Adding predictive variables

INTERMEDIATE PREDICTIVE ANALYTICS IN PYTHON

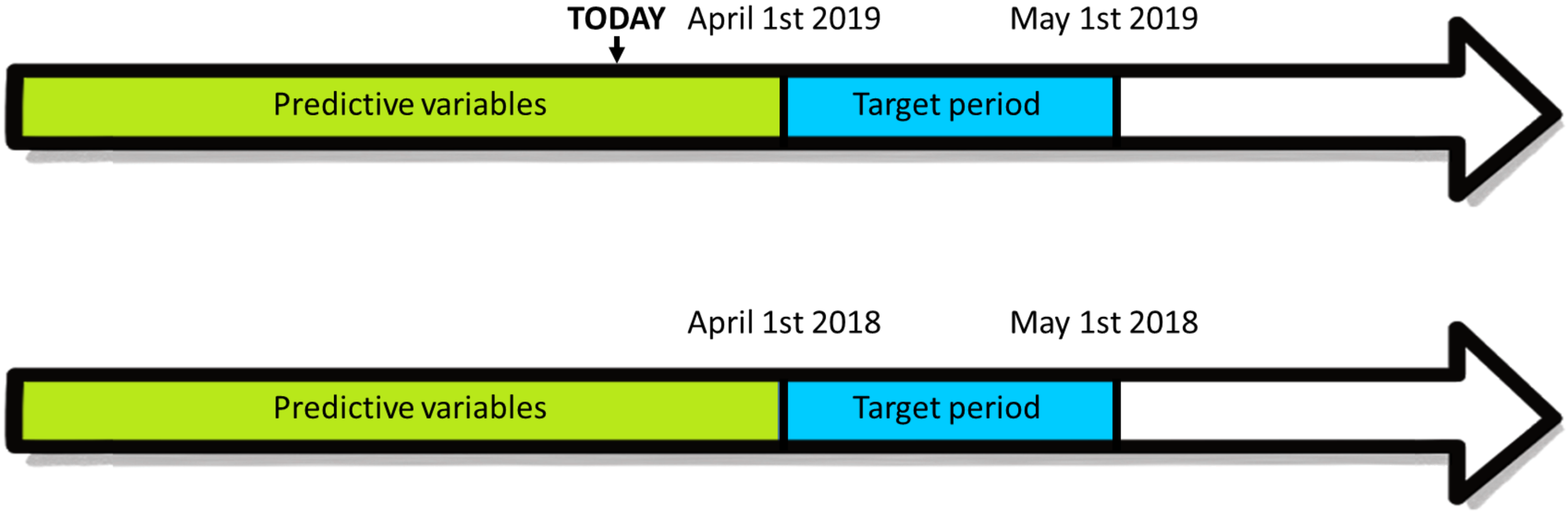
Nele Verbiest
Senior Data Scientist
@PythonPredictions

datacamp

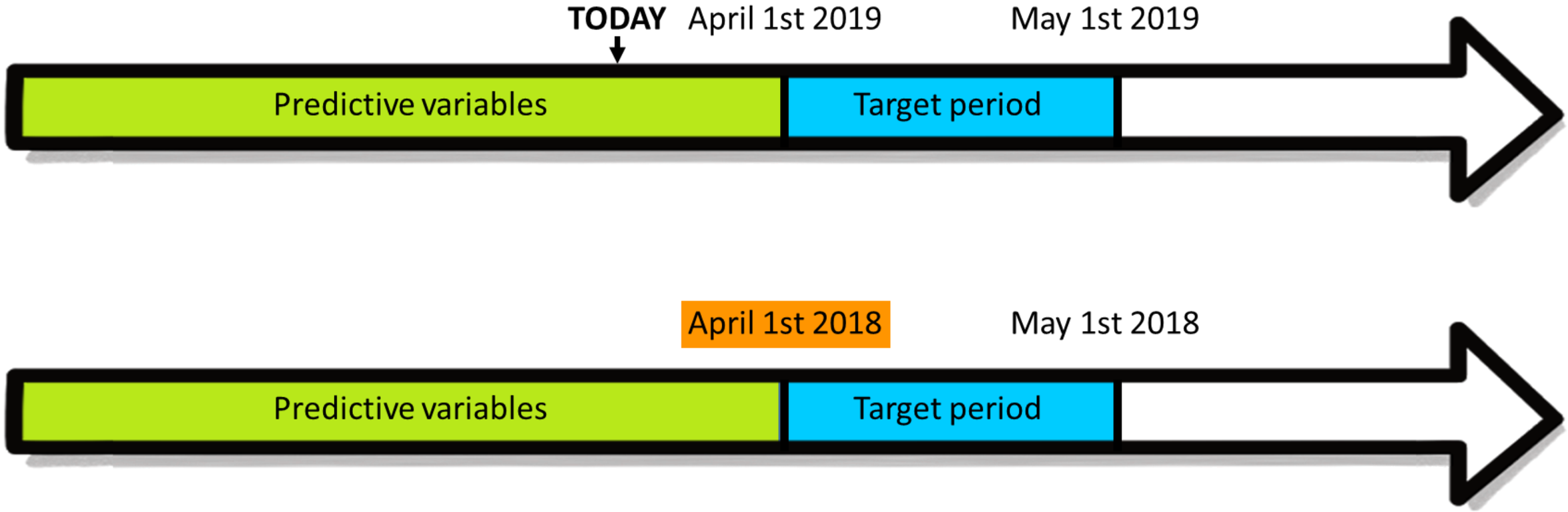
Predictive variables

- Demographics:
  - Age
  - Gender
  - Living place
- Spending behaviour
- Watching behaviour
- Product usage
- Surfing behaviour
- Payment information

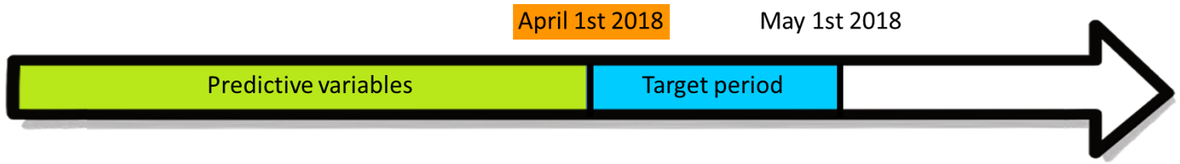
Timeline compliant predictive variables (1)



Timeline compliant predictive variables (2)



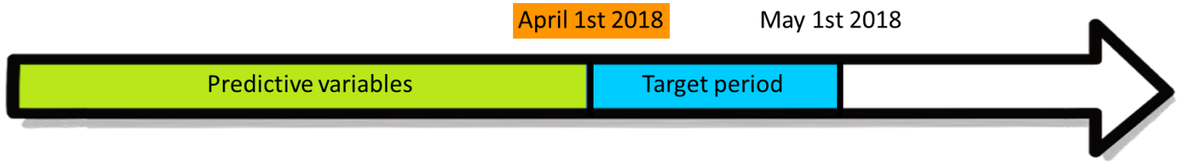
Adding lifetime




```
# Reference date
reference_date = datetime.date(2018,4,1)
# Add lifetime to the basetable
basetable["lifetime"] = reference_date - basetable["member_since"]
print(basetable.head())
```

```
donor_id member_since lifetime
1        2015-02-03   1153
2        2016-01-30   729
3        2016-02-23   768
```

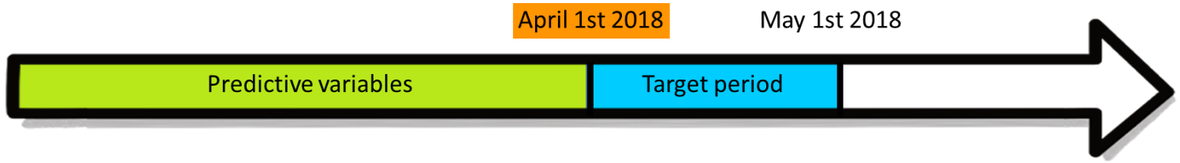
Adding preferred contact channel (1)




```
donor_id start_valid_date end_valid_date  contact_channel
1        2014-02-03       2016-03-04      "phone"
1        2016-03-04       2016-05-08      "e-mail"
2        2016-02-23       2026-02-23      "e-mail"
```

```
reference_date = datetime.date(2018,4,1)
# Select lines compliant with reference data
contact_channel_reference_date
    = living_places[
      (contact_channel["start_valid_date"]<=reference_date) &
      (living_places["end_valid_date"]>reference_date)]
```

Adding preferred contact channel (2)




```
# Add contact channel place to the basetable
basetable =
    pd.merge(
     basetable,
     living_places_reference_date[["donor_ID","contact_channel"]],
     on="donor_ID"
    )
print(basetable.head())
```

```
donor_id contact_channel
1        "phone"
2        "phone"
3        "e-mail"
```

Let's practice!

INTERMEDIATE PREDICTIVE ANALYTICS IN PYTHON

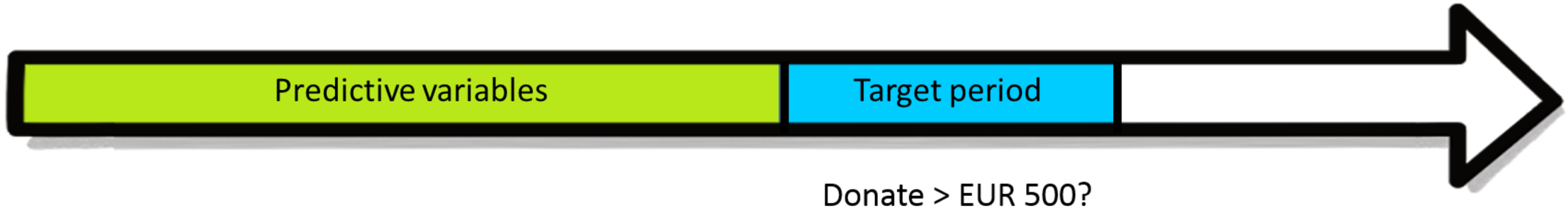
Adding aggregated variables

INTERMEDIATE PREDICTIVE ANALYTICS IN PYTHON

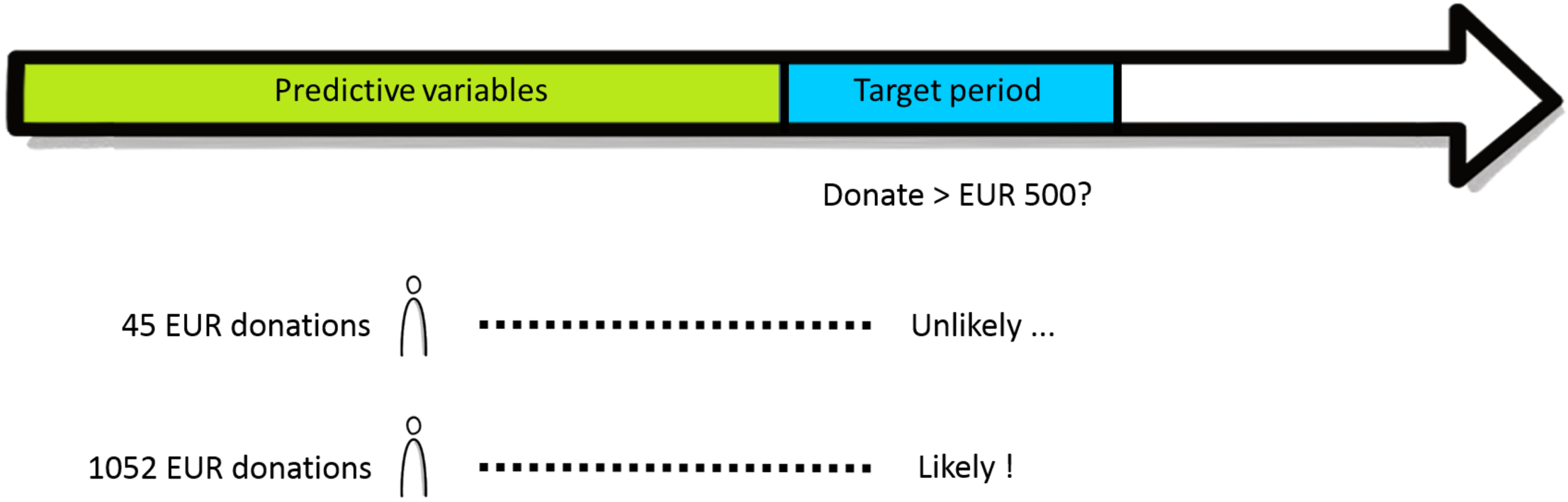
Nele Verbiest
Senior Data Scientist
@PythonPredictions

datacamp

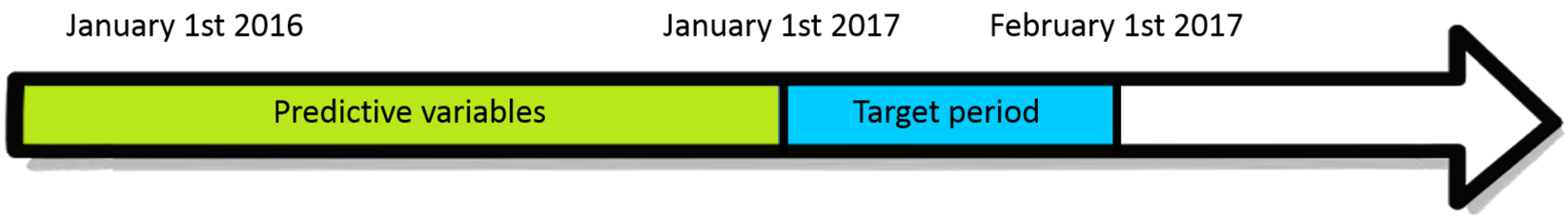
Motivation for aggregated variables (1)



Motivation for aggregated variables (2)



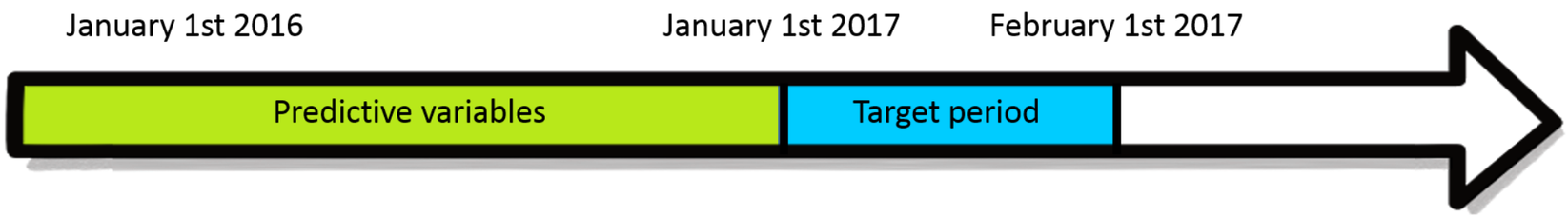
Adding total value last year (1)




```
id    date         amount
1     2015-10-16    75
1     2014-02-11    111
2     2012-03-28    93
```

```
# Start and end date of the aggregation period
start_date = datetime.date(2016,1,1)
end_date = datetime.date(2017,1,1)
# Select gifts made in 2016
gifts_2016 = gifts[(gifts["date"] >= start_date) & (gifts["date"] <= end_date)]
```

Adding total value last year (2)




```
# Sum of gifts per donor in 2016
gifts_2016_bydonor = gifts_2016.groupby(["id"])["amount"].sum().reset_index()
gifts_2016_bydonor.columns = ["donor_ID","sum_2016"]
# Add sum of gifts to the basetable
basetable = pd.merge(basetable, gifts_2016_bydonor, how = "left", on = "donor_ID")
print(basetable.head())
```

```
donor_id sum_2016
1        837
2        29
3        682
```

Adding number of donations to the basetable

```
# Number of gifts per donor in 2016
gifts_2016_bydonor = gifts_2016.groupby(["id"]).size().reset_index()
gifts_2016_bydonor.columns = ["donor_ID","count_2016"]
# Add number of gifts to the basetable
basetable = pd.merge(basetable, gifts_2016_bydonor, how = "left", on = "donor_ID")
print(basetable.head())
```

```
donor_id count_2016
1        4
2        9
3        2
```

Let's practice!

INTERMEDIATE PREDICTIVE ANALYTICS IN PYTHON

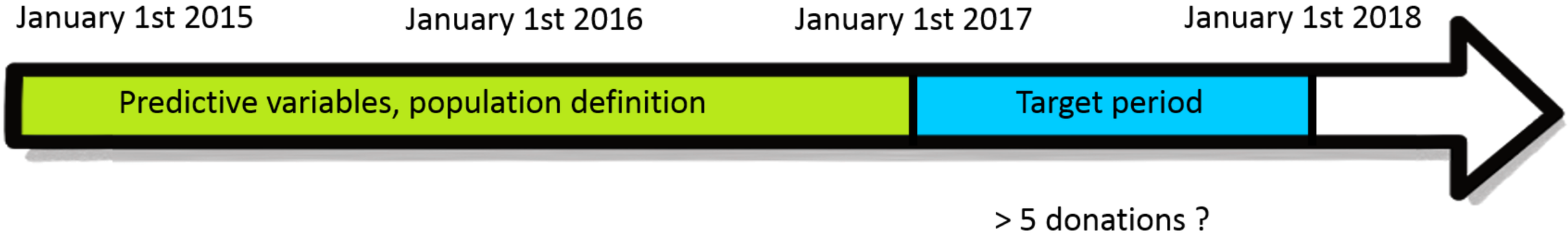
Adding evolutions

INTERMEDIATE PREDICTIVE ANALYTICS IN PYTHON

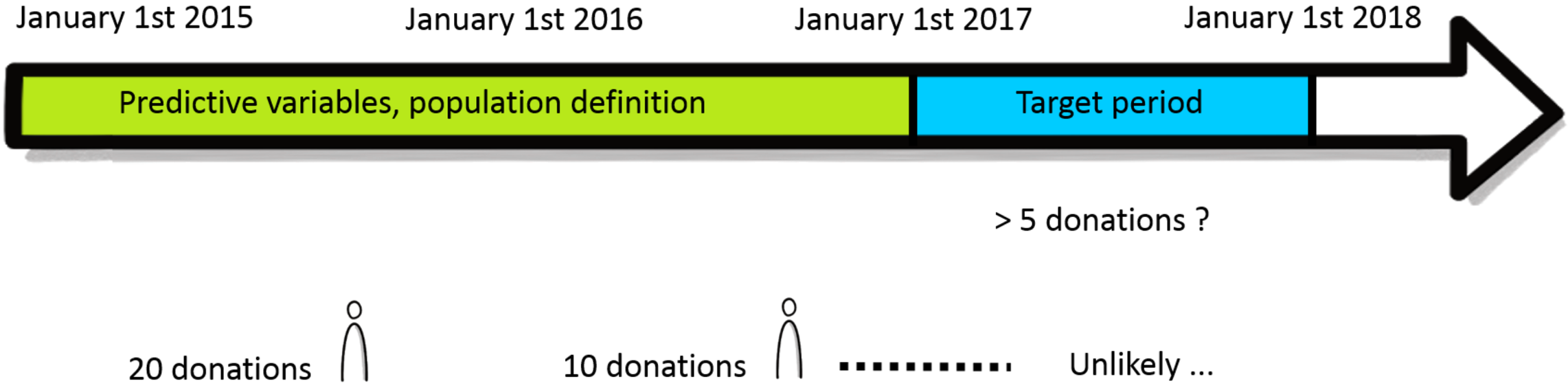
Nele Verbiest
Senior Data Scientist
@PythonPredictions

datacamp

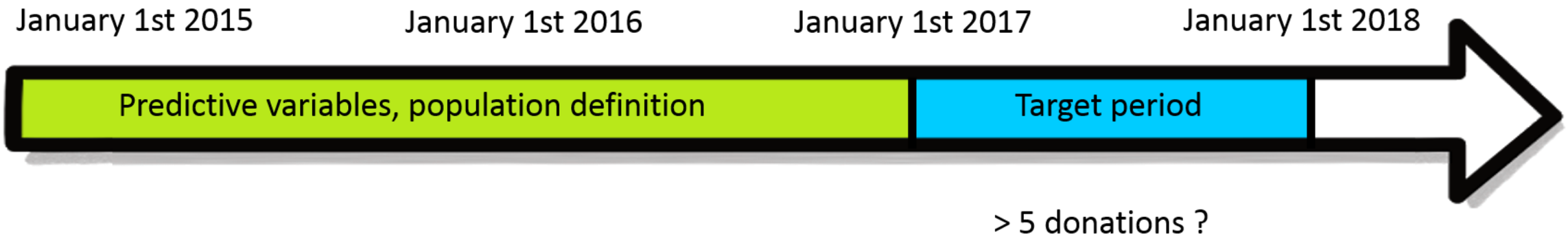
Motivation for evolutions (1)



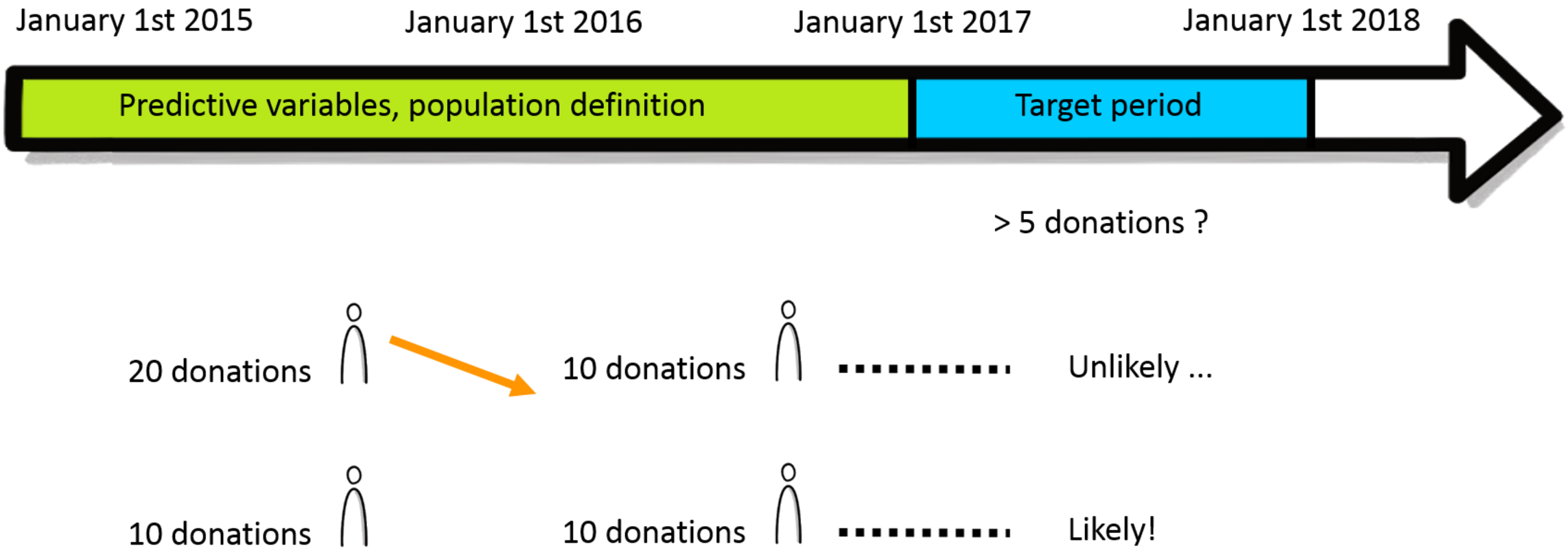
Motivation for evolutions (2)



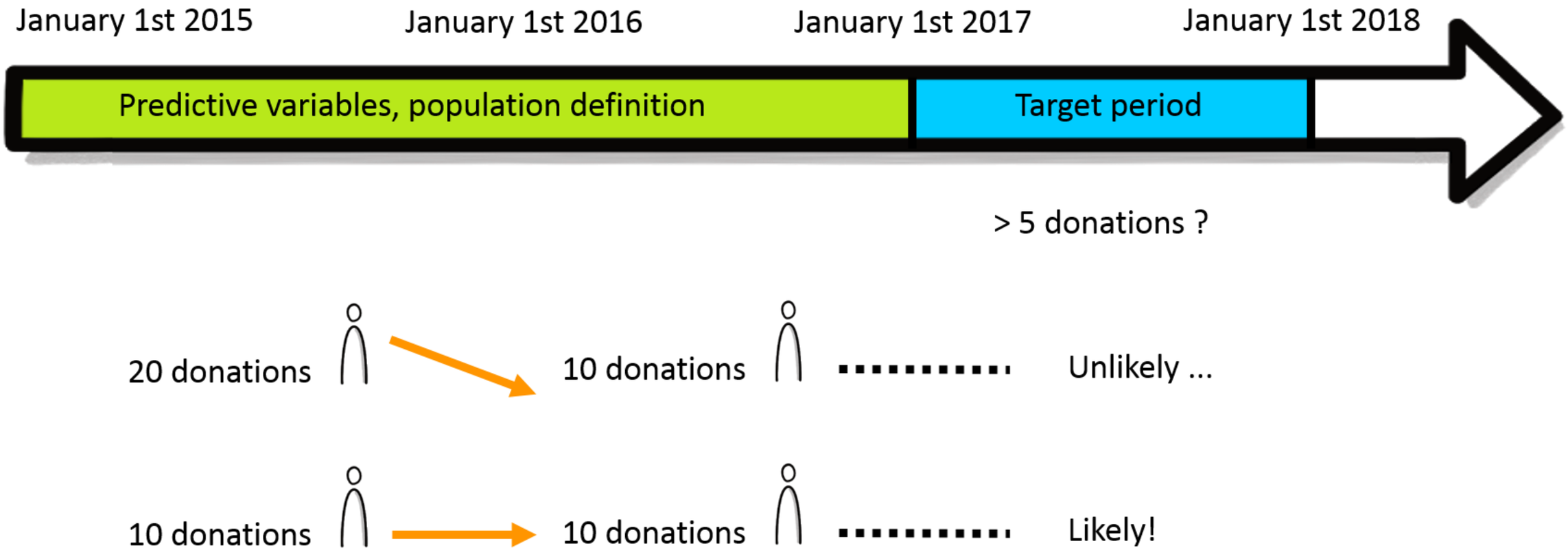
Motivation for evolutions (3)



Motivation for evolutions (4)



Motivation for evolutions (5)



Adding evolutions to the basetable (1)

```
# Reference dates
start_2017 = datetime.date(2017,1,1)
start_2016 = datetime.date(2016,1,1)
start_2015 = datetime.date(2015,1,1)
# Gifts last month and last year
gifts_2016 = gifts[
   (gifts["date"]<start_2017)
    & (gifts["date"]>=start_2016)]

gifts_2015_and_2016 = gifts[
   (gifts["date"]<start_2017)
    & (gifts["date"]>=start_2015)]
```

Adding evolutions to the basetable (2)

```
# Number of gifts in these periods per donor
number_gifts_2016 = gifts_2016.groupby("id")["amount"].size().reset_index()
number_gifts_2016.columns = ["donor_ID", "number_gifts_2016"]
number_gifts_2015_and_2016 =
    gifts_2015_and_2016 .groupby("id")["amount"].size().reset_index()
number_gifts_2015_and_2016.columns = ["donor_ID", "number_gifts_2015_and_2016"]
```

Adding evolutions to the basetable (3)

```
# Add these numbers to the basetable
basetable = pd.merge(basetable,
                     number_gifts_2016,
                     on="donor_ID",
                     how = "left")
basetable = pd.merge(basetable,
                     number_gifts_2015_and_2016,
                     on="donor_ID",
                     how = "left")
# Calculate ratio of last month's and last year's average
basetable["ratio_2015_to_2015_and_2016"] =
    basetable["number_gifts_2016"] /
    basetable["number_gifts_2015_and_2016"]
```

Adding evolutions to the basetable (4)

```
print(basetable.head())
```

```
donor_id number_gifts_2016 number_gifts_2015_and_2016 ratio_2015_to_2015_and_2016
1        Na                5                          Na
2        9                 12                         0.75
3        3                 6                          0.5
```

Let's practice!

INTERMEDIATE PREDICTIVE ANALYTICS IN PYTHON

Using evolution variables

INTERMEDIATE PREDICTIVE ANALYTICS IN PYTHON

Nele Verbiest
Senior Data Scientist
@PythonPredictions

datacamp

Building predictive models

```
# Import the linear_model module
from sklearn import linear_model
# Predictive variables
variables = ["gender","age", "donations_last_year", "ratio_month_year"]
# Select predictors and target
X = basetable[variables]
y = basetable[["target"]]
# Construct the logistic regression model
logreg = linear_model.LogisticRegression()
logreg.fit(X, y)
```

Making predictions

```
# Import the linear_model module
from sklearn import linear_model
# Predictive variables
variables = ["gender","age", "donations_last_year", "ratio_month_year"]
# Select predictors and target
X = basetable[variables]
y = basetable[["target"]]
# Construct the logistic regression model
logreg = linear_model.LogisticRegression()
logreg.fit(X, y)
```

```
# Make predictions
predictions = logreg.predict_proba(X)[:,1]
```

Evaluating predictive models using AUC

```
# Import roc_auc_score module from sklearn.metrics
from sklearn.metrics import roc_auc_score
# Calculate the AUC
auc= roc_auc_score(y, predictions)
print(round(auc,2))
```

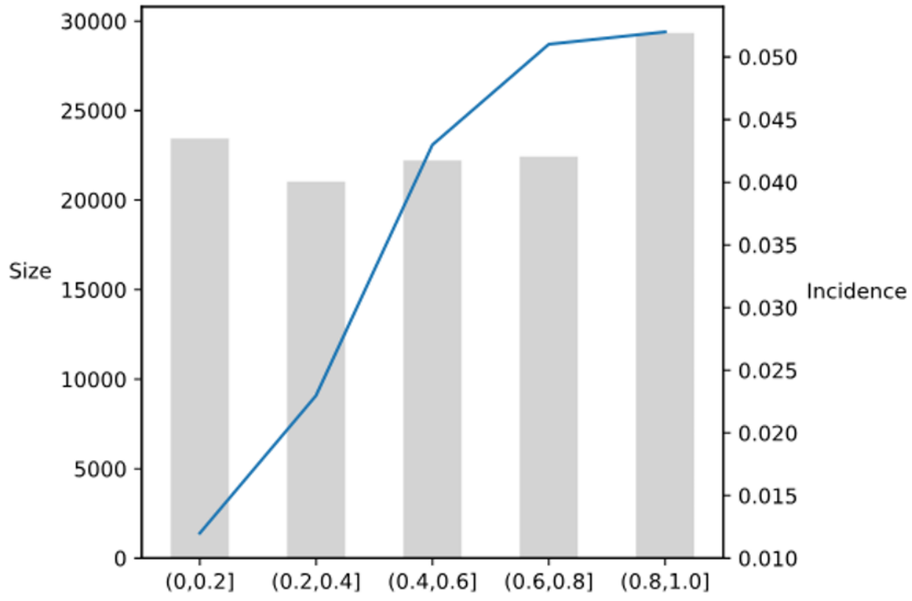
```
0.56
```

The predictor insight graph

```
# Discretize the variable in 5 bins and add to the basetable
basetable["ratio_month_year_disc"] = pd.qcut(basetable["ratio_month_year"], 5)
# Construct the predictor insight graph table
pig_table = create_pig_table(basetable, "target","ratio_month_year_disc")

```{python}
# Plot the predictor insight graph
plot_pig(pig_table, "ratio_month_year_disc")
```
```

Predictor insight graph interpretation



Let's practice!

INTERMEDIATE PREDICTIVE ANALYTICS IN PYTHON