Multiple logistic regression

INTERMEDIATE REGRESSION WITH STATSMODELS IN PYTHON

Maarten Van den Broeck
Content Developer at DataCamp

Bank churn dataset

has_churned	time_since_first_purchase	time_since_last_purchase
0	0.3993247	-0.5158691
1	-0.4297957	0.6780654
0	3.7383122	0.4082544
0	0.6032289	-0.6990435
...	...	...
response	*length of relationship*	*recency of activity*

[1] https://www.rdocumentation.org/packages/bayesQR/topics/Churn

logit()

```
from statsmodels.formula.api import logit

logit("response ~ explanatory", data=dataset).fit()
```

```
logit("response ~ explanatory1 + explanatory2", data=dataset).fit()
```

```
logit("response ~ explanatory1 * explanatory2", data=dataset).fit()
```

The four outcomes

	predicted false	predicted true
actual false	correct	false positive
actual true	false negative	correct

```
conf_matrix = mdl_logit.pred_table()
```

```
print(conf_matrix)
```

```
[[102.  98.]
 [ 53. 147.]]
```

Prediction flow

```
from itertools import product

explanatory1 = some_values
explanatory2 = some_values

p = product(explanatory1, explanatory2)
explanatory_data = pd.DataFrame(p,
                                columns=["explanatory1",
                                         "explanatory2"])
prediction_data = explanatory_data.assign(
  mass_g = mdl_logit.predict(explanatory_data))
```

Visualization

```
prediction_data["most_likely_outcome"] = np.round(prediction_data["has_churned"])

sns.scatterplot(...
                data=churn,
                hue="has_churned",
                ...)

sns.scatterplot(...
                data=prediction_data,
                hue="most_likely_outcome",
                ...)
```

Let's practice!

INTERMEDIATE REGRESSION WITH STATSMODELS IN PYTHON

The logistic distribution

INTERMEDIATE REGRESSION WITH STATSMODELS IN PYTHON

Maarten Van den Broeck
Content Developer at DataCamp

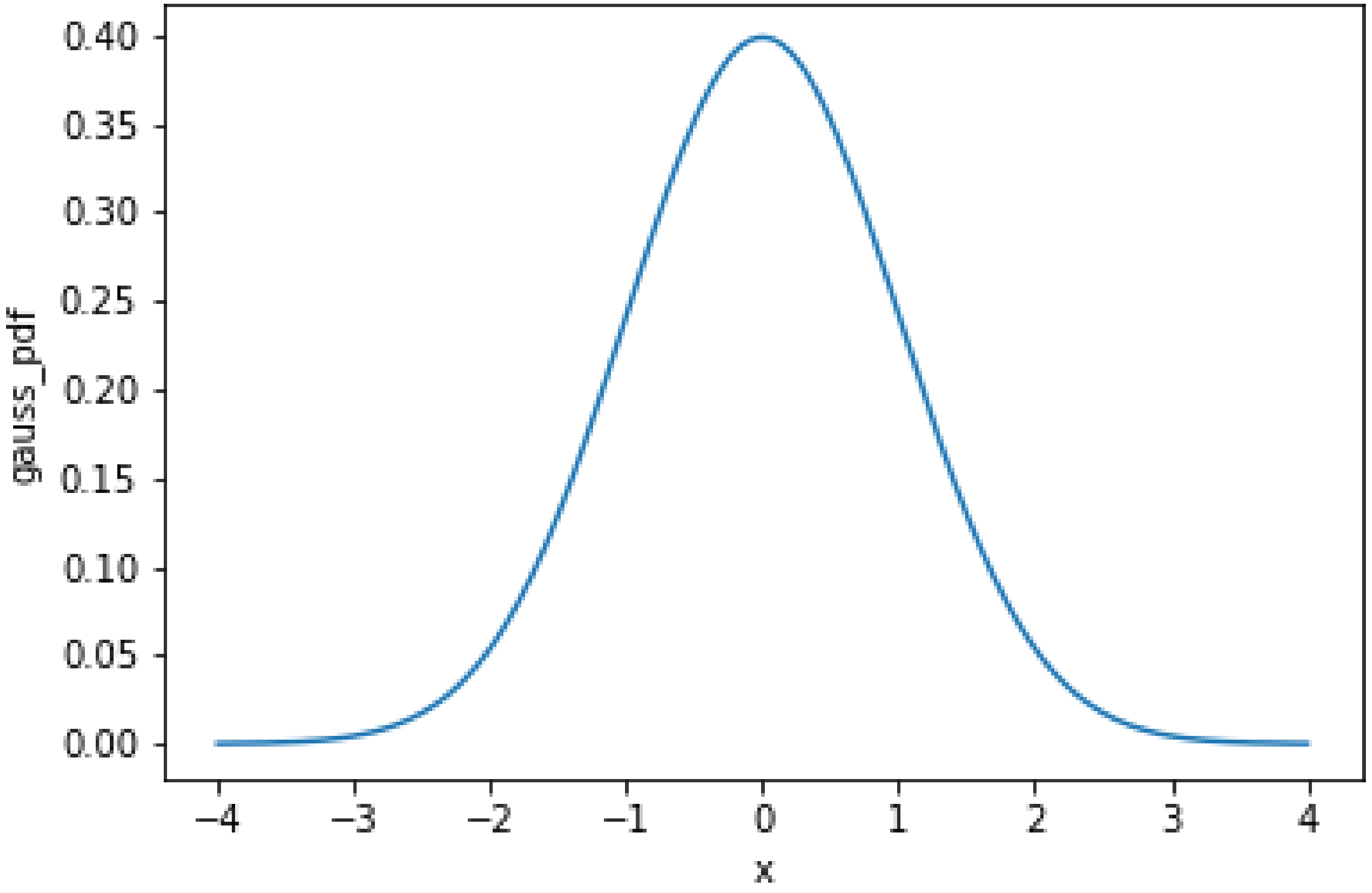
Gaussian probability density function (PDF)

```
from scipy.stats import norm

x = np.arange(-4, 4.05, 0.05)

gauss_dist = pd.DataFrame({
  "x": x,
  "gauss_pdf": norm.pdf(x)}
)
```

```
sns.lineplot(x="x",
             y="gauss_pdf",
             data=gauss_dist)
```

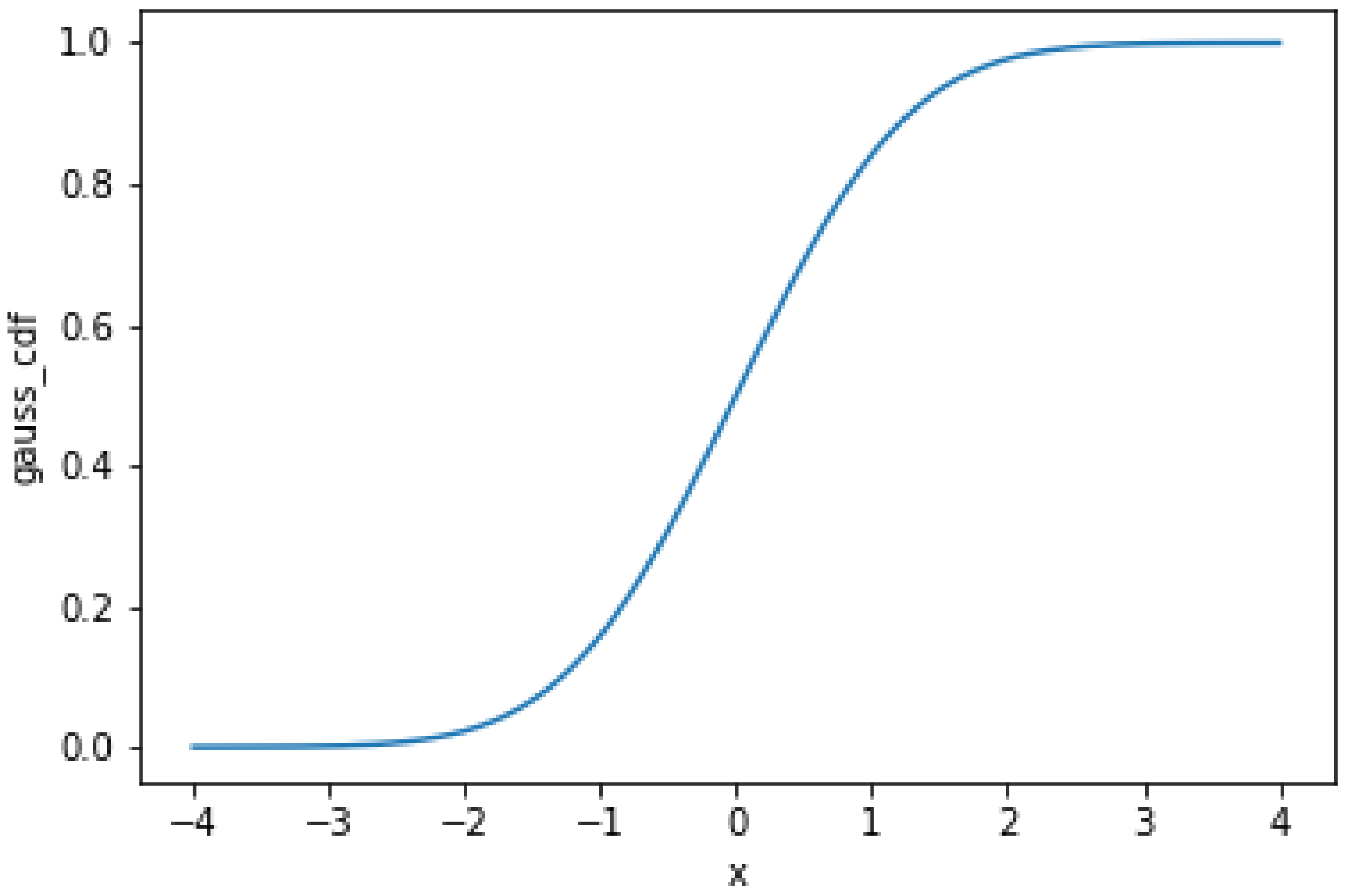


Gaussian cumulative distribution function (CDF)

```
x = np.arange(-4, 4.05, 0.05)

gauss_dist = pd.DataFrame({
  "x": x,
   "gauss_pdf": norm.pdf(x),
   "gauss_cdf": norm.cdf(x)}
)
```

```
sns.lineplot(x="x",
             y="gauss_cdf",
             data=gauss_dist)
```

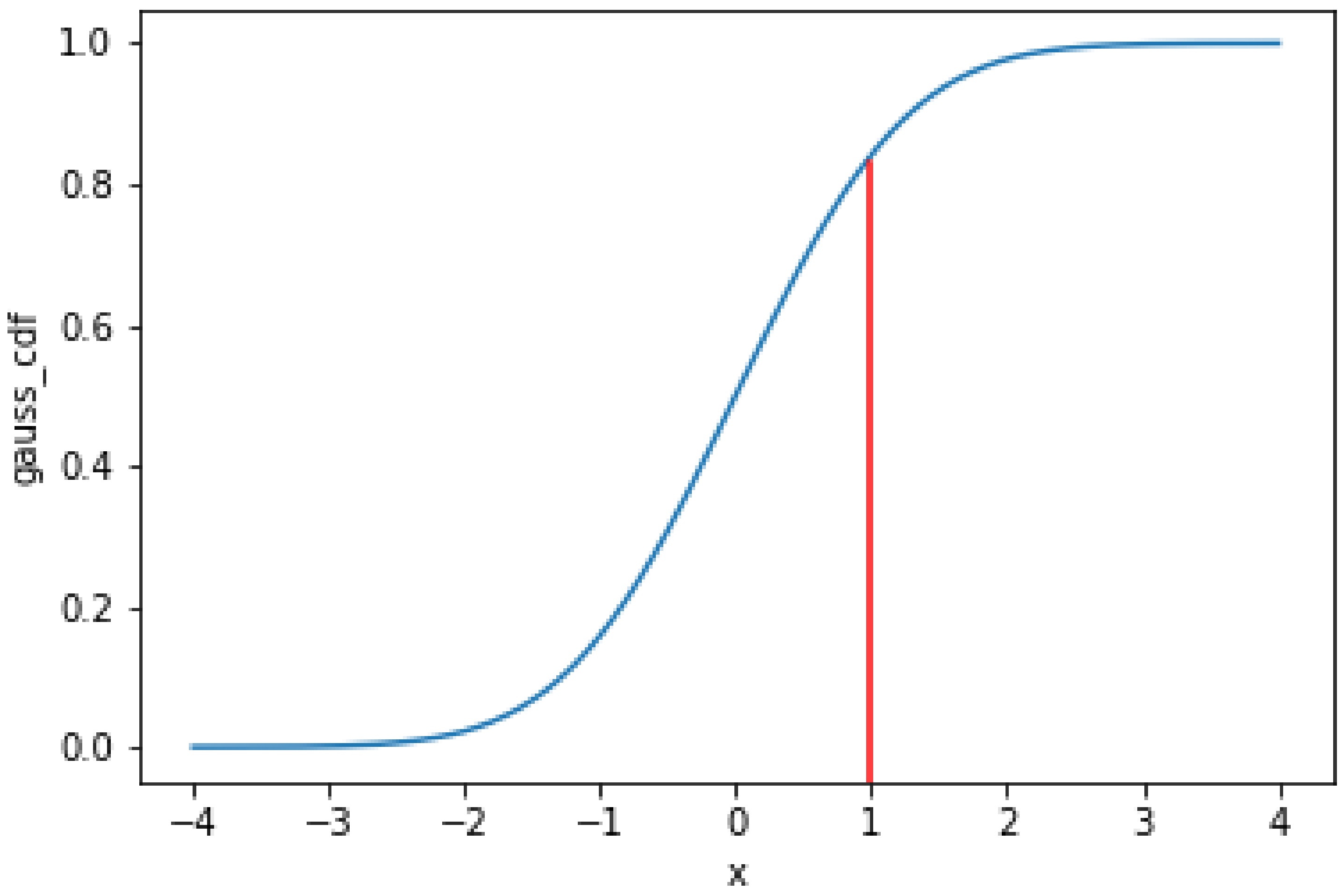


Gaussian cumulative distribution function (CDF)

```
x = np.arange(-4, 4.05, 0.05)

gauss_dist = pd.DataFrame({
  "x": x,
   "gauss_pdf": norm.pdf(x),
   "gauss_cdf": norm.cdf(x)}
)
```

```
sns.lineplot(x="x",
             y="gauss_cdf",
             data=gauss_dist)
```

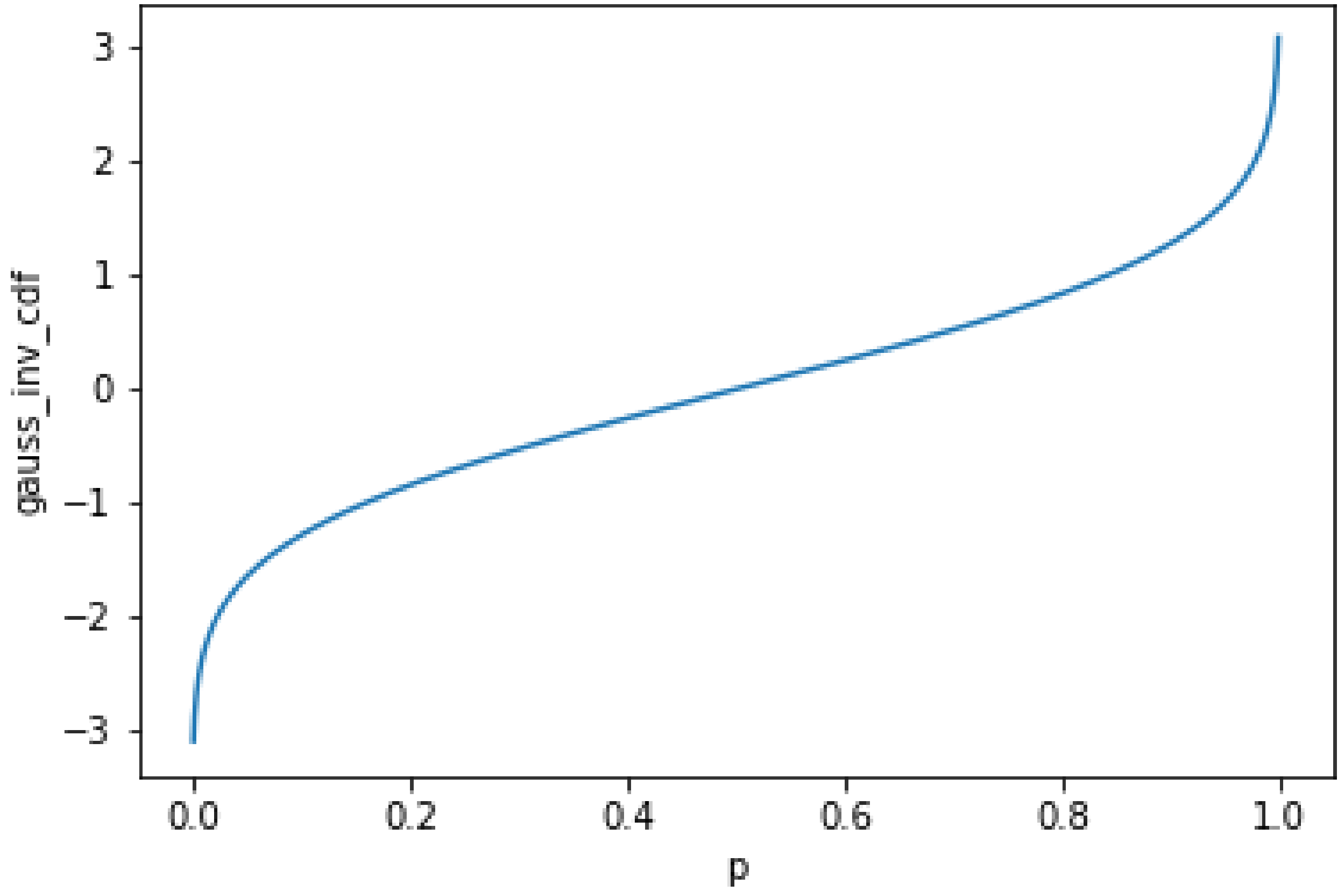


Gaussian inverse CDF

```
p = np.arange(0.001, 1, 0.001)

gauss_dist_inv = pd.DataFrame({
  "p": p,
  "gauss_inv_cdf": norm.ppf(p)}
)
```

```
sns.lineplot(x="p",
             y="gauss_inv_cdf",
             data=gauss_dist_inv)
```



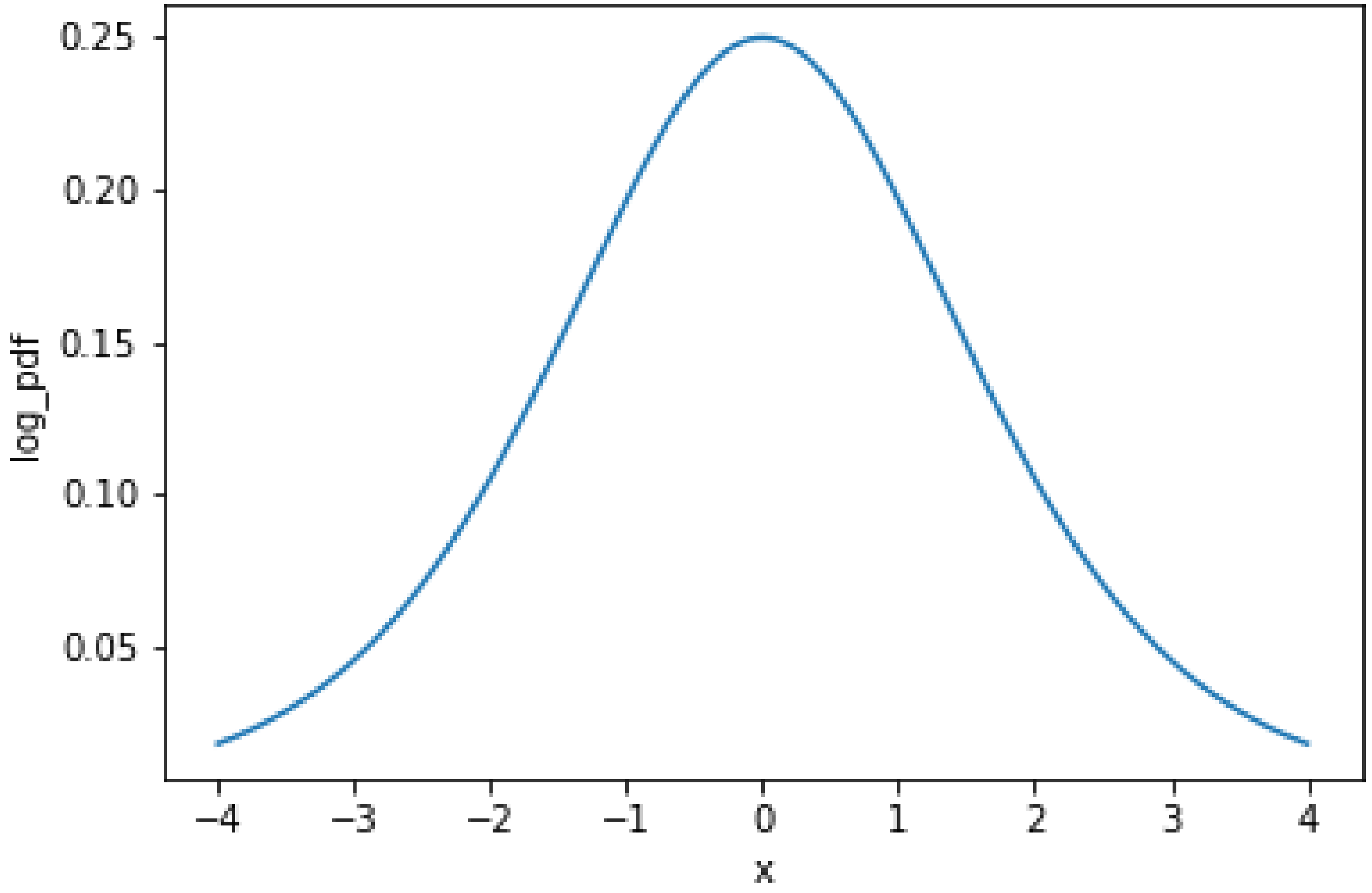
Logistic PDF

```
from scipy.stats import logistic

x = np.arange(-4, 4.05, 0.05)

logistic_dist = pd.DataFrame({
  "x": x,
  "log_pdf": logistic.pdf(x)}
)
```

```
sns.lineplot(x="x",
             y="log_pdf",
             data=logistic_dist)
```



Logistic distribution

- Logistic distribution CDF is also called the *logistic function*.
- $\mathrm{cdf}(x) = \frac{1}{(1+exp(-x))}$
- Logistic distribution inverse CDF is also called the *logit function*.
- $\mathrm{inverse\_cdf}(p) = log(\frac{1}{(1-p)})$

Let's practice!

INTERMEDIATE REGRESSION WITH STATSMODELS IN PYTHON

How logistic regression works

INTERMEDIATE REGRESSION WITH STATSMODELS IN PYTHON

Maarten Van den Broeck
Content Developer at DataCamp

Sum of squares doesn't work

```
np.sum((y_pred - y_actual) ** 2)
```

`y_actual` is always `0` or `1`.

`y_pred` is between `0` and `1`.

There is a better metric than sum of squares.

Likelihood

```
y_pred * y_actual
```

Likelihood

```
y_pred * y_actual + (1 - y_pred) * (1 - y_actual)
```

Likelihood

```
np.sum(y_pred * y_actual + (1 - y_pred) * (1 - y_actual))
```

When `y_actual = 1`

```
y_pred * 1 + (1 - y_pred) * (1 - 1) = y_pred
```

When `y_actual = 0`

```
y_pred * 0 + (1 - y_pred) * (1 - 0) = 1 - y_pred
```

Log-likelihood

- Computing likelihood involves adding many very small numbers, leading to numerical error.
- Log-likelihood is easier to compute.

```
log_likelihood = np.log(y_pred) * y_actual + np.log(1 - y_pred) * (1 - y_actual)
```

Both equations give the same answer.

Negative log-likelihood

Maximizing log-likelihood is the same as minimizing negative log-likelihood.

```
-np.sum(log_likelihoods)
```

Logistic regression algorithm

```
def calc_neg_log_likelihood(coeffs)
  intercept, slope = coeffs
  # More calculation!
```

```
from scipy.optimize import minimize


minimize(
  fun=calc_neg_log_likelihood,
  x0=[0, 0]
)
```

Let's practice!

INTERMEDIATE REGRESSION WITH STATSMODELS IN PYTHON

Congratulations!

INTERMEDIATE REGRESSION WITH STATSMODELS IN PYTHON

Maarten Van den Broeck
Content Developer at DataCamp

You learned things

Chapter 1

- Fit/visualize/predict/assess parallel slopes

Chapter 2

- Interactions between explanatory variables
- Simpson's Paradox

Chapter 3

- Extend to many explanatory variables
- Implement linear regression algorithm

Chapter 4

- Logistic regression with multiple explanatory variables
- Logistic distribution
- Implement logistic regression algorithm

There is more to learn

- Training and testing sets
- Cross validation
- P-values and significance

Advanced regression

- Generalized Linear Models in Python
- Introduction to Predictive Analytics in Python
- Linear Classifiers in Python
- Machine Learning with Tree-Based Models in Python

Have fun regressing!

INTERMEDIATE REGRESSION WITH STATSMODELS IN PYTHON