Predictive Modeling

Series 3

Exercise 3.1

We consider a data set originating from a medical application which is contained in the data file `catheter.csv` . The data frame consists of measurements of the `height` in centimeter and of the `weight` in kilogram of patients. The response variable `catlength` refers to the optimal length of catheters used for the examination of the heart. In this exercise, we intend to predict the optimal catheter length on the basis of the available data set `catheter`.

a) Fit a simple linear regression model for both `catlength` $\sim$ `height` and `catlength` $\sim$ `weight`. Are the predictors significant?

b) Fit a multiple linear regression model `catlength` $\sim$ `height` + `weight`. Is there any influence of the predictors on the response variable ? Is it significant?

c) Test the null hypotheses $H_0 : \beta_1 = 0$ and $H_0 : \beta_2 = 0$. Compare the results with those from the two simple linear regressions. Comment and explain the differences if there are any.

d) For a child that is 120 cm tall and has a weight of 25 kg, compute the 95 % prediction interval

- by means of the multiple regression model,
- by means of the simple regression models (2x).

In practice, a prediction error of $\pm$ 2 cm would be acceptable. Do the data and the models allow for a prediction of `catlength` that is sufficiently precise? Does it make sense to use both predictors? **Python** Hints:

[ ]:
```
""" Hints: """
# two options for defining the factor x0 for prediction:
# 1: x0 = [[1, x0_1, x0_2]]
x0 = [[1, 120, 25]]
# 2: using add_constant(x0, has_constant='add')
x0 = [[120, 25]]
x0 = sm.add_constant(x0, has_constant='add')
```

Exercise 3.2

In this exercise, we would like to analyze to which extent savings differ between countries. The data set `savings.csv` contains data for 50 countries. For each country the values are averaged over the entire population and over the time period 1960 - 1970. The variables have the following meanings:

- `sr` : proportion of the available income that is saved
- `pop15` : proportion of the population that is younger than 15 years
- `pop75` : proportion of the population that is older than 75 years
- `dpi` : per capita income
- `ddpi` : growth rate of `dpi`

a) Fit the model `sr` $\sim$ `pop15` + `pop75` + `dpi` + `ddpi`. Carry out a residual analysis.

b) Identify the three countries having the largest leverage and describe how they differ from the remaining data points. `Python`-Hints:

  - You can use `numpy.argsort(`*`leverage`*`)` to find the sorted indices, which can than be projected on the list of countries.
  - You can use `seaborn.pairplot()` to create the pairwise scatter plots. The easiest way to visualize the three countries, is to add a new feature to our Dataframe, which is filled either with 'rest' or with the name of one of the three countries.

[1]:
```python
""" Hints: """
import seaborn as sns

# Add a new feature, which indicates the last three countries
last3_feat = ["rest" for x in range(savings.shape[0])]
for i in range(3):
    last3_feat[index[-3 + i]] = last3.iloc[i, 0]

# Append this new feature to the DataFrame
savings['last3'] = last3_feat

# Pairplot using sns.pairplot()
sns.pairplot(savings, hue="last3", diag_kind="hist")

# Show plot
plt.show()
```

c) Remove the data point (country) with the largest Cook's distance from the analysis. To what extent do the results change?

d) Now consider the following variable transformations:

- Log-transform `sr`
- Log-transform the predictor variables `dpi` and `ddpi`, but not the response variable `sr`
- Log-transform `sr` and the predictor variables `dpi` and `ddpi`

Fit the corresponding models and analyze the residuals. Finally, decide which model is most appropriate.

Exercise 3.3

The data set `synthetisch.csv` contains the response Y and the predictors X_1 and X_2. Fit a multiple linear regression model. We can assume that the errors ϵ_i are independent and that the majority of the observations are distributed according to $\mathcal{N}(0, \sigma^2)$.

Carry out a residual analysis and generate a 3D plot of the regression hyperplane. Does the 3D plot confirm your conclusion from the residual analysis?

Python–Hints: For creating the 3D plot, you can make use of the below template.

```
[ ]: """ Hints: """
     import matplotlib.pyplot as plt
     from matplotlib import cm
     from mpl_toolkits.mplot3d import Axes3D
     ''' %matplotlib will open all plots in a new window,
     where you can scroll, zoom, rotate, and more. '''
     %matplotlib

     ''' Create data: '''
     n = 20 # points in grid

     # x1 and x2, evenly spaced allong grid
     x_1 = np.linspace( ? , ? , n)
     x_2 = np.linspace( ? , ? , n)

     # predicted values for y at x1 and x2
     y_pred = np.zeros((n, n))
     for i in range(len(x_1)):
         for j in range(len(x_2)):
             # fill y_pred
     # create grid
     x_1, x_2 = np.meshgrid(x_1, x_2)

     # Create Figure
     fig = plt.figure(figsize=(8, 8))
     ax = fig.gca(projection='3d')

     # Plot the surface.
     surf = ax.plot_surface(x_1, x_2, y_pred, cmap=cm.coolwarm)
     # Set Labels
     ax.set_xlabel("x_1"), ax.set_ylabel("x_2"), ax.set_zlabel("y")
     ax.set_title("predicted and true y")

     # 3D scatterplot
     ax.scatter(x['x1'], x['x2'], y, alpha=0.8, linewidth=2)

     plt.tight_layout()
     plt.show()
```

The plot shows up in an interactive window.

Exercise 3.4

A mechanical engineer is investigating the surface finish of metal parts produced on a lathe and its relationship to the speed (in revolutions per minute) of the lathe. 20 measurements of the quality (`Q`), the speed (`S`) and the used cutting tool (`CTT`) are found in the file `lathe.csv`. (Source: Montgomery and Runger, *Applied Statistics and Probability for Engineers*).

a) Generate a scatter plot of `Q` versus `S`. `Python`-Hints:

- You may find it useful later on, to have the data sorted on 'S'. Furthermore, we will add dummy variables converting the 'CTT' type to a categorical feature, being 1 or 0, depending on the type.

[ ]:
```
import pandas as pd
import numpy as np

# Load data
lathe = pd.read_csv('./data/lathe.csv')

# Sort data
lathe.sort_values(by='S', inplace=True)

# Add a column where CTT is mapped to a 1/0 integer
dummies = pd.get_dummies(lathe['CTT'])
```

```
lathe = lathe.join(dummies)
```

- Now you have added this dummy-variables, you can use these columns for the coloring in matplotlib.pyplot:

```
[ ]: # Plot data
     ax.scatter(lathe['S'], lathe['Q'], c=lathe['DM302'])
```

b) Fit a regression model for which the response variable **`Q`** depends linearly on the speed **`S`** of the lathe. The slope should be the same for both types of cutting tools, but the intercept should depend on the tool used. Write the linear regression model using a dummy variable for the cutting tool. Add the regression line to the plot in a).

c) Fit the following model:

```
[ ]: # Define x and y:
     x = pd.DataFrame({
         'S': lathe['S'],
         'DM416': lathe['DM416'],
         'S * DM416': lathe['S'] * lathe['DM416']})
     y = lathe['Q']
```

What is the name of this model? Write the linear regression model using a dummy variable for the cutting tool.

d) Let us assume that the slope depends as well on the cutting tool used. Would a different slope be plausible according to the collected data? Answer the question with a hypothesis test at the 5 % level of significance. What is the probability of a type II error?

Exercise 3.5

The Australian Bureau of Agricultural and Resource Economics conducts an annual survey of the agroindustry. In 1991, 451 farms in New South Wales took part in this survey. The raw data is contained in the file **`farm.csv`**. The variables have the following meanings:

- **`revenue`** : response variable, total revenue of the farm
- **`costs`** : predictor variable, total costs of the farm
- **`region`** : predictor variable, code for different regions within New South Wales

- `industry` : predictor variable, code for cultivation (1 = wheat, 2=wheat, sheep, cattle, 3=sheep, 4=cattle, 5=sheep, cattle).

The aim is to fit a suitable regression model that predicts the `revenue` of a farm. You will need to perform the following steps:

a) Preprocess the data in such a way that factor variables are correctly stored in the data frame.

b) Fit the complete regression model and check the residuals. Does the model fit? If not, carry out a log-transformation of all non-factor variables. Does the model fit now? Continue the analysis with the log-transformed variables.

c) For a cattle farm in `region` 111 with `costs` of 100 000, what is the predicted `revenue`?

d) Test whether `region` has a significant influence on revenue when the other predictors are given.

[ ]:
```
""" Hints:
The easiest solution uses statsmodels.formula.api,
together with sm.stats.anova_lm"""
import statsmodels.formula.api as smf

"""Define model in Formula form,
np.log() is the log,
C() defines Categorical variables. """
model_f = smf.ols(formula='np.log(revenue) ~ \
                  np.log(costs) + C(region) + C(industry)',
                  data=farm).fit()

# Table and print results using anova_lm
table_f = sm.stats.anova_lm(model_f, typ=2)
print(table_f)
```

e) Add an interaction term between `region` and `industry`.

- How many parameters are estimated in total?
- Is the interaction term significant? Carry out a suitable test.
- What is the intuitive meaning of the interaction term, how do we have to interpret it in the context of the model? What is the conclusion?

f) We now have several models of varying complexity. First, there is the model with all predictors and the interaction term between `region` and `industry`. Then, there is the model with the main effects followed by the one without the variable `region`. Last, the least complex model is the one that does not distinguish between `region` and `industry`. Which model is best suited for predicting the `revenue` of a farm?

Result Checker

E 3.2: b) Libya, United States, Japan

E 3.5: c) 154 914

Predictive Modeling

Solutions to Series 3

Solution 3.1

a) In both cases, the predictor variable in the simple linear model is highly significant: **Python** code:

[1]:
```
import pandas as pd

# Load data
catheter = pd.read_csv('./data/catheter.csv')
```

[2]:
```
import statsmodels.api as sm

# Define x and y:
x_1 = catheter['height']
x_2 = catheter['weight']

y = catheter['catlength']
x_1_sm = sm.add_constant(x_1)
x_2_sm = sm.add_constant(x_2)

# Fit the linear models
model_1 = sm.OLS(y, x_1_sm).fit()
model_2 = sm.OLS(y, x_2_sm).fit()

# print summaries
print(model_1.summary(), '\n\n\n', model_2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              catlength   R-squared:                       0.776
Model:                            OLS   Adj. R-squared:                  0.754
Method:                 Least Squares   F-statistic:                     34.73
Date:                Tue, 09 Mar 2021   Prob (F-statistic):           0.000152
Time:                        11:28:23   Log-Likelihood:                -32.594
No. Observations:                  12   AIC:                             69.19
Df Residuals:                      10   BIC:                             70.16
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         12.1271      4.247      2.855      0.017       2.664      21.590
height         0.2377      0.040      5.893      0.000       0.148       0.328
==============================================================================
Omnibus:                        0.408   Durbin-Watson:                   1.790
Prob(Omnibus):                  0.815   Jarque-Bera (JB):                0.073
Skew:                          -0.165   Prob(JB):                        0.964
Kurtosis:                       2.806   Cond. No.                         386.
==============================================================================
                            OLS Regression Results
==============================================================================
Dep. Variable:              catlength   R-squared:                       0.799
Model:                            OLS   Adj. R-squared:                  0.779
Method:                 Least Squares   F-statistic:                     39.86
Date:                Tue, 09 Mar 2021   Prob (F-statistic):           8.75e-05
Time:                        11:28:23   Log-Likelihood:                -31.943
No. Observations:                  12   AIC:                             67.89
Df Residuals:                      10   BIC:                             68.86
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         25.6263      2.003     12.796      0.000      21.164      30.088
weight         0.6161      0.098      6.313      0.000       0.399       0.834
==============================================================================
Omnibus:                        2.618   Durbin-Watson:                   1.854
Prob(Omnibus):                  0.270   Jarque-Bera (JB):                0.500
Skew:                          -0.279   Prob(JB):                        0.779
Kurtosis:                       3.830   Cond. No.                         37.6
==============================================================================
```

b) **Python** code:

[3]:
```python
# Define x and y:
x = catheter[['height', 'weight']]
x_sm = sm.add_constant(x)

# Fit the linear model
model = sm.OLS(y, x_sm).fit()

# print summary
print(model.summary())
```

```
                           OLS Regression Results
==============================================================================
Dep. Variable:              catlength   R-squared:                       0.806
Model:                            OLS   Adj. R-squared:                  0.762
Method:                 Least Squares   F-statistic:                     18.65
Date:                Tue, 09 Mar 2021   Prob (F-statistic):           0.000630
Time:                        11:28:23   Log-Likelihood:                -31.757
No. Observations:                  12   AIC:                             69.51
Df Residuals:                       9   BIC:                             70.97
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         21.0853      8.770      2.404      0.040       1.245      40.925
height         0.0768      0.144      0.533      0.607      -0.249       0.403
weight         0.4275      0.368      1.161      0.275      -0.405       1.260
==============================================================================
Omnibus:                        1.049   Durbin-Watson:                   1.814
Prob(Omnibus):                  0.592   Jarque-Bera (JB):                0.029
```

```
Skew:                          -0.017   Prob(JB):                      0.986
Kurtosis:                       3.239   Cond. No.                       826.
==============================================================================
```

Yes, there is an influence of the predictor variables on the response variable. This influence is measured by means of the global F-test. The corresponding p-value is smaller than 0.01 so that the null hypothesis is rejected at the 1 % level. At least one of the predictors is relevant.

c) As we can see from the summary output (see above), both null hypotheses are retained, i.e. the predictors are not significant. Is this a contradiction to the results obtained in the two simple linear regression models? The answer is no. In multiple linear regression, on the basis of the hypotheses tests we decide whether the predictor variable `height` is required when the value of the predictor `weight` is known. The answer is no and the same holds vice versa. On the other hand, the global F-test indicates that we need at least one of the two predictors. So we do not need to include both predictors simultaneously but we need one of them. This situation occurs when the predictors are strongly correlated. Due to the smaller p-value we would prefer to include the predictor `weight` in the regression model.

d) **Python** code:

[4]:
```python
import numpy as np

# Prediction at point 120 cm, 25 kg
x0 = [[120, 25]]
x0 = sm.add_constant(x0, has_constant='add')

pred = model.get_prediction(x0)
pred_1 = model_1.get_prediction(x0[0][[0, 1]])
pred_2 = model_2.get_prediction(x0[0][[0, 2]])

pred_x0 = []
for prediction in [pred, pred_1, pred_2]:
    # Prediction summary frame
    predict_i = prediction.summary_frame(alpha=0.05)
    # Select mean and prediction interval
    predict_i = predict_i.loc\
        [0, ['mean', 'obs_ci_lower', 'obs_ci_upper']]
    # Append
    pred_x0.append(np.round(predict_i, 4))

pred_x0 = pd.DataFrame(data=pred_x0,
                       index=['full', 'height', 'weight'])

print(pred_x0)
```

```
           mean  obs_ci_lower  obs_ci_upper
full     40.9907       31.5399       50.4415
height   40.6561       31.2089       50.1033
weight   41.0295       32.0616        49.997
```

The prediction intervals differ slightly. We note that the prediction interval obtained on the basis of the multiple regression model is the largest among the three prediction intervals, contrary to what we might have expected. For, the multiple linear regression model includes most information. However, the multiple model requires the estimation of one additional parameter on the basis of the 12 available data points. This is associated with a larger estimation error of each single parameter.

In general, the prediction accuracy increases by including an additional parameter. But in this case the increase of the estimation error has a stronger effect, here a negative one. This is due to the fact that the two predictors are strongly correlated - adding the second predictor when the first one is already present does hardly yield additional information.

In practice, a prediction error of ± 2 cm would be acceptable. Thus, the data and the fitted models do not allow for a prediction of `catlength` that is sufficiently precise.

Solution 3.2 For this solution, we will use a slightly adapted version of the set of definitions used in Session 2:

[1]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.graphics.gofplots import ProbPlot
import random
import pandas as pd
import numpy as np


""" Plot Residuals vs Fitted Values """
def plot_residuals(axes, res, yfit, n_samp=0):
    """ Inputs:
    axes: axes created with matplotlib.pyplot
    x: x values
    ytrue: y values
    yfit: fitted/predicted y values
    res[optional]: Residuals, used for resampling
    n_samp[optional]: number of resamples """
    # For every random resampling
    for i in range(n_samp):
        # 1. resample indices from Residuals
        samp_res_id = random.sample(list(res), len(res))
        # 2. Average of Residuals, smoothed using LOWESS
        sns.regplot(x=yfit, y=samp_res_id,
                    scatter=False, ci=False, lowess=True,
                    line_kws={'color': 'lightgrey', 'lw': 1, 'alpha': 0.8})
        # 3. Repeat again for n_samples

    dataframe = pd.concat([yfit, res], axis=1)
    axes = sns.residplot(x=yfit, y=res, data=dataframe,
                    lowess=True, scatter_kws={'alpha': 0.5},
                    line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})
    axes.set_title('Residuals vs Fitted')
    axes.set_ylabel('Residuals')
    axes.set_xlabel('Fitted Values')


""" QQ Plot standardized residuals """
def plot_QQ(axes, res_standard, n_samp=0):
    """ Inputs:
    axes: axes created with matplotlib.pyplot
    res_standard: standardized residuals
    n_samp[optional]: number of resamples """
    # QQ plot instance
    QQ = ProbPlot(res_standard)
    # Split the QQ instance in the seperate data
    qqx = pd.Series(sorted(QQ.theoretical_quantiles), name="x")
    qqy = pd.Series(QQ.sorted_data, name="y")
    if n_samp != 0:
        # Estimate the mean and standard deviation
        mu = np.mean(qqy)
        sigma = np.std(qqy)
        # For ever random resampling
        for lp in range(n_samp):
            # Resample indices
            samp_res_id = np.random.normal(mu, sigma, len(qqx))
            # Plot
            sns.regplot(x=qqx, y=sorted(samp_res_id),
                        scatter=False, ci=False, lowess=True,
                        line_kws={'color': 'lightgrey', 'lw': 1, 'alpha': 0.8})

    sns.regplot(x=qqx, y=qqy, scatter=True, lowess=False, ci=False,
                scatter_kws={'s': 40, 'alpha': 0.5},
                line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})
    axes.plot(qqx, qqx, '--k', alpha=0.5)
    axes.set_title('Normal Q-Q')
    axes.set_xlabel('Theoretical Quantiles')
    axes.set_ylabel('Standardized Residuals')


""" Scale-Location Plot """
def plot_scale_loc(axes, yfit, res_stand_sqrt, n_samp=0):
    """ Inputs:
    axes: axes created with matplotlib.pyplot
    yfit: fitted/predicted y values
    res_stand_sqrt: Absolute square root Residuals
    n_samp[optional]: number of resamples """
    # For every random resampling
    for i in range(n_samp):
        # 1. resample indices from sqrt Residuals
        samp_res_id = random.sample(list(res_stand_sqrt), len(res_stand_sqrt))
        # 2. Average of Residuals, smoothed using LOWESS
        sns.regplot(x=yfit, y=samp_res_id,
                    scatter=False, ci=False, lowess=True,
                    line_kws={'color': 'lightgrey', 'lw': 1, 'alpha': 0.8})
        # 3. Repeat again for n_samples

    # plot Regression usung Seaborn
    sns.regplot(x=yfit, y=res_stand_sqrt,
                scatter=True, ci=False, lowess=True,
                scatter_kws={'s': 40, 'alpha': 0.5},
                line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})
    axes.set_title('Scale-Location plot')
    axes.set_xlabel('Fitted Sales values')
    axes.set_ylabel('$\sqrt{\|Standardized\ Residuals\|}$')


""" Cook's distance """
def plot_cooks(axes, res_inf_leverage, res_standard, n_pred=1,
               x_lim=None, y_lim=None, n_levels=4):
    """ Inputs:
    axes: axes created with matplotlib.pyplot
    res_inf_leverage: Leverage
    res_standard: standardized residuals
    n_pred: number of predictor variables in x
    x_lim, y_lim[optional]: axis limits
    n_levels: number of levels"""
    sns.regplot(x=res_inf_leverage, y=res_standard,
                scatter=True, ci=False, lowess=True,
                scatter_kws={'s': 40, 'alpha': 0.5},
                line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})
    # Set limits
    if x_lim != None:
        x_min, x_max = x_lim[0], x_lim[1]
    else:
        x_min, x_max = min(res_inf_leverage), max(res_inf_leverage)
    if y_lim != None:
        y_min, y_max = y_lim[0], y_lim[1]
    else:
        y_min, y_max = min(res_standard), max(res_standard)

    # Plot centre line
    plt.plot((x_min, x_max), (0, 0), 'g--', alpha=0.8)
    # Plot contour lines for Cook's Distance levels
    n = 100
    cooks_distance = np.zeros((n, n))
    x_cooks = np.linspace(x_min, x_max, n)
    y_cooks = np.linspace(y_min, y_max, n)

    for xi in range(n):
        for yi in range(n):
            cooks_distance[yi][xi] = \
            y_cooks[yi]**2 * x_cooks[xi] / (1 - x_cooks[xi]) / (n_pred + 1)
    CS = axes.contour(x_cooks, y_cooks, cooks_distance, levels=n_levels, alpha=0.6)

    axes.clabel(CS, inline=0,  fontsize=10)
    axes.set_xlim(x_min, x_max)
    axes.set_ylim(y_min, y_max)
    axes.set_title('Residuals vs Leverage and Cook\'s distance')
    axes.set_xlabel('Leverage')
    axes.set_ylabel('Standardized Residuals')


""" Standard scatter plot and regression line """
def plot_reg(axes, x, y, model, x_lab="x", y_lab="y", title="Linear Regression"):
    """ Inputs:
    axes: axes created with matplotlib.pyplot
    x: (single) Feature
    y: Result
    model: fitted linear sm model  """
    # Plot scatter data
    sns.regplot(x=x, y=y,
                scatter=True, ci=False, lowess=False,
                scatter_kws={'s': 40, 'alpha': 0.5},
                line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})
    # Set labels:
    axes.set_xlabel(x_lab)
    axes.set_ylabel(y_lab)
    axes.set_title(title)
```

Import and inspect the data:

[1]:
```
import pandas as pd

# Load data
savings = pd.read_csv('./data/savings.csv')

# As a first inspection, print the first rows of the data:
print(savings.head()) #, '\n\n', clocks.describe())
# As well as the dimensions of the set:
print('\nSize of savings =\n', savings.shape)
```

```
  Unnamed: 0     sr  pop15  pop75      dpi  ddpi
0  Australia  11.43  29.35   2.87  2329.68  2.87
1    Austria  12.07  23.32   4.41  1507.99  3.93
2    Belgium  13.17  23.80   4.43  2108.47  3.82
3    Bolivia   5.75  41.89   1.67   189.13  0.22
4     Brazil  12.88  42.19   0.83   728.47  4.56

Size of savings =
 (50, 6)
```

a) **Python** code:

[2]:
```
import statsmodels.api as sm
import numpy as np

# Define x and y:
x = savings[['pop15', 'pop75', 'dpi', 'ddpi']]
y = savings['sr']

x_sm = sm.add_constant(x)

# Fit the linear model
model = sm.OLS(y, x_sm).fit()

# Find the predicted values for the original design.
yfit = model.fittedvalues
# Find the Residuals
res = model.resid
# Influence of the Residuals
res_inf = model.get_influence()
# Studentized residuals using variance from OLS
res_standard = res_inf.resid_studentized_internal
# Absolute square root Residuals:
res_stand_sqrt = np.sqrt(np.abs(res_standard))
```

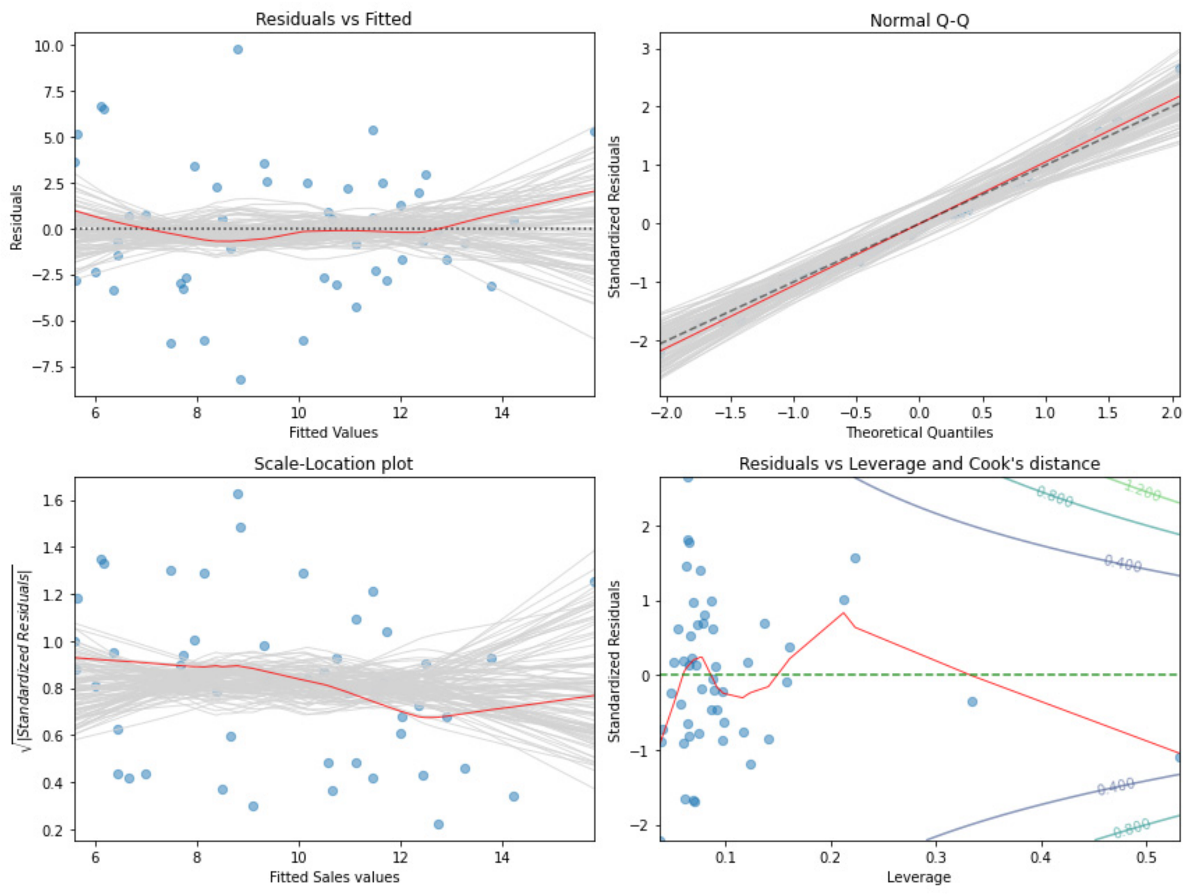
```
# Cook's Distance and leverage:
res_inf_cooks = res_inf.cooks_distance
res_inf_leverage = res_inf.hat_matrix_diag
```

[3]:
```
from MLR_def import *

""" Plots """
# Create Figure and subplots
fig = plt.figure(figsize = (12,9))

# First subplot: Residuals vs Fitted values with 100 resamples
ax1 = fig.add_subplot(2, 2, 1)
plot_residuals(ax1, res, yfit, n_samp = 100)

# Second subplot: QQ Plot with 100 resamples
ax2 = fig.add_subplot(2, 2, 2)
plot_QQ(ax2, res_standard, n_samp = 100)

# Third subplot: Scale-location with 100 resamples
ax3 = fig.add_subplot(2, 2, 3)
plot_scale_loc(ax3, yfit, res_stand_sqrt, n_samp = 100)

# Fourth subplot: Residuals vs Fitted values with 100 resamples
ax4 = fig.add_subplot(2, 2, 4)
plot_cooks(ax4, res_inf_leverage, res_standard, n_pred = 4)


# Show plot
plt.tight_layout()
plt.show()
```

The residual plots don't indicate any serious violation of the model assumptions, hence the model seems appropriate. There are a few points with large leverage but none of these points is influential since Cook's distance is smaller than 0.5 for all points.

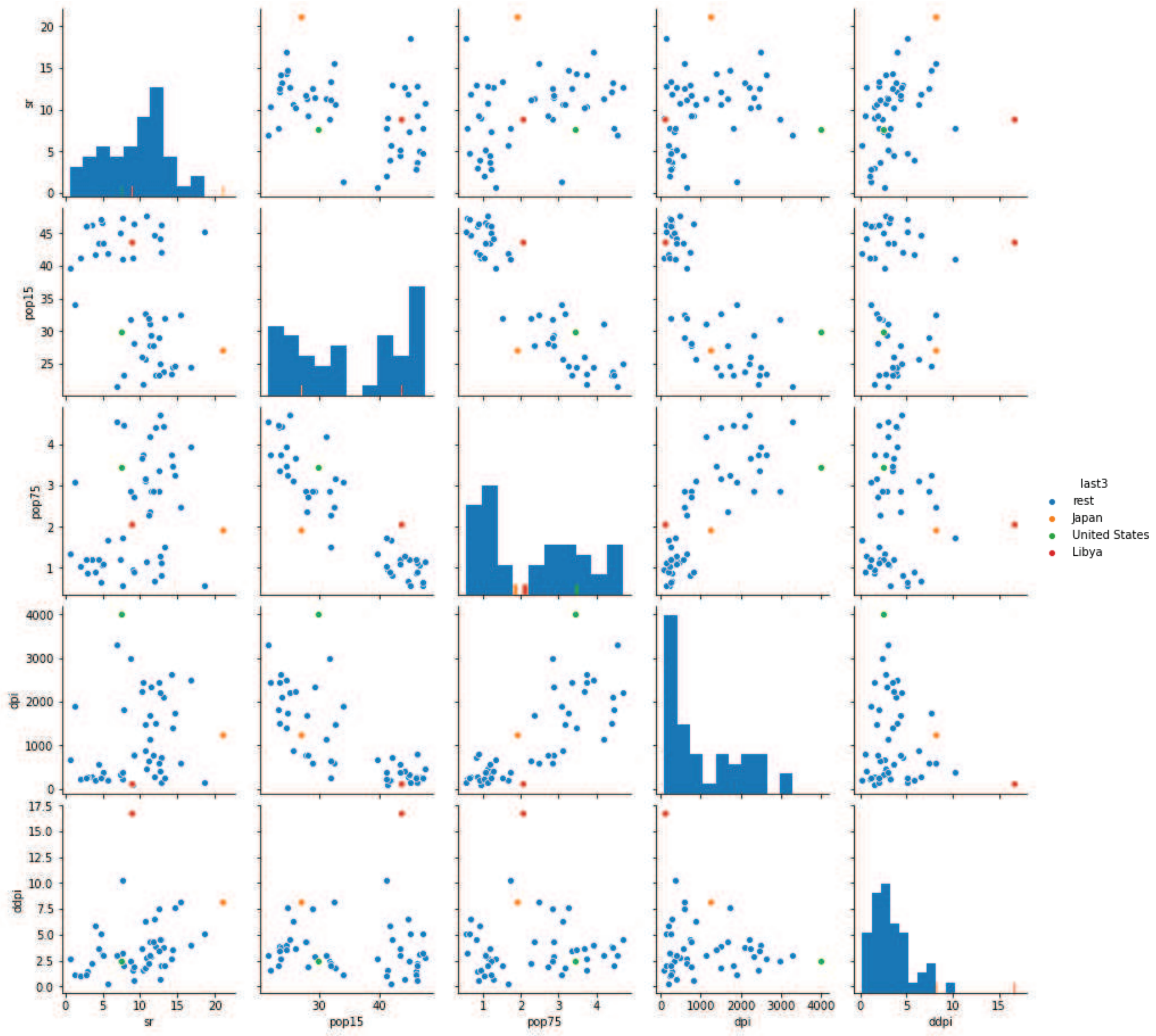
b) **Python** code:

[4]:
```
# Find sorted indices
index = np.argsort(res_inf_leverage)
# Select last three countries and leverages:
last3 = {'Country': savings.iloc[index, 0].iloc[-3:],
         'Leverage': res_inf_leverage[index][-3:]}

last3 = pd.DataFrame(data=last3)

print(last3)
```



```
           Country  Leverage
22           Japan  0.223310
43   United States  0.333688
48           Libya  0.531457
```

[5]:
```
""" Hints: """
import seaborn as sns

# Add a new feature, which indicates the last three countries
last3_feat = ["rest" for x in range(savings.shape[0])]
for i in range(3):
    last3_feat[index[-3 + i]] = last3.iloc[i, 0]

# Append this new feature to the DataFrame
savings['last3'] = last3_feat

# Pairplot using sns.pairplot()
sns.pairplot(savings, hue="last3", diag_kind="hist")

# Show plot
plt.show()
```



The three countries with the largest leverage are Libya, the USA, and Japan. The simplest way to see why these points have conspicuous predictor configurations is to plot pairwise scatter plots.

Python code:

```
[1]:
```

Libya has a very low value of `dpi`, but a very large value of `ddpi`. The USA have the largest `dpi` value and a relatively large value of `pop75`. Japan, on the other hand, lies at the border in several scatter plots but is not extraordinary with respect to a single feature.

c) **Python** code:

[6]:
```python
# We know the index of Libya from the DataFrame last3:
savings_red = savings.drop(last3.index[-1])

# Define x and y:
x = savings_red[['pop15', 'pop75', 'dpi', 'ddpi']]
y = savings_red['sr']

x_sm = sm.add_constant(x)

# Fit the linear model
model_2 = sm.OLS(y, x_sm).fit()

# Print summaries:
print(model.summary(), '\n\n\n', model_2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                     sr   R-squared:                       0.338
Model:                            OLS   Adj. R-squared:                  0.280
Method:                 Least Squares   F-statistic:                     5.756
Date:                Tue, 09 Mar 2021   Prob (F-statistic):           0.000790
Time:                        15:03:48   Log-Likelihood:                -135.10
No. Observations:                  50   AIC:                             280.2
Df Residuals:                      45   BIC:                             289.8
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         28.5661      7.355      3.884      0.000      13.753      43.379
pop15         -0.4612      0.145     -3.189      0.003      -0.753      -0.170
pop75         -1.6915      1.084     -1.561      0.126      -3.874       0.491
dpi           -0.0003      0.001     -0.362      0.719      -0.002       0.002
ddpi           0.4097      0.196      2.088      0.042       0.015       0.805
==============================================================================
Omnibus:                        0.866   Durbin-Watson:                   1.934
Prob(Omnibus):                  0.649   Jarque-Bera (JB):                0.493
Skew:                           0.241   Prob(JB):                        0.782
Kurtosis:                       3.064   Cond. No.                     2.04e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly␣
↪specified.
[2] The condition number is large, 2.04e+04. This might indicate that there are
strong multicollinearity or other numerical problems.


                            OLS Regression Results
==============================================================================
Dep. Variable:                     sr   R-squared:                       0.355
Model:                            OLS   Adj. R-squared:                  0.297
Method:                 Least Squares   F-statistic:                     6.065
Date:                Tue, 09 Mar 2021   Prob (F-statistic):           0.000562
Time:                        15:03:48   Log-Likelihood:                -132.24
No. Observations:                  49   AIC:                             274.5
Df Residuals:                      44   BIC:                             283.9
Df Model:                           4
Covariance Type:            nonrobust
```

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         24.5240      8.224      2.982      0.005       7.950      41.098
pop15         -0.3914      0.158     -2.479      0.017      -0.710      -0.073
pop75         -1.2809      1.145     -1.118      0.269      -3.589       1.027
dpi           -0.0003      0.001     -0.343      0.733      -0.002       0.002
ddpi           0.6103      0.269      2.271      0.028       0.069       1.152
==============================================================================
Omnibus:                        0.449   Durbin-Watson:                   1.994
Prob(Omnibus):                  0.799   Jarque-Bera (JB):                0.254
Skew:                           0.176   Prob(JB):                        0.881
Kurtosis:                       2.968   Cond. No.                     2.28e+04
==============================================================================
```

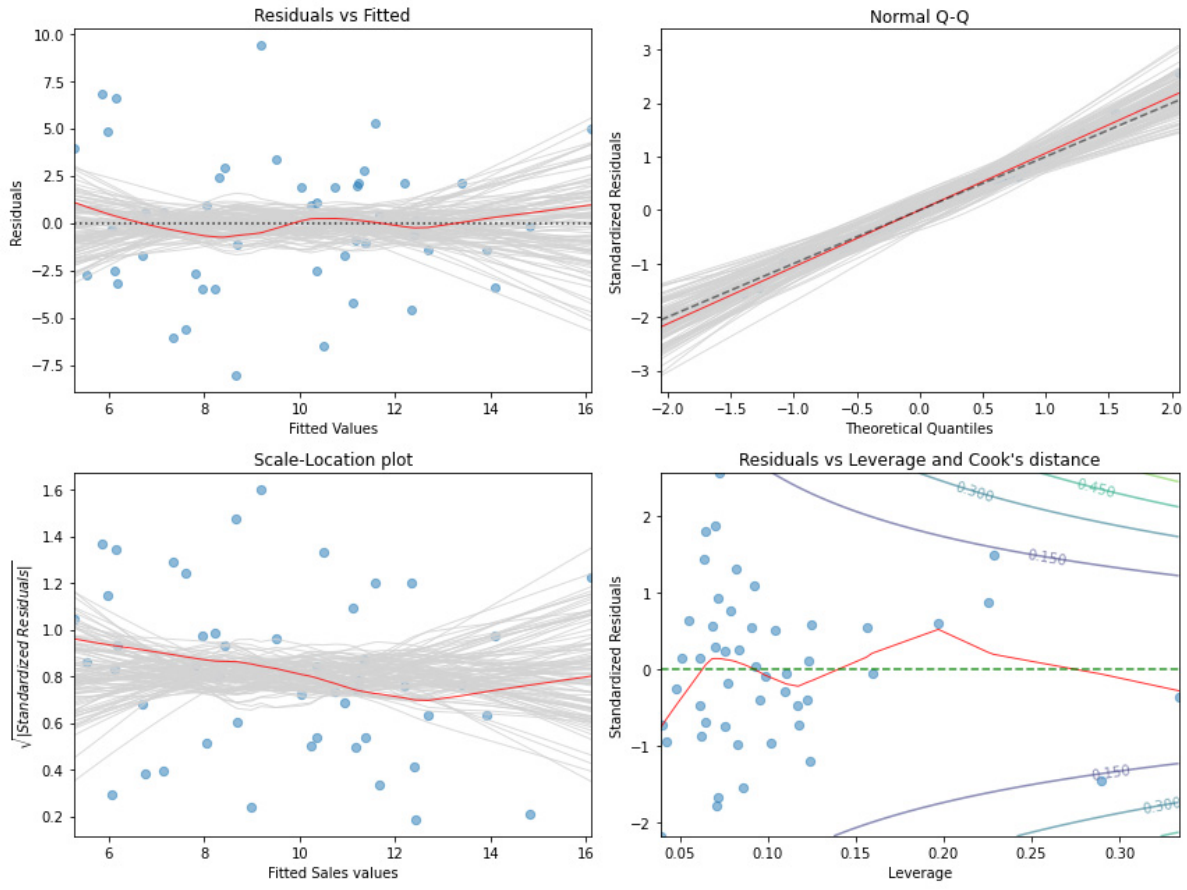


Figure 1: Plots are created exactly the same as in a).

The results only change slightly. The coefficients have similar magnitudes and the same predictors are significant. Also the residual analysis does not yield entirely new insights. This is not surprising as we have seen that Libya does not have a large influence, even though its leverage is high.

d) A transformation may be appropriate for the response variable `sr` as well as for `dpi` and `ddpi`. We shall experiment with three different models. In the first one we transform the response but not the predictors. In the second one we transform both predictors but not the response. In the third model, we transform both the response and the predictors.

[9]:
```python
# Define x and y:
x = savings[['pop15', 'pop75', 'dpi', 'ddpi']]
y = np.log(savings['sr'])
```

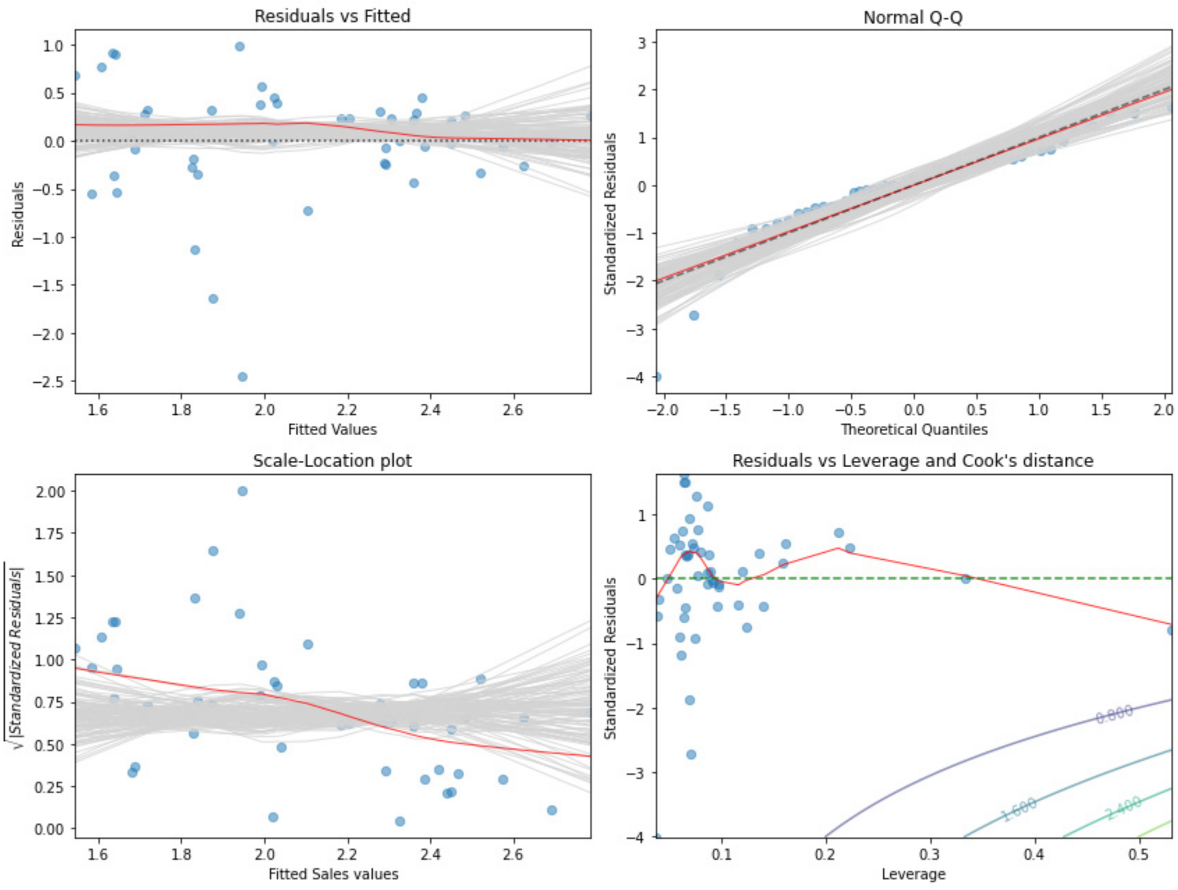


Figure 2: Log transformed sr. Plots are created exactly the same as in a).

[11]:
```python
# Define x and y:
x_log = np.log(savings[['dpi', 'ddpi']])
x = savings[['pop15', 'pop75']]
x = pd.concat((x_log, x), axis=1)
y = savings['sr']
```

[13]:
```python
# Define x and y:
x_log = np.log(savings[['dpi', 'ddpi']])
x = savings[['pop15', 'pop75']]
x = pd.concat((x_log, x), axis=1)
y = np.log(savings['sr'])
```

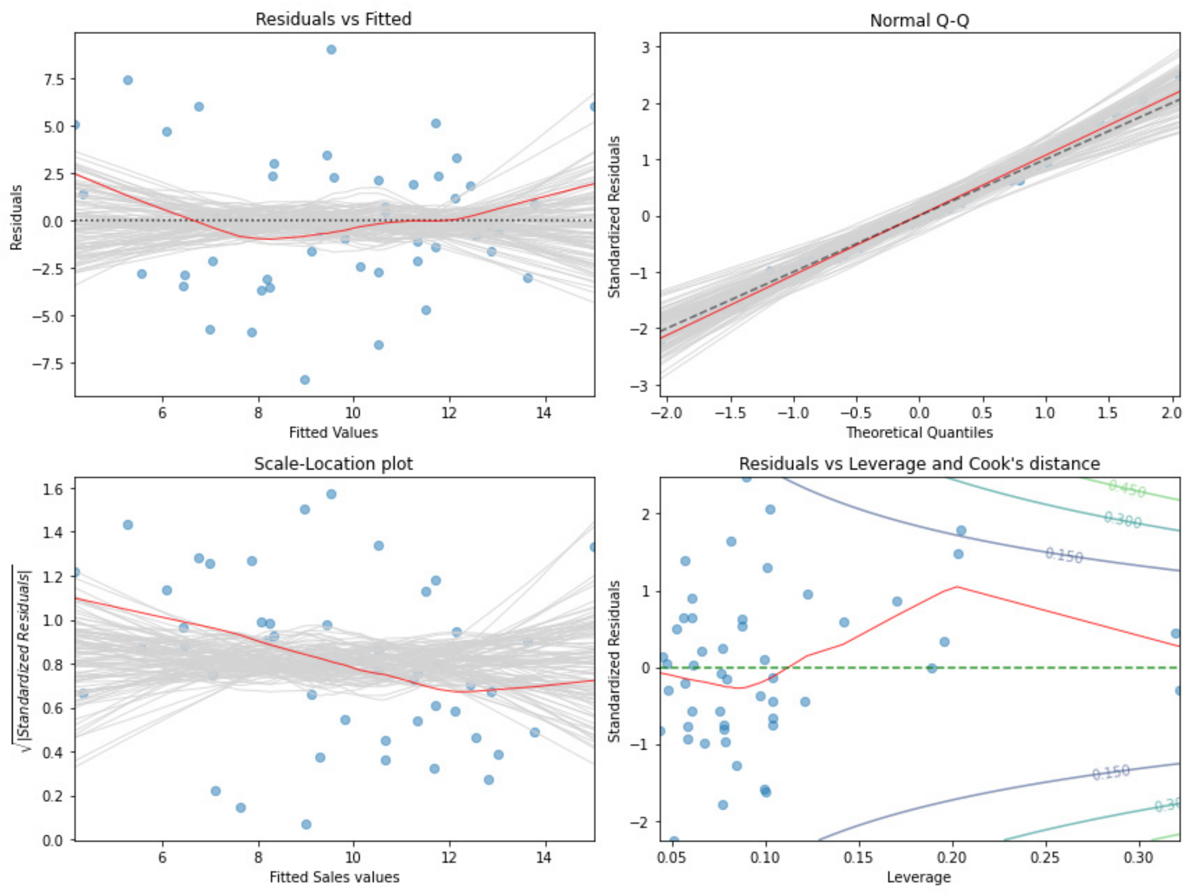


Figure 3: og-transform the predictor variables **dpi** and **ddpi**, but not the response variable **sr**. Plots are created exactly the same as in a).

The residual plots do not show any improvement with respect to the first model. Therefore, we will use the original model without transformations.

Solution 3.3 Python code:

[1]:
```
import pandas as pd

# Load data
synthetic = pd.read_csv('./data/synthetisch.csv')

# As a first inspection, print the first rows of the data:
print(synthetic.head()) #, '\n\n', clocks.describe())
# As well as the dimensions of the set:
print('\nSize of synthetic =\n', synthetic.shape)
```

```
   Unnamed: 0      y     x1    x2
0           1  33.50  19.19 -3.42
1           2  27.29  17.57 -4.52
2           3  22.60  18.57 -6.82
```

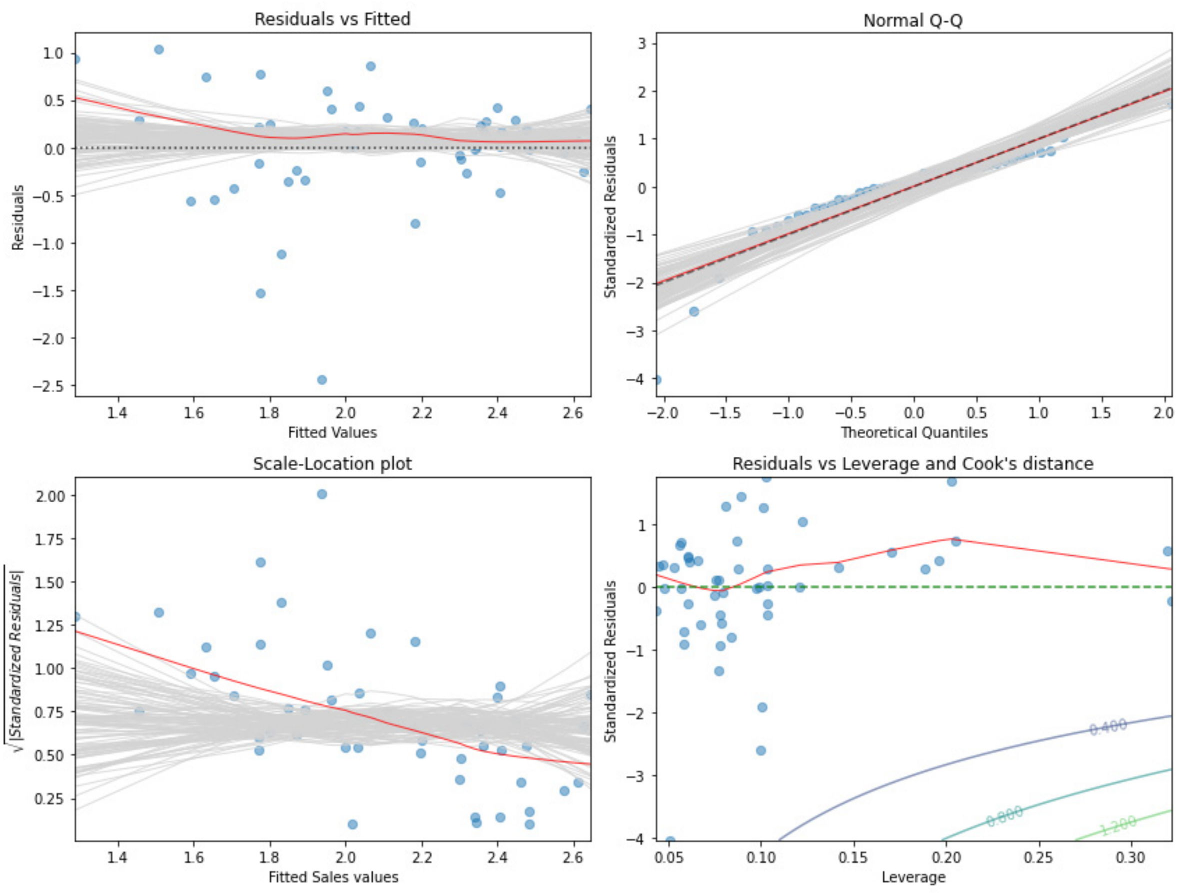


Figure 4: Log-transform sr and the predictor variables dpi and ddpi. Plots are created exactly the same as in a).

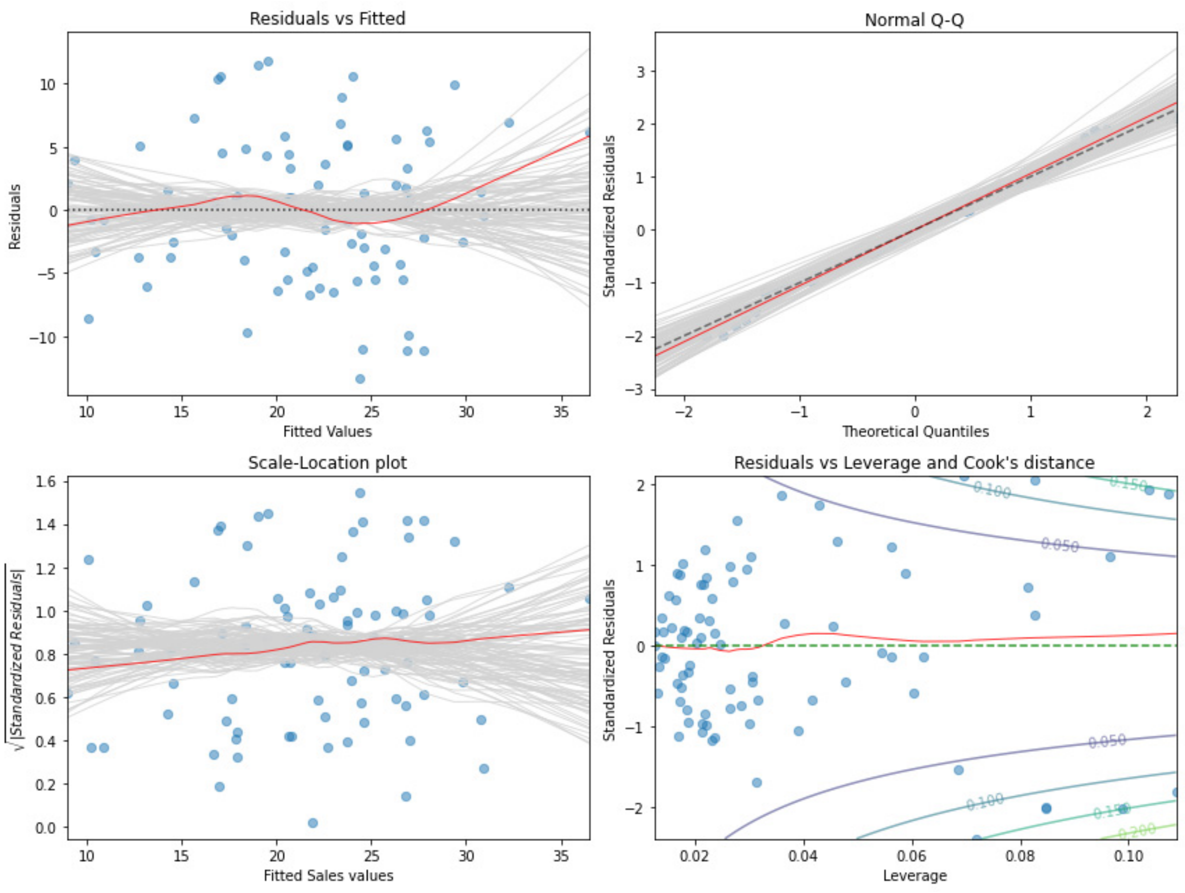
```
3            4  13.39  22.06 -12.33
4            5  20.71  18.09  -7.09

Size of synthetic =
 (83, 4)
```

[2]:
```
import statsmodels.api as sm
import numpy as np

# Define x and y:
x = synthetic[['x1', 'x2']]
y = synthetic['y']

x_sm = sm.add_constant(x)

# Fit the linear model
model = sm.OLS(y, x_sm).fit()
```

```
# Find the predicted values for the original design.
yfit = model.fittedvalues
# Find the Residuals
res = model.resid
# Influence of the Residuals
res_inf = model.get_influence()
# Studentized residuals using variance from OLS
res_standard = res_inf.resid_studentized_internal
# Absolute square root Residuals:
res_stand_sqrt = np.sqrt(np.abs(res_standard))
# Cook's Distance and leverage:
res_inf_cooks = res_inf.cooks_distance
res_inf_leverage = res_inf.hat_matrix_diag
```

[3]:
```
from MLR_def import *

""" Plots """
# Create Figure and subplots
fig = plt.figure(figsize = (12,9))

# First subplot: Residuals vs Fitted values with 100 resamples
ax1 = fig.add_subplot(2, 2, 1)
plot_residuals(ax1, res, yfit, n_samp = 100)

# Second subplot: QQ Plot with 100 resamples
ax2 = fig.add_subplot(2, 2, 2)
plot_QQ(ax2, res_standard, n_samp = 100)

# Third subplot: Scale-location with 100 resamples
ax3 = fig.add_subplot(2, 2, 3)
plot_scale_loc(ax3, yfit, res_stand_sqrt, n_samp = 100)

# Fourth subplot: Residuals vs Fitted values with 100 resamples
ax4 = fig.add_subplot(2, 2, 4)
plot_cooks(ax4, res_inf_leverage, res_standard, n_pred = 2)


# Show plot
plt.tight_layout()
plt.show()
```

[4]:
```
import matplotlib.pyplot as plt
from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D
%matplotlib

n = 20

# Create data
```



```
# x1 and x2
x_1 = np.linspace(x['x1'].min(), x['x1'].max(), n)
x_2 = np.linspace(x['x2'].min(), x['x2'].max(), n)
# predicted values for y
y_pred = np.zeros((n, n))
for i in range(len(x_1)):
    for j in range(len(x_2)):
        y_pred[j][i] = model.predict([1, x_1[i], x_2[j]])
# grid
x_1, x_2 = np.meshgrid(x_1, x_2)

# Create Figure
fig = plt.figure(figsize=(8, 8))
ax = fig.gca(projection='3d')

# Plot the surface.
surf = ax.plot_surface(x_1, x_2, y_pred, cmap=cm.coolwarm)
# Set Labels
ax.set_xlabel("x_1"), ax.set_ylabel("x_2"), ax.set_zlabel("y")
ax.set_title("predicted and true y")
```

```
# 3D scatterplot
ax.scatter(x['x1'], x['x2'], y, alpha=0.8, linewidth=2)

plt.tight_layout()
plt.show()
```

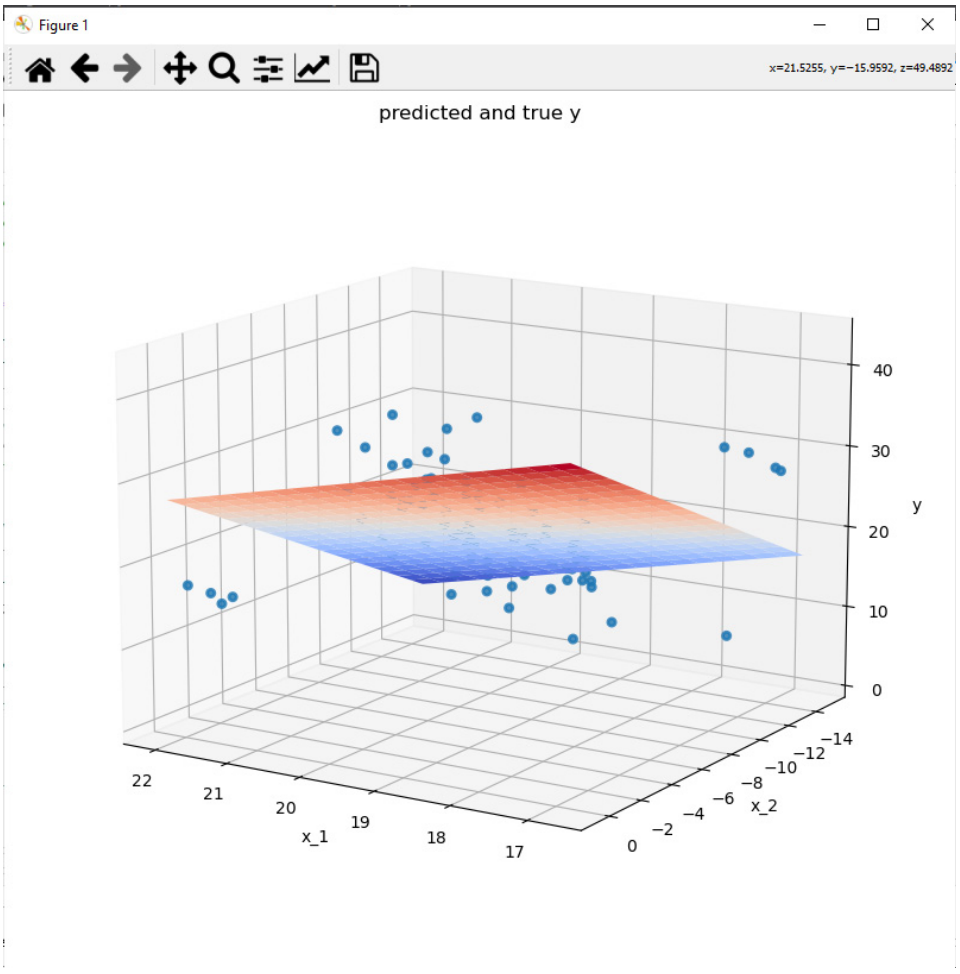


Figure 5: Regression plane and Data points, *Note:* the 3D plot sadly is not yet managing the different layers well, it always puts the surface on top.

The residual plots look fine. There is a slight but acceptable deviation in the Tukey-Anscombe plot and a slight but as well acceptable indication of increasing variance in the scale-location plot.

The 3D plot contradicts the above conclusion. The majority of the data points scatter around a plane that is completely different from the least squares solution. In addition, there are a few high leverage points that lie outside the point cloud on both sides of the regression plane. The least squares fit tries to accommodate all observations as good as possible. This results in the residuals having a similar size, i.e. the least squares plane does not fit well anywhere but it does not fit badly anywhere, either. Robust methods could account for this and weigh influential observations less strongly.

This example shows that the least squares solution can yield bad results which are not detected by the model diagnostics tools. This situation can occur when outliers occur in groups instead of as single observations. In such a setting, Cook's distance does not work reliably as a measure for influential data points as leaving out a single observation does not change much (and Cook's distance is based on this change).

While this example has been constructed artificially and is certainly an extreme one, it does show that robust methods constitute an important additional tool in regression analysis. Unfortunately, studying these methods is beyond the scope of this course.

Solution 3.4

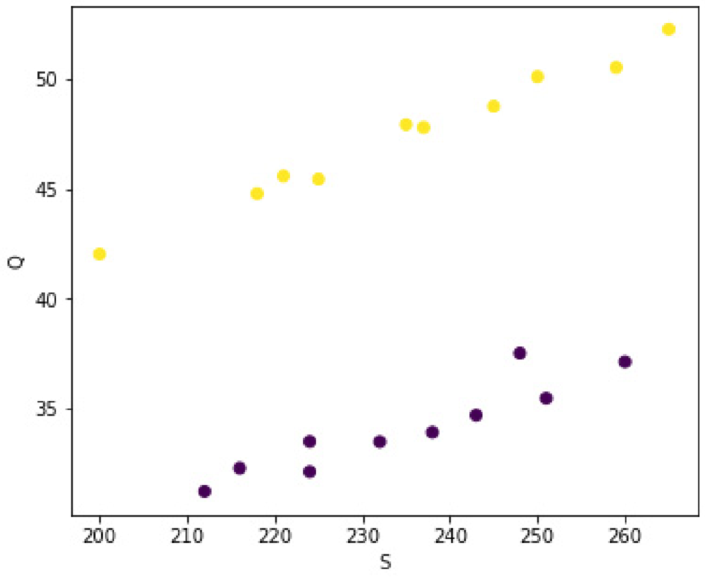
a) **Python** code:

[1]:
```
import pandas as pd
import numpy as np

# Load data
lathe = pd.read_csv('./data/lathe.csv')

# Sort data
lathe.sort_values(by='S', inplace=True)
# Add a column where CTT is mapped to 1/0 integer
dummies = pd.get_dummies(lathe['CTT'])
lathe = lathe.join(dummies)

# As a first inspection, print the first rows of the data:
print(lathe.head()) #, '\n\n', clocks.describe())
# As well as the dimensions of the set:
print('\nSize of lathe =\n', lathe.shape)
```

```
        Q     S    CTT  DM302  DM416
1   42.03   200  DM302      1      0
11  31.23   212  DM416      0      1
19  32.29   216  DM416      0      1
9   44.78   218  DM302      1      0
8   45.58   221  DM302      1      0

Size of lathe =
 (20, 5)
```

[2]:
```
import matplotlib.pyplot as plt

# Create Figure
fig = plt.figure(figsize=(6, 5))
```

```
ax = fig.add_subplot(1, 1, 1)

# Plot data
ax.scatter(lathe['S'], lathe['Q'], c=lathe['DM302'])

# Set labels and show
ax.set_xlabel("S"), ax.set_ylabel("Q")

plt.show()
```



The data points scatter around two almost parallel straight lines, depending on the used cutting tool.

b) **Python** code:

[3]:
```
import statsmodels.api as sm

# Define x and y:
x = lathe[['S', 'DM416']]
y = lathe['Q']

x_sm = sm.add_constant(x)

# Fit the linear model
model = sm.OLS(y, x_sm).fit()

print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Q   R-squared:                       0.992
```

```
Model:                            OLS   Adj. R-squared:                  0.991
Method:                 Least Squares   F-statistic:                     1104.
Date:                Wed, 10 Mar 2021   Prob (F-statistic):           1.02e-18
Time:                        11:36:10   Log-Likelihood:                -18.955
No. Observations:                  20   AIC:                             43.91
Df Residuals:                      17   BIC:                             46.90
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         14.2762      2.091      6.827      0.000       9.864      18.688
S              0.1411      0.009     15.979      0.000       0.123       0.160
DM416        -13.2802      0.303    -43.847      0.000     -13.919     -12.641
==============================================================================
Omnibus:                        1.989   Durbin-Watson:                   1.778
Prob(Omnibus):                  0.370   Jarque-Bera (JB):                1.484
Skew:                           0.647   Prob(JB):                        0.476
Kurtosis:                       2.675   Cond. No.                     3.26e+03
==============================================================================
```

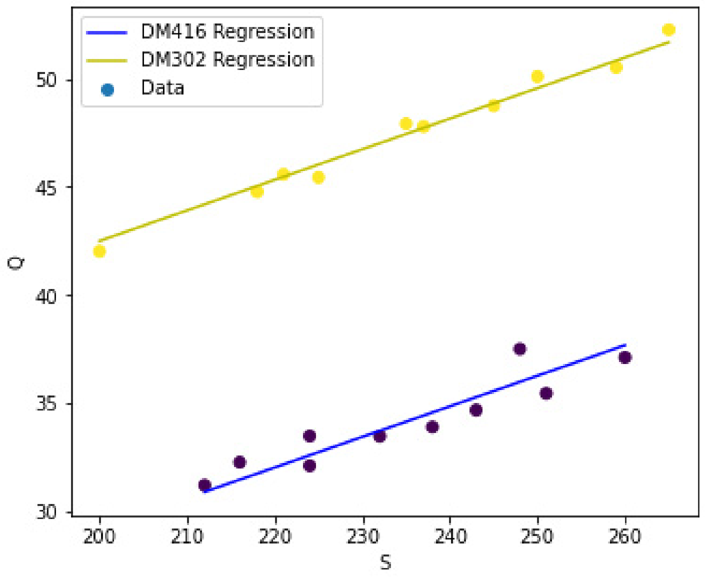
[4]:

```
# Create prediction
y_pred = (model.params['S'] * x['S'] +
          model.params['DM416'] * x['DM416'] +
          model.params['const'] )

# Create Figure
fig = plt.figure(figsize=(6, 5))
ax = fig.add_subplot(1, 1, 1)

# Plot data
ax.scatter(lathe['S'], lathe['Q'], c=lathe['DM302'], label='Data')

# Plot regression lines:
plt.plot(x['S'].loc[(lathe['DM416'] == 1)],
         y_pred.loc[(lathe['DM416'] == 1)],
         'b-', label='DM416 Regression')

plt.plot(x['S'].loc[(lathe['DM302'] == 1)],
         y_pred.loc[(lathe['DM302'] == 1)],
         'y-', label='DM302 Regression')

# Set labels and show
ax.set_xlabel("S"), ax.set_ylabel("Q")
plt.legend()

plt.show()
```

Since the type of cutting tool likely affects the surface finish, we will fit the model

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

where Y is the surface finish, X_1 is the lathe speed in revolutions per minute,



and X_2 is an indicator variable denoting the type of cutting tool used; that is,

$$X_2 = \begin{cases} 0 & \text{for tool type 302} \\ 1 & \text{for tool type 416} \end{cases}$$

The parameters in this model may be easily interpreted. If $X_2 = 0$, the model becomes

$$Y = \beta_0 + \beta_1 X_1 + \epsilon$$

which is a straight-line model with slope β_0 and intercept β_1. However, if $X_2 = 1$, the model becomes

$$Y = \beta_0 + \beta_1 X_1 + \beta_2(1) + \epsilon = (\beta_0 + \beta_2) + \beta_1 X_1 + \epsilon$$

which is a straight-line model with slope β_1 and intercept $\beta_0 + \beta_2$. Thus, the model

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

implies that surface finish is linearly related to lathe speed and that the slope β_1 does not depend on the type of cutting tool used. However, the type of cutting tool does affect the intercept, and β_2 indicates the change in the intercept associated with a change in tool type from 302 to 416

c) **`Python`** code:

```
[5]: # Define x and y:
     x = pd.DataFrame({
         'S': lathe['S'],
         'DM416': lathe['DM416'],
         'S * DM416': lathe['S'] * lathe['DM416']})
     y = lathe['Q']
```

```
[6]: x_sm = sm.add_constant(x)

     # Fit the linear model
     model = sm.OLS(y, x_sm).fit()

     print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Q   R-squared:                       0.994
Model:                            OLS   Adj. R-squared:                  0.992
Method:                 Least Squares   F-statistic:                     832.3
Date:                Wed, 10 Mar 2021   Prob (F-statistic):           9.00e-18
Time:                        11:36:10   Log-Likelihood:                -17.130
No. Observations:                  20   AIC:                             42.26
Df Residuals:                      16   BIC:                             46.24
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         11.5029      2.504      4.593      0.000       6.194      16.812
S              0.1529      0.011     14.428      0.000       0.130       0.175
DM416         -6.0942      4.025     -1.514      0.149     -14.626       2.437
S * DM416     -0.0306      0.017     -1.790      0.092      -0.067       0.006
==============================================================================
Omnibus:                       11.749   Durbin-Watson:                   1.760
Prob(Omnibus):                  0.003   Jarque-Bera (JB):                9.575
Skew:                           1.313   Prob(JB):                      0.00833
Kurtosis:                       5.144   Cond. No.                     8.30e+03
==============================================================================
```

The last part S * DM416 is the so called interaction between the variables **`S`** and **`CTT`**. Since **`CTT`** is a dummy variable, it means that **`S`** has a -0.0306 times larger (which means a 0.0306 times smaller) slope for **`CTT`**=DM416.

It is also possible to use indicator variables to investigate whether tool type affects both the slope and intercept. Let the model be

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 \cdot X_2 + \epsilon$$

where X_2 is the indicator variable. Now if tool type 302 is used, $X_2 = 0$, and the model is

$$Y = \beta_0 + \beta_1 X_1 + \epsilon$$

If tool type 416 is used, $X_2 = 1$, and the model becomes

$$Y = (\beta_0 + \beta_2) + (\beta_1 + \beta_3)X_1 + \epsilon$$

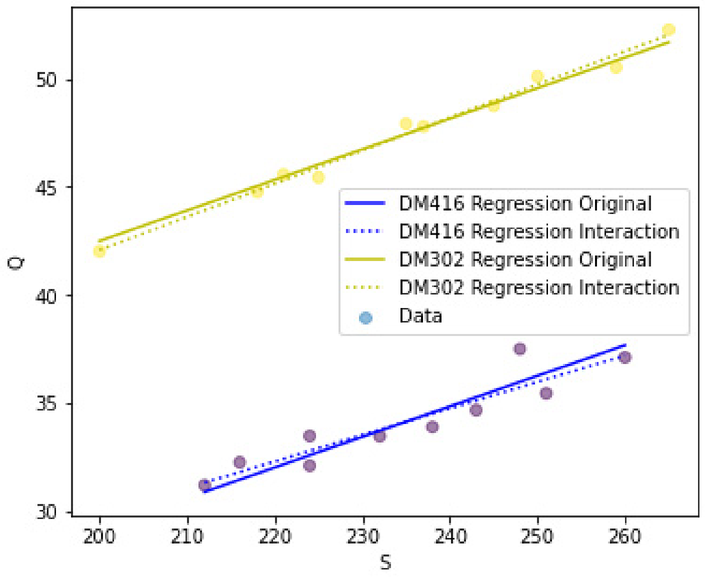
Note that β_2 is the change in the intercept and that β_3 is the change in slope produced by a change in tool type.

[7]:
```
# Create prediction
y_pred_2 = (model.params['S'] * x['S'] +
            model.params['DM416'] * x['DM416'] +
            model.params['S * DM416'] * x['S * DM416'] +
            model.params['const'] )

# Create Figure
fig = plt.figure(figsize=(6, 5))
ax = fig.add_subplot(1, 1, 1)

# Plot data
ax.scatter(lathe['S'], lathe['Q'], c=lathe['DM302'],
           alpha=0.5, label='Data')

# Plot regression lines:
plt.plot(x['S'].loc[(lathe['DM416'] == 1)],
         y_pred.loc[(lathe['DM416'] == 1)],
         'b-', label='DM416 Regression Original')
plt.plot(x['S'].loc[(lathe['DM416'] == 1)],
         y_pred_2.loc[(lathe['DM416'] == 1)],
         'b:', label='DM416 Regression Interaction')
plt.plot(x['S'].loc[(lathe['DM302'] == 1)],
         y_pred.loc[(lathe['DM302'] == 1)],
         'y-', label='DM302 Regression Original')
plt.plot(x['S'].loc[(lathe['DM302'] == 1)],
         y_pred_2.loc[(lathe['DM302'] == 1)],
         'y:', label='DM302 Regression Interaction')
# Set labels and show
ax.set_xlabel("S"), ax.set_ylabel("Q")
plt.legend()

plt.show()
```

The differences are visible but not very large.

d) Since the p-value for the differences of the slopes is 0.0924, the difference is not significant at the 5 % level, although it would be significant at the 10 % level.

We may have commited a type II error, which means that the null hypothesis is retained although the alternative hypothesis is correct. Since we do not have any information about the alternative hypothesis (we only know the value is not



equal 0), we cannot calculate the distribution of the test statistic assuming the alternative is true and that is why we cannot provide more information about the probability of a type II error than it lies between 0 and 1.

Solution 3.5

a) First, we check the structure of the data frame:

[1]:
```
import pandas as pd
import numpy as np

# Load data
farm = pd.read_csv('./data/farm.csv')

# As a first inspection, print the first rows of the data:
print(farm.head()) #, '\n\n', clocks.describe())

# As well as the dimensions of the set:
print('\nSize of farm =\n', farm.shape)

# As well as the types of data:
print('\ndtypes in farm =')
for i in range(farm.shape[1]):
    print(type(farm.iloc[i, 1]))

# Number of different values for `region`
print('\nUnique values for \'region\' ->', farm['region'].unique())

# Number of different values for `industry`
print('\nUnique values for \'industry\' ->', farm['industry'].unique())
```

```
   region  industry   costs  revenue
0     111         3  115096   147652
1     111         5   75443    82920
2     111         2  378857   442726
3     111         1  433590   649628
4     111         2  347417   407836

Size of farm =
 (451, 4)

dtypes in farm =
<class 'numpy.int64'>
<class 'numpy.int64'>
<class 'numpy.int64'>
<class 'numpy.int64'>

Unique values for 'region' -> [111 121 122 123 131 132]

Unique values for 'industry' -> [3 5 2 1 4]
```

All variables are of data type *int*. This is incorrect for the factor variables `region` and `industry` and would lead to incorrect regression results. We define the factor variables as follows:

[2]:
```
# Note that re-running this cell will raise an error
# Add a column where industry is mapped to a dummy variable
dummies_reg = pd.get_dummies(farm['region'], drop_first = True, prefix = 'reg')
dummies_ind = pd.get_dummies(farm['industry'], drop_first = True, prefix = 'ind')

farm = farm.join((dummies_reg, dummies_ind))

print(farm.head())
```

```
   region  industry   costs  revenue  reg_121  reg_122  reg_123  reg_131
0     111         3  115096   147652        0        0        0        0
1     111         5   75443    82920        0        0        0        0
2     111         2  378857   442726        0        0        0        0
3     111         1  433590   649628        0        0        0        0
4     111         2  347417   407836        0        0        0        0

   reg_132  ind_2  ind_3  ind_4  ind_5
0        0      0      1      0      0
1        0      0      0      0      1
2        0      1      0      0      0
3        0      0      0      0      0
4        0      1      0      0      0
```

For the advanced reader: We now check whether there are sufficiently many observations for all levels of the factor variables. It is recommended that there are at least five observations for each level.

[3]:
```
print(farm.sum())
```

```
region           55774
industry          1410
costs        115856032
revenue      135697768
reg_121             95
reg_122            103
reg_123            108
reg_131             81
reg_132             34
ind_2              137
ind_3              100
ind_4               86
ind_5               91
dtype: int64
```

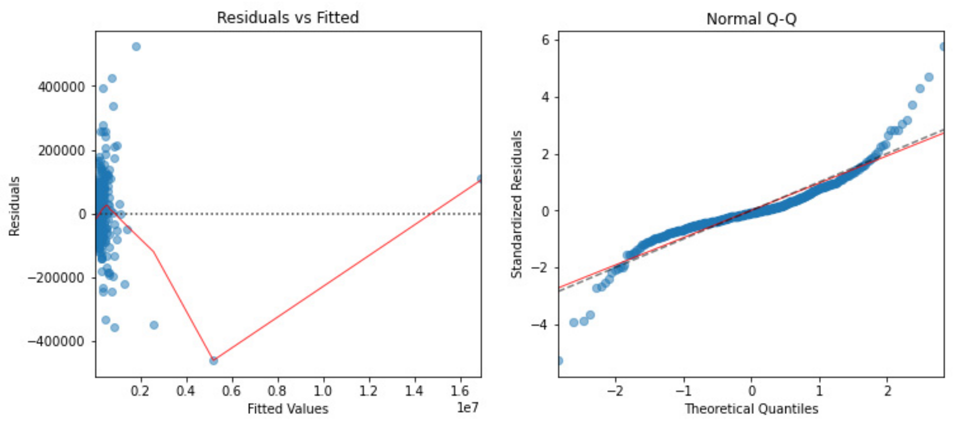
The number of observations are sufficient for all levels of the factor variables.

b) `Python` code:

[4]:
```python
import statsmodels.api as sm

# Define x and y:
x = farm.drop(labels = ['region', 'industry', 'revenue'], axis=1)
y = farm['revenue']

x_sm = sm.add_constant(x)

# Fit the linear model
model = sm.OLS(y, x_sm).fit()

# Find the predicted values for the original design.
yfit = model.fittedvalues
# Find the Residuals
res = model.resid
# Influence of the Residuals
res_inf = model.get_influence()
# Studentized residuals using variance from OLS
res_standard = res_inf.resid_studentized_internal
```

[5]:
```python
from MLR_def import *

""" Plots """
# Create Figure and subplots
fig = plt.figure(figsize = (12,5))

# First subplot: Residuals vs Fitted values
ax1 = fig.add_subplot(1, 2, 1)
plot_residuals(ax1, res, yfit)

# Second subplot: QQ Plot
```

```
ax2 = fig.add_subplot(1, 2, 2)
plot_QQ(ax2, res_standard)

plt.show()
```



The Tukey-Anscombe plot and the normal plot clearly show that the model assumptions are violated. A variable transformation is absolutely necessary.

We log-transform the predictor variable `costs` and the response variable `revenue`.

[6]:
```
# Define x and y:
x['costs'] = np.log(x['costs'])
y = np.log(farm['revenue'])

x_sm = sm.add_constant(x)

# Fit the linear model
model = sm.OLS(y, x_sm).fit()
print(model.summary())

# Find the predicted values for the original design.
yfit = model.fittedvalues
# Find the Residuals
res = model.resid
# Influence of the Residuals
res_inf = model.get_influence()
# Studentized residuals using variance from OLS
res_standard = res_inf.resid_studentized_internal
```

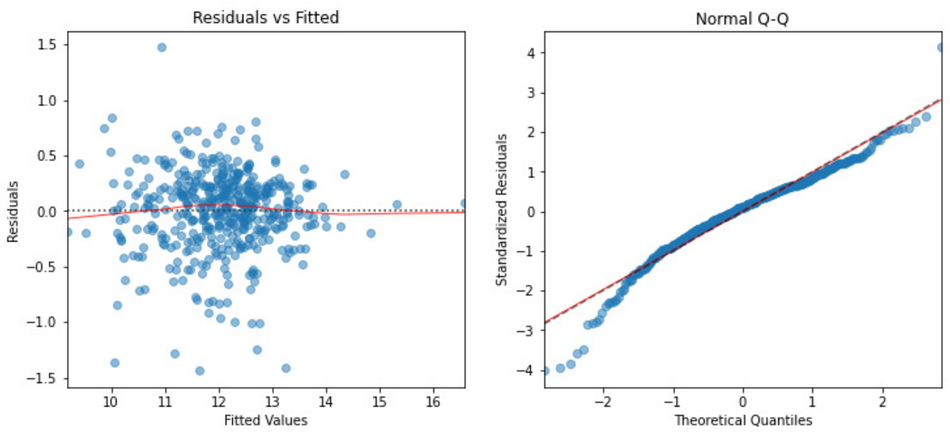
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                revenue   R-squared:                       0.871
Model:                            OLS   Adj. R-squared:                  0.868
Method:                 Least Squares   F-statistic:                     297.7
Date:                Wed, 13 Nov 2024   Prob (F-statistic):          8.29e-189
Time:                        12:44:05   Log-Likelihood:                -175.13
```

```
No. Observations:                 451   AIC:                             372.3
Df Residuals:                     440   BIC:                             417.5
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.3796      0.248      5.553      0.000       0.891       1.868
costs          0.9180      0.019     49.306      0.000       0.881       0.955
reg_121       -0.0769      0.077     -0.994      0.321      -0.229       0.075
reg_122       -0.0830      0.077     -1.079      0.281      -0.234       0.068
reg_123       -0.0367      0.076     -0.482      0.630      -0.186       0.113
reg_131       -0.0039      0.080     -0.048      0.961      -0.161       0.153
reg_132       -0.2439      0.101     -2.426      0.016      -0.442      -0.046
ind_2         -0.1556      0.068     -2.288      0.023      -0.289      -0.022
ind_3         -0.2229      0.071     -3.121      0.002      -0.363      -0.083
ind_4          0.0026      0.076      0.035      0.972      -0.146       0.152
ind_5         -0.1711      0.073     -2.346      0.019      -0.314      -0.028
==============================================================================
Omnibus:                       59.533   Durbin-Watson:                   1.897
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              131.920
Skew:                          -0.707   Prob(JB):                     2.26e-29
Kurtosis:                       5.241   Cond. No.                         185.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly␣
↪specified.
```

[7]:
```python
from MLR_def import *

""" Plots """
# Create Figure and subplots
fig = plt.figure(figsize = (12,5))

# First subplot: Residuals vs Fitted values
ax1 = fig.add_subplot(1, 2, 1)
plot_residuals(ax1, res, yfit)

# Second subplot: QQ Plot with
ax2 = fig.add_subplot(1, 2, 2)
plot_QQ(ax2, res_standard)

plt.show()
```

The Tukey-Anscombe plot does not indicate the presence of any systematic error. The normal plot shows that the distribution of the residuals is skewed to the left and there is one large positive outlier no. 43). In summary, the assumptions seem to be fulfilled to a sufficient degree but not entirely.

c) Using `model.predict()` we obtain the prediction on the log scale. We thus need to transform the value back to the original scale. So the predicted `revenue` is 154 914.2 Dollar.



```
[8]: # quickest way to copy the shape and columns of x:
     x0 = x.iloc[0, :].copy()
     x0[:] = 0

     # Set predictor variables
     x0['costs'] = np.log(100000)
     x0['ind_4'] = 1
     # Add constant 1 for offset
     x0 = pd.concat([pd.Series(1), x0], axis=0)

     # Predict
     pred = model.predict([x0])
     pred = np.exp(pred)

     print('Predicted Revenue:\n', np.round(pred[0],0))

     # or, without rounding
     print('Predicted Revenue:\n', pred[0])

     # OR: directly introducing a list for the predictors
     # Predict
     pred = model.predict([1, np.log(100000), 0, 0, 0, 0, 0, 0, 0, 1, 0])
     pred = np.exp(pred)
     print('Predicted Revenue:\n', pred[0])
```

```
Predicted Revenue:
 154914.0
Predicted Revenue:
 154914.21558794827
Predicted Revenue:
 154914.21558794827
```

d) **Python** code:

[9]:
```
""" Hints:
The easiest solution uses statsmodels.formula.api,
together with sm.stats.anova_lm"""
import statsmodels.formula.api as smf

"""Define model in Formula form,
np.log() is the log,
C() defines Categorical variables. """
model_f = smf.ols(formula='np.log(revenue) ~ \
                  np.log(costs) + C(region) + C(industry)',
                  data=farm).fit()

# Table and print results using anova_lm
table_f = sm.stats.anova_lm(model_f, typ=2)
print(table_f)
```

```
                   sum_sq     df            F         PR(>F)
C(region)        1.363906    5.0     2.090604   6.550739e-02
C(industry)      2.766513    4.0     5.300664   3.542108e-04
np.log(costs)  317.207862    1.0  2431.092268  2.541722e-181
Residual        57.411009  440.0          NaN            NaN
```

The predictor variable **region** is not significant as can be seen from the p-value 0.0655 of the partial F-test.

e)
- 31 parameters are estimated since the model has 420 degrees of freedom and there are 451 observations.
- We have sufficiently many observations since there are more than five observations for every estimated parameter.
- To test the interaction term we need to do a partial F-test. We could do this explicitly with the **Python**-function **sm.stats.anova_lm()**:

[10]:
```
model_f = smf.ols(formula='np.log(revenue) ~ \
                  np.log(costs) + C(region) + C(industry) \
                  + C(region) * C(industry)',
                  data=farm).fit()

# Table and print results using anova_lm
table_f = sm.stats.anova_lm(model_f, typ=2)
print(table_f)
```

```
                             sum_sq     df            F         PR(>F)
C(region)                  1.363906    5.0     2.100608   6.439816e-02
C(industry)                2.766513    4.0     5.326029   3.421181e-04
C(region):C(industry)      2.870583   20.0     1.105276   3.404492e-01
np.log(costs)            303.466942    1.0  2336.910900  1.045538e-173
Residual                  54.540426  420.0          NaN            NaN
```

The interaction term is not significant and can be excluded from the model.

- The interaction term can be interpreted in an intuitive way as follows: `region` and `industry` do not influence `revenue` independently and additively, but the influence of `industry` differs between regions. However, as we have seen this is not the case for this data set.

f) The interaction term is not significant as we have seen above. So we exclude it and are left with the main effects model (model without interaction term). As we have concluded for the main effects model, the predictor variable `region` is not significant, so we will exclude it as well. Thus, the log-transformed response variable `revenue` is predicted by the log-transformed predictor variable `costs` and by the predictor `industry`. In this model both predictors are significant. Hence, we select this model.