Linear Models 1: Cats Lab

Dr. Luisa Barbanti and Dr. Matteo Tanadini

Applied Machine Learning and Predictive Modelling 1, HS25 (HSLU)

Contents

1 Import packages

```
## Import data sets
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.formula.api as smf
import numpy as np
import statsmodels.api as sm
import matplotlib.patches as mpatches
```

2 Load data set

```
## Load dataset
d_cats = pd.read_csv("../../Datasets/Cats.csv")

## Rename col names because "." causes troubles
d_cats.rename(columns = {'Body.weight': 'Body_weight',
                         'Heart.weight': 'Heart_weight'},
              inplace = True)

## Inspect structure
print(d_cats.info())
```

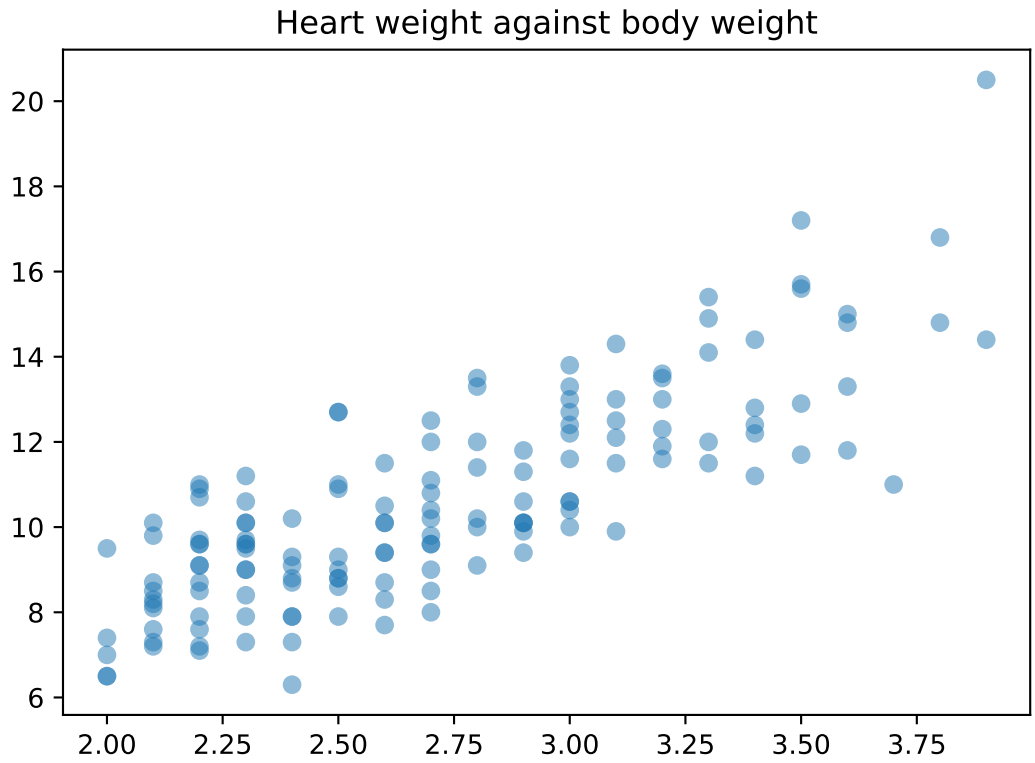
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144 entries, 0 to 143
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Sex           144 non-null    object
 1   Body_weight   144 non-null    float64
 2   Heart_weight  144 non-null    float64
dtypes: float64(2), object(1)
memory usage: 3.5+ KB
None
```

```
print(d_cats.head())
```

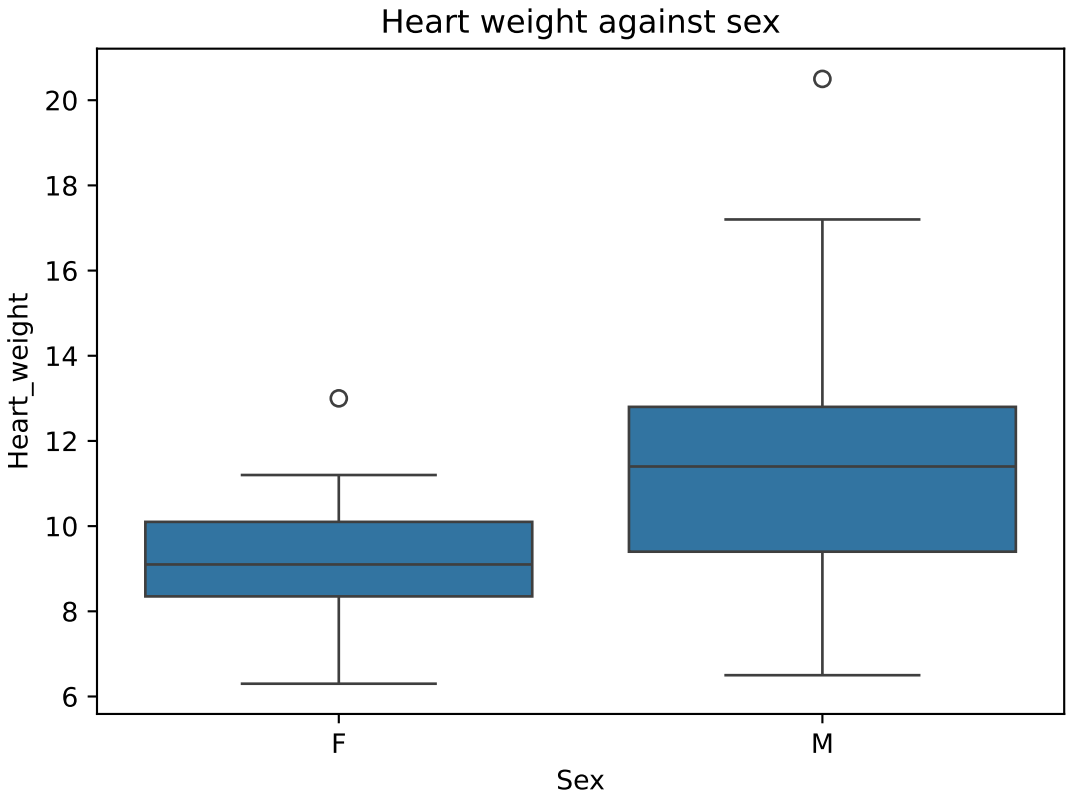
```
  Sex  Body_weight  Heart_weight
0   F          2.0           7.0
1   F          2.0           7.4
2   F          2.0           9.5
3   F          2.1           7.2
4   F          2.1           7.3
```

3 Graphical Analysis

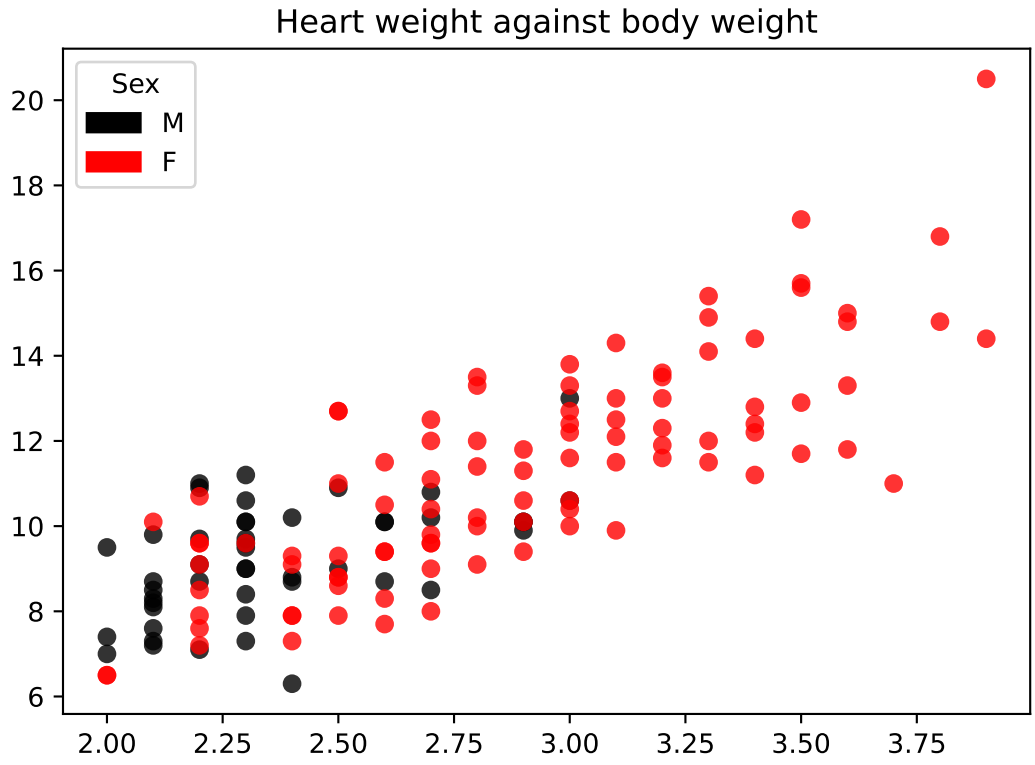
```
## Clean figure object
plt.clf()
## Scatterplot of Heart_weight vs. Body_weight
plt.scatter(d_cats['Body_weight'], d_cats['Heart_weight'], alpha = 0.5)
plt.title("Heart weight against body weight")

## If you want to change axes name
# plt.xlabel("Body weight")
# plt.ylabel("Heart weight")
```



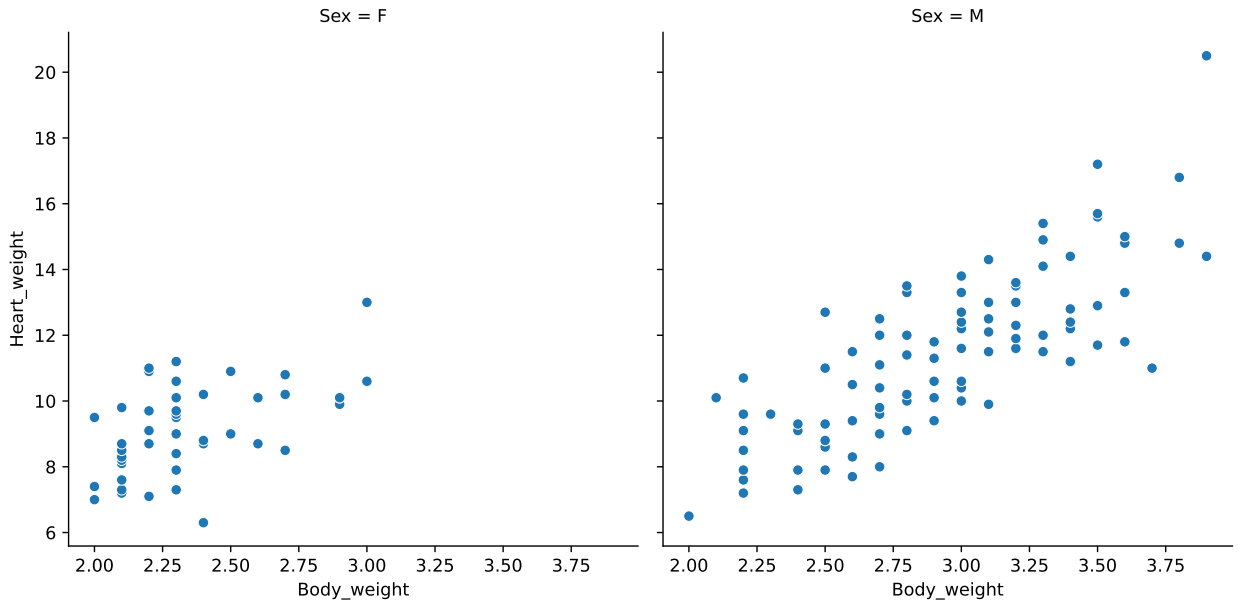

```
## Clean figure object
plt.clf()
## Boxplot of Heart_weight by Sex
sns.boxplot(x = 'Sex', y = 'Heart_weight', data = d_cats)
plt.title("Heart weight against sex")

## If you want to change y label
# plt.ylabel("Heart weight")
```



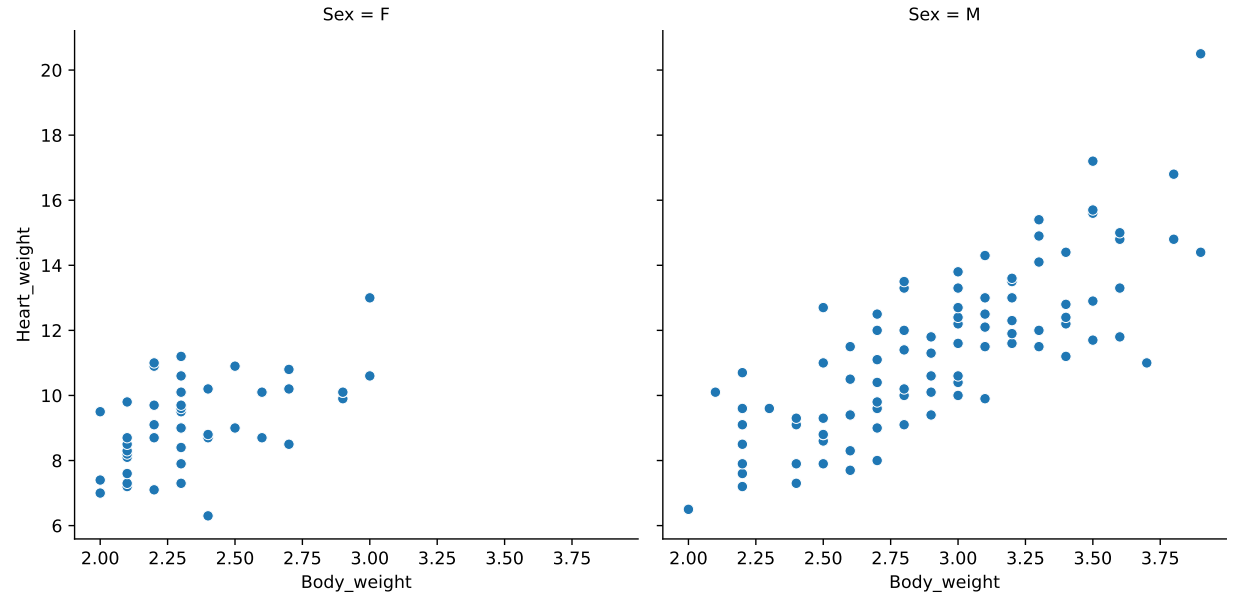

```
## Clean figure object
plt.clf()

## Set colors for sex groups
colors = {'F': 'black', 'M': 'red'}
## Scatterplot with color-coded Sex
plt.scatter(d_cats['Body_weight'], d_cats['Heart_weight'],
            c = d_cats['Sex'].map(colors), label = d_cats['Sex'], alpha = 0.8)
plt.title("Heart weight against body weight")

## Set legend to display both M and F. If not, it only displays F.
legend_handles = [mpatches.Patch(color = 'black', label = 'M'),
                  mpatches.Patch(color = 'red', label = 'F')]
plt.legend(handles = legend_handles, title = "Sex", loc = 'upper left')

## If you want to change axes name
# plt.xlabel("Body weight")
# plt.ylabel("Heart weight")
```




```
## Clean figure object
plt.clf()

## Scatterplot with panelling by Sex
g = sns.FacetGrid(d_cats, col = 'Sex', height = 5, aspect = 1)
g.map(sns.scatterplot, 'Body_weight', 'Heart_weight')
```

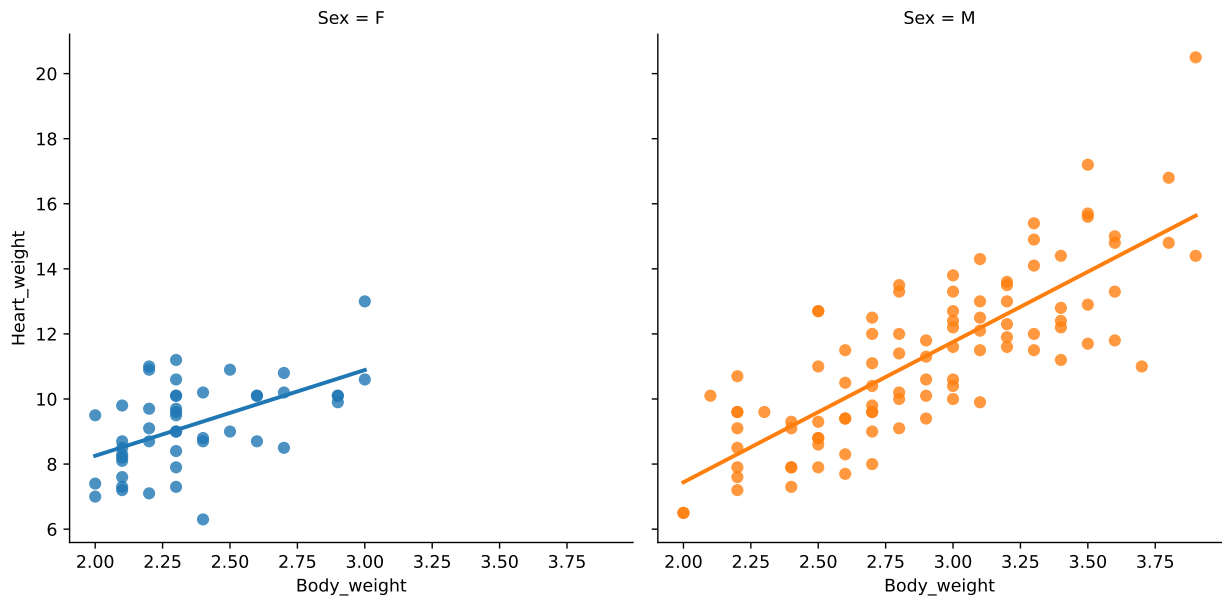



```
## If you want to color the points by groups
# g.set_titles("{col_name}")

## If you want to change the axes name
# g.set_axis_labels("Body weight", "Heart weight")
```




```
## Clean figure object
plt.clf()

## Scatterplot with regression line in each panel
g = sns.lmplot(x = 'Body_weight', y = 'Heart_weight', hue = 'Sex',
               data = d_cats, col = 'Sex', ci = None, height = 5, aspect = 1)
```

```
##  If you want to color the points by groups
# g.set_titles("{col_name}")
## If you want to change the axes name
# g.set_axis_labels("Body weight", "Heart weight")
```



4 Fitting models

```
## Fit a linear model: Hwt ~ Bwt
lm_cats = smf.ols('Heart_weight ~ Body_weight', data = d_cats).fit()
print(lm_cats.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Heart_weight   R-squared:                       0.647
Model:                            OLS   Adj. R-squared:                  0.644
Method:                 Least Squares   F-statistic:                     259.8
Date:                Thu, 11 Sep 2025   Prob (F-statistic):           6.97e-34
Time:                        14:40:04   Log-Likelihood:                -257.06
No. Observations:                 144   AIC:                             518.1
Df Residuals:                     142   BIC:                             524.1
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      -0.3567      0.692     -0.515      0.607      -1.725       1.012
Body_weight     4.0341      0.250     16.119      0.000       3.539       4.529
==============================================================================
Omnibus:                        4.637   Durbin-Watson:                   1.580
```

```
Prob(Omnibus):                  0.098   Jarque-Bera (JB):                4.217
Skew:                           0.408   Prob(JB):                        0.121
Kurtosis:                       3.191   Cond. No.                         17.8
==============================================================================

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

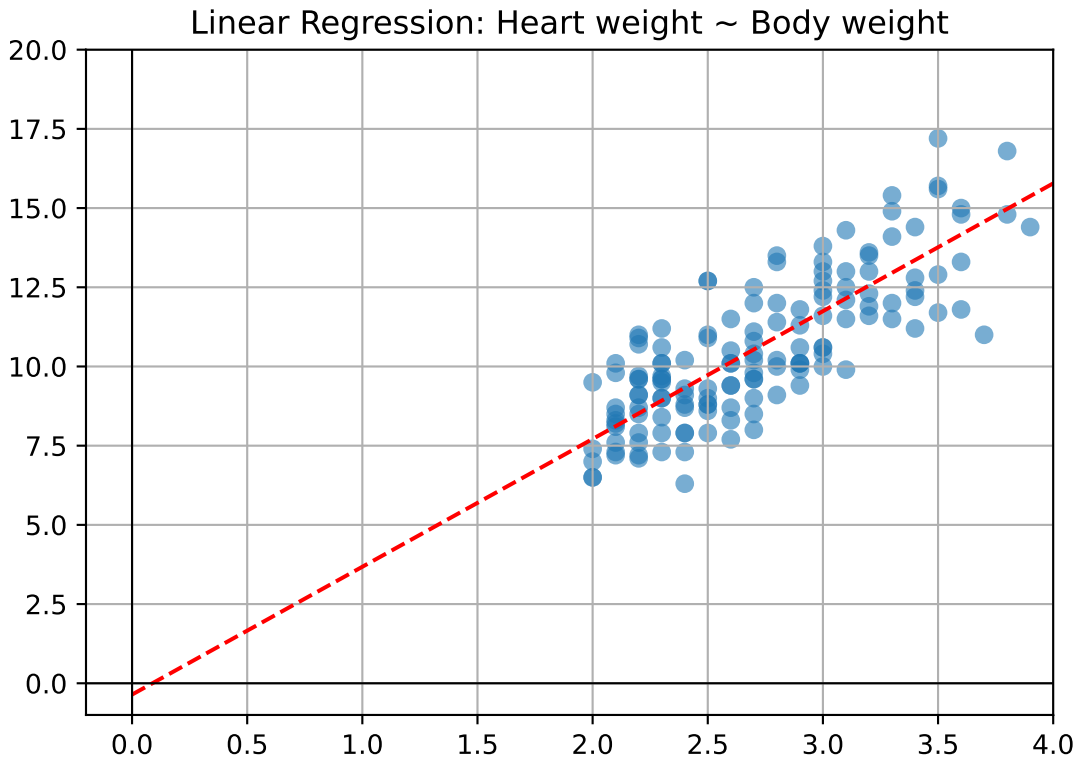
```
## coefficients
print(lm_cats.params)
```

```
Intercept     -0.356662
Body_weight    4.034063
dtype: float64
```

```
## Clean figure object
plt.clf()

## Scatter plot for the observed data
plt.scatter(d_cats['Body_weight'], d_cats['Heart_weight'], alpha = 0.6)

## Set limits for the plot
plt.ylim(-1, 20)
```

```
plt.xlim(-0.2, 4)
```

```
plt.grid(True)


## Create data set for regression line
lo = 0
hi = 4
n = 20
x = [(hi - lo) / (n - 1) * i for i in range(n)]
x_array = np.array(x)
intercept, slope = lm_cats.params

## Plot regression line
plt.plot(x_array, intercept + slope * x_array,
         color = "red", linestyle = "dashed")

plt.axvline(0, color = "black", linewidth = 0.8)
plt.axhline(0, color = "black", linewidth = 0.8)

plt.title("Linear Regression: Heart weight ~ Body weight")

## If you want to change axes name
# plt.xlabel("Body weight")
# plt.ylabel("Heart weight")
```



```
## Let's zoom in and concentrate on the two regression coefficients
##
## Divide the plotting region into two
fig, axes = plt.subplots(1, 2, figsize = (12, 6))

## Set common title
fig.suptitle("Regression coefficients of the 'lm.cats' model")

############################
### First plot: Intercept ###
ax1 = axes[0]
ax1.set_title(f"Intercept: {intercept:.2f}")


############################
## 2) Add grid and regression line
ax1.grid(True)
ax1.axhline(0, color = "black", linewidth = 0.8)
ax1.axvline(0, color = "black", linewidth = 0.8)
ax1.plot(x_array, intercept + slope * x_array,
         color = "red", linestyle = "dashed")

############################
## 3) Add arrow for intercept
ax1.annotate("", xy = (0, intercept), xytext = (0, 0),
             arrowprops = dict(facecolor = 'blue',
             edgecolor = 'blue', linewidth = 1))


## zoom in
# ax1.set_xlim(-0.01, 0.2)
# ax1.set_ylim(-0.5, 2)
ax1.set_xlim(-0.05, 0.55)
```

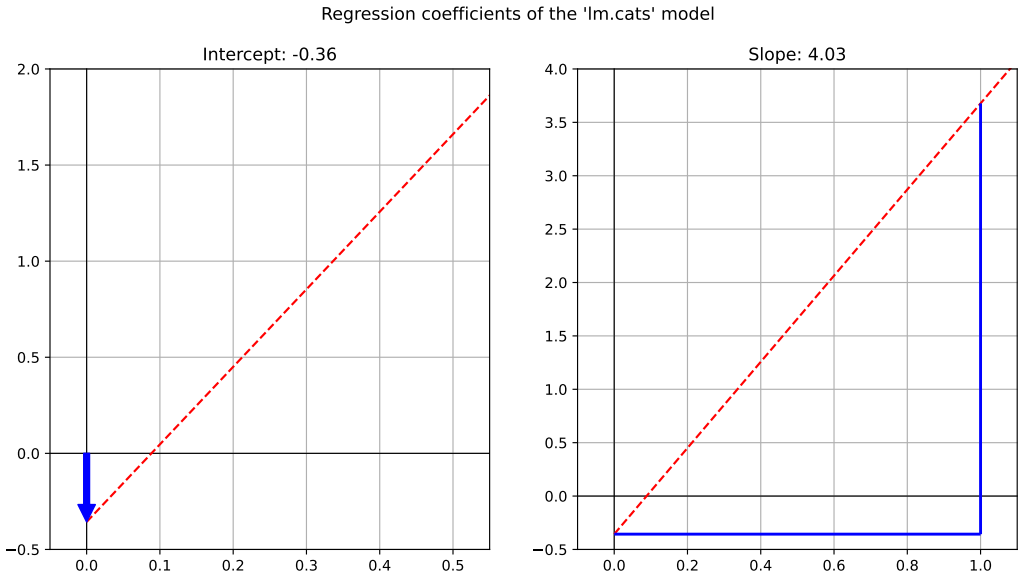
```
ax1.set_ylim(-0.5, 2)
```

```
############################
### Second plot: Slope ###
ax2 = axes[1]

ax2.set_title(f"Slope: {slope:.2f}")

## Add grid and regression line
ax2.grid(True)
ax2.axhline(0, color = "black", linewidth = 0.8)
ax2.axvline(0, color = "black", linewidth = 0.8)
ax2.plot(x_array, intercept + slope * x_array,
         color = "red", linestyle = "dashed")

## Indicate slope with segments
ax2.hlines(y = intercept, xmin = 0, xmax = 1, color = "blue", linewidth = 2)
ax2.vlines(x = 1, ymin = intercept,
           ymax = intercept + slope, color = "blue", linewidth = 2)
```

```
##########################
## zoom in
ax2.set_xlim(-0.1, 1.1)
```

```
ax2.set_ylim(-0.5, 4)
```

```
## Plot it
plt.show()
```



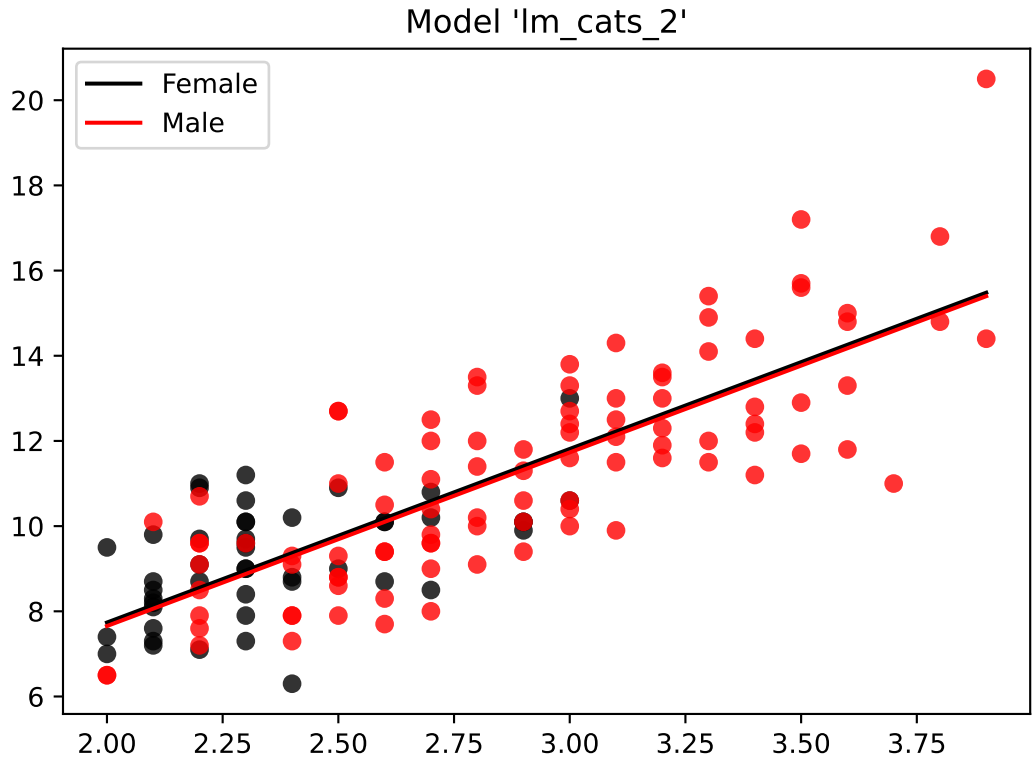
4.1 Including the *Sex* predictor

```
lm_cats_2 = smf.ols('Heart_weight ~ Body_weight + Sex', data = d_cats).fit()
print(lm_cats_2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Heart_weight   R-squared:                       0.647
Model:                            OLS   Adj. R-squared:                  0.642
Method:                 Least Squares   F-statistic:                     129.1
Date:                Thu, 11 Sep 2025   Prob (F-statistic):           1.37e-32
Time:                        14:40:05   Log-Likelihood:                -257.02
No. Observations:                 144   AIC:                             520.0
Df Residuals:                     141   BIC:                             529.0
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      -0.4150      0.727     -0.571      0.569      -1.853       1.023
```

```
Sex[T.M]       -0.0821      0.304     -0.270      0.788      -0.683       0.519
Body_weight     4.0758      0.295     13.826      0.000       3.493       4.659
==============================================================================
Omnibus:                        4.665   Durbin-Watson:                   1.581
Prob(Omnibus):                  0.097   Jarque-Bera (JB):                4.245
Skew:                           0.410   Prob(JB):                        0.120
Kurtosis:                       3.192   Cond. No.                         19.6
==============================================================================

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

```
## Clean figure object
plt.clf()

## Visualise regression lines for each sex

## color points based on sex
colors = {'F': 'black', 'M': 'red'}

## Plot the data
plt.scatter(d_cats['Body_weight'],
            d_cats['Heart_weight'],
            c = d_cats['Sex'].map(colors),
            alpha = 0.8)

plt.title("Model 'lm_cats_2'")

# plt.xlabel("Body_weight")
# plt.ylabel("Heart_weight")

## Add regression lines for each sex
intercept_female = lm_cats_2.params['Intercept']
slope2 = lm_cats_2.params['Body_weight']

## Regression line for female
plt.plot(d_cats['Body_weight'], intercept_female + slope2 * d_cats['Body_weight'],
        color = "black", label = "Female")

## Intercept for male
intercept_male = intercept_female + lm_cats_2.params['Sex[T.M]']
## Regression line for male
plt.plot(d_cats['Body_weight'],
         intercept_male + slope2 * d_cats['Body_weight'],
         color = "red", label = "Male")
plt.legend(loc = 'upper left')
```




```
print("Female intercept:", intercept_female)
```

```
Female intercept: -0.41495262809572275
```

```
print("Male intercept:", intercept_male)
```

```
Male intercept: -0.49704946327762756
```

```
## Generate the summary of the linear model as a string
lm_summary = lm_cats_2.summary().as_text()

## Split the summary into lines and extract the desired lines
## (example indices for similar output)
summary_lines = lm_summary.split("\n")
## Adjust indices as necessary for the specific content
desired_lines = summary_lines[11:18]

# Print the extracted lines
for line in desired_lines:
    print(line)
```

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
```

```
Intercept      -0.4150      0.727     -0.571      0.569      -1.853       1.023
Sex[T.M]       -0.0821      0.304     -0.270      0.788      -0.683       0.519
Body_weight     4.0758      0.295     13.826      0.000       3.493       4.659
==============================================================================
```

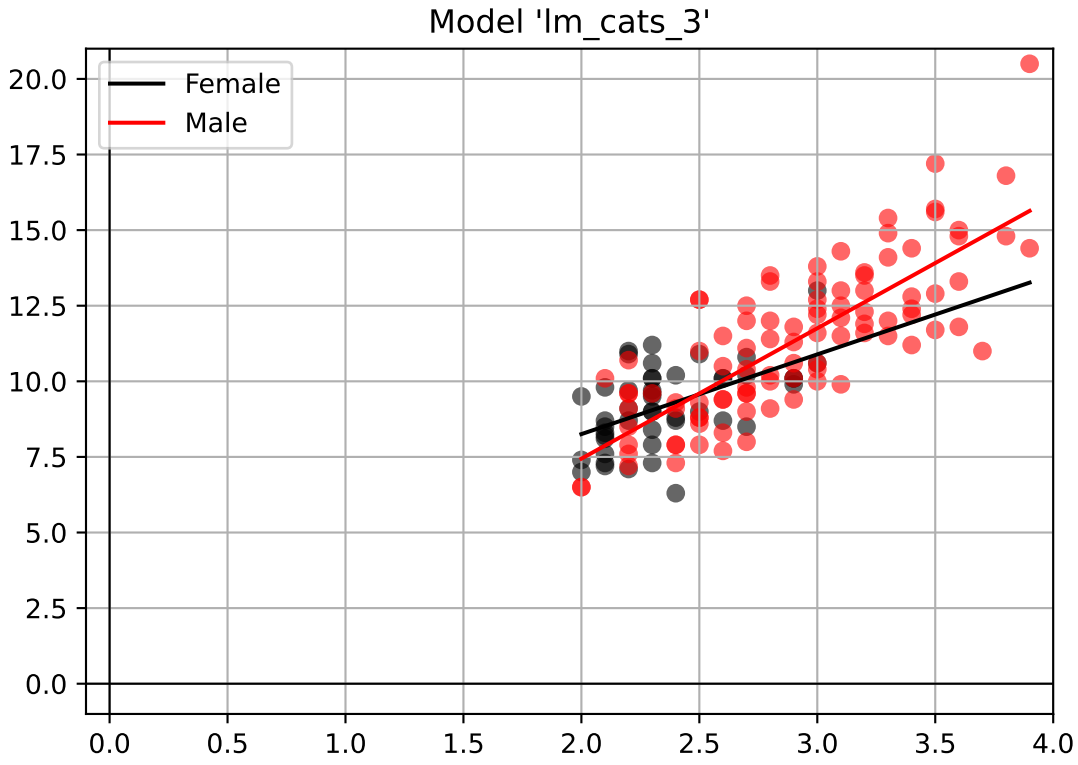
4.2 Include *Sex:Body_weight* interaction

```
lm_cats_3 = smf.ols('Heart_weight ~ Body_weight * Sex', data = d_cats).fit()
```

```
## Clean current figure
plt.clf()

# Visualise regression with interaction
plt.scatter(d_cats['Body_weight'], d_cats['Heart_weight'],
            c = d_cats['Sex'].map(colors), alpha = 0.6)

plt.title("Model 'lm_cats_3'")
plt.xlim(-0.1, 4)
```

```
plt.ylim(-1, 21)
```

```
plt.grid(True)
plt.axhline(0, color = "black", linewidth = 0.8)
plt.axvline(0, color = "black", linewidth = 0.8)

# Add regression lines
intercept = lm_cats_3.params['Intercept']
slope = lm_cats_3.params['Body_weight']

# For females
plt.plot(d_cats['Body_weight'], intercept + slope * d_cats['Body_weight'],
color = "black", label = "Female")

# For males
slope_male = slope + lm_cats_3.params['Body_weight:Sex[T.M]']
intercept_male = intercept + lm_cats_3.params['Sex[T.M]']
plt.plot(d_cats['Body_weight'],
         intercept_male + slope_male * d_cats['Body_weight'],
         color = "red", label = "Male")
plt.legend(loc = 'upper left')
```




```
## Regression coefficients
coefficients_interaction = lm_cats_3.params
print("Regression Coefficients:")
```

```
Regression Coefficients:
```

```
print(coefficients_interaction)
```

```
Intercept               2.981312
Sex[T.M]               -4.165400
Body_weight             2.636414
Body_weight:Sex[T.M]    1.676265
dtype: float64
```

```
## Confidence intervals
confidence_intervals_interaction = lm_cats_3.conf_int()
print("\nConfidence Intervals:")
```

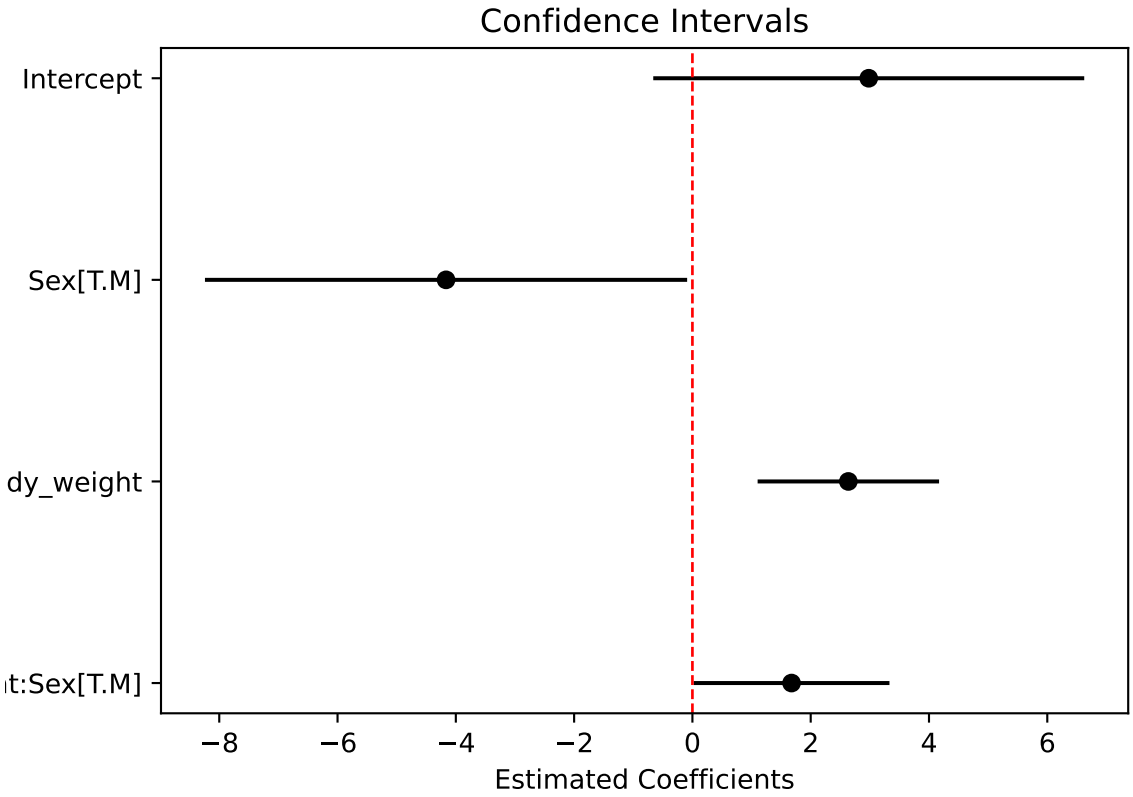
```
Confidence Intervals:
```

```
print(confidence_intervals_interaction)
```

```
                            0         1
Intercept            -0.662080  6.624705
Sex[T.M]             -8.241601 -0.089199
Body_weight           1.102414  4.170414
Body_weight:Sex[T.M]  0.020827  3.331702
```

```
## Clean figure object
plt.clf()

## Visualise confidence intervals
coef_names = lm_cats_3.params.index[::-1]
coef_values = lm_cats_3.params[::-1]
conf_int_values = confidence_intervals_interaction[::-1]

plt.errorbar(coef_values, range(len(coef_values)),
             xerr = [coef_values - conf_int_values[0],
             conf_int_values[1] - coef_values],
             fmt = 'o', color = 'black')
plt.axvline(0, linestyle = "dashed", color = "red", linewidth = 1)
plt.yticks(range(len(coef_values)), coef_names)
```

```
plt.xlabel("Estimated Coefficients")
plt.title("Confidence Intervals")
```



5 Appendix

```
## Model with no interaction
print("Model with no interaction formula:")
```

```
Model with no interaction formula:
```

```
print(lm_cats_2.model.formula)
```

```
Heart_weight ~ Body_weight + Sex
```

```
r_squared_no_interaction = lm_cats_2.rsquared
print("\nR-squared (no interaction):")
```

```
R-squared (no interaction):
```

```
print(r_squared_no_interaction)
```

```
0.6468035413329445
```

```
## Model with interaction
print("\nModel with interaction formula:")
```

```
Model with interaction formula:
```

```
print(lm_cats_3.model.formula)
```

```
Heart_weight ~ Body_weight * Sex
```

```
r_squared_interaction = lm_cats_3.rsquared
print("\nR-squared (with interaction):")
```

```
R-squared (with interaction):
```

```
print(r_squared_interaction)
```

```
0.6566329434116425
```

```
# Adjusted R-squared for the model with no interaction
adj_r_squared_no_interaction = lm_cats_2.rsquared_adj
print("Adjusted R-squared (no interaction):")
```

```
Adjusted R-squared (no interaction):
```

```
print(adj_r_squared_no_interaction)
```

```
0.6417936624866032
```

```
# Adjusted R-squared for the model with interaction
adj_r_squared_interaction = lm_cats_3.rsquared_adj
print("\nAdjusted R-squared (with interaction):")
```

```
Adjusted R-squared (with interaction):
```

```
print(adj_r_squared_interaction)
```

```
0.6492750779133205
```

5.1 Fitted values

```
## Get fitted values for the model
fitted_cats = lm_cats.fittedvalues

# Display the first few fitted values (similar to head() in R)
print("\nFirst few fitted values:")
```

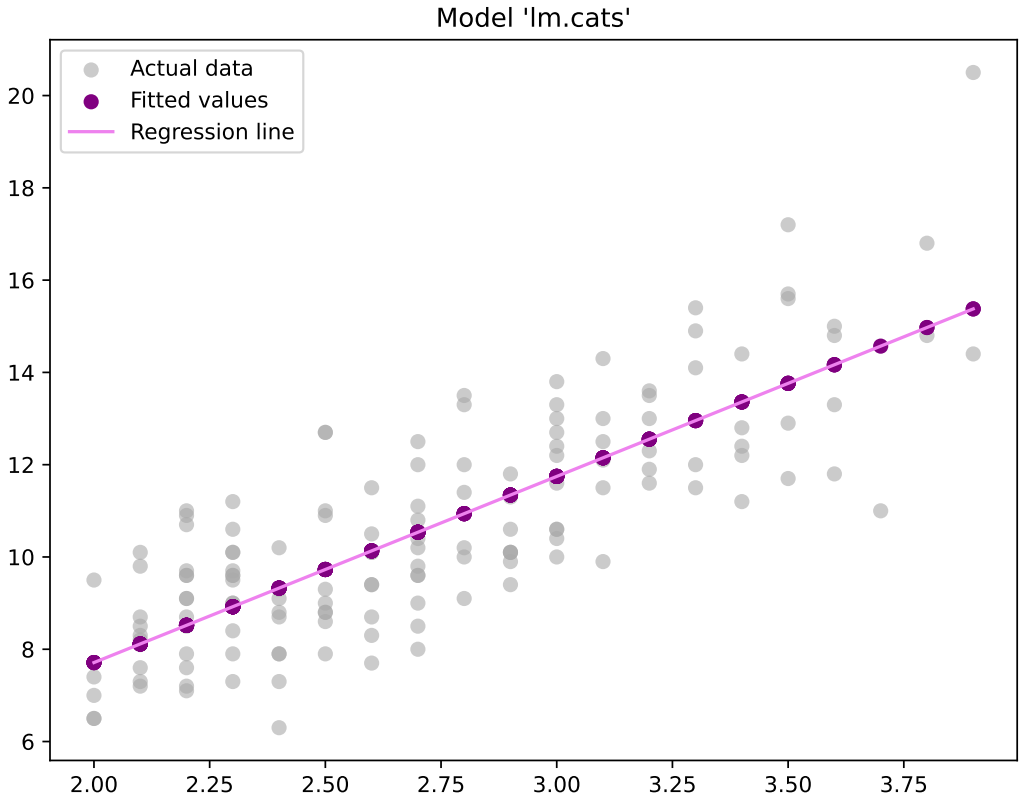
```
First few fitted values:
```

```
print(fitted_cats.head())
```

```
0    7.711463
1    7.711463
2    7.711463
3    8.114869
4    8.114869
dtype: float64
```

```
## Clean figure object
plt.clf()

## Plot the actual data points (Heart_weight vs. Body_weight)
plt.figure(figsize = (8, 6))
plt.scatter(d_cats['Body_weight'], d_cats['Heart_weight'],
            color = 'darkgray', label = "Actual data", alpha = 0.6)

## Plot the fitted values (predicted values) from the model (lm_cats)
plt.scatter(d_cats['Body_weight'], fitted_cats,
            color = 'purple', label = "Fitted values", zorder = 5)

## Add the regression line
```

```
plt.plot(d_cats['Body_weight'], lm_cats.fittedvalues,
         color = 'violet', label = "Regression line", zorder = 10)

## Add title and labels
plt.title("Model 'lm.cats'")
# plt.xlabel("Body weight")
# plt.ylabel("Heart weight")

## Show legend
plt.legend()
```



5.2 Residuals

```
# Residuals are the difference between the actual values and the fitted values
resid_cats = d_cats['Heart_weight'] - lm_cats.fittedvalues

# Length of the residuals
len_resid_cats = len(resid_cats)

# First few residuals
head_resid_cats = resid_cats.head()
```

```python
# Output the results
print("Length of residuals:", len_resid_cats)
```

```
Length of residuals: 144
```

```python
print("First few residuals:\n", head_resid_cats)
```

```
First few residuals:
 0   -0.711463
1   -0.311463
2    1.788537
3   -0.914869
4   -0.814869
dtype: float64
```

```python
# Set seed for reproducibility
np.random.seed(20)

# Sample 5 random indices from the range 1 to 144 (Python uses 0-indexing)
sample_indices = np.random.choice(range(144), size = 5, replace = False)

# Get the corresponding residuals and fitted values using the sampled indices
sample_residuals = resid_cats.iloc[sample_indices]
sample_fitted = fitted_cats.iloc[sample_indices]

# Output the results
print("Sampled residuals:\n", sample_residuals)
```

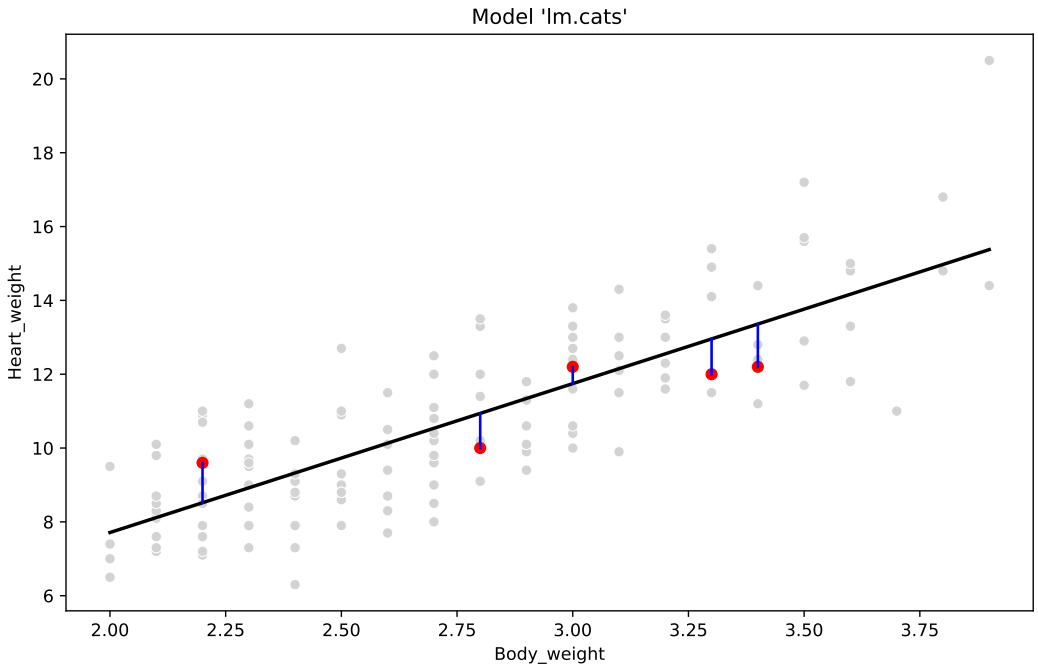
```
Sampled residuals:
 88    -0.938713
126   -1.159151
56     1.081724
103    0.454474
121   -0.955744
dtype: float64
```

```python
print("Sampled fitted values:\n", sample_fitted)
```

```
Sampled fitted values:
 88     10.938713
126    13.359151
56      8.518276
103    11.745526
121    12.955744
dtype: float64
```

```python
## Clean figure object
plt.clf()

# Create the plot
```

```
plt.figure(figsize = (10, 6))

# 1. Plot Heart_weight vs Body_weight in light gray
sns.scatterplot(data = d_cats, x = 'Body_weight', y = 'Heart_weight',
                color = 'lightgray')

# 2. Plot the regression line
plt.plot(d_cats['Body_weight'], fitted_cats, color = 'black', lw = 2)

# 3. Plot red points for the selected indices
plt.scatter(d_cats.iloc[sample_indices]['Body_weight'],
d_cats.iloc[sample_indices]['Heart_weight'], color = 'red')

# 4. Add blue segments (difference between actual and fitted values)
for i in sample_indices:
    plt.plot([d_cats.iloc[i]['Body_weight'], d_cats.iloc[i]['Body_weight']],
             [fitted_cats[i], d_cats.iloc[i]['Heart_weight']], color = 'blue')

# Set the title
plt.title("Model 'lm.cats'")
```

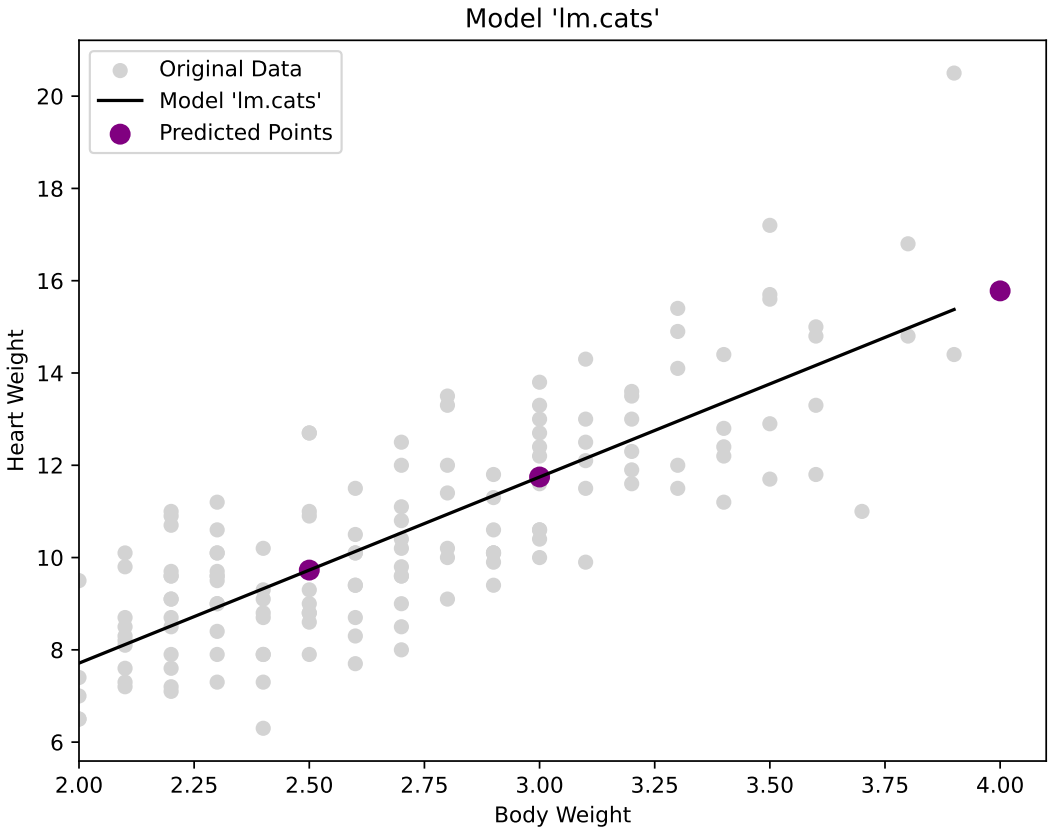


5.3 Predicted values

```
## Clean figure object
plt.clf()

## Fit the model
```

```
lm_cats = smf.ols('Heart_weight ~ Body_weight', data = d_cats).fit()

## Plot the data with model fit
plt.figure(figsize = (8, 6))
plt.scatter(d_cats['Body_weight'], d_cats['Heart_weight'],
            color = 'lightgray', label = 'Original Data')
plt.plot(d_cats['Body_weight'], lm_cats.fittedvalues,
         color = 'black', label = "Model 'lm.cats'")
plt.title("Model 'lm.cats'")
plt.xlabel('Body Weight')
plt.ylabel('Heart Weight')


## New data and predictions
new_data_cats = pd.DataFrame({'Body_weight': [4, 2.5, 3]})
pred_new_cats = lm_cats.predict(new_data_cats)

## Add predictions to the plot
plt.scatter(new_data_cats['Body_weight'], pred_new_cats,
            color = 'purple', s = 75, label = 'Predicted Points')

## Adjust plot limits and show
plt.xlim(2, 4.1)
```

```
plt.legend()
```




```
## Predictions for new data
new_data_cats = pd.DataFrame({'Body_weight': [4, 2.5, 3]})
pred_new_cats = lm_cats.predict(new_data_cats)
print(pred_new_cats)
```

```
0    15.779588
1     9.728494
2    11.745526
dtype: float64
```

```
## Predictions with confidence intervals
# pred_new_cats_ci = lm_cats.get_prediction(new_data_cats).summary_frame()
pred_new_cats_ci = lm_cats.get_prediction(new_data_cats).conf_int(alpha = 0.05)
print(pred_new_cats_ci)
```

```
[[15.10432673 16.45484999]
 [ 9.46490162  9.99208701]
 [11.46995403 12.02109729]]
```

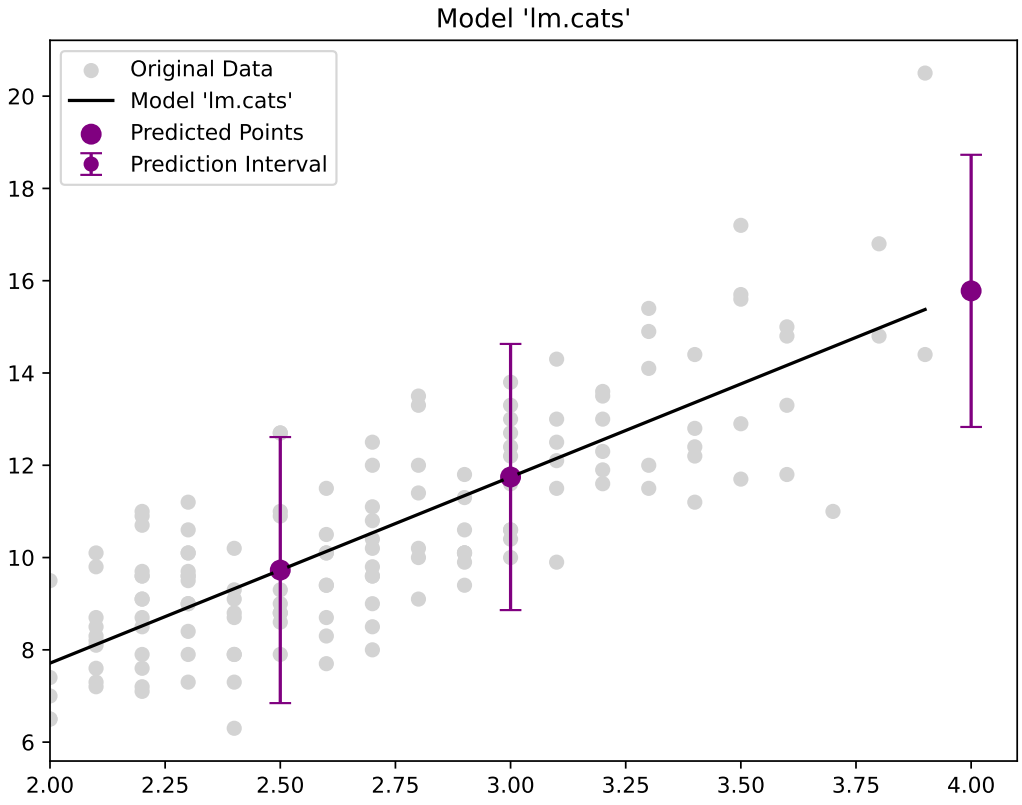
```
## Predictions with confidence intervals
# pred_new_cats_ci = lm_cats.get_prediction(new_data_cats).summary_frame()
pred_new_cats_ci = lm_cats.get_prediction(new_data_cats).conf_int(alpha = 0.05,
                                                                  obs = True)
print(pred_new_cats_ci)
```

```
[[12.83018038 18.72899634]
 [ 6.84535188 12.61163675]
 [ 8.86126338 14.62978794]]
```

```
## Clean figure object
plt.clf()

## Plot the data with model fit
plt.figure(figsize = (8, 6))
```

```
plt.scatter(d_cats['Body_weight'], d_cats['Heart_weight'],
            color = 'lightgray', label = 'Original Data')
plt.plot(d_cats['Body_weight'], lm_cats.fittedvalues,
         color = 'black', label = "Model 'lm.cats'")
plt.title("Model 'lm.cats'")
# plt.xlabel('Body Weight')
# plt.ylabel('Heart Weight')

## New data and predictions
new_data_cats = pd.DataFrame({'Body_weight': [4, 2.5, 3]})
pred_new_cats = lm_cats.predict(new_data_cats)

## Compute prediction intervals
pred_new_cats_ci = lm_cats.get_prediction(new_data_cats).conf_int(alpha = 0.05,
                                                                  obs = True)
lower_bounds = pred_new_cats_ci[:, 0]
upper_bounds = pred_new_cats_ci[:, 1]

## Add predictions to the plot
plt.scatter(new_data_cats['Body_weight'], pred_new_cats,
            color = 'purple', s = 75, label = 'Predicted Points')

## Add prediction intervals as error bars
plt.errorbar(new_data_cats['Body_weight'], pred_new_cats,
             yerr=[pred_new_cats - lower_bounds, upper_bounds - pred_new_cats],
             fmt='o', color='purple', capsize=5, label='Prediction Interval')

## Adjust plot limits and show
plt.xlim(2, 4.1)
```

```
plt.legend()
```



5.4 Treatment contrasts

```
## Set the first 10 observations to 'Unknown'
d_cats.loc[:9, 'Sex'] = 'unknown'
```

```
lm_cats_Newsex = smf.ols('Heart_weight ~ Sex', data = d_cats).fit()
```

```
## Regression coefficients
coefficients_unkown = lm_cats_Newsex.params
print("Regression Coefficients:")
```

```
Regression Coefficients:
```

```
print(coefficients_unkown)
```

```
Intercept         9.551351
Sex[T.M]          1.771329
Sex[T.unknown]   -1.641351
dtype: float64
```

5.5 Changing the reference level

```
## Change the reference level (set 'M' as the reference)
## Before change
d_cats['Sex'] = pd.Categorical(d_cats['Sex'], categories = ['M', 'F', 'unknown'])
```

```
## After change
d_cats['Sex'] = d_cats['Sex'].cat.reorder_categories(['M', 'F', 'unknown'],
                                                     ordered = True)
```

```
lm_cats_Newsex2 = smf.ols('Heart_weight ~ Sex', data = d_cats).fit()
```

```
## Regression coefficients
coefficients = lm_cats_Newsex2.params
print("Regression Coefficients:")
```

```
Regression Coefficients:
```

```
print(coefficients)
```

```
Intercept         11.322680
Sex[T.F]          -1.771329
Sex[T.unknown]    -3.412680
dtype: float64
```

5.6 How are regression coefficients estimated?

```
print(f"Formula: {lm_cats.model.formula}")
```

```
Formula: Heart_weight ~ Body_weight
```

```
print("\nCoefficients:")
```

```

Coefficients:
```

```
print(lm_cats.params)
```

```
Intercept     -0.356662
Body_weight    4.034063
dtype: float64
```

```
## Create dummy variables for 'Sex'
X = pd.get_dummies(d_cats[['Body_weight', 'Sex']], drop_first = True)

## # Add constant for intercept
X = sm.add_constant(X)
X_matrix = X.values
print("Shape of X (design matrix):", X_matrix.shape)
```

```
Shape of X (design matrix): (144, 4)
```

```
print("\nFirst few rows of X (design matrix):")
```

```
First few rows of X (design matrix):
```

```
print(X_matrix[:5])  # Display the first 5 rows
```

```
[[1.0 2.0 False True]
 [1.0 2.0 False True]
 [1.0 2.0 False True]
 [1.0 2.1 False True]
 [1.0 2.1 False True]]
```