# kodechunk kmeans

# Steg 1: Antall klynger
antall_klynger = 10

# Steg 2: Initialiser k-means algoritmen
kmeans = KMeans(n_clusters = antall_klynger, random_state = 1)

# Steg 3: Tilpass modellen
kmeans.fit(images)

# sentroidene (a)
sentroider = kmeans.cluster_centers_
```

Problem 3b) (3P)

(i) Tegn sentroidene av de 10 klyngene fra K -gjennomsnitt modellen. Tilpass koden oppover (ovenfor problem 3a).

(ii) Synes du at grupperingen i klynger er relevant og nyttig? Forklar. Maks 3 setninger.

(iii) Vi har valgt $K = 10$ for dette eksempelet fordi vi hadde håpet å finne klynger som representerer de 10 sifferene 0-9. Men generelt er K vilkårlig. Kom opp med en forslag for hvordan man (generelt, ikke nødvendigvis her) kan best velge K . (Se her, for eksempel: <https://medium.com/analytics-vidhya/how-to-determine-the-optimal-k-for-k-means-708505d204eb>). Beskriv i egne ord med maks 3 setninger!

Løsning:

(i)

Se kodecellen under.

(ii)

Vi vet hvilke 10 grupperinger vi vil ha i dette scenarioet, nemlig 0-9 - her ser vi at de fleste tallene har kommet tydelig frem, med unntak av tallene 5 og 4. Vi ser noe som kan se ut som omrisset til 5 bak 1eren på bilde nr 6, og antar også at den ene 9-eren er 4-ere og 9-ere slått sammen. Grupperingen er absolutt relevant og nyttig, men kunne vært enda bedre hvis disse to tallene hadde kommet tydeligere frem.

(iii)

Vi kjenner til to metoder for å finne optimal K , albuemetoden og silhuettmetoden, som man kan kombinere for best effekt. I albuemetoden regner vi ut "within-cluster-sum of squared errors", plotter disse opp mot k og velger den k -verdien der grafen har skarpest kant, altså en albue. Dog kan denne metoden være litt naiv dersom vi ikke har så klare grenser, og burde derfor kombineres med silhuettmetoden.

```
# Tilhører oppgave (i)

features = np.array(sentroider)
features = features.reshape(features.shape[0], 28,28)

fig = plt.figure(figsize=(10,10))

for i in range(antall_klynger):
    fig.add_subplot(1, 10, i+1)

    plt.imshow(features[i], cmap = 'gray')

    plt.xticks([])
    plt.yticks([])
    plt.tight_layout()
```



Hierarkisk klyngeanalyse

Vi fortsetter nå med å bruke hierarkisk klyngeanalyse for *mnist* datasettet. Vi gjør *Agglomerative Clustering* ved bruk av `sklearn.cluster` pakken. (Agglomerative Clustering er noe vi har lært om i undervisningen, men se også her hvis du har lyst til å vite mer: https://en.wikipedia.org/wiki/Hierarchical_clustering)

Fordi hierarkisk gruppering er tregt for store datasett, og særlig for grafiske data, ble et tilfeldig utvalg på 20 bilder valgt fra det originale datasettet for å bruke denne modellen for illustrasjon.

Problem 3c (3P):

(i) Vurder dendrogrammet nedenfor. Synes du at den hierarkiske gruppering-algoritmen har laget gode/meningfulle grupper av bildene?

(ii) I koden under har vi brukt gjeonnsnittskobling (`method = 'average'`). Hvordan fungerer gjeonnsnittskobling? Maks 2 setninger!

(iii) Velg en annen måte å koble klyngene sammen (vi har lært om dette i undervisningen) og lag et nytt dendrogram ved å tilpasse koden nedenfor. Kommenter resultatene. Ser det bedre/verre ut?

Løsning:

(i)

Det ser ut som dendrogrammet har greid å lage meningsfulle grupperinger av tallet 1, men ikke så veldig bra for andre tall. Det ser vi fordi bildene av 1 er gruppert ved siden av hverandre, og har lav avstand i diagrammet. Alle andre grupperinger inneholder ikke samme tall direkte, og har generelt stor avstand i diagrammet.

(ii)

Gjennomsnittkobling kobler klyngene basert på gjennomsnittlig avstand mellom elementene i klyngen. Det vil si at koblingspunktene vil ligge mer eller mindre i midten av klyngene, men ikke nødvendigvis på sentroiden som ved sentroidekobling.

(iii)

Vi har valgt å bruke singel kobling som alternativ måte å koble klyngene på. Det er fordi komplett kobling gir for mye oppmerksomhet til uteliggere, som det vil være veldig mye ved hånskrevne tall.

Se kodecelle markert med "Tilhører oppgave (iii)" under det første dendrogrammet.

Her ser vi fra dendrogrammet at koblingene er mer meningsfulle. Spesielt for tallene 1, 2 og 3, som gjennomsnittskoblingen fant en sterkere kobling for enn singel koblingen gjorde. Dessuten ser vi at på y-aksen at distansen for absolutt alle bildene er lenger i singel kobling, som er ønsket.

Dermed kan vi konkludere at singel koblingen ser bedre ut enn gjennomsnittskobling for denne modellen.

```
##Cluster

n_image = 20

sample = images.sample(n = n_image, random_state = 1)

sampleimg = np.array(sample).reshape(sample.shape[0], 28,28)

plt.figure(figsize=(10,10))
ax = plt.subplot()

# Bruk gjennomsnittskobling (method='average')
link = linkage(y = sample, method = 'average', metric = 'euclidean')

dendro = dendrogram(link)

dcoord = np.array(dendro["dcoord"])
icoord = np.array(dendro["icoord"])
leaves = np.array(dendro["leaves"])

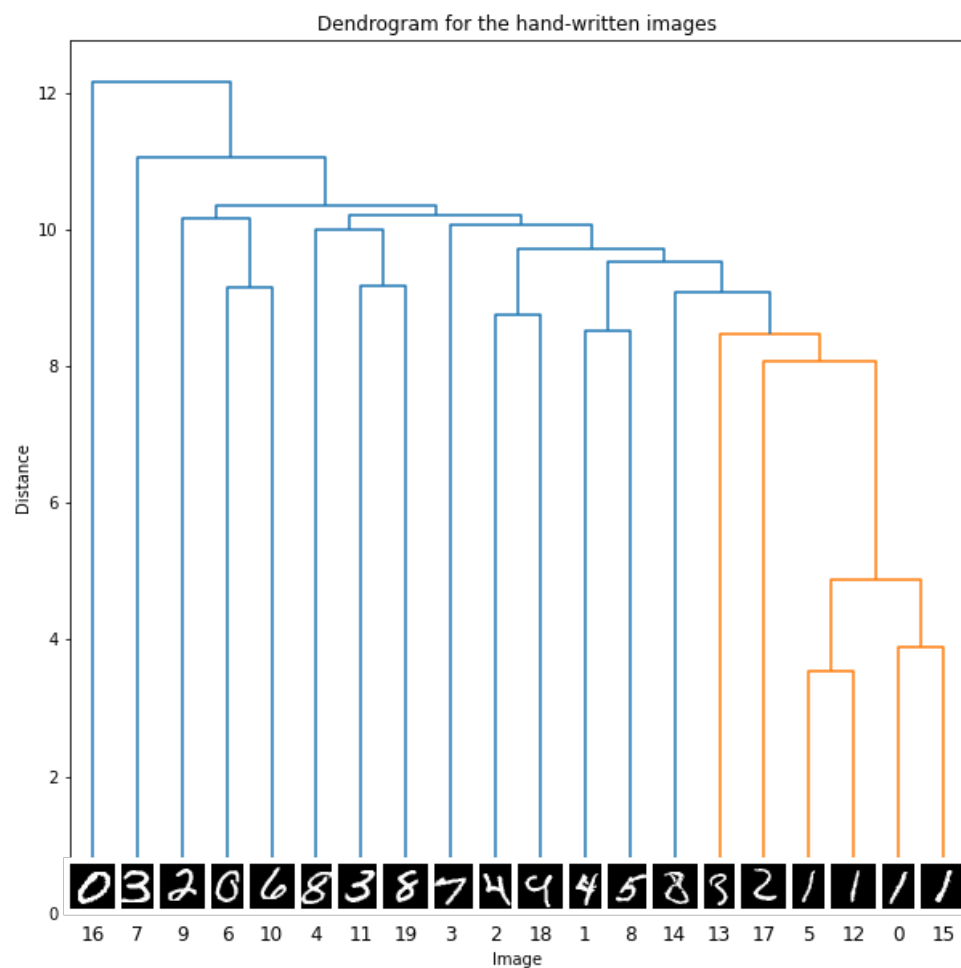
idx = np.argsort(dcoord[:, 2])

dcoord = dcoord[idx, :]
icoord = icoord[idx, :]

idx = np.argsort(link[:, :2].ravel())
label_pos = icoord[:, 1:3].ravel()[idx][:n_image]

for i in range(n_image):
    imagebox = OffsetImage(sampleimg[i], cmap = 'gray', interpolation = "bilinear")
    ab = AnnotationBbox(imagebox, (label_pos[i], 0), box_alignment=(0.5, -0.1),
                        bboxprops={"edgecolor" : "none"})
    ax.add_artist(ab)

plt.title('Dendrogram for the hand-written images')
plt.xlabel('Image')
plt.ylabel('Distance')
plt.show()
```



```
# Tilhører oppgave (iii)

n_image = 20

sample = images.sample(n = n_image, random_state = 1)

sampleimg = np.array(sample).reshape(sample.shape[0], 28,28)

plt.figure(figsize=(10,10))
ax = plt.subplot()

# Bruker nå singel kobling (method='single')
link = linkage(y = sample, method = 'single', metric = 'euclidean')

dendro = dendrogram(link)

dcoord = np.array(dendro["dcoord"])
icoord = np.array(dendro["icoord"])
leaves = np.array(dendro["leaves"])

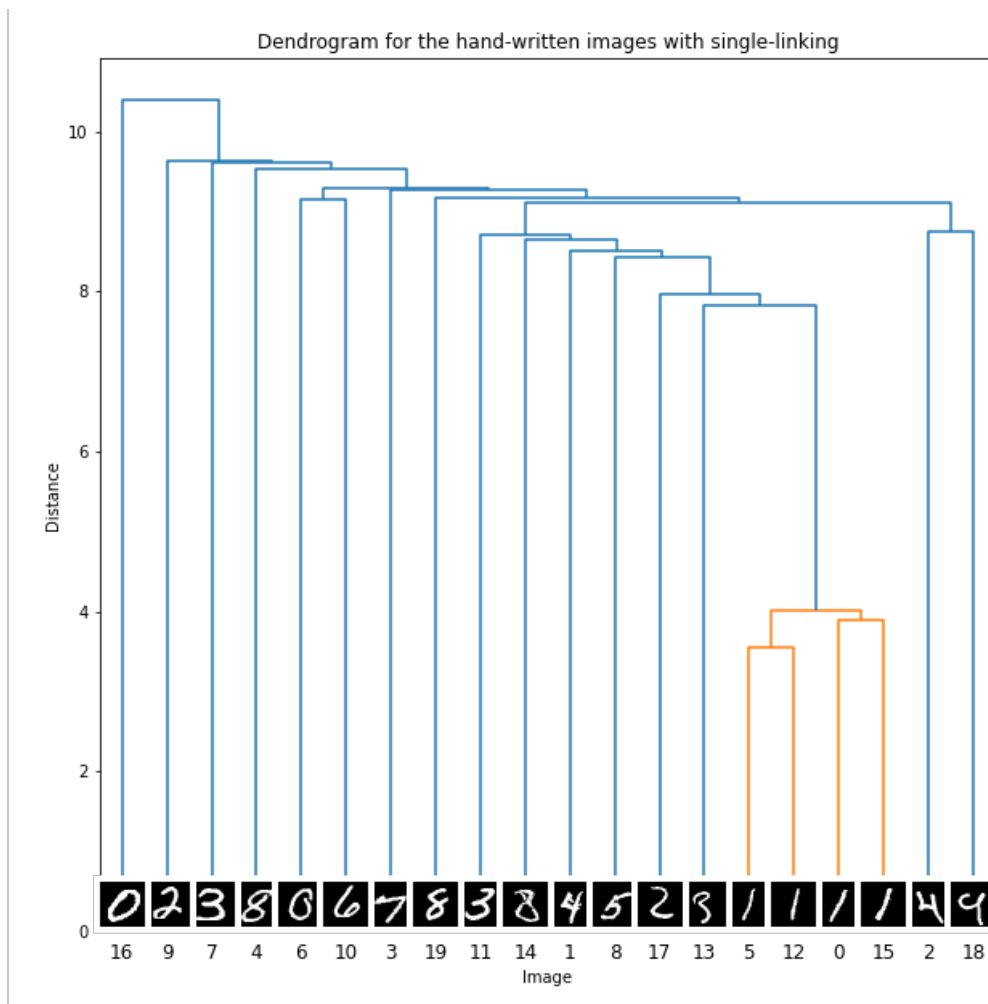
idx = np.argsort(dcoord[:, 2])

dcoord = dcoord[idx, :]
icoord = icoord[idx, :]

idx = np.argsort(link[:, :2].ravel())
label_pos = icoord[:, 1:3].ravel()[idx][:n_image]

for i in range(n_image):
    imagebox = OffsetImage(sampleimg[i], cmap = 'gray', interpolation = "bilinear")
    ab = AnnotationBbox(imagebox, (label_pos[i], 0), box_alignment=(0.5, -0.1),
                        bboxprops={"edgecolor" : "none"})
    ax.add_artist(ab)

plt.title('Dendrogram for the hand-written images with single-linking')
plt.xlabel('Image')
plt.ylabel('Distance')
plt.show()
```



Problem 3d) (2P)

(i) Med tanke på at verken de 10 klyngene funnet med K -gjennomsnittsalgoritmen, eller med hierarkisk klyngeanalyse, ser ut til å representere de 10 sifferene 0-9 veldig godt, tror du at klyngeanalyse (som er en ikke-veiledet metode) var en god metode for å bruke her?

(ii) Hvilke andre metoder vil du anbefale når målet er å klassifisere (predikere) siffer fra et håndskrevet tall?

Løsning:

(i)

Med tanke på at både K -gjennomsnittsalgoritmen og hierarkisk klyngeanalyse bearbeidet dataene dårlig virker det ikke som om klyngeanalyse er en god metode for disse dataene.

(ii)

Å analysere håndskrevne tall viser seg å være krevende for ikke-veiledet klyngeanalyse. Med disse dataene kunne vi derfor benyttet oss av en veiledet klyngeanalyse, selv om det ville krevd mer jobb.