

# Why you need logistic regression

INTRODUCTION TO 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
...	...	...
<i>response</i>	<i>length of relationship</i>	<i>recency of activity</i>

<sup>1</sup> <https://www.rdocumentation.org/packages/bayesQR/topics/Churn>

# Churn vs. recency: a linear model

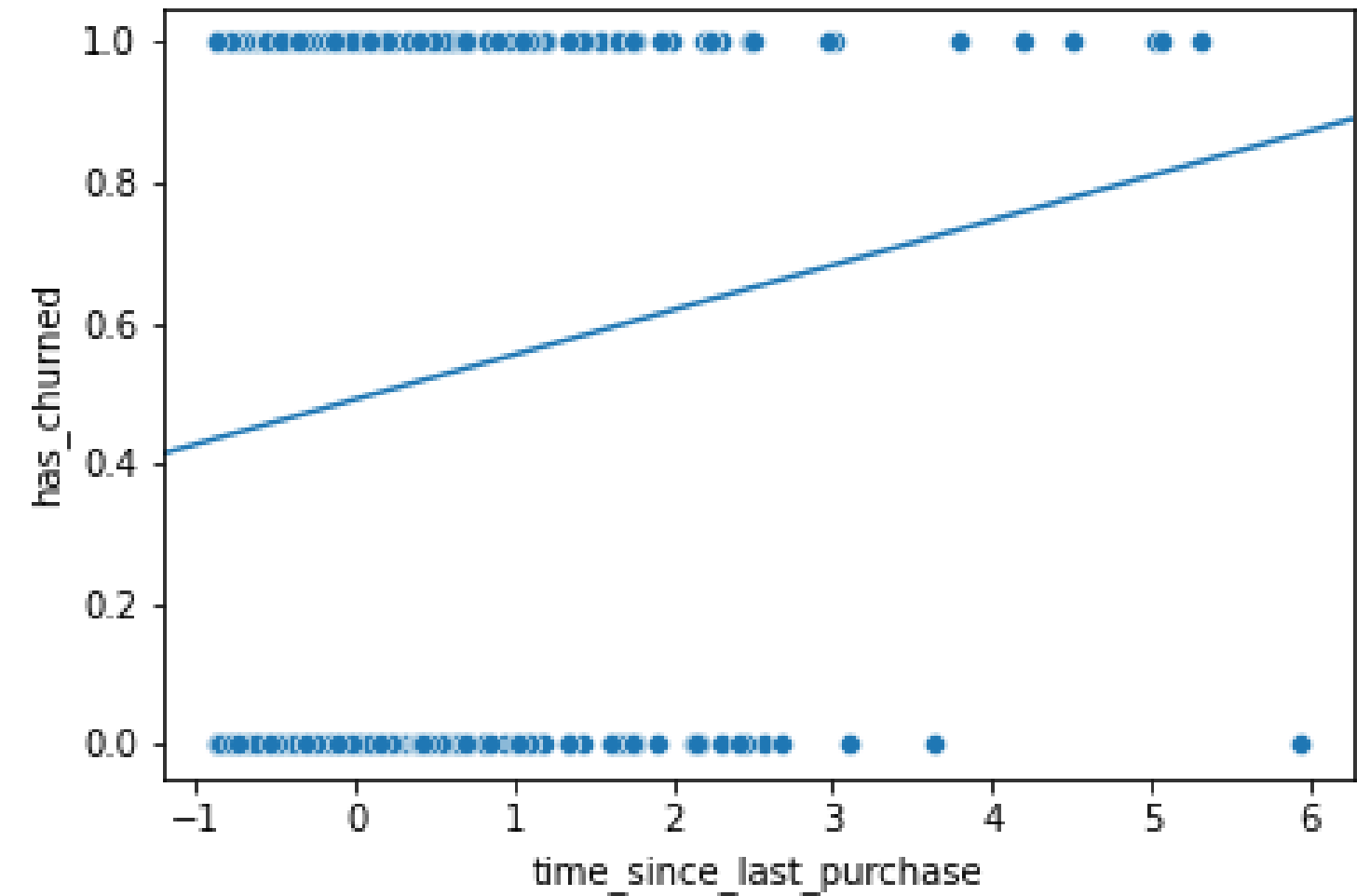
```
mdl_churn_vs_recency_lm = ols("has_churned ~ time_since_last_purchase",  
                               data=churn).fit()  
  
print(mdl_churn_vs_recency_lm.params)
```

```
Intercept          0.490780  
time_since_last_purchase  0.063783  
dtype: float64
```

```
intercept, slope = mdl_churn_vs_recency_lm.params
```

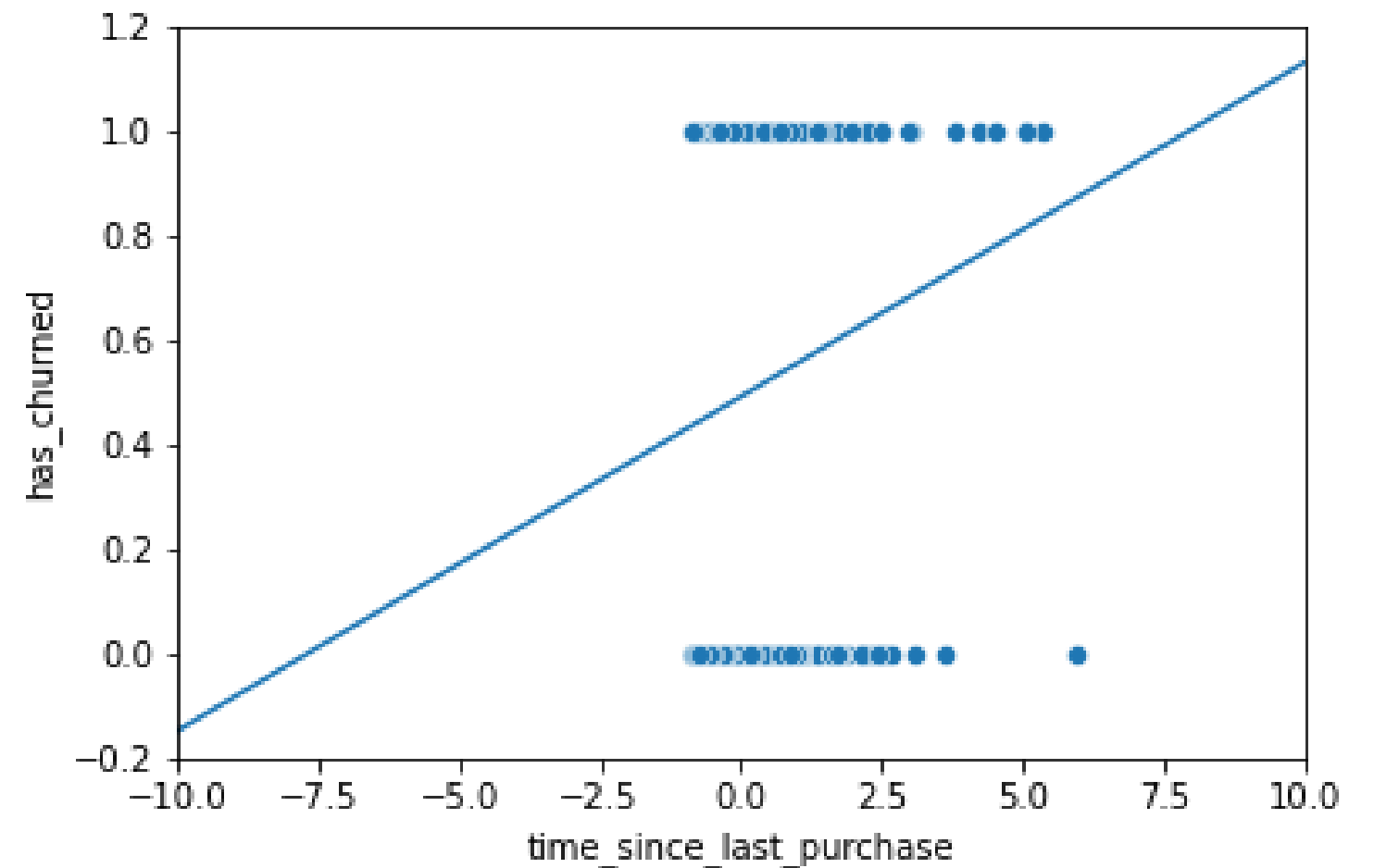
# Visualizing the linear model

```
sns.scatterplot(x="time_since_last_purchase",  
               y="has_churned",  
               data=churn)  
  
plt.axline(xy1=(0, intercept),  
           slope=slope)  
  
plt.show()
```



# Zooming out

```
sns.scatterplot(x="time_since_last_purchase",  
               y="has_churned",  
               data=churn)  
  
plt.axline(xy1=(0, intercept),  
           slope=slope)  
  
plt.xlim(-10, 10)  
plt.ylim(-0.2, 1.2)  
plt.show()
```



# What is logistic regression?

- Another type of generalized linear model.
- Used when the response variable is logical.
- The responses follow logistic (S-shaped) curve.

# Logistic regression using logit()

```
from statsmodels.formula.api import logit
mdl_churn_vs_recency_logit = logit("has_churned ~ time_since_last_purchase",
                                   data=churn).fit()
print(mdl_churn_vs_recency_logit.params)
```

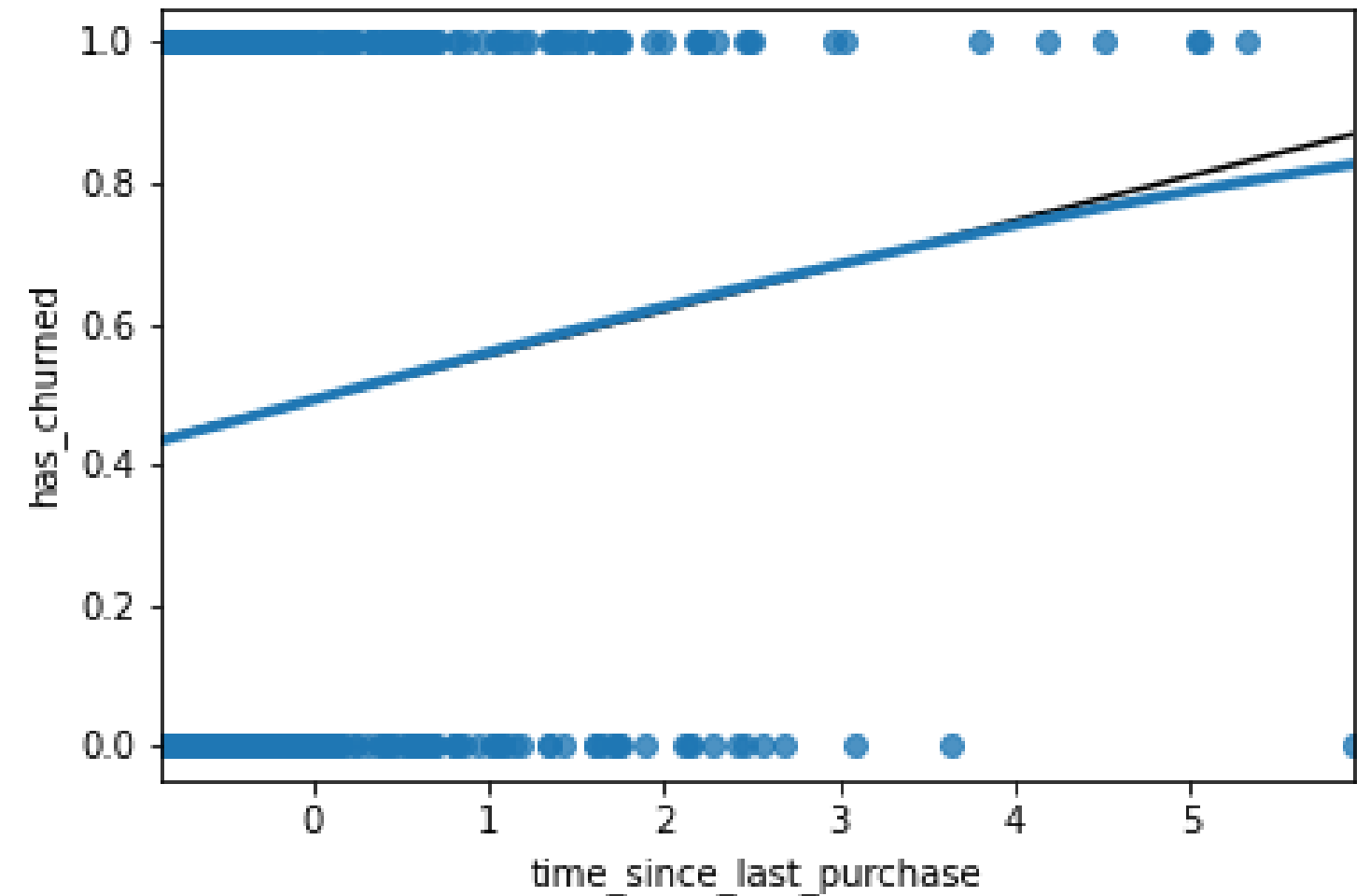
```
Intercept          -0.035019
time_since_last_purchase  0.269215
dtype: float64
```

# Visualizing the logistic model

```
sns.regplot(x="time_since_last_purchase",  
            y="has_churned",  
            data=churn,  
            ci=None,  
            logistic=True)
```

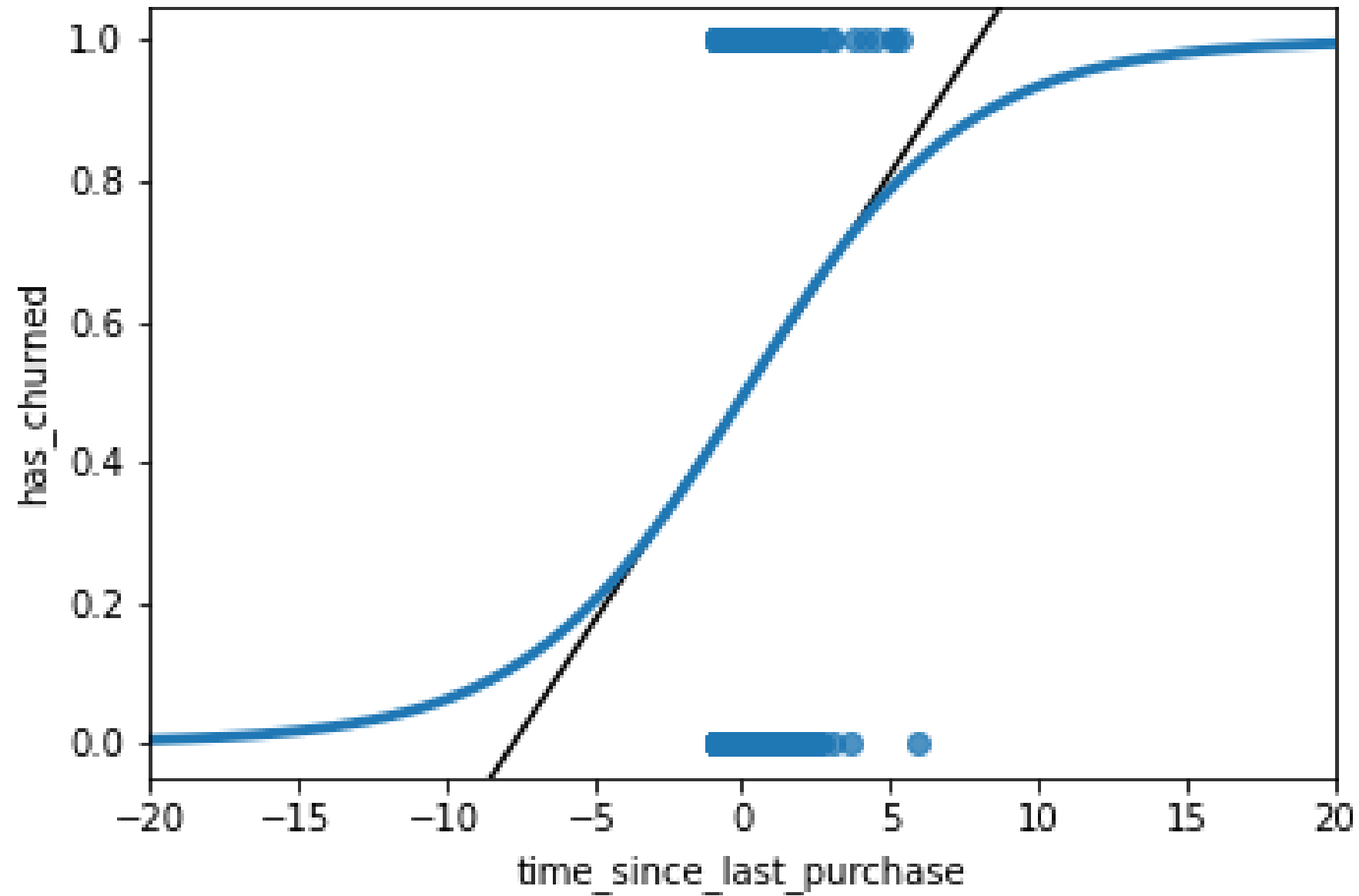
```
plt.axline(xy1=(0, intercept),  
           slope=slope,  
           color="black")
```

```
plt.show()
```





# Zooming out



# Let's practice!

INTRODUCTION TO REGRESSION WITH STATSMODELS IN PYTHON

# Predictions and odds ratios

INTRODUCTION TO REGRESSION WITH STATSMODELS IN PYTHON

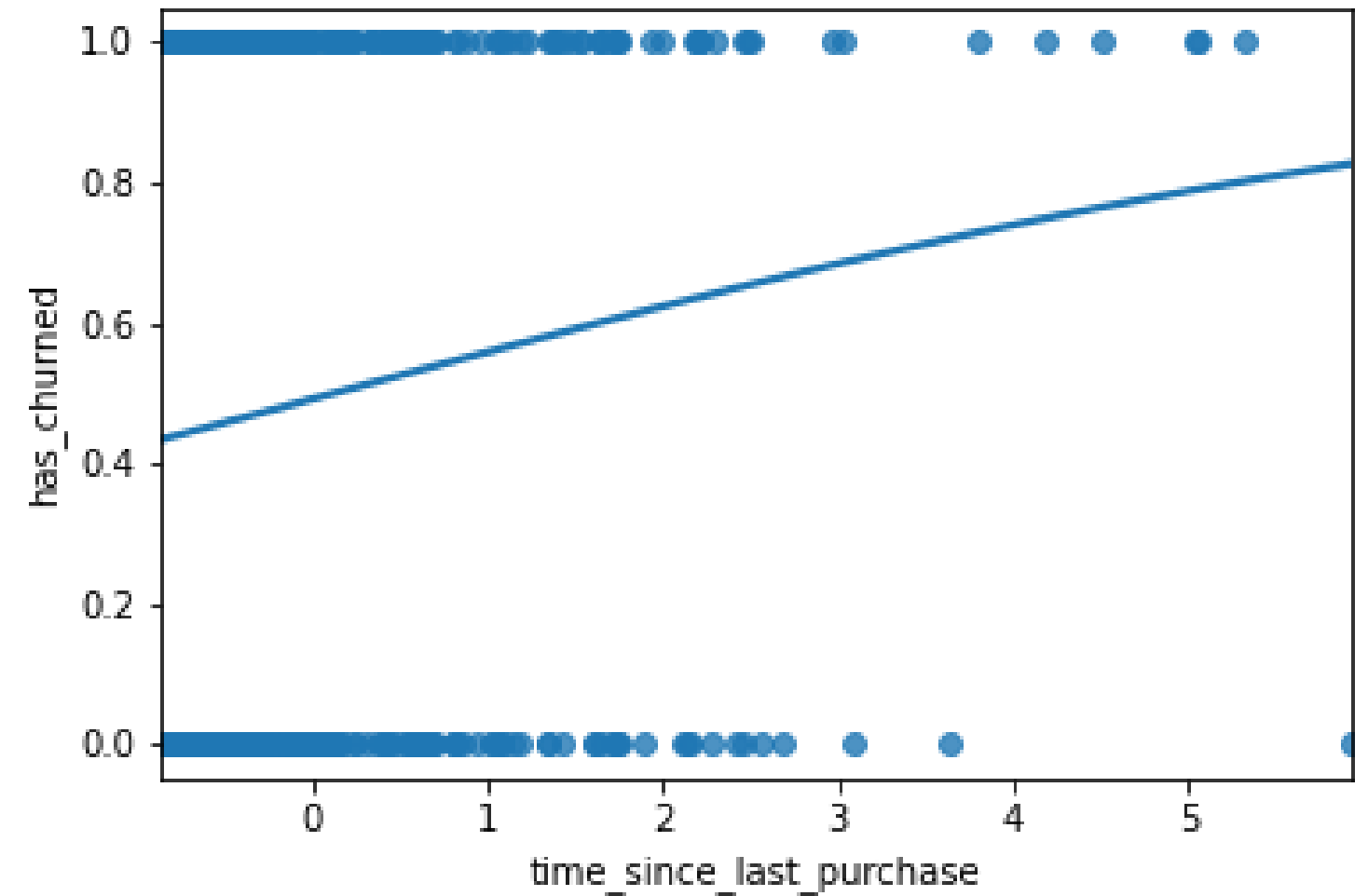


**Maarten Van den Broeck**

Content Developer at DataCamp

# The regplot() predictions

```
sns.regplot(x="time_since_last_purchase",  
            y="has_churned",  
            data=churn,  
            ci=None,  
            logistic=True)  
  
plt.show()
```

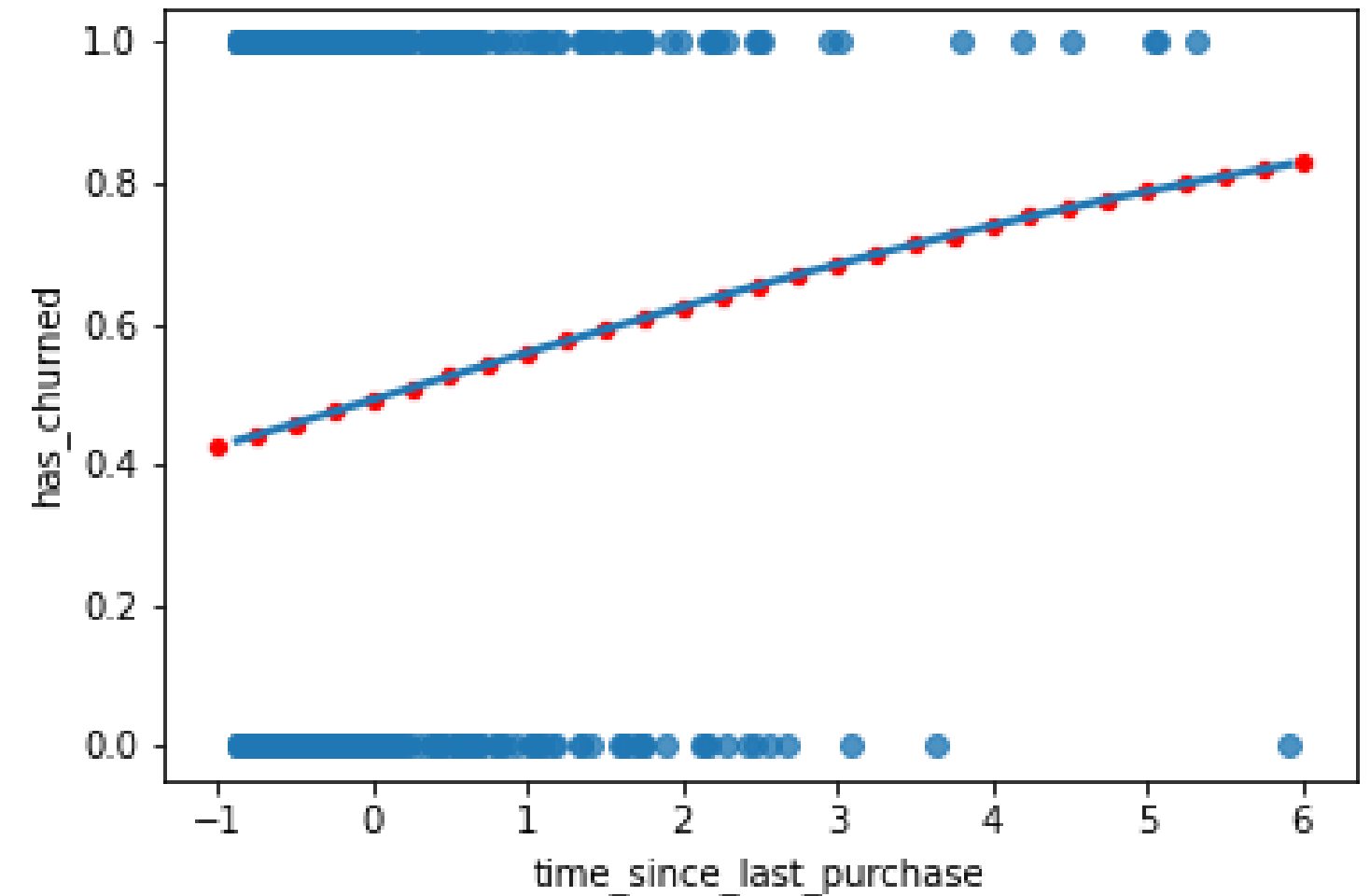


# Making predictions

```
mdl_recency = logit("has_churned ~ time_since_last_purchase",  
                    data = churn).fit()  
  
explanatory_data = pd.DataFrame(  
    {"time_since_last_purchase": np.arange(-1, 6.25, 0.25)})  
  
prediction_data = explanatory_data.assign(  
    has_churned = mdl_recency.predict(explanatory_data))
```

# Adding point predictions

```
sns.regplot(x="time_since_last_purchase",  
            y="has_churned",  
            data=churn,  
            ci=None,  
            logistic=True)  
  
sns.scatterplot(x="time_since_last_purchase",  
                y="has_churned",  
                data=prediction_data,  
                color="red")  
  
plt.show()
```

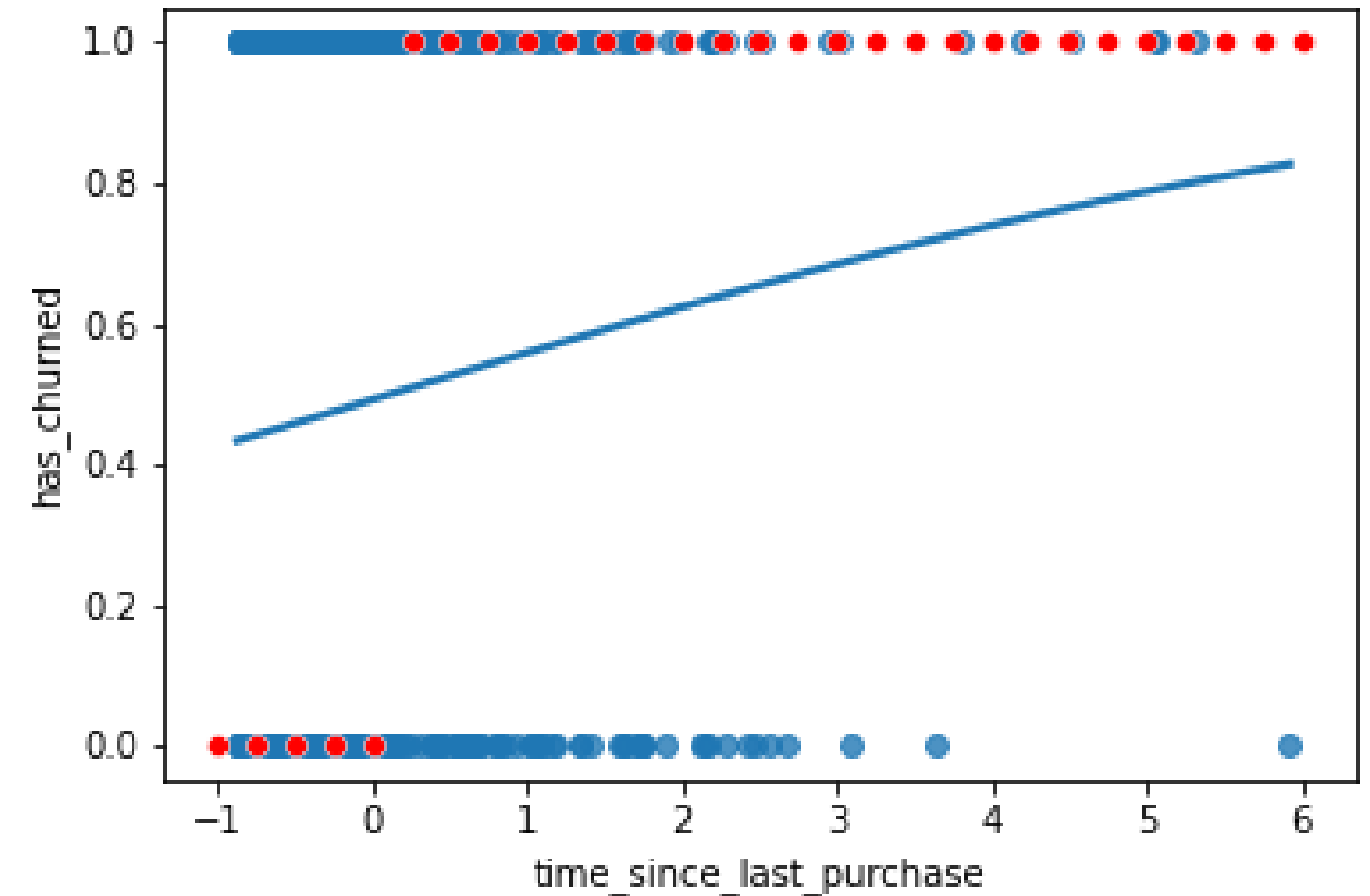


# Getting the most likely outcome

```
prediction_data = explanatory_data.assign(  
    has_churned = mdl_recency.predict(explanatory_data))  
prediction_data["most_likely_outcome"] = np.round(prediction_data["has_churned"])
```

# Visualizing most likely outcome

```
sns.regplot(x="time_since_last_purchase",  
            y="has_churned",  
            data=churn,  
            ci=None,  
            logistic=True)  
  
sns.scatterplot(x="time_since_last_purchase",  
                y="most_likely_outcome",  
                data=prediction_data,  
                color="red")  
  
plt.show()
```



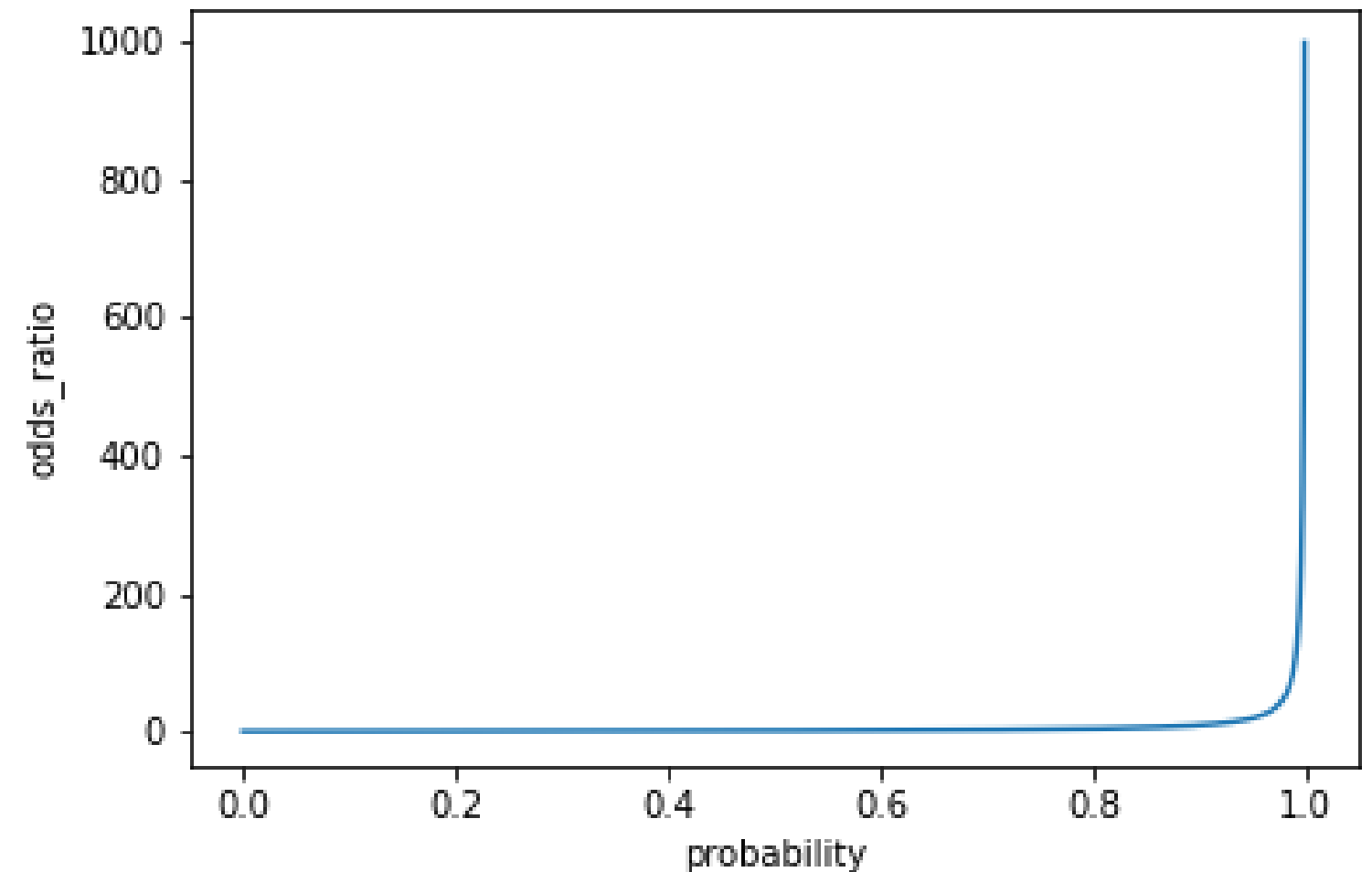


# Odds ratios

*Odds ratio* is the probability of something happening divided by the probability that it doesn't.

$$\text{odds\_ratio} = \frac{\text{probability}}{(1 - \text{probability})}$$

$$\text{odds\_ratio} = \frac{0.25}{(1 - 0.25)} = \frac{1}{3}$$

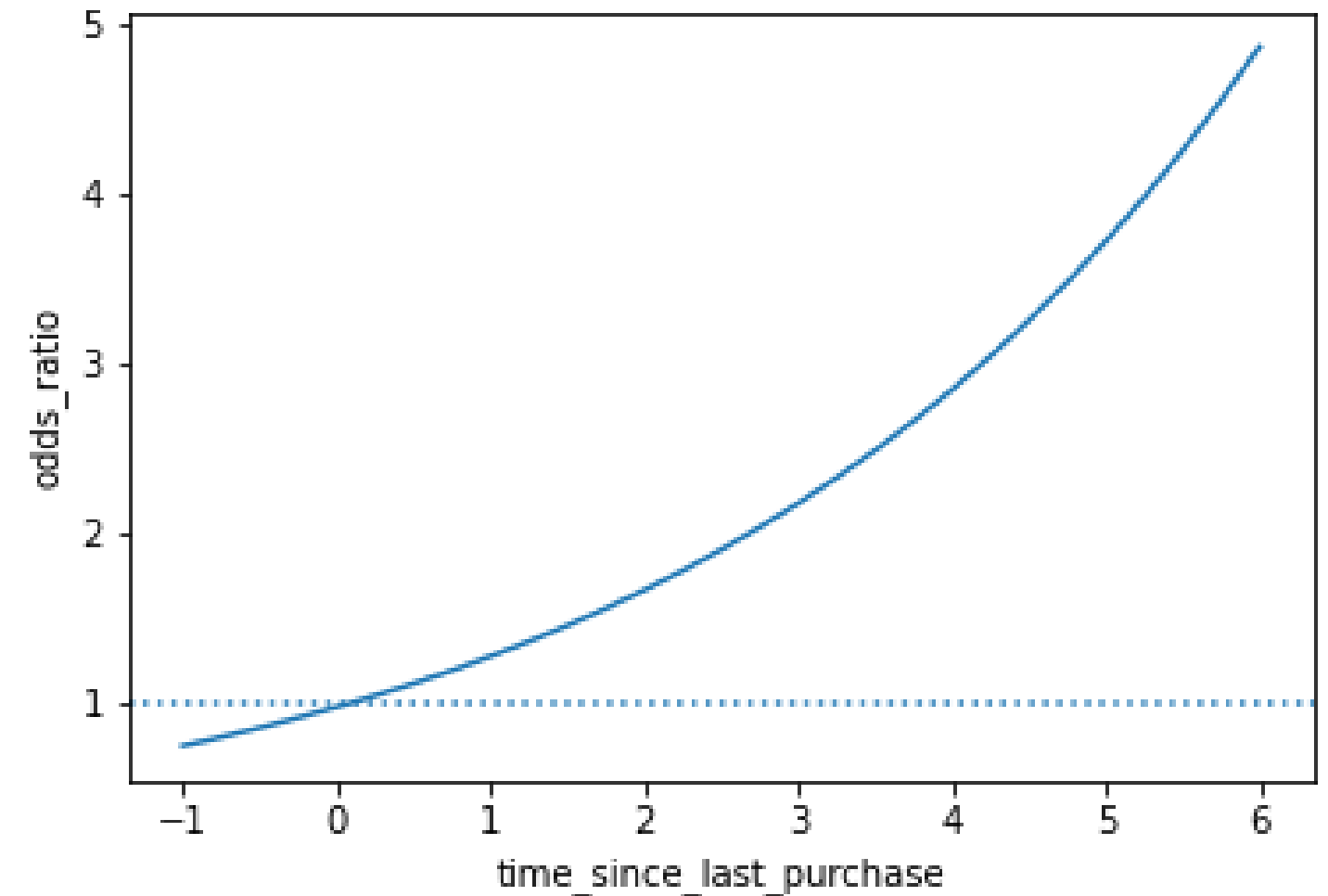


# Calculating odds ratio

```
prediction_data["odds_ratio"] = prediction_data["has_churned"] /  
                                (1 - prediction_data["has_churned"])
```

# Visualizing odds ratio

```
sns.lineplot(x="time_since_last_purchase",  
             y="odds_ratio",  
             data=prediction_data)  
  
plt.axhline(y=1,  
            linestyle="dotted")  
  
plt.show()
```



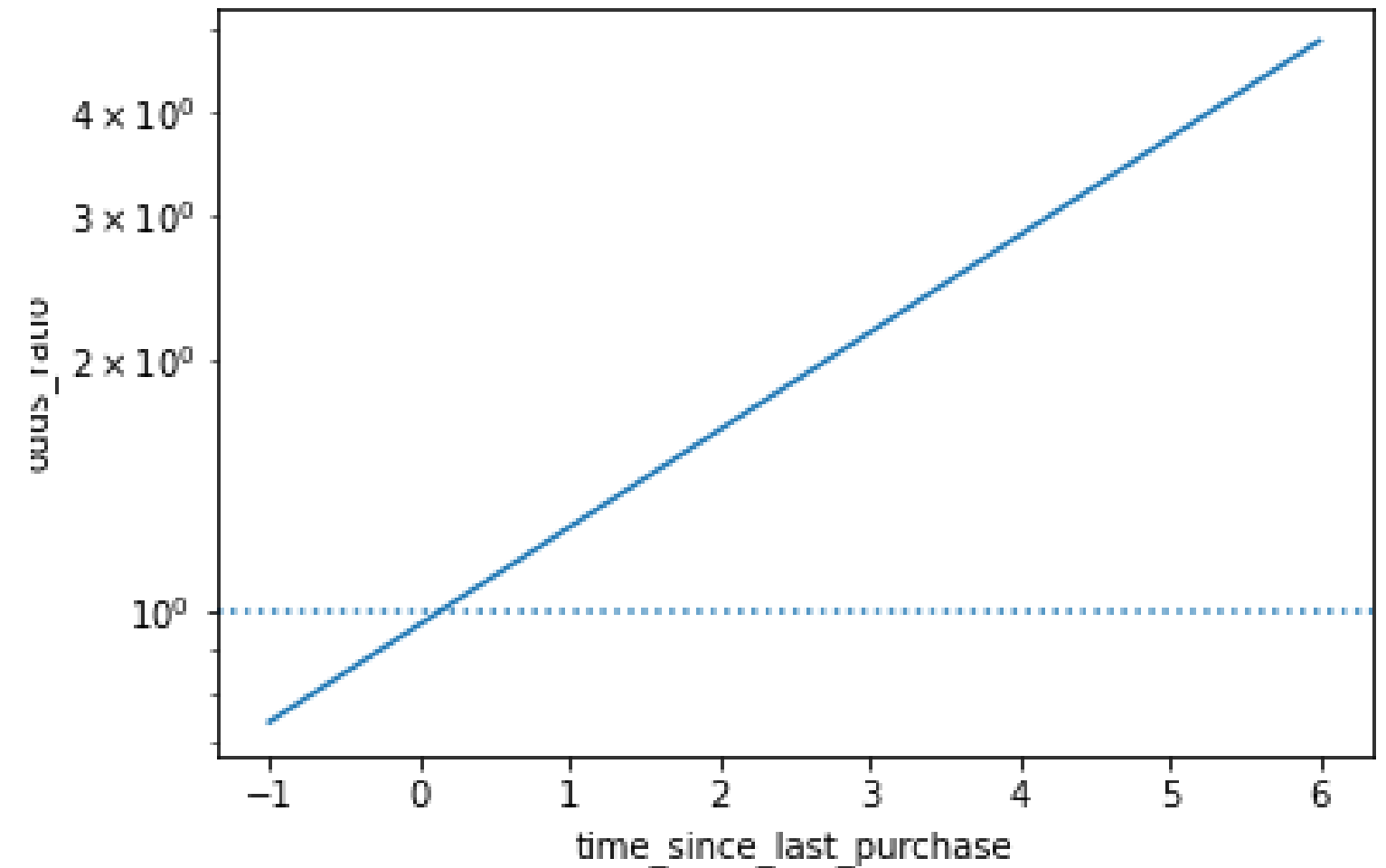
# Visualizing log odds ratio

```
sns.lineplot(x="time_since_last_purchase",  
             y="odds_ratio",  
             data=prediction_data)
```

```
plt.axhline(y=1,  
            linestyle="dotted")
```

```
plt.yscale("log")
```

```
plt.show()
```



# Calculating log odds ratio

```
prediction_data["log_odds_ratio"] = np.log(prediction_data["odds_ratio"])
```

# All predictions together

time_since_last_prchs	has_churned	most_likely_rspns	odds_ratio	log_odds_ratio
0	0.491	0	0.966	-0.035
2	0.623	1	1.654	0.503
4	0.739	1	2.834	1.042
6	0.829	1	4.856	1.580
...	...	...	...	...

# Comparing scales

Scale	Are values easy to interpret?	Are changes easy to interpret?	Is precise?
Probability	✓	✗	✓
Most likely outcome	✓✓	✓	✗
Odds ratio	✓	✗	✓
Log odds ratio	✗	✓	✓

# Let's practice!

INTRODUCTION TO REGRESSION WITH STATSMODELS IN PYTHON



# Quantifying logistic regression fit

INTRODUCTION TO REGRESSION WITH STATSMODELS IN PYTHON



**Maarten Van den Broeck**

Content Developer at DataCamp

# The four outcomes

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

# Confusion matrix: counts of outcomes

```
actual_response = churn["has_churned"]
```

```
predicted_response = np.round mdl_recency.predict())
```

```
outcomes = pd.DataFrame({"actual_response": actual_response,  
                          "predicted_response": predicted_response})
```

```
print(outcomes.value_counts(sort=False))
```

actual_response	predicted_response	
0	0.0	141
	1.0	59
1	0.0	111
	1.0	89

# Visualizing the confusion matrix

```
conf_matrix = mdl_recency.pred_table()

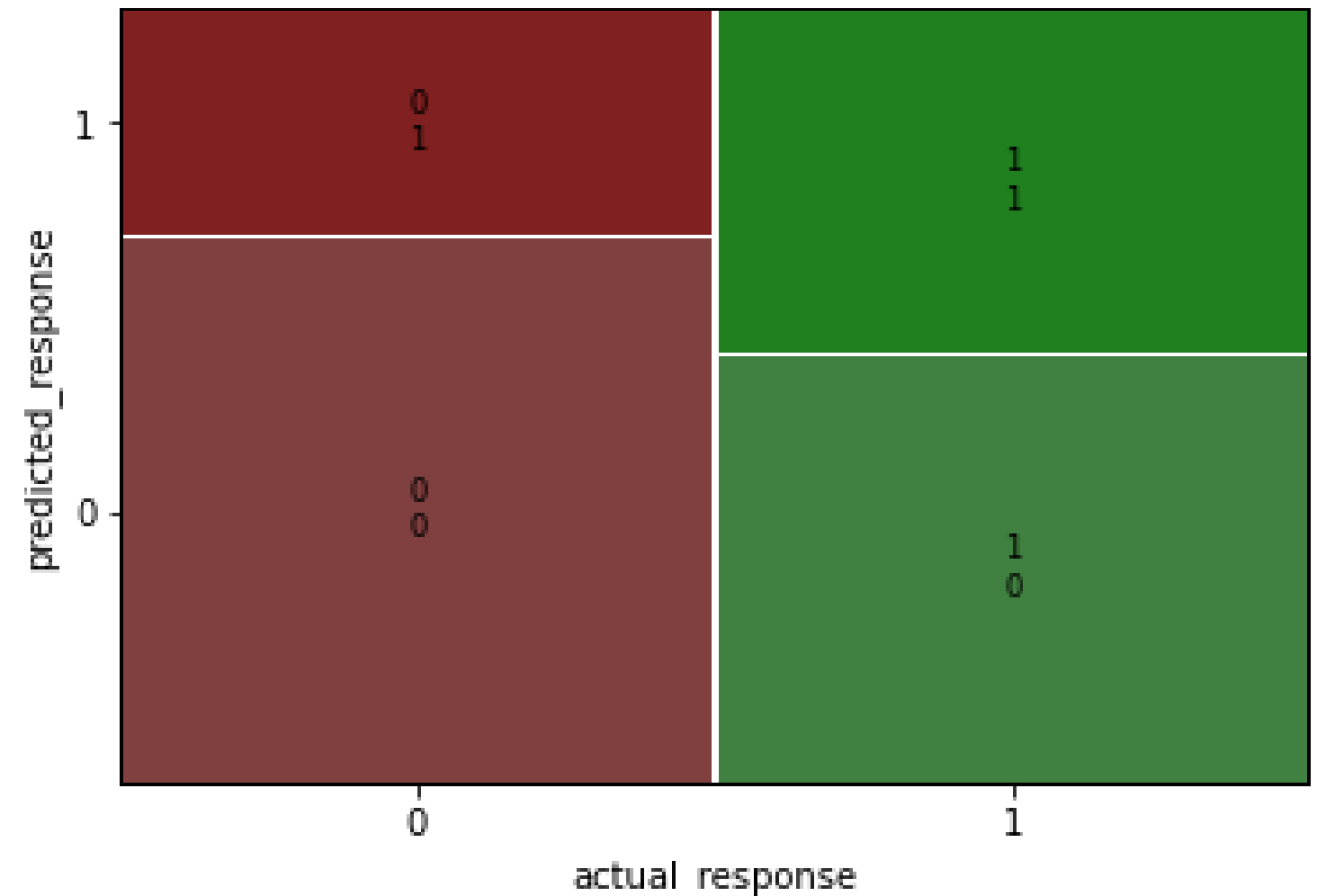
print(conf_matrix)
```

```
[[141.    59.]
 [111.    89.]]
```

true negative	false positive
false negative	true positive

```
from statsmodels.graphics.mosaicplot
import mosaic

mosaic(conf_matrix)
```



# Accuracy

*Accuracy* is the proportion of correct predictions.

$$\text{accuracy} = \frac{TN + TP}{TN + FN + FP + TP}$$

```
[[141.,  59.],  
 [111.,  89.]]
```

```
TN = conf_matrix[0,0]  
TP = conf_matrix[1,1]  
FN = conf_matrix[1,0]  
FP = conf_matrix[0,1]
```

```
acc = (TN + TP) / (TN + TP + FN + FP)  
print(acc)
```

```
0.575
```

# Sensitivity

*Sensitivity* is the proportion of true positives.

$$\text{sensitivity} = \frac{TP}{FN + TP}$$

```
[[141.,  59.],  
 [111.,  89.]]
```

```
TN = conf_matrix[0,0]  
TP = conf_matrix[1,1]  
FN = conf_matrix[1,0]  
FP = conf_matrix[0,1]
```

```
sens = TP / (FN + TP)  
print(sens)
```

```
0.445
```

# Specificity

*Specificity* is the proportion of true negatives.

$$\text{specificity} = \frac{TN}{TN + FP}$$

```
[[141.,  59.],  
 [111.,  89.]]
```

```
TN = conf_matrix[0,0]  
TP = conf_matrix[1,1]  
FN = conf_matrix[1,0]  
FP = conf_matrix[0,1]
```

```
spec = TN / (TN + FP)  
print(spec)
```

```
0.705
```

# Let's practice!

INTRODUCTION TO REGRESSION WITH STATSMODELS IN PYTHON



# Congratulations

INTRODUCTION TO REGRESSION WITH STATSMODELS IN PYTHON



**Maarten Van den Broeck**

Content Developer at DataCamp

# You learned things

## Chapter 1

- Fit a simple linear regression
- Interpret coefficients

## Chapter 3

- Quantifying model fit
- Outlier, leverage, and influence

## Chapter 2

- Make predictions
- Regression to the mean
- Transforming variables

## Chapter 4

- Fit a simple logistic regression
- Make predictions
- Get performance from confusion matrix

# Multiple explanatory variables

Intermediate Regression with statsmodels in Python

# Unlocking advanced skills

- Generalized Linear Models in Python
- Introduction to Predictive Analytics in Python
- Linear Classifiers in Python

# Happy learning!

INTRODUCTION TO REGRESSION WITH STATSMODELS IN PYTHON