Prosjekt1003_2021_solution

November 29, 2021

1 Løsning til tellende prosjekt i ISTx1003 - 2021

2 Oppgave 1: Regresjon (16 poeng)

[82]:
```
# 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"
```

2.1 Datasettet

[83]:
```
# 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]
```

[84]:
```
# 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
```

[85]:
```
# Få oversikt over datasettet

df.describe()
```

[85]:

	age	numco	edu	scoma	totcst	\
count	8142.000000	8142.000000	6680.000000	8142.000000	8142.000000	
mean	63.020722	1.901498	11.756587	11.648858	30865.642767	
std	15.537342	1.352183	3.429399	24.258079	45717.962369	
min	18.120000	0.000000	0.000000	0.000000	420.310000	
25%	53.250000	1.000000	10.000000	0.000000	5958.347500	
50%	65.145000	2.000000	12.000000	0.000000	14484.290000	
75%	74.400000	3.000000	14.000000	9.000000	36146.585000	
max	101.850000	9.000000	31.000000	100.000000	633212.000000	

	meanbp	hrt	resp	temp	pafi
count	8142.000000	8142.000000	8142.000000	8142.000000	6042.000000
mean	84.775608	97.709862	23.525669	37.100981	240.604037
std	26.725561	30.638716	9.464641	1.244434	110.482380
min	22.000000	0.000000	0.000000	31.700000	12.000000
25%	63.000000	72.000000	18.000000	36.200000	155.022500
50%	77.000000	100.000000	24.000000	36.700000	225.220000
75%	107.000000	120.000000	28.000000	38.090000	306.630000
max	195.000000	300.000000	90.000000	41.700000	890.380000

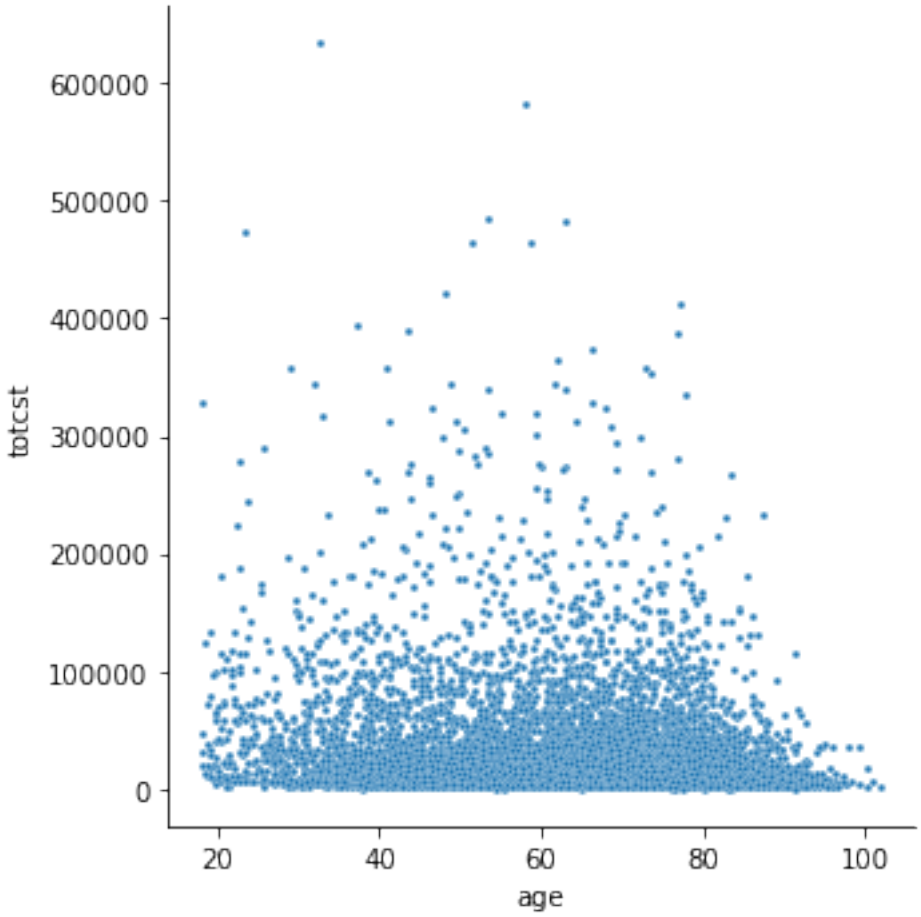
[86]:
```
# Sjekk datatyper for alle variabler

df.dtypes
```

[86]:
```
age         float64
dzgroup    category
numco         int64
edu         float64
income     category
scoma         int64
totcst      float64
race       category
meanbp        int64
```

```
hrt         float64
resp          int64
temp        float64
pafi        float64
dtype: object
```

2.2 Enkel lineær regresjon

[87]:
```
sns.relplot(x = 'age', y = 'totcst', kind = 'scatter', s=8, data = df)
plt.show()
```



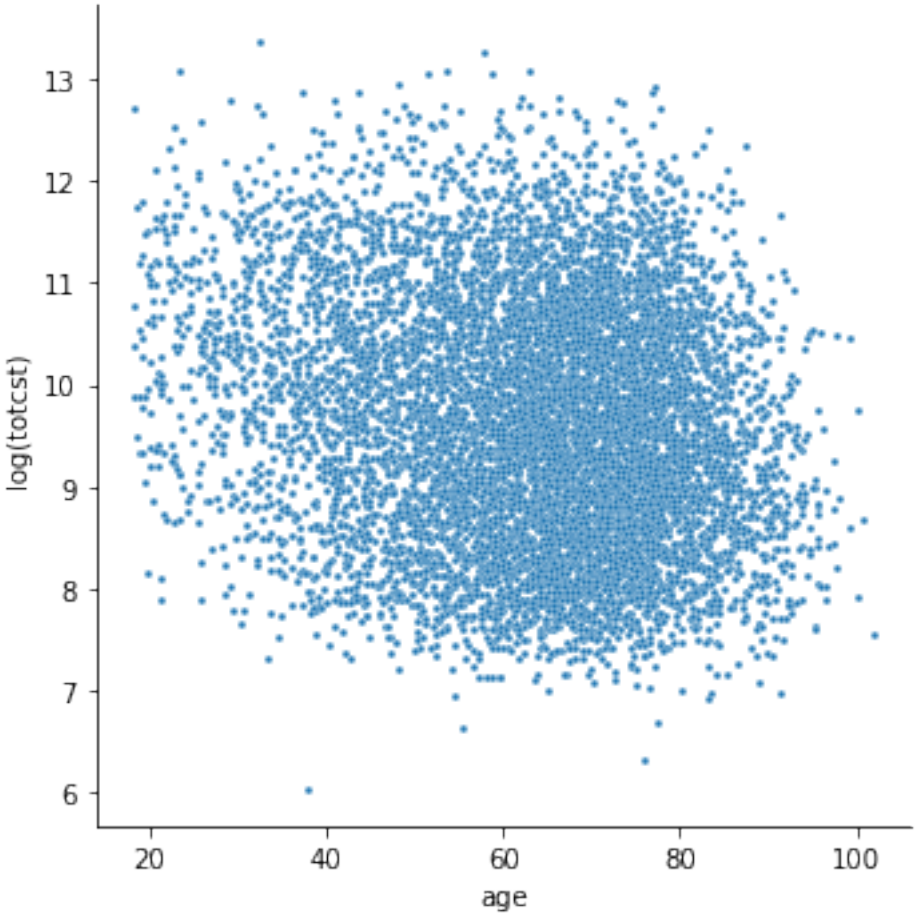
2.2.1 Problem 1a) (1P)

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

2.2.2 Løsning:

Here the students should realize that a linear model does not fit, as there is no linear relation visible (1P). A linea rmodel with totcst as response is therefore an invalid model. Potentially they will realize that the residuals would not be normally distributed, but this is not needed to get the point.

[88]:
```
# 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()
```



2.2.3 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.

2.2.4 Løsning:

Now the students should see/understand and therefore answer that a linear relationship fits much better than in 1a).

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 multippel) 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.

[89]:
```
# kodechunk Steg2-4

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

# Steg 3: Initaliser 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:                Thu, 21 Oct 2021   Prob (F-statistic):           1.83e-60
Time:                        12:07:00   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).

2.2.5 Problem 1c) (2P)

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

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

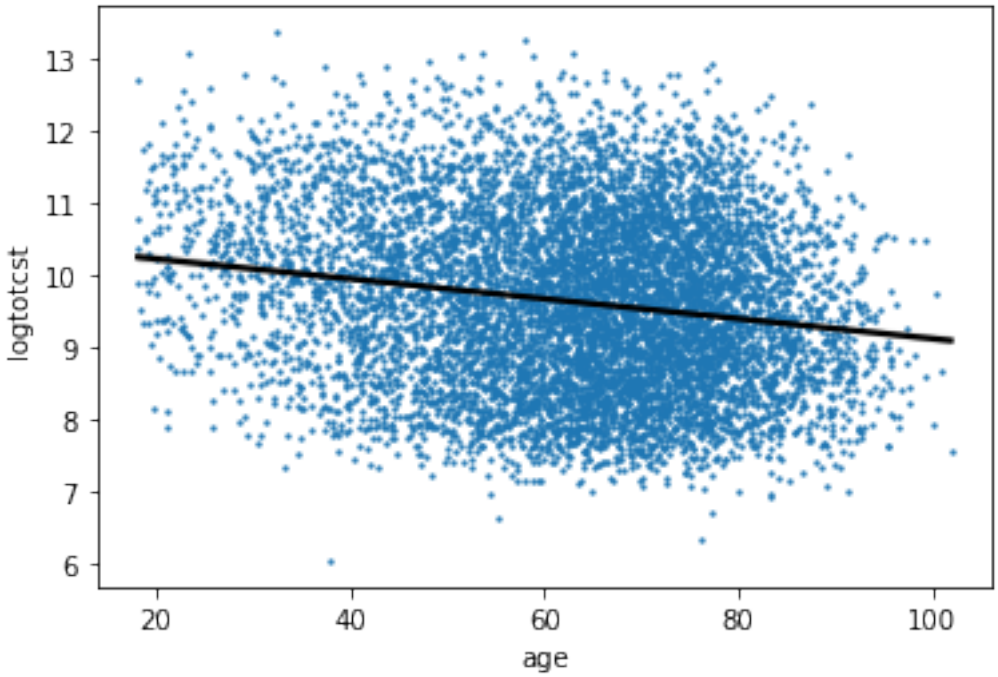
2.2.6 Løsning:

(i) $\hat{y} = 10.4962 - 0.0138 \cdot x + \epsilon$, where it should be made clear that y represents `logtotcst` and x is `age`. Important that there is a hat on y, but not on x, otherwise -0.5 points.

(ii) $R^2_{adj} = 0.032$, which means that only 3.2% of the variance of y is explained (0.5P). That is a *small* value (0.5P). Note that actually we should better look at R^2, as there is anyway only one covariate, but R^2_{adj} and R^2 are anyway essentially the same here, so this does not really matter.

[90]:
```
sns.regplot(x=df['age'],y=df['logtotcst'],line_kws={"color":␣
↪"black"},scatter_kws={"s":2})
```

[90]:
```
<AxesSubplot:xlabel='age', ylabel='logtotcst'>
```



2.2.7 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 sammenhang med `logtotcost`. Hvorfor (ikke)?

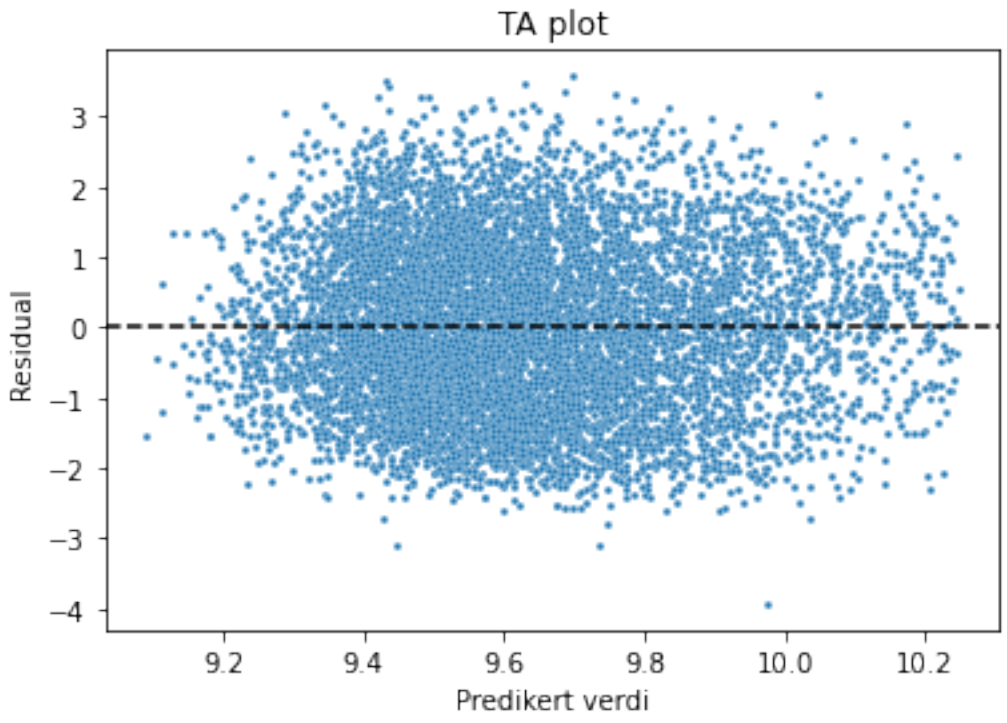
2.2.8 Løsning:

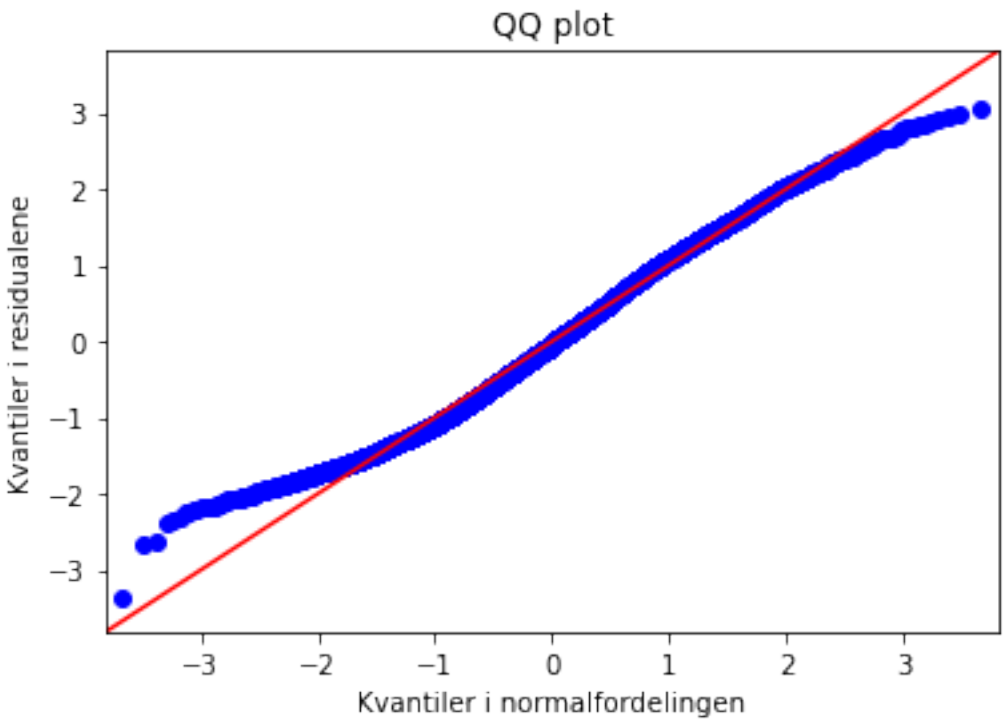
(i) $\hat{\beta}_{age} = -0.0138$ means that if a person is one year older, the response `logtotcst` is expected to decrease by a value of -0.0138 (0.5P for exactly this argument). Older patients are *less* expensive (0.5P).

(ii) CI is calulated as $\hat{\beta}_{age} \pm 1.96 \cdot SE(\hat{\beta}_{age})$, where $SE(\hat{\beta}_{age}) = 0.001$ (this is obviously a rounded value, but it's ok that the student use it; also ok if they use 2 instead of 1.96). Therefore CI=$[-0.01576, -0.01184]$ (0.5P). Interpretation: all values between -0.01576 and -0.01184 are plausible/compatible with the data, so we can be *quite sure* that the effect in fact is exists and is negative (0.5P).

(iii) $p = 0.000$ according to the python output. Attention: p=0 is, strictly speaking, not possible, so the students should notice that it would be a small, but non-zero value. In any case, it looks like we can be very sure about the linear relation, given that p must be very small (for sure $p < 0.0005$). **Attention**: We deduct -0.5P if the students think p=0, even if the answer otherwise is correct.

[91]:
```
# 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()
```





2.2.9 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.

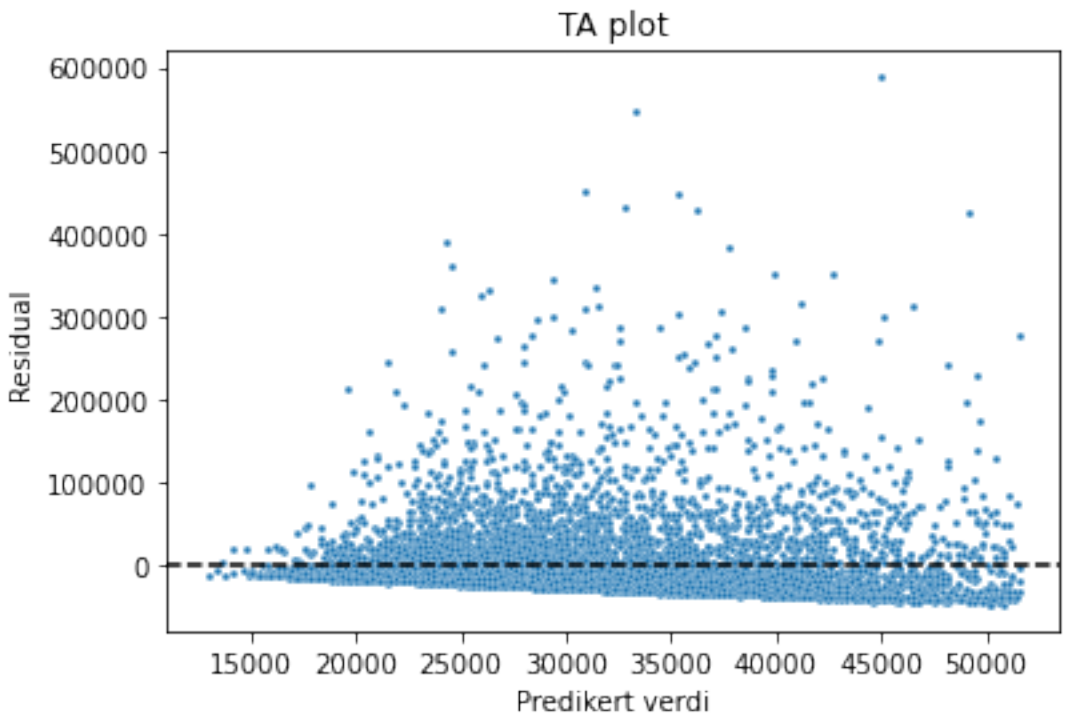
2.2.10 Løsning:

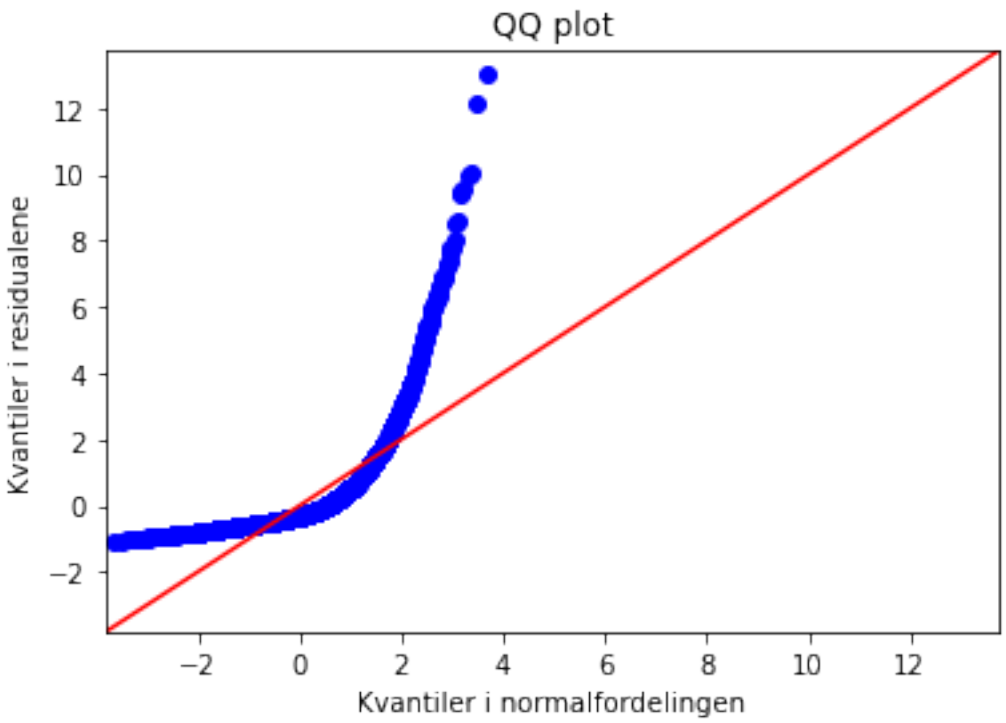
(i) The assumptions seem to be met: The TA plot shows points centered around the 0-line, without any special structure in the residuals (no funnel etc). In the QQ plot there are some small deviations in the tails, but this is quite common and not problematic. If students are a bit worried about this, that is still ok (no points deducted), because it is not an exact science.

(ii) 1P for the coding (see below), 1P for the answer, where the interpreation should be that the modeling assumptions are heavily violated, both indicated by the TA and QQ plots.

[92]:
```
# Steg 2: spesifiser matematisk modell
formel='totcst ~ age'
```

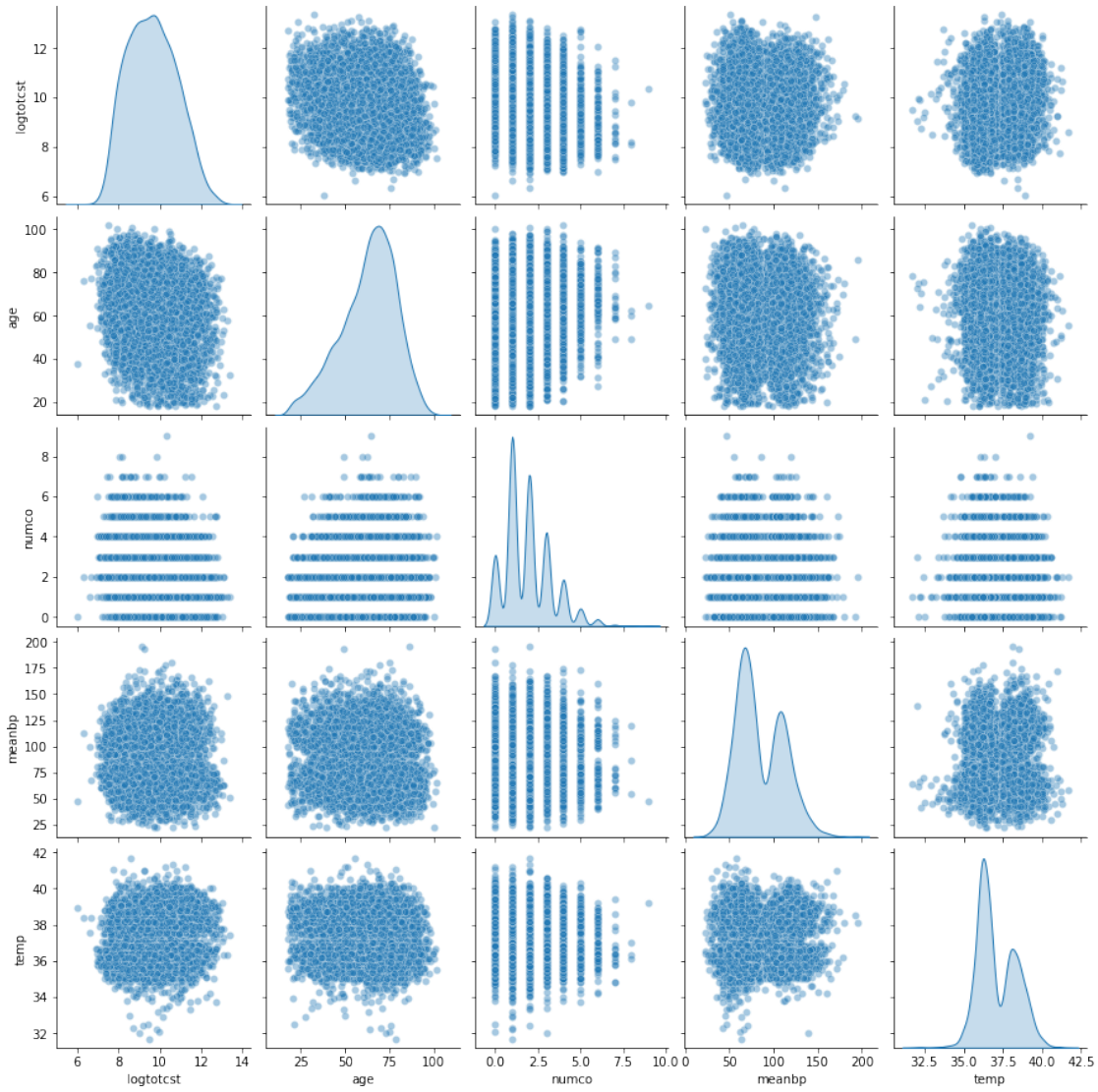
```
# Steg 3: Initaliser og tilpass en enkel lineær regresjonsmodell
# først initialisere
modell = smf.ols(formel,data=df)
# deretter tilpasse
resultat = modell.fit()

# 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()
```

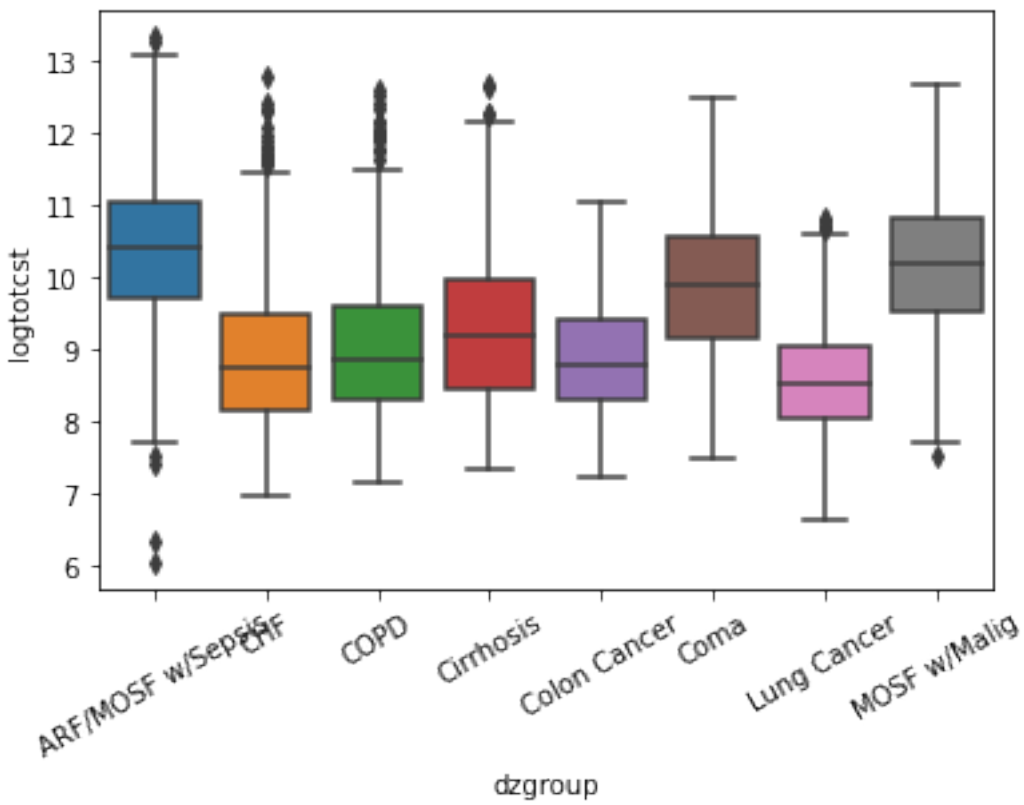


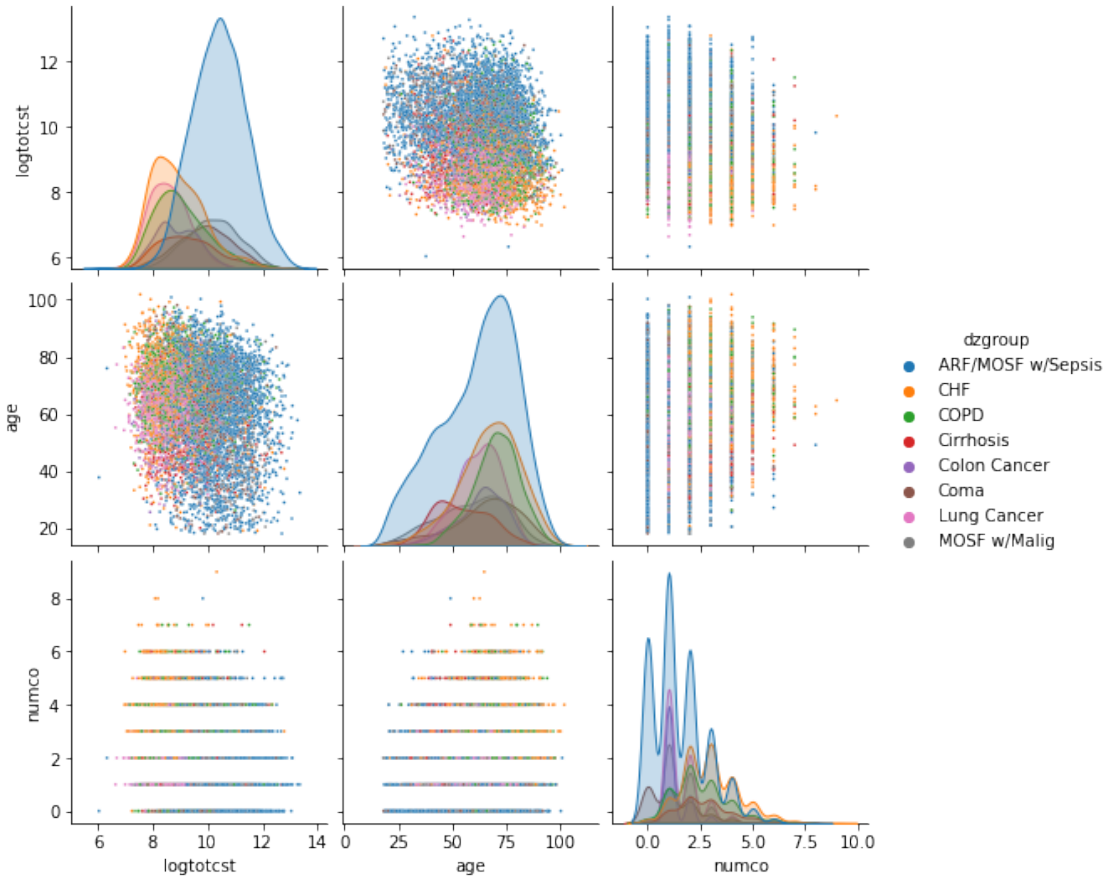


2.3 Multippel lineær regresjon

[93]:
```
# 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()
```


logtotcst
age
numco
meanbp
temp
logtotcst
age
numco
meanbp
temp


logtotcst
13
12
11
10
9
8
7
6
ARF/MOSF w/Sepsis
CHF
COPD
Cirrhosis
Colon Cancer
Coma
Lung Cancer
MOSF w/Malig
dzgroup



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

2.3.1 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 (lungesykdommen kols), som begge er like gamle, har samme antall comorbiditeter (`numco`), samme gjennomsnittig 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..

2.3.2 Løsning:

(i) (1P) See code below. The point is only given if the code produces exactly the correct results.

(ii) (1P) 12 (Intercept plus 11 slope parameters).

(iii) We see that the dummy variable for `Lung cancer` is estimated as $\hat{\beta}_{LungCancer} = -1.7661$, whereas for `COPD` we get $\hat{\beta}_{COPD} = -1.2107$. This thus means that Lung Cancer patients have a higher negative cost deviation from the reference level (which is `ARF/MOSF w/Sepsis`). Thus COPD patients are expected to be more expensive (0.5P for choosing the right disease (COPD is more expensive), and 0.5P for the right reasoning).

(iv) The calculation goes as $\exp(8.6679 - 1.5160 + (-0.0084) \cdot 50 + (-0.0546) \cdot 1 + (-0.0013) \cdot 130 + 0.0651 \cdot 36.2) = 7079$, although this contains some rounding error. Students who use the model without rounding will get something around 7145. Both are correct, as long as the explanation is given as here. If the exp() is forgotton and everything else is correctly described and calculated, give 0.5 points. Otherwise 0 points.

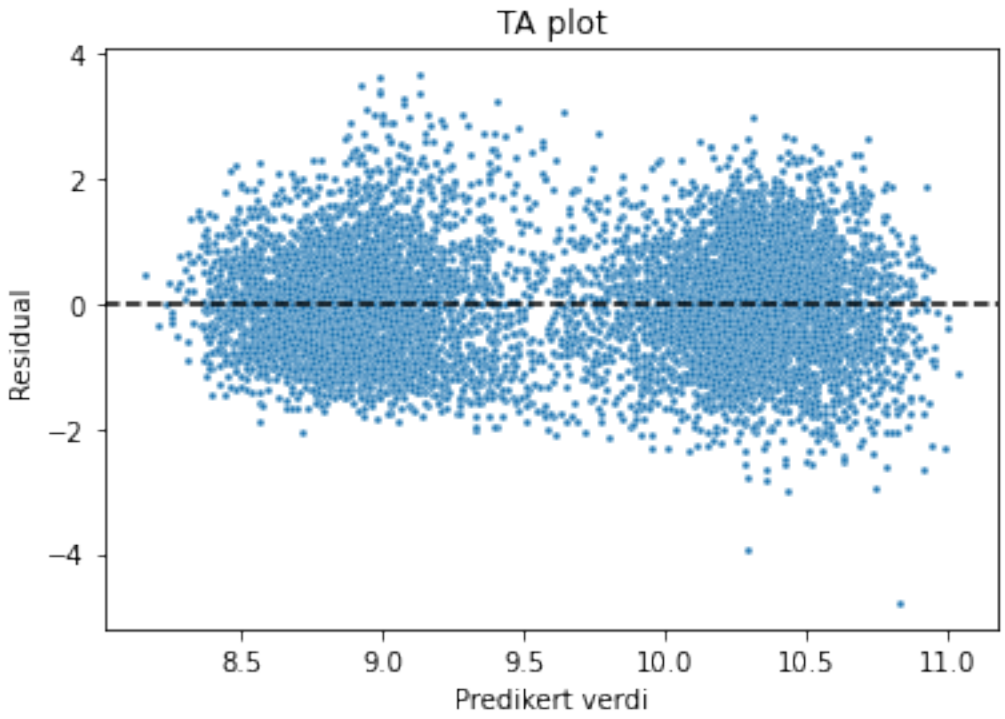
[96]:
```
# Steg 2: spesifiser matematisk modell
formel='logtotcst ~ age + dzgroup + numco +  meanbp + temp'

# Steg 3: Initaliser 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())
```

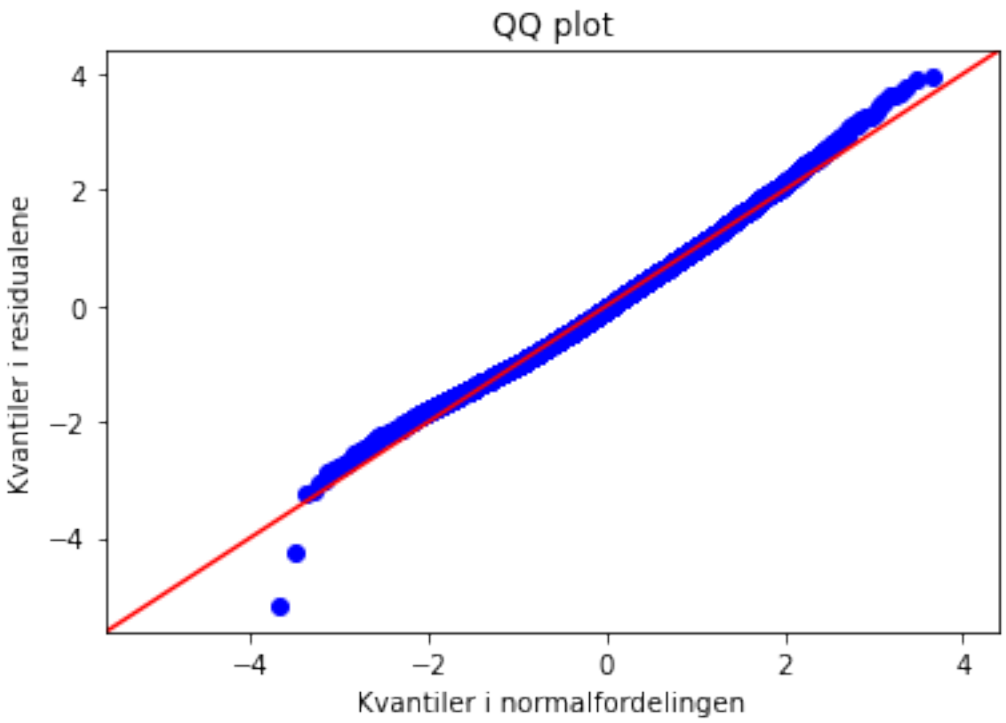
```
                            OLS Regression Results
==============================================================================
Dep. Variable:              logtotcst   R-squared:                       0.394
Model:                            OLS   Adj. R-squared:                  0.393
Method:                 Least Squares   F-statistic:                     480.0
Date:                Thu, 21 Oct 2021   Prob (F-statistic):               0.00
Time:                        13:00:40   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.
```

[95]:
```python
# 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()
```





2.3.3 Problem 1g) (2P)

(i) Forklaringsvariablen `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(iii). Var det en god idé å bruke multippel lineær regresjon, eller var enkel linear regresjon godt nok?

2.3.4 Løsning:

(i) Yes (0.5P), all diseases have a negative deviation from the reference group, and all p-values are small ($p = 0.000$ with rounding from python, which means that $p < 0.0005$ for sure (0.5P for the rasoning with the p-value, otherwise this half point cannot be given).

(ii) We now have R^2 and $R^2_{adj} = 0.39$ (0.5P for reporting either of them). This is *much* better than for simple regression, thus simple regression was not good enough (0.5P). Alternatively, say that multiple linear regression is much better.

3 Oppgave 2: Klassifikasjon (14 poeng)

3.1 Del 1: Logistisk regresjon

[97]:
```
# Vi begynner igjen med å importere pakker og funksjoner vi trenger i oppgaven

import pandas as pd
import numpy as np
import random

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
```

3.2 Spamfilter datasett

[98]:
```
# 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='\t',header=None)

# Gi nye navner til den første kolonnen (slik at den heter 'y') og den andre␣
 ↪kolonnen (som skal hete 'text').
# 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]
```

3.2.1 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.

3.2.2 Løsning:

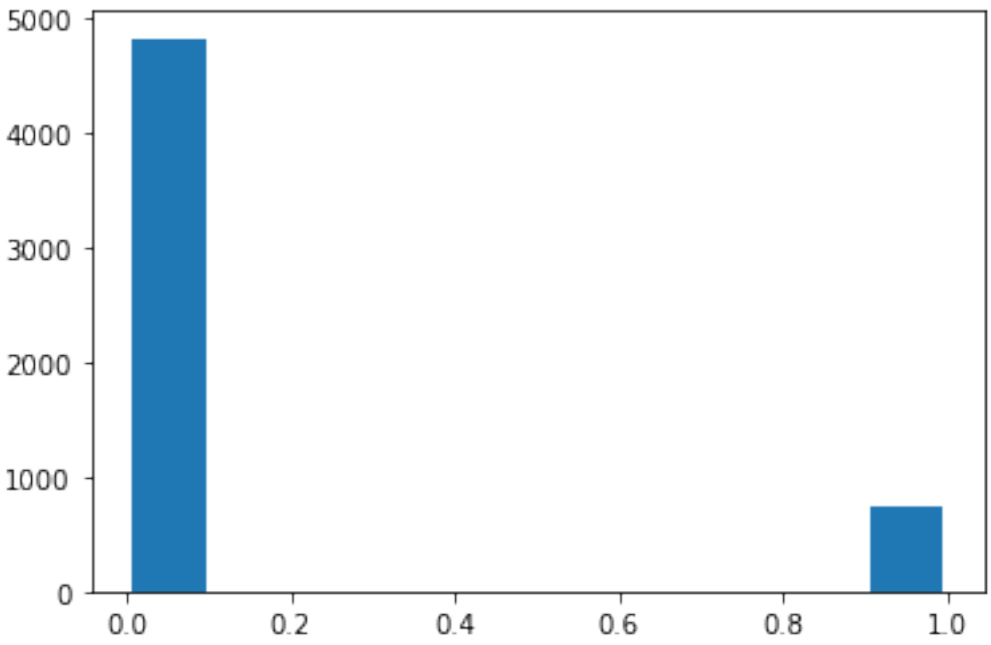
(i) See code below (0.5P for the code). We have 747 spam and 4825 ham (0.5P for the right answer; this 0.5 point is not given if spam/ham is mixed up or if the values are wrong).

(ii) 1P for the right code and histogram.

[99]:
```
# Code for (i) and (ii)
print(df["y"].value_counts())
plt.hist(df['y'],histtype='bar',rwidth=0.9)
```

```
0    4825
1     747
Name: y, dtype: int64
```

[99]:
```
(array([4825.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
         747.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <BarContainer object of 10 artists>)
```



3.2.3 Trening, validerings og testsett

[100]:
```
# 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_state=10)

# Og så del testsettet igjen i et test- og et valideringssett slik at begge␣
 ↪inneholder 15% (50% av de 30%)
X_test_raw, X_val_raw, y_test, y_val =␣
 ↪train_test_split(X_test_raw,y_test,test_size=0.5,random_state=10)
```

3.3 Logistisk regresjon

3.3.1 Problem 2b) (1P)

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

3.3.2 Løsning:

Here the students should understand that 'text' is not useable as a covariate. The problem is that each SMS contains another text, so it's impossible to fit a model. Similar explanations are accepted.

[102]:
```
# 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)
```

[103]:
```
# 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?

3.3.3 Løsning:

3900 SMS messages (0.5P) and 7186 different words (0.5P). No point is given if the numbers are mixed up or wrong.

[104]:
```
# Tilpass en logistisk regression med treningsdata:

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

[104]:
```
LogisticRegression()
```

[107]:
```
## 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
```

3.3.4 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!

3.3.5 Løsning:

(i) The error rate is 0.00231 (rounded), thus the proportion correctly classified messages is 1-0.00231 = 0.99769 (1P *only* for correct answer, otherwise 0P).

(ii) See code below where the validation dataset is used to predict from the model that was fitted to the trainings data (0.5P, 0P if the wrong model or dataset is used). The error rate is now *considerably higher* than for the trainingsset because the model is (too well) adapted (or *overfitted*) to the trainingsdata (0.5P for the argument along the lines that the trainingsdata is used to train the model, thus the error rate is expected to be smaller than for a new dataset).

[109]:
```
### 2d) (ii) Error rate for the validation set

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

```
Logistic regression feilrate:  0.02153
```

3.3.6 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.

3.3.7 Løsning:

(i) Correct answer: cutoff = 0.2 (0.5P), error rate 0.01675 (0.5P). Just running the code does not give points.

(ii) 0.5P for the code below (2e (ii)). Error rate in the test set: 0.02392 (0.5P).

(iii) The trainingsset is used to adjust the model, and the validation set is used to select the cutoff, thus both datasets were already used and the model "optimized" to them. We give 0.5P for the correct reasoning for the training set, 0.5P for the reasoning regarding the validation set.

[113]:
```
# For å få sannsynlighet P(y=1) for at en SMS var spam
val_prob = lr.predict_proba(X=x_val)[:,[1]]

# 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)

    # 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))
```

```
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
```

[114]:
```
# 2e) (ii)
cutoff = 0.2
```

```
test_prob = lr.predict_proba(X=x_test)[:,[1]]

# Prediker sannsynlighet for spam for testsett
y_testpred = np.where(test_prob > cutoff, 1, 0)

# Finn andel korrekte klassifikasjoner
print("Feilrate:", round(1-accuracy_score(y_true=y_test, y_pred=y_testpred),5))
```

```
Feilrate: 0.02392
```

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

[9]:
```
# 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
```

3.5 Caravan datasettet

[10]:
```
df = pd.read_csv('https://www.math.ntnu.no/emner/IST100x/ISTx1003/Caravan.csv',␣
↪delimiter=',')

df.shape
df.describe
```

[10]:
```
<bound method NDFrame.describe of        MOSTYPE  MAANTHUI   MGEMOMV  MGEMLEEF
MOSHOOFD    MGODRK    MGODPR  \
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]>
```

[11]:
```python
# 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
```

3.5.1 Problem 2f) (1P)

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

3.5.2 Løsning:

The proportion of customers that buys an insurance is $348/(5822) = 0.05977327$ **(only right/wrong answers possible, 0P for anything wrong in the calculation).**

[12]:
```
# 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)
```

[13]:
```
# Start med k=1
k=1
knn = KNeighborsClassifier(n_neighbors=k,p=2)
knn.fit(X_tren, df_tren['Purchase'])

# Feilrate på testsettet:
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']
)
```

[13]:

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

[14]:
```
# Se på antall kjøpere og ikke-kjøpere i valideringssettet:

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

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

3.5.3 Problem 2g) (2P)

(i) Vi antar at alle potensielle kunder som predikeres at de skal kjøpe en forsikring i valideringssettet **(pred:yes)** blir kontaktert fra 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.

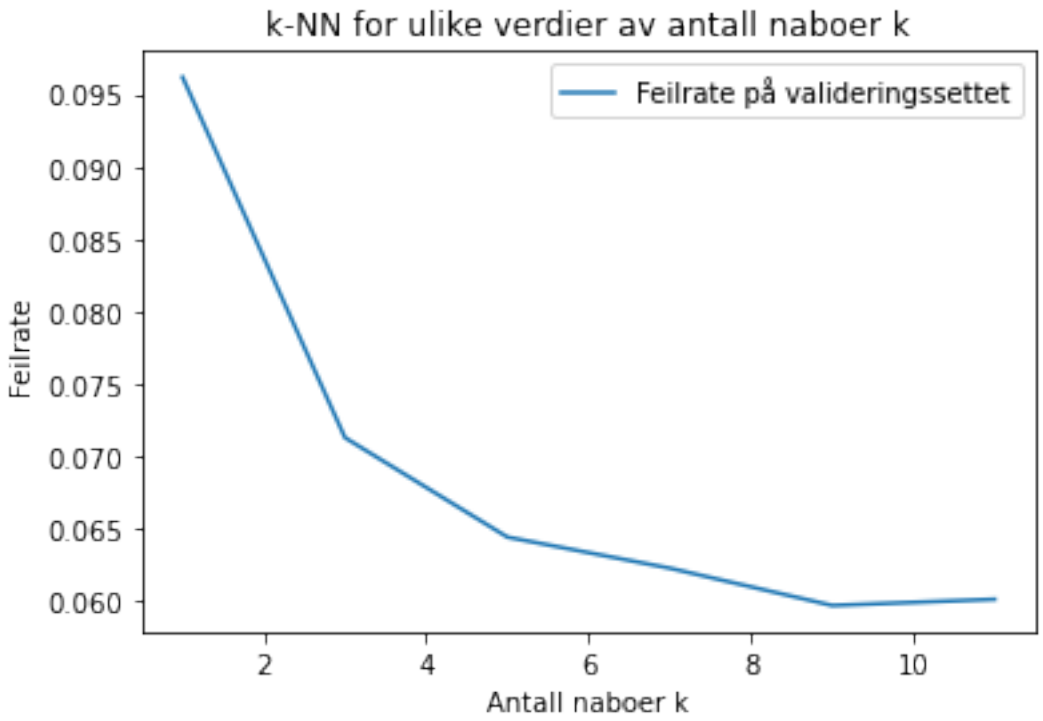
(ii) Sammenlign andelen fra (i) med andelen kjøpere i valideringssttet. Har suksesraten blitt bedre, sammenlignet med et tilfeldig utvalg av kunder fra valideringssettet? Kommentér.

3.5.4 Løsning:

(i) The right calculation is $18/(18+103) = 0.149$. No partial points, eithers it's right or wrong.

(ii) The proportion of buyers in the validation set is $139/(139+2190) = 0.0597$ (0.5P), thus by using our model we have considerably higher success rate among all the phone calls that are done (0.5P for understanding that the success rate is higher thanks to the model).

[15]:
```
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 euklidsk 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]]
```

3.5.5 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)?

3.5.6 Løsning:

(i) The lowest error rate is $k = 9$ (0.5P) and the confusion matrix is seen below (0.5P) (no points are given for the calculation of the error rate, because we specifically ask for the confusion matrix!).

(ii) No, it is not so smart to use this value, because almost everybody is predicted to be a non-buyer (thus the error rate is essentially the same as the proportion of buyers; 0.5P). Moreover, despite a success rate of 50% for the contacted customers (2 out of 4 of those that are predicted as "yes"), we will only reach 2 potential buyers (0.5P).

[16]:
```
# 2h (ii) Best: k=9
k=9
knn = KNeighborsClassifier(n_neighbors=k,p=2)
knn.fit(X_tren, df_tren['Purchase'])

# Feilrate på testsettet:
round(1-knn.score(X_val, df_val['Purchase']),4)
```

[16]: 0.0597

[17]:
```
# 2h (ii) 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']
)
```

[17]:

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

3.6 Oppgave 3: Klyngeanalyse (10 poeng)

[1]:
```
# 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
```

3.7 Les inn datasettet

[2]:
```
## 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_col = 0)

images.head()
```

[2]:
```
   V1  V2  V3  V4  V5  V6  V7  V8  V9  V10  …  V775  V776  V777  V778  V779  \
1   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
3   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
5   0   0   0   0   0   0   0   0   0    0  …     0     0     0     0     0

   V780  V781  V782  V783  V784
1     0     0     0     0     0
2     0     0     0     0     0
3     0     0     0     0     0
4     0     0     0     0     0
5     0     0     0     0     0

[5 rows x 784 columns]
```

[3]:
```
## Standardisering av pikselne

images = images/255
```

[4]:
```
# 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'>
```

[5]:
```
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()
```

3.7.1 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])

3.7.2 Løsning:

(i) 60000 (1P)

(ii) See code below, it looks like a "3" (1P; here we can deduct -0.5 if they choose frame number 51, referring to `features[50]`; in all other cases 0P).

[6]:
```
## 3a (ii)

plt.imshow(features[49], cmap = 'gray')

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

3.8 Klyngeanalyse med K-gjennomsnitt

[7]:
```
# kodechunk kmeans

# Steg 1: Antall klynger
antall_klynger = 10

# Steg 2: Initaliser 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_
```

3.8.1 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 forslåg 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!

3.8.2 Løsning:

(i) See code below, 1P for the right solution, -0.5P for every (small) error.

(ii) The most important thing to notice is that the 10 centroides apparently do not correspond to the 10 different digits 0-9 (0.5P). Some of them seem to be represented several times (two shapes of 0s), others are not represented at all. The clusters therefore do *not* seem very useful (0.5P for the "not useful" answer) when the aim is to classify a new picture of a digit.

(iii) Here the students will most likely choose to describe the "Elbow method" from the link above: One tries all numbers of clusters from K=1 to some reasonably large value, and then plots the within-cluster sum-of-squared errors against K. After som K, the error will not really further decrease and the plot looks like it has an "elbow". Choose K at the elbow. Some students might want to choose the "Silhouette" method or any other ideas that are perhaps also meaningful.

[8]:
```
# Kode for 3b (i)
features = np.array(sentroider)
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()
```

3.9 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.

3.9.1 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 gjeonnomsnittskobling (`method = 'average'`). Hvordan fungerer gjeonnomsnittskobling? Maks 2 setninger!

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

3.9.2 Løsning:

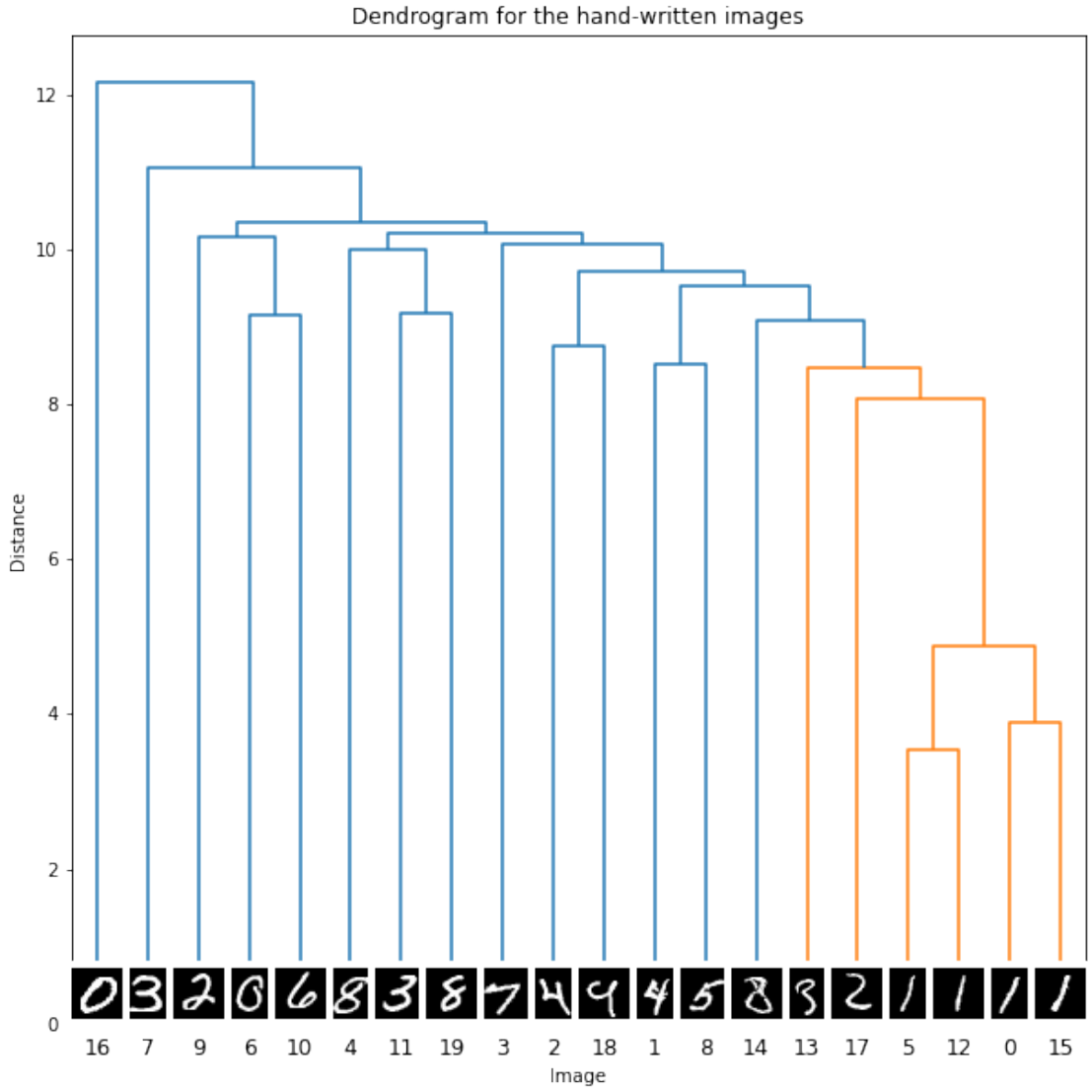
(i) Of course it depends a bit where one cuts the dendogram, and some of the same numbers are grouped together. However, in general the answer is clearly NO: the clustering does not seem very meaningful, almost irrespective of where one cuts. Note: Here the picture in the original question set looked different, but the answer is actually the same. The only thing that is half-way ok is the clustering of the four "1" pictures", but this is still not enough to get a reasonable clusterization.

(ii) In the average linkage (gjoennomsnittskobling) we use the average distance between any pair of points in two clusters to determine the cluster distance. (Here it is important to understand that each PAIR of points is used to calcualte the average).

(iii) See the code below. It looks like complete linkage (maybe?) works a bit better, whereas single or centroid linkage seem at least as bad as average linkage. But in essence, none of the linkages works. Here the students should just try at least one of the methods and run the code again, and then report that the results are still not satisfying. Deduct -0.5p for each wrong aspect in the (otherwise ok) reasoning.

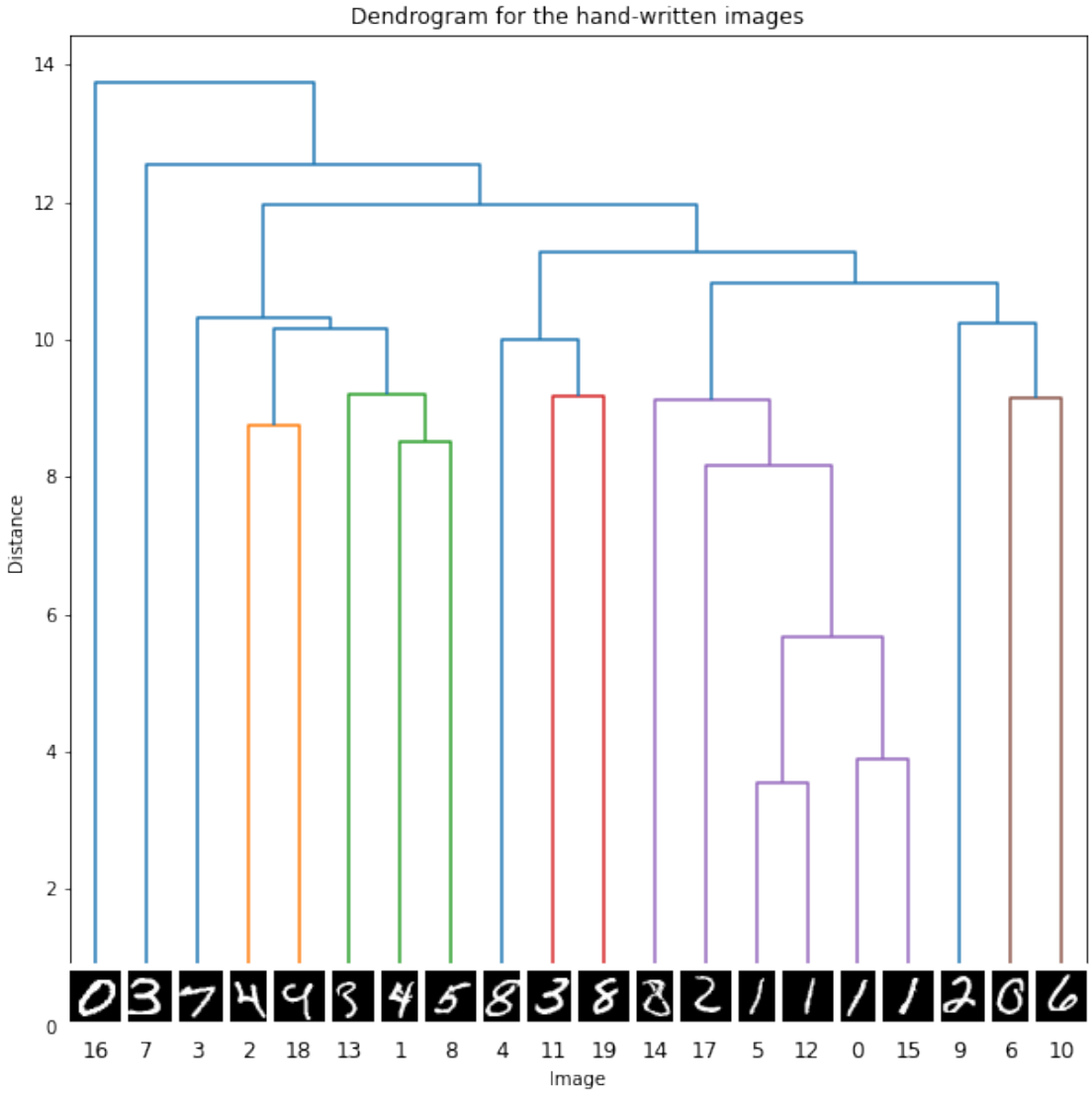
[9]:
```
##Cluster

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()
```



[10]:
```
# Code for 3c) (iii); Velg enten method='single', method='complete' eller␣
↪method='centroid'

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

# Bruk gjennomsnittskobling (men bytt method='...')
link = linkage(y = sample, method = 'complete', metric = 'euclidean')
#link = linkage(y = sample, method = 'single', metric = 'euclidean')
#link = linkage(y = sample, method = 'centroid', 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()
```



3.9.3 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 metode vil du anbefale når målet er å klassifisere (predikere) siffer fra et håndskrevet tall?

3.9.4 Løsning:

(i) Cluster analysis was not a very good method of the aim is to get good predictions.

(ii) For prediction we would need a classification method, like logistic regression (maybe KNN, although the dimensionality might be too high). Full point given for any classification method that is given, but we have to subtract -0.5P for each unsuitable method that is listed in addition.

[ ]: