codecademy

Multiple Linear Regression

Multiple Linear Regression

Multiple linear regression is an extension of simple linear regression used to model the relationship between a quantitative response variable and two or more predictors, which may be quantitative, categorical, or a mix of both. This allows us to control for confounding variables, which may distort the perceived relationship between two variables if not accounted for.

Multiple Regression Coefficient

Instead of a single slope, the multiple linear regression equation has a “slope,” called a partial regression coefficient, for each predictor. For example, a multiple linear regression equation predicting sales from the predictors temperature and day might look like:

$$sales = -256.8 + 31.5 * temperatur$$

The “slopes” are 31.5 and -22.3, while -256.8 is the intercept.

Multiple Regression in Python

Multiple linear regression models can be implemented in Python using the statsmodels function OLS.from_formula() and adding each additional predictor to the formula preceded by a + . For example, the example code shows how we could fit a model predicting income from variables for age, highest education completed, and region.

```
import statsmodels.api as sm

model = sm.OLS.from_formula('income ~ age
+ highest_edu + region', data).fit()
```

codecademy

Binary Predictors in Multiple Regression

Suppose that we fit a multiple regression model and calculate the following regression equation:

$$sales = 250 - 50 * rain + 2 * temper$$

If `rain` is a binary categorical variable that is equal to `1` when it rains and `0` when it does not rain, we can write the following two regression equations:
When it rains:

$$sales = 200 + 2 * temperature$$

When it doesn't rain:

$$sales = 250 + 2 * temperature$$

Therefore, the coefficient on `rain` (`-50`) is the difference in expected sales for rain days compared to non-rain days.

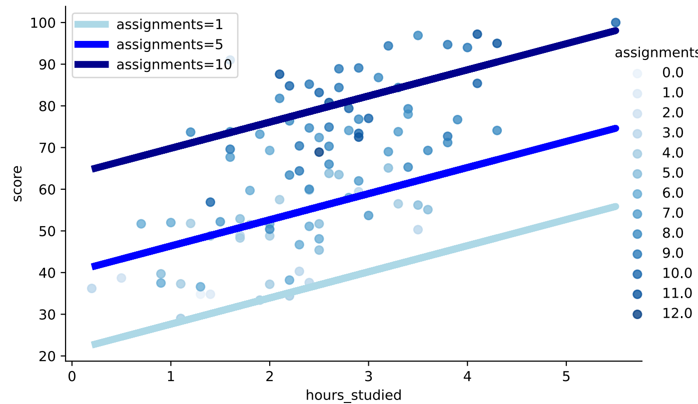
Multiple Linear Regression Interpretation

In multiple linear regression:

- The intercept is the expected value of the response variable when all predictors equal zero.
- The slope coefficient on each predictor is the expected difference in the outcome variable for a one-unit increase of the predictor, holding all other predictors constant.

Visualizing a Multiple Regression Model

We can visualize and understand multiple linear regression as creating a new regression equation for each value of a predictor. For example, the provided image shows a visualization of the following regression model: `'score = hours_studied + assignments'` . In the plot, there are three regression lines, each for a different value of `assignments` .



Coefficients in Multiple Regression

Suppose that we fit a regression model to predict `sales` using `temperature` as a predictor. If we fit another model predicting `sales` using both `temperature` and `rain` as predictors, the coefficient on `temperature` will likely be different in the two models.

Multicollinearity Assumption

The assumptions for multiple regression are the same as for simple linear regression, except for the additional assumption that the predictors are not highly correlated with one another (no multicollinearity).

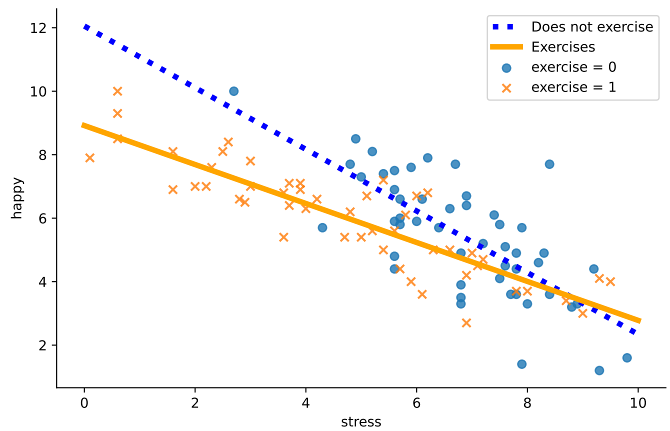
Multicollinearity in Python

Multicollinearity can be checked visually in Python using seaborn's `heatmap()` function. The provided code shows how we can create a heat map of quantitative variable correlations for a dataset called `flowers` .

```
import seaborn as sns
# get table of variable correlations
var_corr = flowers.corr()
# plot the heatmap
sns.heatmap(var_corr,
xticklabels=var_corr.columns,
yticklabels=var_corr.columns, annot=True)
plt.show()
```

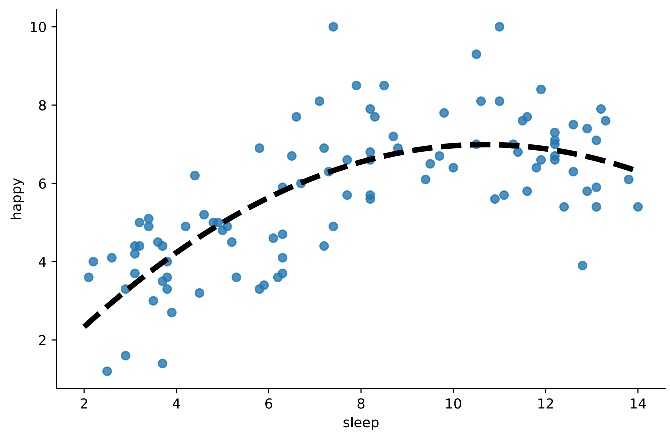
Interaction Terms

In multiple linear regression, we can use an interaction term when the relationship between two variables is moderated by a third variable. This allows the slope coefficient for one variable to vary depending on the value of the other variable.
For example, this scatter plot shows happiness level on the y-axis against stress level on the x-axis. If we fit a model to predict `happy` using `stress`, `exercise`, and `stress:exercise` (an interaction between `stress` and `exercise`) as predictors, we can calculate the pictured regression lines (one for `exercise = 0` and one for `exercise = 1`), which each have a different slope to model the relationship between `stress` and `happy`.



Polynomial Terms

In multiple linear regression, we can use a polynomial term to model non-linear relationships between variables. For example, this plot shows a curved relationship between `sleep` and `happy`, which could be modeled using a polynomial term.
The coefficient on a polynomial term can be difficult to interpret directly; however, the picture is useful. In this example, we see that more sleep is associated with higher happiness levels up to some point, after which more sleep is associated with lower happiness.



Interactions in Python

In the Python library `statsmodels.api`, interaction terms can be added to a multiple regression model formula by adding a term that has both predictors with a colon between them. For example, to fit a multiple regression model predicting `income` from the variables `age`, `region`, and the interaction of `age` and `region`, we could use the example code shown here. This creates a new predictor, which is the product of `age` and `religion`.

```python
import statsmodels.api as sm
model = sm.OLS.from_formula('income ~ age
+ region + age:region', data).fit()
```

Polynomial Terms in Python

In the Python library `statsmodels.api` , polynomial terms can be added to a multiple linear regression model formula by adding a term with the predictor of interest raised to a higher power. To do this, we use the NumPy function `np.power()` and specify the predictor name and degree. For example, the example code shows how to fit a model predicting `nights` from the variable `trip_length` with a quadratic (squared) term for `trip_length` .

```
import statsmodels.api as sm
import numpy as np

model = sm.OLS.from_formula('nights ~ trip_length + np.power(trip_length,2)', data).fit()
```

codecademy

Interactions with Binary and Quantitative

Consider the following multiple regression equation, where `rain` is equal to 1 if it rained and 0 otherwise:

$$sales = 300 + 34 * temperature - 49$$

On days where `rain` = 0, the regression equation becomes:

$$sales = 300 + 34 * temperature$$

On days where `rain` = 1, the regression equation becomes:

$$sales = 251 + 36 * temperature$$

Therefore, the coefficient on `rain` (`-49`) means that the intercept for rain days is 49 units lower than for non-rain days. Meanwhile, the slope on `temperature:rain` (`2`) means that the slope on `temperature` is 2 units higher for rain days than for non-rain days.

Interactions with Two Quantitative

Consider the following multiple regression equation:

$$sales = 300 + 4 * temp + 3 * humidit$$

On days where `humidity` = 0, the regression equation becomes:

$$sales = 300 + 4 * temp$$

On days where `humidity` = 1, the regression equation becomes:

$$sales = 303 + 6 * temp$$

Therefore, the coefficient on `humidity` (`3`) means that the intercept increases by 3 for every additional unit of humidity. Meanwhile, the slope on `temp:humidity` (`2`) means that the slope on `temp` is 2 units higher for every additional unit of humidity.

Polynomials in Regression Equation

In a multiple linear regression equation, a polynomial term will appear as the predictor raised to a higher exponent (such as squared, cubed, to the 4th, etc.). The output of a multiple linear regression predicting `nights` from the variable `trip_length` and the square of `trip_length` is shown.

```
# Output:
# Intercept                  4.6
# trip_length                6.3
# np.power(trip_length,2)   -0.4
```

Based on this output, the regression equation would have a term that raises `trip_length` to the second power:

$$nights = 4.6 + 6.3 * trip\_length - 0.$$

Print   Share