Conversion_Analysis_Case_notebook_PDF

August 15, 2022

[ ]:
```
%config InlineBackend.figure_format='retina'
import pandas as pd
import numpy as np
import seaborn as sbn
import matplotlib.pyplot as plt
from scipy import stats as st

import statsmodels.api as sm

from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()

from statsmodels.stats.outliers_influence import variance_inflation_factor
```

[ ]:
```
# Det fulde dataset importeres som df_full
df_full = pd.read_csv("DataScientist-Case-Dataset.txt")
df_full.head()
```

[ ]:

	customer_id	converted	customer_segment	gender	age	related_customers	family_size	initial_fee_level	credit_account_id	branch
0	15001	0	13	male	22.0	1	0	14.5000	9b2d5b4678781e53038e91ea5324530a03f27dc1d0e5f6…	Helsinki
1	15002	1	11	female	38.0	1	0	142.5666	afa2dc179e46e8456ffff9016f91396e9c6adf1fe20d17…	Tampere
2	15003	1	13	female	26.0	0	0	15.8500		
3	15004	1	11	female	35.0	1	0	106.2000		
4	15005	0	13	male	35.0	0	0	16.1000		

```
2  9b2d5b4678781e53038e91ea5324530a03f27dc1d0e5f6…  Helsinki
3  abefcf257b5d2ff2816a68ec7c84ec8c11e0e0dc4f3425…  Helsinki
4  9b2d5b4678781e53038e91ea5324530a03f27dc1d0e5f6…  Helsinki
```

[ ]:
```
df_full.shape
```

[ ]:
```
(891, 10)
```

[ ]:
```
#customer_id og credit_account_id søjlerne er irrelevante for analysen

df = df_full.drop(['customer_id', 'credit_account_id'], axis=1)
df.head()
```

[ ]:
```
   converted  customer_segment  gender   age  related_customers  family_size  \
0          0                13    male  22.0                  1            0
1          1                11  female  38.0                  1            0
2          1                13  female  26.0                  0            0
3          1                11  female  35.0                  1            0
4          0                13    male  35.0                  0            0

   initial_fee_level    branch
0            14.5000  Helsinki
1           142.5666   Tampere
2            15.8500  Helsinki
3           106.2000  Helsinki
4            16.1000  Helsinki
```

Datasættet består nu af de kategoriske variable coverted, customer_segment, gender og branch. Lad os undersøge om der skal ryddes yderligere op i datasættet for analysen

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   converted          891 non-null    int64
 1   customer_segment   891 non-null    int64
 2   gender             891 non-null    object
 3   age                714 non-null    float64
 4   related_customers  891 non-null    int64
 5   family_size        891 non-null    int64
 6   initial_fee_level  891 non-null    float64
 7   branch             889 non-null    object
dtypes: float64(2), int64(4), object(2)
memory usage: 55.8+ KB
```

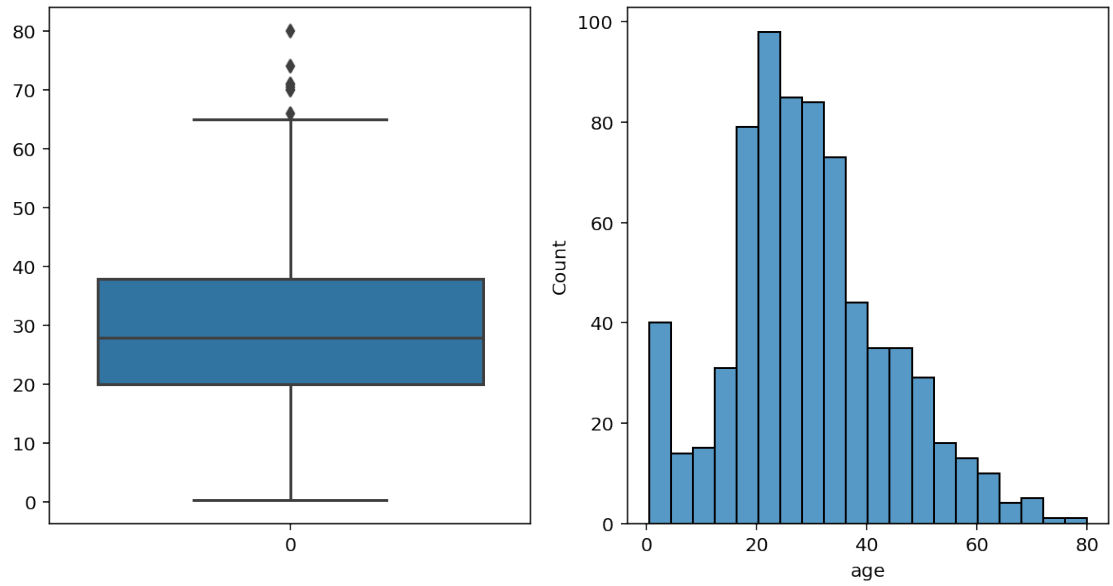
Det mangler værdier i både branch og age søjlerne. Siden der kun mangler 2 værdier for branch i 2 observation fjernes disse blot fra sættet. Fra age mangler der data i tæt ved 20% af observationerne. For ikke at formindske datasættet for meget, udfylder vi de manglende værdier. De mest almindelige måder at gøre det er ved at indsætte middelværdien eller medianen for søjlen. Hvilken metode der er bedst afhænger af fordelingen af dataen.

[ ]:
```
# Eliminer rækkerne hvor branch info mangler
df.dropna(subset=['branch'],inplace = True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 889 entries, 0 to 890
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   converted           889 non-null    int64
 1   customer_segment    889 non-null    int64
 2   gender              889 non-null    object
 3   age                 712 non-null    float64
 4   related_customers   889 non-null    int64
 5   family_size         889 non-null    int64
 6   initial_fee_level   889 non-null    float64
 7   branch              889 non-null    object
dtypes: float64(2), int64(4), object(2)
memory usage: 62.5+ KB
```

[ ]:
```
# Evaluer fordelingen af age
fig, axes = plt.subplots(1,2,figsize = (10,5))
sbn.boxplot(ax = axes[0],data=df.age)
p1 = sbn.histplot(df.age,ax = axes[1])
plt.savefig("age_distribution.pdf",bbox_inches = "tight")
p1
```

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



Vi kan se, at age data er let højreskæv, om end ikke voldsomt meget. Derfor er den mest oplagte taktik til at udfylde de manglende data at indsætte medianen.

[ ]:
```
df['age'].fillna(df['age'].median(skipna=True), inplace = True)
```

Vi fjerner også alle observationer, der er duplikater så præcision af modellen ikke overestimeres.

[ ]:
```
df.drop_duplicates(inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 773 entries, 0 to 890
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   converted          773 non-null    int64
 1   customer_segment   773 non-null    int64
 2   gender             773 non-null    object
 3   age                773 non-null    float64
 4   related_customers  773 non-null    int64
 5   family_size        773 non-null    int64
 6   initial_fee_level  773 non-null    float64
 7   branch             773 non-null    object
dtypes: float64(2), int64(4), object(2)
memory usage: 54.4+ KB
```

Vi har nu et "rent" datasæt at arbejde med. Lad os undesøge hvilke værdier datasættet indeholder

[ ]:
```
for col in df:
    print(df[col].unique())
```

```
[0 1]
[13 11 12]
['male' 'female']
[22.   38.   26.   35.   28.   54.    2.   27.   14.    4.   58.   20.
 39.   55.   31.   34.   15.    8.   19.   40.   66.   42.   21.   18.
  3.    7.   49.   29.   65.   28.5   5.   11.   45.   17.   32.   16.
 25.    0.83 30.   33.   23.   24.   46.   59.   71.   37.   47.   14.5
 70.5  32.5  12.    9.   36.5  51.   55.5  40.5  44.    1.   61.   56.
 50.   36.   45.5  20.5  62.   41.   52.   63.   23.5   0.92 43.   60.
 10.   64.   13.   48.    0.75 53.   57.   80.   70.   24.5   6.    0.67
 30.5   0.42 34.5  74.  ]
[1 0 3 4 2 5 8]
[0 1 2 5 3 4 6]
[  14.5     142.5666   15.85    106.2      16.1      16.9166  103.725
   42.15     22.2666   60.1416   33.4      53.1      62.55     15.7084
   32.       58.25     26.       36.       14.45     52.       16.0584
   71.       62.775   526.       15.7584   15.7916   55.4416  293.0416
   15.5      21.      164.3416  104.       14.4584   22.4834   18.95
   42.       83.1584   31.       43.3584   35.6      79.375    15.6
  153.4584  123.9584   55.5      93.8     166.95     55.8      30.4916
   16.3166   17.325   147.       28.9084  112.9916   15.3      58.
   24.95     18.       19.       15.575    94.2      31.7      68.75
  122.35     41.15     69.3084  126.7166   46.      154.575    17.3084
   15.55     48.3      19.65     28.9166  495.0416   14.2834   44.7166
   13.95     14.1      29.       30.0916   52.5666   18.4334  158.4
   13.5      23.       73.5      15.5916   25.05    133.2      14.625
  122.7584   15.4666  139.1      32.2      31.5      41.05    110.
   51.85     67.       61.3916   50.9334   57.425     0.       30.1
   78.       44.05    100.       16.8084   12.9916   20.925    37.575
   62.      226.55     54.      152.5834  180.       18.7      27.
   15.1      52.5      24.55     14.25    105.1084   40.425   173.
 1024.6584  159.3     306.925   271.2666   39.       59.4     155.9166
   40.5     157.7     182.1584   25.75     17.7     303.1      61.
   46.5      24.7     221.7666  217.8      48.      113.8584  166.3166
  524.75     28.      329.7334  269.       12.475   115.9584   57.
  267.3      31.8      18.45     70.      150.5     138.6     110.8834
  423.        8.025   455.05     31.4834   15.4584   24.      240.
   25.3      37.5      13.7166   65.       15.75     28.8     111.8
   16.225   163.7166   38.5166   39.9334  178.2084   77.       15.45
   27.5834   19.675    14.0916   15.0416   24.575    19.175    99.0084
  156.5334   30.2      15.2584   45.05     52.575   118.8      14.9916
   68.0416  187.      443.5584  212.85     99.      142.       27.725
   15.6584   79.2      34.8     102.9584   52.775    60.       80.25
   17.425    30.       66.       84.8      31.1     130.       64.6416
```

```
   14.1084   16.8666   51.175    19.6834   16.275    20.3416  422.675
  114.       26.8334   15.4834   18.9666   15.475    16.725    46.9
   51.8584   17.3666   17.0334   15.775    74.0084   12.9      13.9
   16.6      12.875    78.8      28.2166   27.7166  100.9916   10.
   19.6916   21.0334]
['Helsinki' 'Tampere' 'Turku' 'Turku}']
```

Vi ser med der samme hvad der må være en tastefejl i branch data, hvor en eller flere branches er angivet som 'Turku}'. Det retter vi til 'Turku'.

[ ]:
```
df.replace(['Turku}'],'Turku',inplace=True)
for col in df:
    print(df[col].unique())
```

```
[0 1]
[13 11 12]
['male' 'female']
[22.   38.   26.   35.   28.   54.    2.   27.   14.    4.   58.   20.
 39.   55.   31.   34.   15.    8.   19.   40.   66.   42.   21.   18.
  3.    7.   49.   29.   65.   28.5   5.   11.   45.   17.   32.   16.
 25.    0.83 30.   33.   23.   24.   46.   59.   71.   37.   47.   14.5
 70.5  32.5  12.    9.   36.5  51.   55.5  40.5  44.    1.   61.   56.
 50.   36.   45.5  20.5  62.   41.   52.   63.   23.5   0.92 43.   60.
 10.   64.   13.   48.    0.75 53.   57.   80.   70.   24.5   6.    0.67
 30.5   0.42 34.5  74.  ]
[1 0 3 4 2 5 8]
[0 1 2 5 3 4 6]
[  14.5     142.5666   15.85    106.2      16.1      16.9166  103.725
   42.15     22.2666   60.1416   33.4      53.1      62.55     15.7084
   32.       58.25     26.       36.       14.45     52.       16.0584
   71.       62.775   526.       15.7584   15.7916   55.4416  293.0416
   15.5      21.      164.3416  104.       14.4584   22.4834   18.95
   42.       83.1584   31.       43.3584   35.6      79.375    15.6
  153.4584  123.9584   55.5      93.8     166.95     55.8      30.4916
   16.3166   17.325   147.       28.9084  112.9916   15.3      58.
   24.95     18.       19.       15.575    94.2      31.7      68.75
  122.35     41.15     69.3084  126.7166   46.      154.575    17.3084
   15.55     48.3      19.65     28.9166  495.0416   14.2834   44.7166
   13.95     14.1      29.       30.0916   52.5666   18.4334  158.4
   13.5      23.       73.5      15.5916   25.05    133.2      14.625
  122.7584   15.4666  139.1      32.2      31.5      41.05    110.
   51.85     67.       61.3916   50.9334   57.425     0.       30.1
   78.       44.05    100.       16.8084   12.9916   20.925    37.575
   62.      226.55     54.      152.5834  180.       18.7      27.
   15.1      52.5      24.55     14.25    105.1084   40.425   173.
 1024.6584  159.3     306.925   271.2666   39.       59.4     155.9166
   40.5     157.7     182.1584   25.75     17.7     303.1      61.
   46.5      24.7     221.7666  217.8      48.      113.8584  166.3166
```

```
  524.75     28.      329.7334  269.       12.475   115.9584   57.
  267.3      31.8      18.45     70.      150.5     138.6     110.8834
  423.        8.025   455.05     31.4834   15.4584   24.      240.
   25.3      37.5      13.7166   65.       15.75     28.8     111.8
   16.225   163.7166   38.5166   39.9334  178.2084   77.       15.45
   27.5834   19.675    14.0916   15.0416   24.575    19.175    99.0084
  156.5334   30.2      15.2584   45.05     52.575   118.8      14.9916
   68.0416  187.      443.5584  212.85     99.      142.       27.725
   15.6584   79.2      34.8     102.9584   52.775    60.       80.25
   17.425    30.       66.       84.8      31.1     130.       64.6416
   14.1084   16.8666   51.175    19.6834   16.275    20.3416  422.675
  114.       26.8334   15.4834   18.9666   15.475    16.725    46.9
   51.8584   17.3666   17.0334   15.775    74.0084   12.9      13.9
   16.6      12.875    78.8      28.2166   27.7166  100.9916   10.
   19.6916   21.0334]
['Helsinki' 'Tampere' 'Turku']
```

Der lader ikke til at være flere problemer med manglende eller "forkert" data. For at lave log. regression skal de kategoriske variable erstattes med dummy-variable:

[ ]:
```
df_dummy = pd.get_dummies(df, columns=['gender', 'customer_segment','branch'],drop_first=True)
df_dummy
```

[ ]:
```
     converted   age  related_customers  family_size  initial_fee_level  \
0            0  22.0                  1            0            14.5000
1            1  38.0                  1            0           142.5666
2            1  26.0                  0            0            15.8500
3            1  35.0                  1            0           106.2000
4            0  35.0                  0            0            16.1000
..         …   …                  …          …                …
885          0  39.0                  0            5            58.2500
887          1  19.0                  0            0            60.0000
888          0  28.0                  1            2            46.9000
889          1  26.0                  0            0            60.0000
890          0  32.0                  0            0            15.5000

     gender_male  customer_segment_12  customer_segment_13  branch_Tampere  \
0              1                    0                    1               0
1              0                    0                    0               1
2              0                    0                    1               0
3              0                    0                    0               0
4              1                    0                    1               0
..           …                    …                    …             …
885            0                    0                    1               0
887            0                    0                    0               0
888            0                    0                    1               0
889            1                    0                    0               1
```

```
890            1                    0                    1                 0

     branch_Turku
0               0
1               0
2               0
3               0
4               0
..            …
885             1
887             0
888             0
889             0
890             1

[773 rows x 10 columns]
```

Vi har som afhængig variabel converted, og alle andre uafhængige:

[ ]:
```
x_cols = df_dummy.columns.to_list()[1:]
x = df_dummy[x_cols]
y = df_dummy['converted']

# For at lave regression skal vi tilføje et konstant led til de uafhængelige␣
 ↪variable
x_const = sm.add_constant(x, prepend=False)
```

Logistisk regression kræver at flere antagelser om data er opfyldt. Vi starter med at tjekke, at tjekke linearitet af log-odds ifht. de kontinuerte variable. Dette gøres med en Box-Tidwell transformation ved at tilføje et ikke-lineært term for hver af af de kontinuerte variable og undersøge om disse termer er signifikante i regressionen. Er de ikke det, er antagelse opfyldt.

[ ]:
```
cont_vars = ['age','related_customers','family_size','initial_fee_level']
df_2 = df.copy()
for var in cont_vars:
    df_2.drop(df_2[df_2[var]==0].index,inplace = True)
df_2_logt = df_2.copy()
for var in cont_vars:
    df_2_logt[f'{var}:Log'] = df_2_logt[var].apply(lambda x: x * np.log(x))
cols_keep = cont_vars + df_2_logt.columns.tolist()[-len(cont_vars):]
x_logt = df_2_logt[cols_keep]
y_logt = df_2_logt['converted']
x_logt_const = sm.add_constant(x_logt,prepend=False)

logit_results = sm.GLM(y_logt,x_logt_const, family=sm.families.Binomial()).fit()
print(logit_results.summary())
```

```
                 Generalized Linear Model Regression Results
```

```
==============================================================================
Dep. Variable:              converted   No. Observations:                  133
Model:                            GLM   Df Residuals:                      124
Model Family:                Binomial   Df Model:                            8
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -70.721
Date:                Mon, 15 Aug 2022   Deviance:                       141.44
Time:                        12:42:20   Pearson chi2:                     177.
No. Iterations:                     6   Pseudo R-squ. (CS):             0.2689
Covariance Type:            nonrobust
==============================================================================
=========
                             coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
---------
age                       -0.1949      0.129     -1.514      0.130      -0.447
0.057
related_customers          0.7811      2.439      0.320      0.749      -3.998
5.561
family_size                0.2668      1.652      0.161      0.872      -2.972
3.505
initial_fee_level          0.0493      0.035      1.421      0.155      -0.019
0.117
age:Log                    0.0366      0.031      1.176      0.239      -0.024
0.098
related_customers:Log     -1.0351      1.350     -0.767      0.443      -3.681
1.611
family_size:Log           -0.3175      0.891     -0.356      0.722      -2.064
1.429
initial_fee_level:Log     -0.0070      0.005     -1.284      0.199      -0.018
0.004
const                     -0.2207      3.149     -0.070      0.944      -6.392
5.951
==============================================================================
=========
```

Det ses på p-værdierne at ingen af de ikke lineære termer, dem med log, er signifikante på et 0.05-niveau. Dermed er antagelsen om linearitet opfyldt.

Den næste antagelse er, at der ikke nogen outliers med stærk indflydelse på modellen. Det tjekkes med Cook's D, der måler måler ændringen i modellen hvis den i'te observation fjernes. En observation vurderes som havende stærk inflydelse hvis $D > 4/n$, hvor n er antal observationer.

[ ]:
```
# Først udføres regressionen
logreg_model = sm.GLM(y, x_const, family=sm.families.Binomial())
logreg_result =logreg_model.fit()
# Inflydelsen "hentes"
```

```
infl = logreg_result.get_influence()
summ_df = infl.summary_frame()

# Cooks' D hentes
diag_df = summ_df.loc[:,['cooks_d']]

# and append the absolute values of the standardized residuals
# og absolutværdien af de standarddiseres residualer vedhæftes
diag_df['std_resid'] = st.zscore(logreg_result.resid_pearson)
diag_df['std_resid'] = diag_df.loc[:,'std_resid'].apply(lambda x: np.abs(x))

# Til sidst sorteres efter størrelse af Cook's D
diag_df.sort_values("cooks_d", ascending=False)
diag_df
```

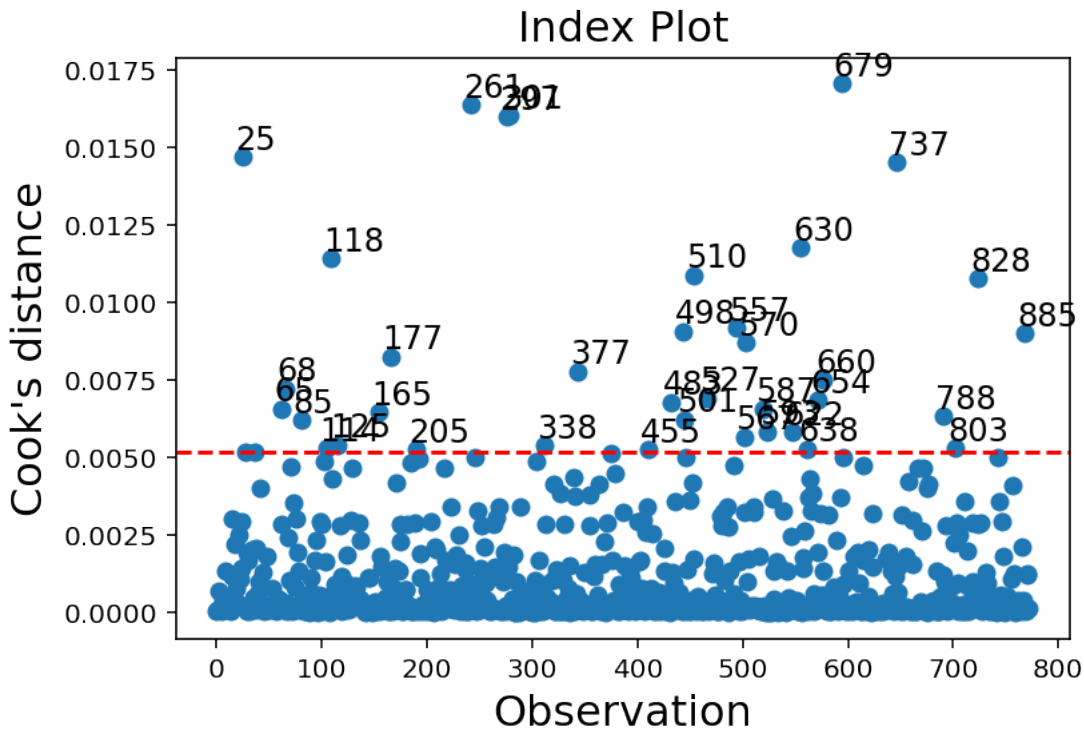
[ ]:

	cooks_d	std_resid
0	0.000040	0.360438
1	0.000079	0.296285
2	0.000686	0.761798
3	0.000115	0.353883
4	0.000031	0.330266
..	…	…
885	0.008996	0.850262
887	0.000034	0.217387
888	0.001237	0.988926
889	0.001203	0.763677
890	0.000142	0.360225

[773 rows x 2 columns]

[ ]:
```
# Grænsen for Cook's D sættes til n / 4
cook_thres = 4 / len(x)

# Cook's D for hver obs. plottes med grænsen angivet med den røde linje
fig = infl.plot_index(y_var="cooks", threshold=cook_thres)
plt.axhline(y=cook_thres, ls="--", color='red');
```



Det lader til at være en mindre del af datasættet der er stærk inflydelse. Vi kan finde andelen:

[ ]:
```
outl = diag_df[diag_df['cooks_d']>cook_thres]
prop_outl = round(100*(len(outl)/len(x)),1)
print(f"Andel af observationerne, der har stærk inflydeæse = {prop_outl}%")
```

```
Andel af observationerne, der har stærk inflydeæse = 4.8%
```

Vi skal nu bestemme, hvor stor en del af disse, der også er outliers dvs. er $> 3\ \sigma$.

[ ]:
```
extreme = diag_df[(diag_df['cooks_d'] > cook_thres) &
                  (diag_df['std_resid'] > 3)]
prop_extreme = round(100*(len(extreme) / len(x)),1)
print(f'Andel af observationer, der er stærkt inflydelsesrige outliers =␣
  ↪{prop_extreme}%')
```

```
Andel af observationer, der er stærkt inflydelsesrige outliers = 0.6%
```

Denne andel af outliers med stærk inflydelse er forventelig, og der er derfor umiddelbart ingen grund til at fjerne dem. Vi kan inspicere de specifikke observationer, for anomelier

[ ]:
```
df_dummy.iloc[list(extreme.index)]
```

[ ]:
```
     converted   age  related_customers  family_size  initial_fee_level  \
281          0  28.0                 0            0            15.7084
```

```
322          1  30.0                   0            0           24.7000
327          1  36.0                   0            0           26.0000
372          0  19.0                   0            0           16.1000
562          0  28.0                   0            0           27.0000

     gender_male  customer_segment_12  customer_segment_13  branch_Tampere  \
281            1                    0                    1               0
322            0                    1                    0               0
327            0                    1                    0               0
372            1                    0                    1               0
562            1                    1                    0               0

     branch_Turku
281             0
322             1
327             0
372             0
562             0
```

Vi kan se, at initial_fee_level er noget under middelværdien der er:

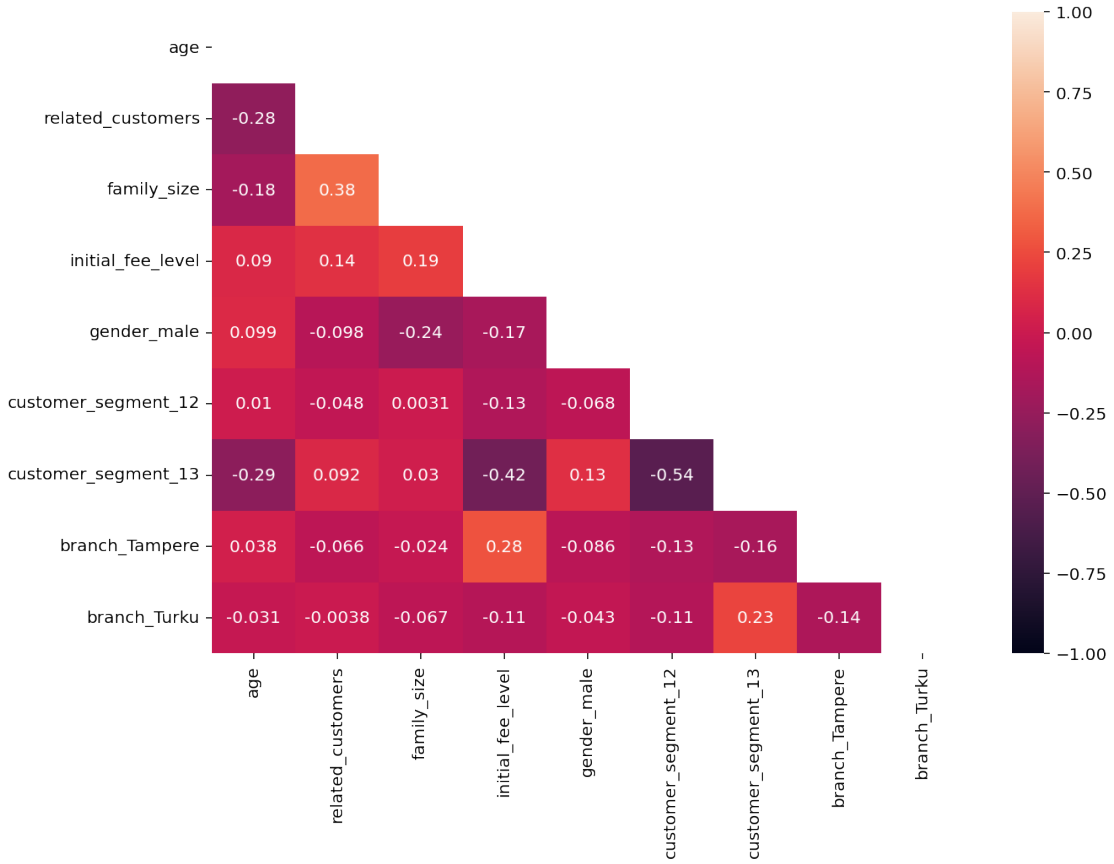
[ ]:
```
df_dummy['initial_fee_level'].mean()
```

[ ]:
```
69.52331772315652
```

vi har dog ingen grund til at tro, at dette er forkert og de beholdes derfor.

Som det næste tjekkes det at antagelsen om at der ikke multikollinearitet. Først ses der på korrelationsmatricen

[ ]:
```
plt.figure(figsize=(10,7))
mask = np.triu(np.ones_like(x.corr(), dtype=bool))
sbn.heatmap(x.corr('pearson'),annot=True,mask=mask,vmin = -1, vmax = 1);
```



Ved første øjekast, er der ingen bekymrende høje korrelationer. Alle pånær mellem customer_segment 12 og 13 er mindre end 0.5. Korreltioner fortæller dog ikke hele historien, og vi ser derfor også på "variance inflation factors" for alle variablene

[ ]:
```
vif_data = pd.DataFrame()
vif_data["feature"] = x.columns

vif_data["VIF"] = [variance_inflation_factor(x.values, i) for i in range(len(x.
  ↪columns))]

print(vif_data)
```

```
              feature       VIF
0                 age  3.658398
1   related_customers  1.575036
2         family_size  1.597341
3   initial_fee_level  1.987811
4         gender_male  2.753254
5  customer_segment_12  1.551195
6  customer_segment_13  2.725730
```

```
7        branch_Tampere  1.361887
8          branch_Turku  1.172358
```

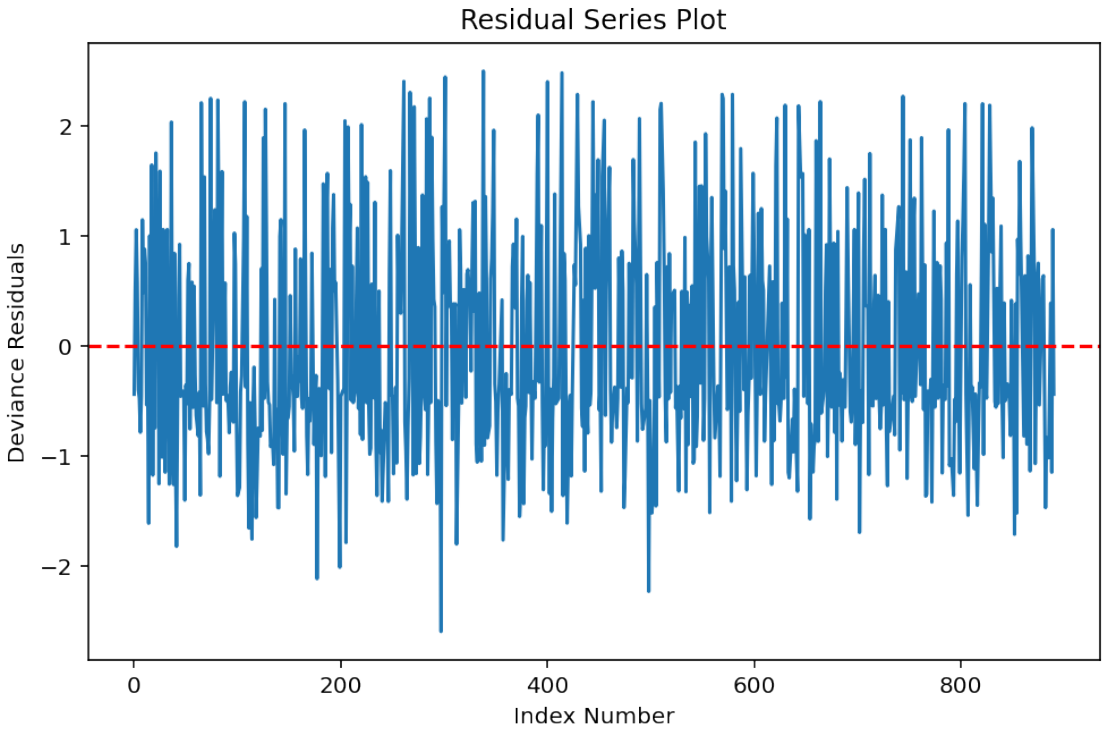
Da ingen af VIF-værdierne foroven er større end 5, konkluderes det at antagelsen om ingen multikollinearitet er opfyldt:

Slutteligt undersøges det at observationerne er uafh. For at gøre dette udføres log. regression for a få residualerne

[ ]:
```
# Generate regression model
log_model = sm.GLM(y,x_const, family=sm.families.Binomial())
log_res = log_model.fit()
log_res.summary()

# Generate residual series plot
fig = plt.figure(figsize=(8,5))
ax = fig.add_subplot(111, title="Residual Series Plot",
                     xlabel="Index Number", ylabel="Deviance Residuals")

ax.plot(x.index.tolist(), st.zscore(log_res.resid_deviance))
plt.axhline(y=0, ls="--", color='red');
```



Residualerne ligner hvid støj i tilstrækkelig grad til, at konludere at antagelsen om uafh. er opfyldt.

Vi er nu klar til, at udføre regression. Vi har grundmodellen:

```
[ ]: log_res.summary()
```

```
[ ]: <class 'statsmodels.iolib.summary.Summary'>
"""
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:                  773
Model:                            GLM   Df Residuals:                      763
Model Family:                Binomial   Df Model:                            9
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -360.61
Date:                Mon, 15 Aug 2022   Deviance:                       721.22
Time:                        12:42:21   Pearson chi2:                     774.
No. Iterations:                     5   Pseudo R-squ. (CS):             0.3441
Covariance Type:            nonrobust
==============================================================================
=======
                           coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
-------
age                     -0.0387      0.008     -4.918      0.000      -0.054
-0.023
related_customers       -0.3133      0.116     -2.694      0.007      -0.541
-0.085
family_size             -0.0932      0.119     -0.785      0.432      -0.326
0.140
initial_fee_level        0.0009      0.001      0.721      0.471      -0.001
0.003
gender_male             -2.5025      0.205    -12.188      0.000      -2.905
-2.100
customer_segment_12     -0.7986      0.302     -2.648      0.008      -1.390
-0.207
customer_segment_13     -2.0452      0.303     -6.742      0.000      -2.640
-1.451
branch_Tampere           0.4146      0.243      1.708      0.088      -0.061
0.890
branch_Turku             0.0717      0.374      0.192      0.848      -0.661
0.805
const                    3.5161      0.455      7.727      0.000       2.624
4.408
==============================================================================
=======
"""
```

Modellen reducers ved at fjerne variable, der har en p-værdi > 0.5, da de er insignifikante. Den største p-værdi er for branch_Turku, der derfor fjernes. Da denne er kædet sammen med

branch_Tampere, fjernes denne samtidig

[ ]:
```
x_reduced = x_const.drop(['branch_Tampere','branch_Turku'], axis=1)
log_model = sm.GLM(y,x_reduced, family=sm.families.Binomial())
log_res = log_model.fit()
log_res.summary()
```

[ ]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:                  773
Model:                            GLM   Df Residuals:                      765
Model Family:                Binomial   Df Model:                            7
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -362.06
Date:                Mon, 15 Aug 2022   Deviance:                       724.12
Time:                        12:42:21   Pearson chi2:                     780.
No. Iterations:                     5   Pseudo R-squ. (CS):             0.3416
Covariance Type:            nonrobust
==============================================================================
=======
                           coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
-------
age                     -0.0394      0.008     -5.026      0.000      -0.055
-0.024
related_customers       -0.3334      0.116     -2.884      0.004      -0.560
-0.107
family_size             -0.0974      0.117     -0.829      0.407      -0.328
0.133
initial_fee_level        0.0012      0.001      0.988      0.323      -0.001
0.003
gender_male             -2.5144      0.203    -12.363      0.000      -2.913
-2.116
customer_segment_12     -0.8844      0.298     -2.973      0.003      -1.468
-0.301
customer_segment_13     -2.0964      0.297     -7.060      0.000      -2.678
-1.514
const                    3.6712      0.446      8.224      0.000       2.796
4.546
==============================================================================
=======
"""
```

[ ]:
```
x_reduced.drop(['family_size'], axis=1, inplace=True)
log_model = sm.GLM(y,x_reduced, family=sm.families.Binomial())
log_res = log_model.fit()
log_res.summary()
```

[ ]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:                  773
Model:                            GLM   Df Residuals:                      766
Model Family:                Binomial   Df Model:                            6
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -362.41
Date:                Mon, 15 Aug 2022   Deviance:                       724.82
Time:                        12:42:21   Pearson chi2:                     786.
No. Iterations:                     5   Pseudo R-squ. (CS):             0.3410
Covariance Type:            nonrobust
==============================================================================
=======
                           coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
-------
age                     -0.0393      0.008     -5.014      0.000      -0.055
-0.024
related_customers       -0.3585      0.112     -3.202      0.001      -0.578
-0.139
initial_fee_level        0.0010      0.001      0.847      0.397      -0.001
0.003
gender_male             -2.4823      0.199    -12.472      0.000      -2.872
-2.092
customer_segment_12     -0.9095      0.295     -3.078      0.002      -1.489
-0.330
customer_segment_13     -2.1288      0.294     -7.250      0.000      -2.704
-1.553
const                    3.6522      0.444      8.218      0.000       2.781
4.523
==============================================================================
=======
"""
```

[ ]:
```
x_reduced.drop(['initial_fee_level'], axis=1, inplace=True)
log_model = sm.GLM(y,x_reduced, family=sm.families.Binomial())
log_res = log_model.fit()
log_res.summary()
```

[ ]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:                  773
Model:                            GLM   Df Residuals:                      767
Model Family:                Binomial   Df Model:                            5
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -362.79
Date:                Mon, 15 Aug 2022   Deviance:                       725.57
Time:                        12:42:21   Pearson chi2:                     785.
No. Iterations:                     5   Pseudo R-squ. (CS):             0.3404
Covariance Type:            nonrobust
==============================================================================
=======
                           coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
-------
age                     -0.0398      0.008     -5.099      0.000      -0.055
-0.025
related_customers       -0.3430      0.110     -3.114      0.002      -0.559
-0.127
gender_male             -2.4961      0.198    -12.587      0.000      -2.885
-2.107
customer_segment_12     -1.0213      0.266     -3.845      0.000      -1.542
-0.501
customer_segment_13     -2.2632      0.249     -9.090      0.000      -2.751
-1.775
const                    3.8232      0.398      9.596      0.000       3.042
4.604
==============================================================================
=======
"""
```

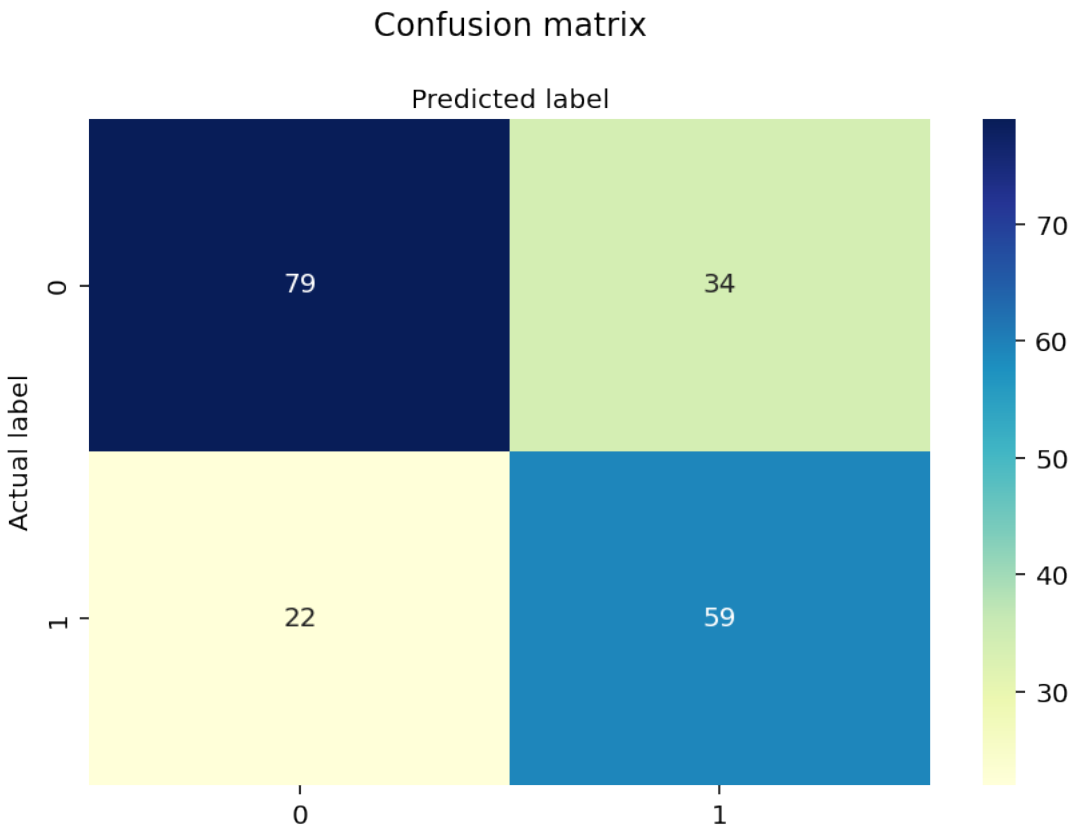
Modellen indeholder nu kun signifikante variable. Et første estimat på kvaliteten af modellen er værdien for Pseudo R-squ foroven. Her benyttes Cox-Snell formlen, hvor en tommelfingerregel er at en god model har en værdi på 0.2-0.4 eller derover. Dette er dog stærkt situationsafhængigt. Derfor vil vi i stedet forsøge at benytte modellen til at forudsige hvilke customers der er converted baseret på de uafh. variable. For dette splitter vi datasættet op og benytter 75% til at træne modellen og de sidste 25% til at teste.

[ ]:
```
# Træning af modellen og forudsigelse på test data
x_train, x_test, y_train, y_test = train_test_split(x_reduced,y,test_size=0.25,␣
 ↪random_state=1)
x_train = scale.fit_transform(x_train)
x_test = scale.transform(x_test)
```

```
log_model_train = sm.GLM(y_train,x_train, family=sm.families.Binomial())
log_res_train = log_model_train.fit()
y_hat = log_res_train.predict(x_test)
y_pred = list(map(round, y_hat))
```

Vi kan få en ide om præcisionen af modellen ud fra en "confusion matrix", der har korrekt placerede ikke konverterede og konverterede i hhv. øverste venstre og nederste højre celle

[ ]:
```
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sbn.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label');
plt.savefig("confusion_matrix.pdf",bbox_inches='tight')
```



Vi kan få yderligere metrikker på præcisionen af modellen

```
[ ]: target_names = ['Not converted', 'Converted']
     print(metrics.classification_report(y_test, y_pred, target_names=target_names))
```

```
                precision    recall  f1-score   support

Not converted       0.78      0.70      0.74       113
    Converted       0.63      0.73      0.68        81

     accuracy                           0.71       194
    macro avg       0.71      0.71      0.71       194
 weighted avg       0.72      0.71      0.71       194
```

Det ses at modellen forudsiger ikke-konverteret korrekt i 78% af tilfælde og konverteret i 63% af tilfældene. Da klasserne er rimeligt balanceret, kan vi også se på accuracy, der er på 0.71. Det er ganske godt og modellen accepteres.

Som et sidste skridt ses på odds ratio for at forså hvorledes de uafhængige variable påvirker sandsynligheden for at en customer er converted.

```
[ ]: params = log_res.params
     conf = log_res.conf_int()
     conf['Odds Ratio'] = params
     conf.columns = ['5%', '95%','Odds Ratio']
     np.exp(conf)
```

```
[ ]:                            5%        95%  Odds Ratio
     age                  0.946341   0.975774    0.960945
     related_customers    0.571794   0.880639    0.709608
     gender_male          0.055869   0.121554    0.082408
     customer_segment_12  0.213974   0.606056    0.360112
     customer_segment_13  0.063856   0.169453    0.104022
     const               20.953058  99.895247   45.750529
```

Det ses at højere alder og flere related_customers begge medfører en mindre sandsynlighed for at en customer er converted. Det ses yderligere, at det er meget mindre sandsynligt at en mand er konverteret end en kvinde og ligeledes for customer_segment 12 og 13 ifht 11.

Det kan konkluderes at de vigtigste parametre for at forudsige om en customer er converted er: age, related_customers, gender og customer_segment, og at hvis cornverted er målet, bør man fokusere på yngre kvinder i customer_segment 11 med få eller ingen related_customers.