Statistik für Fortgeschrittene: Bonusprogramm

Dieses Jupyter Notebook ist unsere Abgabe für das Statistik für Fortgeschrittene Bonusprogramm im WS22/23 @KIT.

Thema:

- Wie reagieren Schätzer auf Ausreißer in Daten? (OLS/LAD)

Autoren:

- Katharina Jacob, MATRIKELNR
- Peter Fabisch, MATRIKELNR
- Alessio Negrini, 2106547

Import der benötigten Libraries

In [1]:

```
# Convert notebook to pdf
!jupyter nbconvert --to webpdf --allow-chromium-download ols_lad_regression.ipynb
```

```
[NbConvertApp] Converting notebook ols_lad_regression.ipynb to webpdf
[NbConvertApp] Building PDF
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 585879 bytes to ols_lad_regression.pdf
```

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
import statsmodels.formula.api as smf
import seaborn as sns
sns.set_theme(style="whitegrid")
```

Mit der statsmodel.api können wir sowohl eine OLS Regression, als auch eine LAD Regression ausführen.
Beachte, dass eine LAD Regression eine Quantil Regression ist mit dem Parameter $q = 0.5$. Eine LAD Regression ist somit auch in statsmodels.api implementiert, jedoch als Quantil Regression.

Die restlichen Libraries sind dafür Gedacht, Daten zu manipulieren und zu visualisieren.

Load Data

The data shows the relationship between income and expenditures on food for a sample of working class Belgian households in 1857 (the Engel data).

In [3]:

```
# Load data as pandas df
df = sm.datasets.engel.load_pandas().data.round(2)

# Information
print("Shape of the data:", df.shape)

# Save exogen and response variable
X = df["income"]
```

```
y = df["foodexp"]

# Add constant to exogen variables to add intercept in model
X = sm.add_constant(X)

# Preview Data
df.head(5)
```

```
Shape of the data: (235, 2)
```

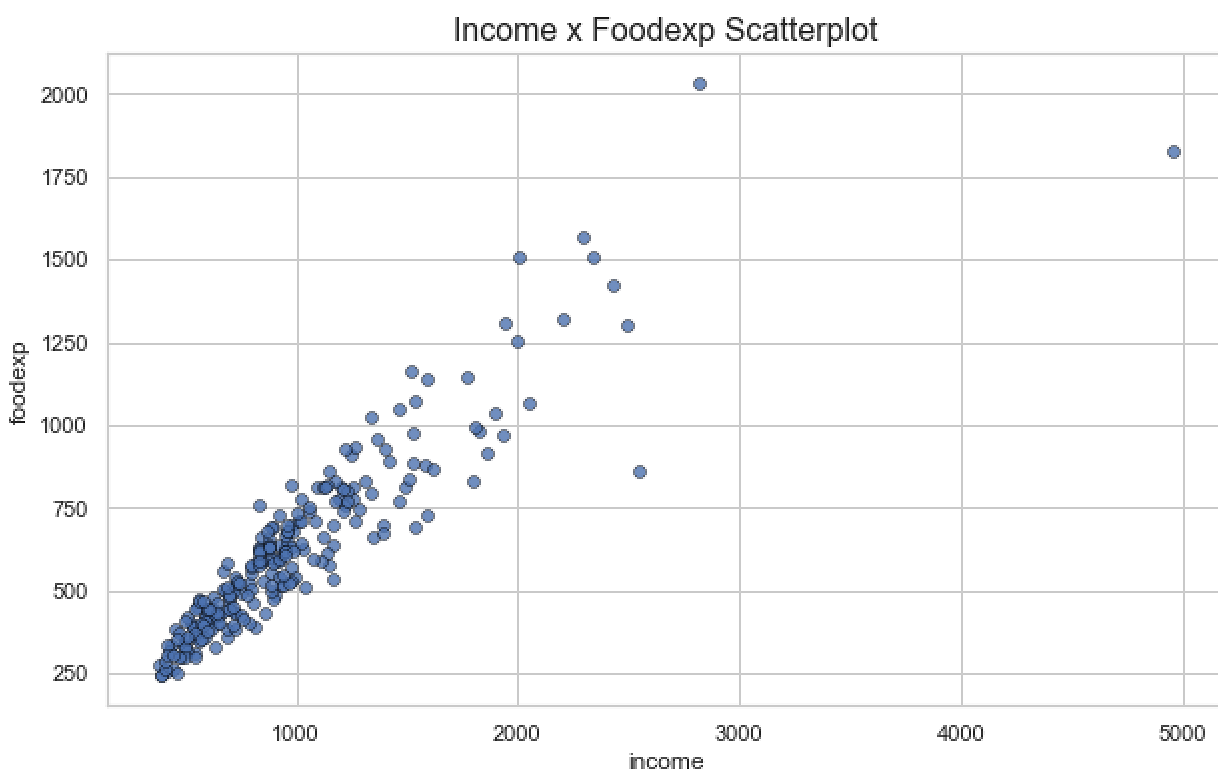
Out[3]:

	income	foodexp
0	420.16	255.84
1	541.41	310.96
2	901.16	485.68
3	639.08	403.00
4	750.88	495.56

Visualize Data

In [4]:

```
plt.figure(figsize=(10,6))
plt.title("Income x Foodexp Scatterplot", size=16)
sns.scatterplot(data=df, x="income", y="foodexp", edgecolor="black", alpha=0.8)
plt.show()
```



In [14]:

```
df.describe()
```

Out[14]:

	income	foodexp
count	235.000000	235.000000
mean	982.473191	624.150511
std	519.230382	276.456821
min	377.060000	242.320000

25%	638.875000	429.690000
50%	883.980000	582.540000
75%	1163.985000	743.885000
max	4957.810000	2032.680000

In [15]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 235 entries, 0 to 234
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   income   235 non-null    float64
 1   foodexp  235 non-null    float64
dtypes: float64(2)
memory usage: 3.8 KB
```

OLS Regression

The following function is the target function of the OLS regression:

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2$$

Which can be solved using linear algebra, i.e. compute the derivation, set to 0 and solve for β:

$$\hat{\beta} = (X'X)^{-1}X'y$$

In [5]:
```
# Instantiate model
ols_model = sm.OLS(endog=y, exog=X)

# Train model, i.e. fit estimators
results = ols_model.fit()

# Show summary
print(results.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                foodexp   R-squared:                       0.830
Model:                            OLS   Adj. R-squared:                  0.830
Method:                 Least Squares   F-statistic:                     1141.
Date:                Tue, 24 Jan 2023   Prob (F-statistic):           9.92e-92
Time:                        12:58:58   Log-Likelihood:                -1445.7
No. Observations:                 235   AIC:                             2895.
Df Residuals:                     233   BIC:                             2902.
Df Model:                           1                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        147.4757     15.957      9.242      0.000     116.037     178.914
income         0.4852      0.014     33.772      0.000       0.457       0.513
==============================================================================
Omnibus:                       68.110   Durbin-Watson:                   1.411
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              927.668
Skew:                          -0.670   Prob(JB):                    3.63e-202
Kurtosis:                      12.641   Cond. No.                     2.38e+03
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
[2] The condition number is large, 2.38e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

LAD Regression

LAD regression is very similar to OLS regression, but instead of trying to minimize the squared residuals, we minimize the absolute residuals, i.e. the difference between the actual response variable y and our estimated values $\hat{y}$.

So the objective function is defined as:

$$\min_{\beta} \sum_{i=1}^{n} |y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j|$$

Unlike the OLS solution, we can not solve this analytically. So there exists many approaches. The statsmodel.api uses the **Iterative Weighted Least Squares** approach.

In [6]:
```
# Instantiate model
lad_model = smf.quantreg("foodexp ~ income", df)

# Fit model with q=0.5, i.e. LAD Regression
lad_results = lad_model.fit(q=0.5)

# Show summary
print(lad_results.summary())
```

```
                         QuantReg Regression Results
==============================================================================
Dep. Variable:                foodexp   Pseudo R-squared:               0.6206
Model:                       QuantReg   Bandwidth:                       64.51
Method:                 Least Squares   Sparsity:                        209.3
Date:                Tue, 24 Jan 2023   No. Observations:                  235
Time:                        12:58:58   Df Residuals:                      233
                                        Df Model:                            1
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     81.4823     14.635      5.568      0.000      52.649     110.315
income         0.5602      0.013     42.516      0.000       0.534       0.586
==============================================================================

The condition number is large, 2.38e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Add Outliers

In [7]:
```
# Manually create outliers
outliers = [(4000, 300, "Outlier"),
            (3900, 350, "Outlier"),
            (4300, 400, "Outlier"),
            (4200, 370, "Outlier"),
            (3600, 370, "Outlier")]

# Append to df and plot
cols = list(df.columns)
```

```python
# Add flag column
cols.append("flag")
outlier_df = pd.DataFrame(outliers, columns=cols)
outlier_df.head(5)

df_copy = df.copy()

# Add flag column to df
df_copy["flag"] = "Normal"

# Create df with outlier
df_with_outlier = pd.concat([df_copy, outlier_df], axis=0)
df_with_outlier.reset_index(drop=True, inplace=True)
df_with_outlier
```

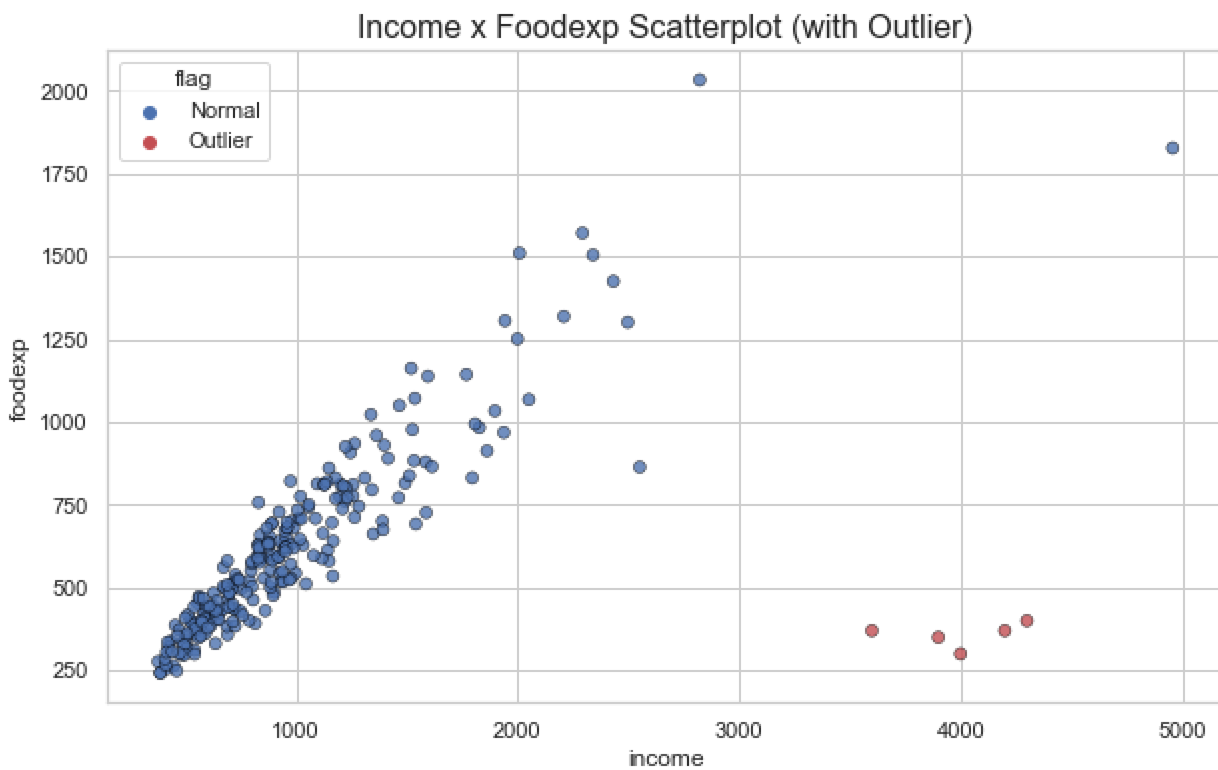
Out[7]:

	income	foodexp	flag
0	420.16	255.84	Normal
1	541.41	310.96	Normal
2	901.16	485.68	Normal
3	639.08	403.00	Normal
4	750.88	495.56	Normal
...	...	...	...
235	4000.00	300.00	Outlier
236	3900.00	350.00	Outlier
237	4300.00	400.00	Outlier
238	4200.00	370.00	Outlier
239	3600.00	370.00	Outlier

240 rows × 3 columns

In [8]:

```python
plt.figure(figsize=(10,6))
plt.title("Income x Foodexp Scatterplot (with Outlier)", size=16)
sns.scatterplot(data=df_with_outlier,
                x="income",
                y="foodexp",
                edgecolor="black",
                hue="flag",
                palette = ["C0", "C3"],
                alpha=0.8)
plt.show()
```



OLS with Outlier

In [9]:

```
# Instantiate model
ols_model_outlier = sm.OLS(endog=df_with_outlier["foodexp"], exog=sm.add_constant(df_wit

# Train model, i.e. fit estimators
ols_results_outlier = ols_model_outlier.fit()

# Show summary
print(ols_results_outlier.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                foodexp   R-squared:                       0.362
Model:                            OLS   Adj. R-squared:                  0.359
Method:                 Least Squares   F-statistic:                     134.9
Date:                Tue, 24 Jan 2023   Prob (F-statistic):           5.33e-25
Time:                        12:58:59   Log-Likelihood:                -1635.3
No. Observations:                 240   AIC:                             3275.
Df Residuals:                     238   BIC:                             3281.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        360.2151     26.434     13.627      0.000     308.140     412.290
income         0.2472      0.021     11.614      0.000       0.205       0.289
==============================================================================
Omnibus:                       81.398   Durbin-Watson:                   0.956
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              806.751
Skew:                          -1.008   Prob(JB):                    6.55e-176
Kurtosis:                      11.753   Cond. No.                     2.30e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
```

```
[2] The condition number is large, 2.3e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

LAD with Outlier

In [10]:

```
# Instantiate model
lad_model_outlier = smf.quantreg("foodexp ~ income", df_with_outlier)

# Fit model with q=0.5, i.e. LAD Regression
lad_results_outlier = lad_model_outlier.fit(q=0.5)

# Show summary
print(lad_results_outlier.summary())
```

```
                         QuantReg Regression Results
==============================================================================
Dep. Variable:                foodexp   Pseudo R-squared:               0.4352
Model:                       QuantReg   Bandwidth:                       69.70
Method:                 Least Squares   Sparsity:                        235.2
Date:                Tue, 24 Jan 2023   No. Observations:                  240
Time:                        12:58:59   Df Residuals:                      238
                                        Df Model:                            1
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    145.6372     14.058     10.360      0.000     117.943     173.331
income         0.4799      0.011     42.401      0.000       0.458       0.502
==============================================================================

The condition number is large, 2.3e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Comparison

In the following section we compare both Regression models (OLS/LAD) and after adding outliers.

Visualize both Regression lines

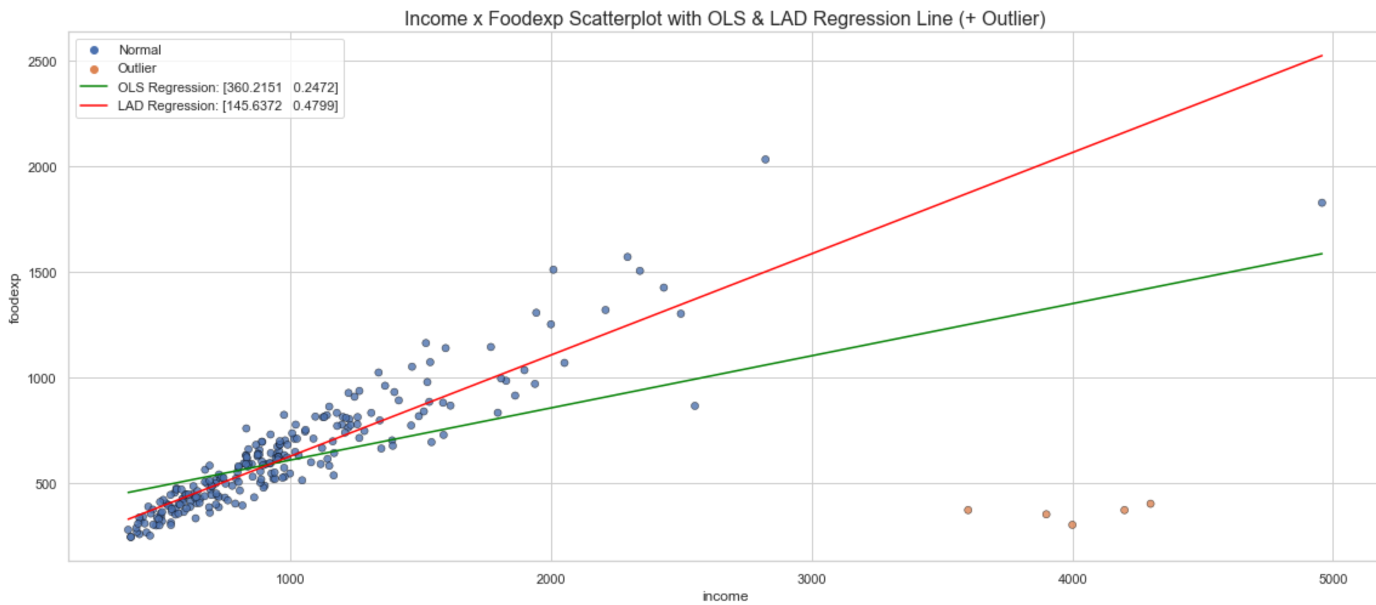
In [11]:

```
def regression_line(results, x) -> float:
    """
    Function to get the estimated value given the exogen variable x.
    First gets the estimated betas from the fitted model and then computes the estimated
    params: results Results of the fitted model (OLS / LAD)
    params: x Scalar of the exogen variable
    returns: Estimated value y_hat
    """

    # Get fitted values
    intercept = results.params[0]
    beta_1 = results.params[1]

    # Compute y_hat, i.e. estimated value
    y_hat = intercept + x * beta_1
    return y_hat
```

In [12]:

```
def plot_comparison(df, ols_results, lad_results, ols_results_outlier, lad_results_outli
    """
    Plots comparison between OLS and LAD Regression as well as with outlier added
    :param: df
```

```
    :param: ols_results
    :param: lad results
    :param: ols_results_outlier
    :param: lad_results_outlier
    """

    # Surpress scientific notation
    np.set_printoptions(suppress=True)

    fig, ax = plt.subplots(nrows=2, ncols=1, figsize=(16,14))

    ax[0].set_title("Income x Foodexp Scatterplot with OLS & LAD Regression Line", size=
    sns.scatterplot(data=df[df.flag == "Normal"], x="income", y="foodexp", edgecolor="bl
    # Add regression line for OLS
    sns.lineplot(x=df["income"],
                 y=df["income"].apply(lambda x: regression_line(ols_results, x)),
                 color="green",
                 label=f"OLS Regression: {np.around(ols_results.params.values, 4)}", ax
    # Add regression line for LAD
    sns.lineplot(x=df["income"],
                 y=df["income"].apply(lambda x: regression_line(lad_results, x)),
                 color="red",
                 label=f"LAD Regression: {np.around(lad_results.params.values, 4)}", ax

    # With Outlier:
    ax[1].set_title("Income x Foodexp Scatterplot with OLS & LAD Regression Line (+ Outl
    sns.scatterplot(data=df, x="income", y="foodexp", edgecolor="black", alpha=0.8, hue=
    # Add regression line for OLS
    sns.lineplot(x=df["income"],
                 y=df["income"].apply(lambda x: regression_line(ols_results_outlier, x))
                 color="green",
                 label=f"OLS Regression: {np.around(ols_results_outlier.params.values, 4
    # Add regression line for LAD
    sns.lineplot(x=df["income"],
                 y=df["income"].apply(lambda x: regression_line(lad_results_outlier, x))
                 color="red",
                 label=f"LAD Regression: {np.around(lad_results_outlier.params.values, 4
    plt.tight_layout()
    plt.show()


plot_comparison(df=df_with_outlier,
         ols_results=results,
         lad_results=lad_results,
         ols_results_outlier=ols_results_outlier,
         lad_results_outlier=lad_results_outlier)
```



Visualize Regression Plot with OLS

In [13]:
```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(16,14))
# OLS Residual Plot Settings
sns.residplot(y=results.resid, x=results.fittedvalues, ax=axes[0,0], lowess=True, line_k
axes[0,0].set_title("Residual Plot OLS", size=16)
axes[0,0].axhline(y=0, color="green", linestyle="--")
axes[0,0].set_xlabel("Fitted values $\hat{y}_i$")
axes[0,0].set_ylabel("Residuals $e_i$")
# LAD Residual Plot Settings
sns.residplot(y=lad_results.resid, x=lad_results.fittedvalues, ax=axes[0,1], lowess=True
axes[0,1].set_title("Residual Plot LAD", size=16)
axes[0,1].axhline(y=0, color="green", linestyle="--")
axes[0,1].set_xlabel("Fitted values $\hat{y}_i$")
axes[0,1].set_ylabel("Residuals $e_i$")
# OLS Residual Plot Settings (+ Outlier)
sns.residplot(y=ols_results_outlier.resid, x=ols_results_outlier.fittedvalues, ax=axes[1
axes[1,0].set_title("Residual Plot OLS (with Outlier)", size=16)
axes[1,0].axhline(y=0, color="green", linestyle="--")
axes[1,0].set_xlabel("Fitted values $\hat{y}_i$")
axes[1,0].set_ylabel("Residuals $e_i$")
# LAD Residual Plot Settings (+ Outlier)
sns.residplot(y=lad_results_outlier.resid, x=lad_results_outlier.fittedvalues, ax=axes[1
axes[1,1].set_title("Residual Plot LAD (with Outlier)", size=16)
axes[1,1].axhline(y=0, color="green", linestyle="--")
```

```
axes[1,1].set_xlabel("Fitted values $\hat{y}_i$")
axes[1,1].set_ylabel("Residuals $e_i$")
fig.tight_layout()
plt.show()
```