The basetable timeline

INTERMEDIATE PREDICTIVE ANALYTICS IN PYTHON

Nele Verbiest Ph. D.
Senior Data Scientist @PythonPredictions

The predictive modeling process

Foundations of predictive analytics I:

- Build predictive models
- Evaluate predictive models
- Present predictive models to business stakeholders

Foundations of predictive analytics II:

- Construct the basetable

The basetable (1)

The basetable (2)

Population

The basetable (3)

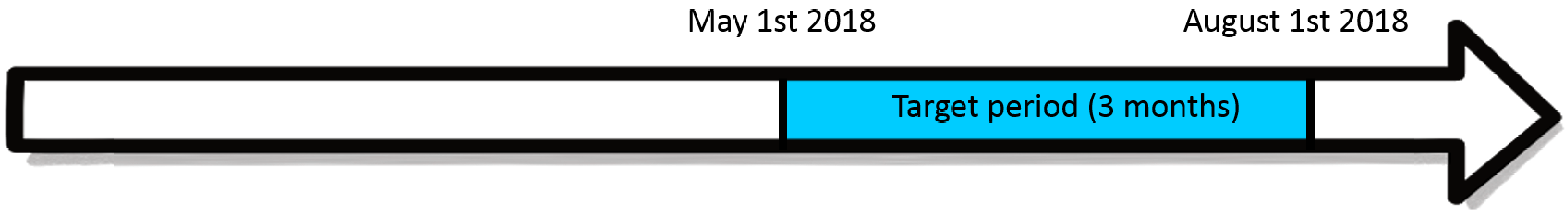
Population

The basetable (4)

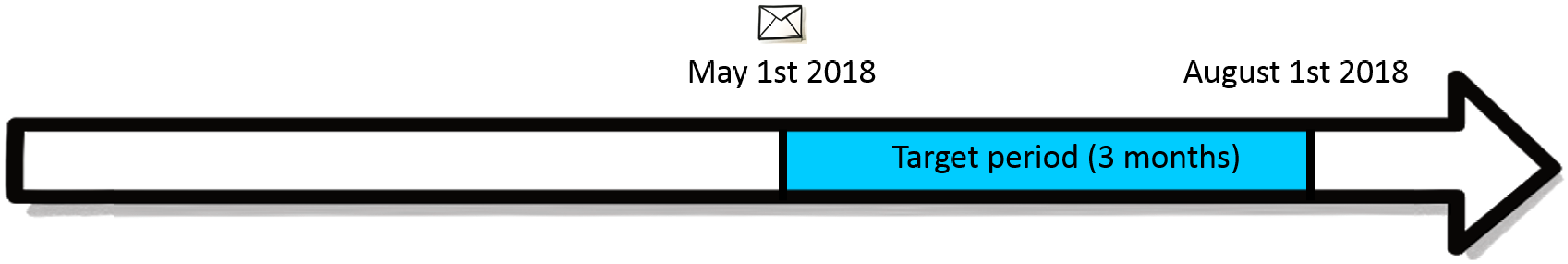
Population

The timeline (1)

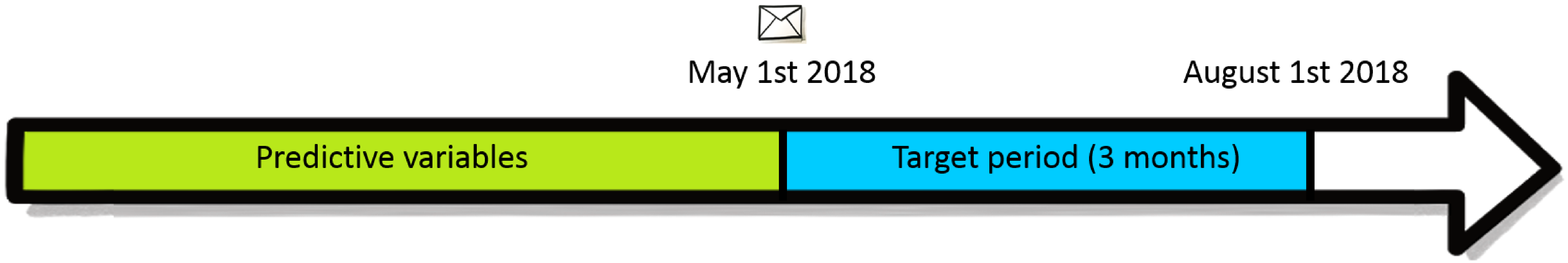
The timeline (2)



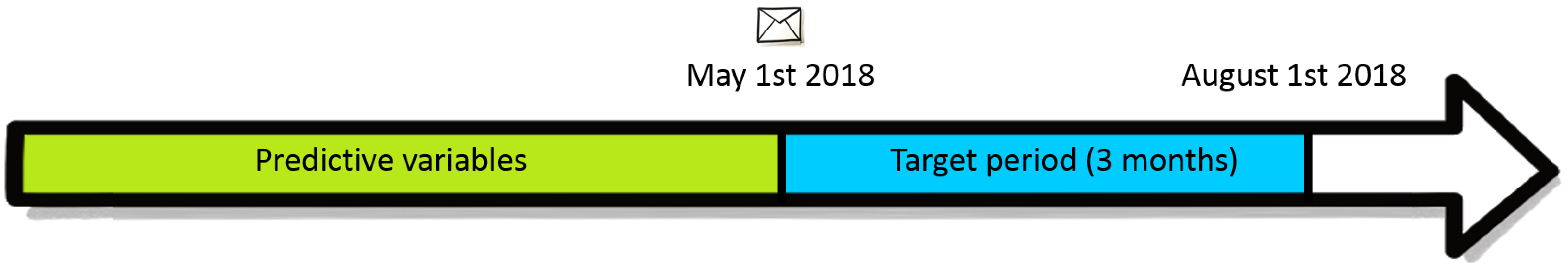
The timeline (3)



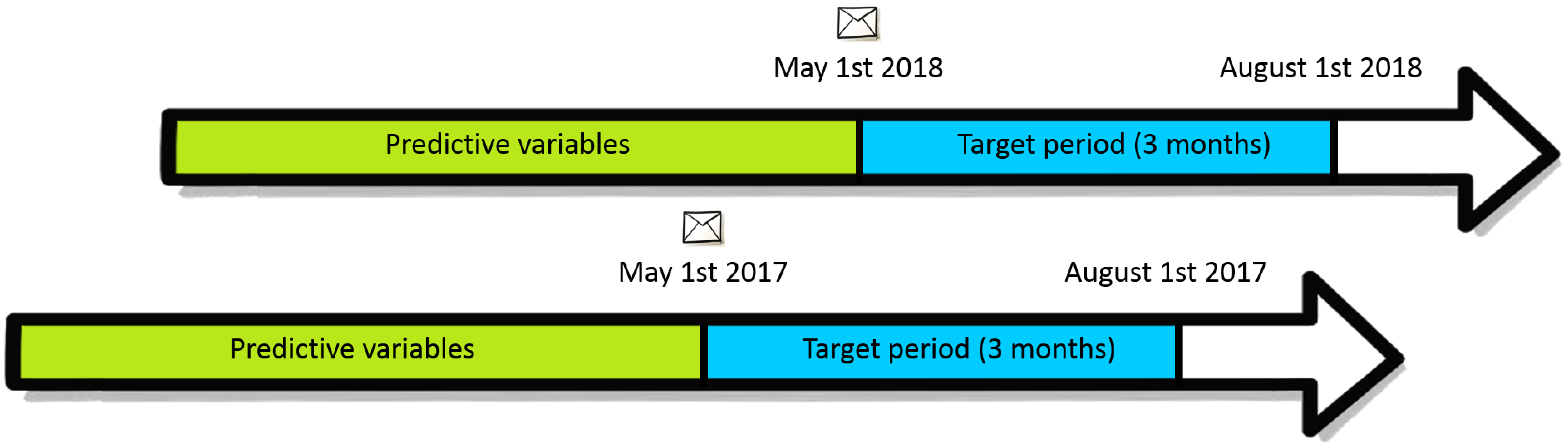
The timeline (4)



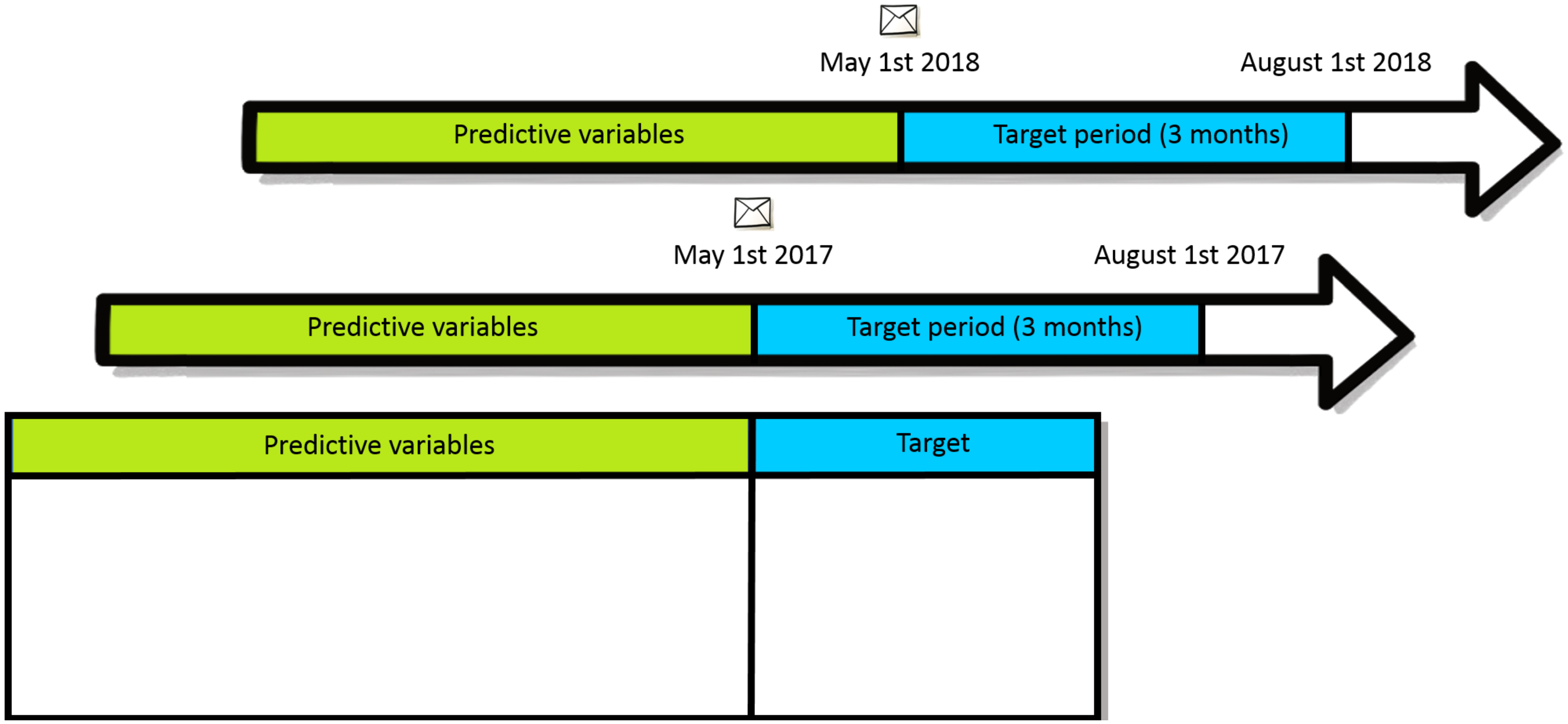
Reconstructing history (1)



Reconstructing history (2)



Reconstructing history (3)



Selecting relevant data in Python

```
Import pandas as pd
gifts = pd.read_csv(“gifts.csv”)
gifts["date"] = pd.to_datetime(gifts["date"])
print(gifts.head())
```

```
   id       date  amount
0   1 2015-10-16    75.0
1   1 2014-02-11   111.0
2   1 2012-03-28    93.0
3   1 2013-12-13   113.0
4   1 2012-01-10    93.0
```

```
start_target = datetime(year = 2018, month = 5, day = 1)
end_target = datetime(year = 2018, month = 8, day = 1)
gifts_target = gifts[(gifts["date"]>=start_target) & (gifts["date"]<end_target)]
gifts_pred_variables = gifts[(gifts["date"]<start_target]
```

Let's practice!

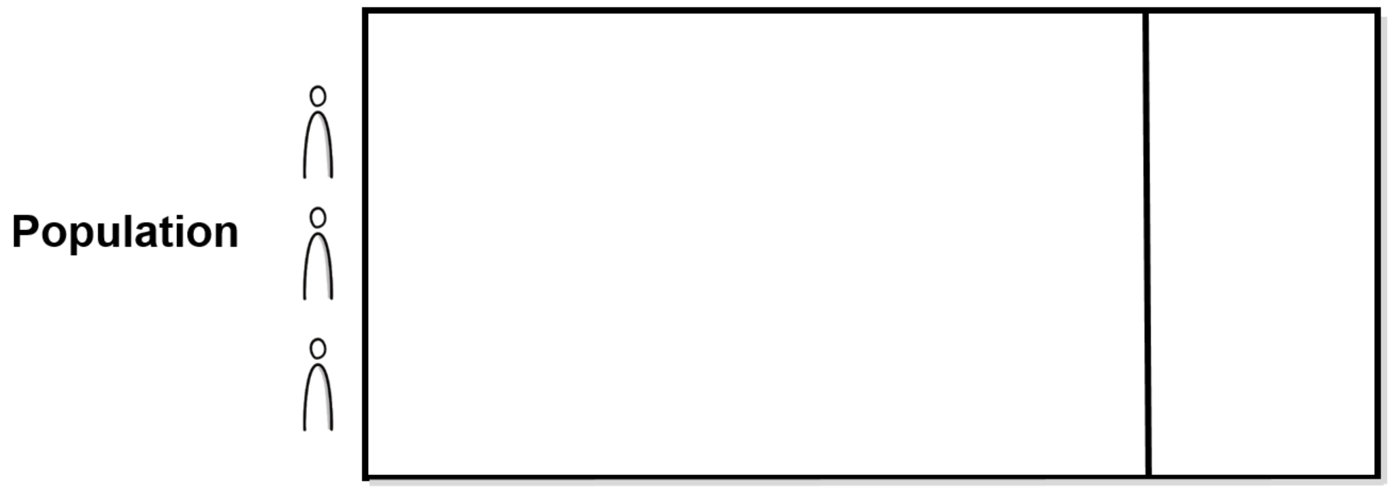
INTERMEDIATE PREDICTIVE ANALYTICS IN PYTHON

The population

INTERMEDIATE PREDICTIVE ANALYTICS IN PYTHON

Nele Verbiest Ph. D.
Senior Data Scientist @PythonPredictions

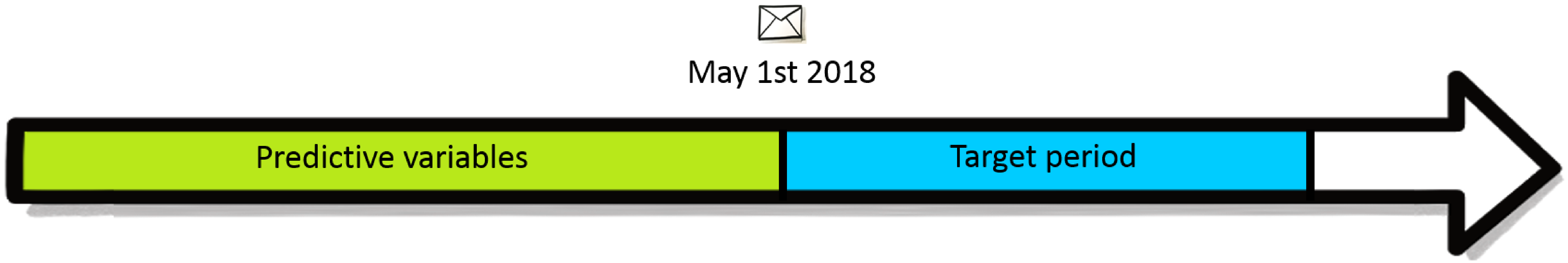
Population requirements



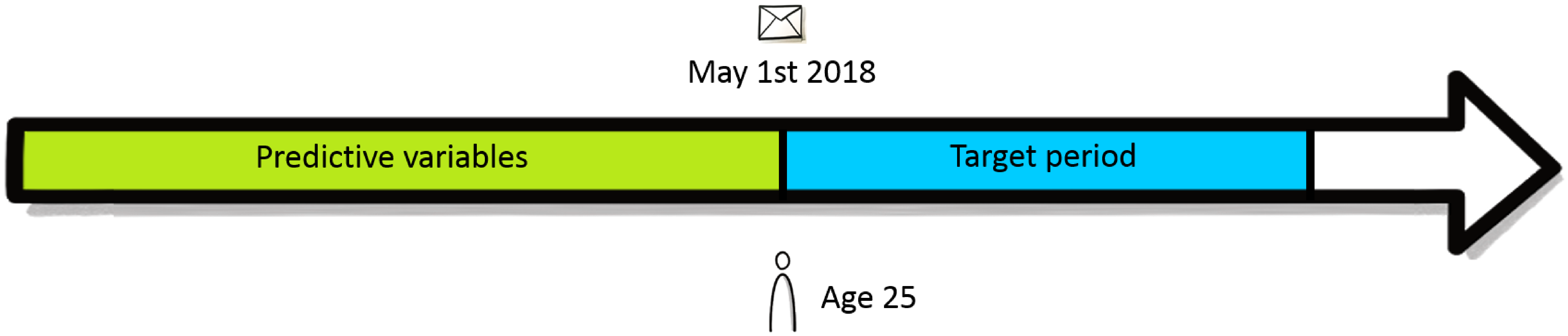
Population should be eligible for being target:

- Address available
- Privacy settings
- ...

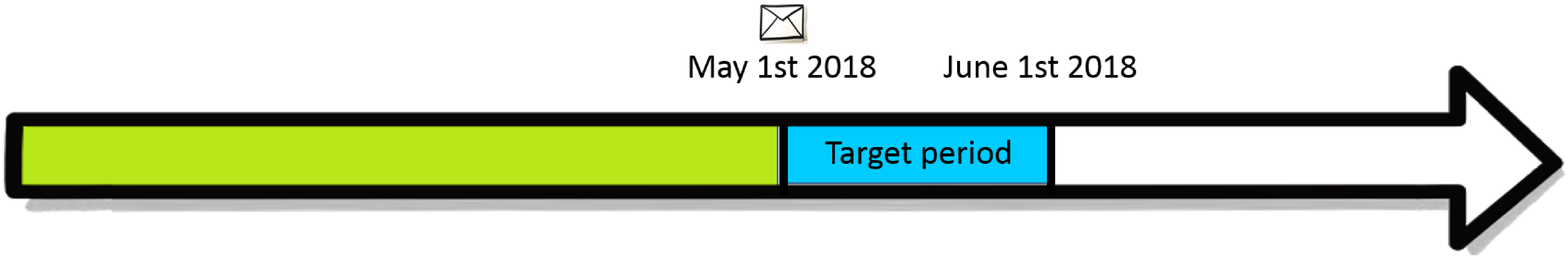
Timeline compliant population: age (1)



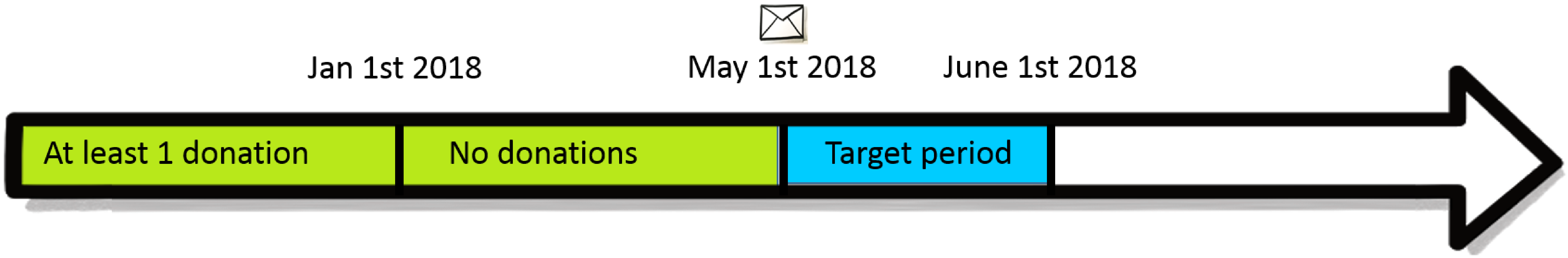
Timeline compliant population: age (2)



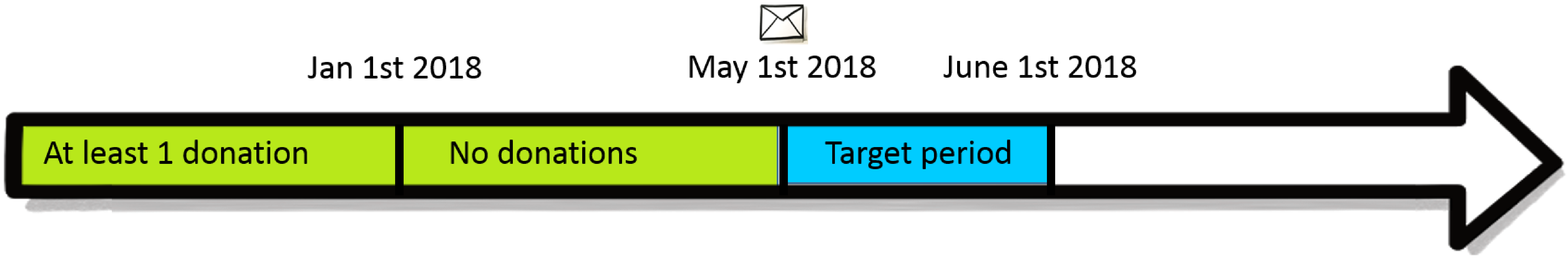
Timeline compliant population: donations (1)

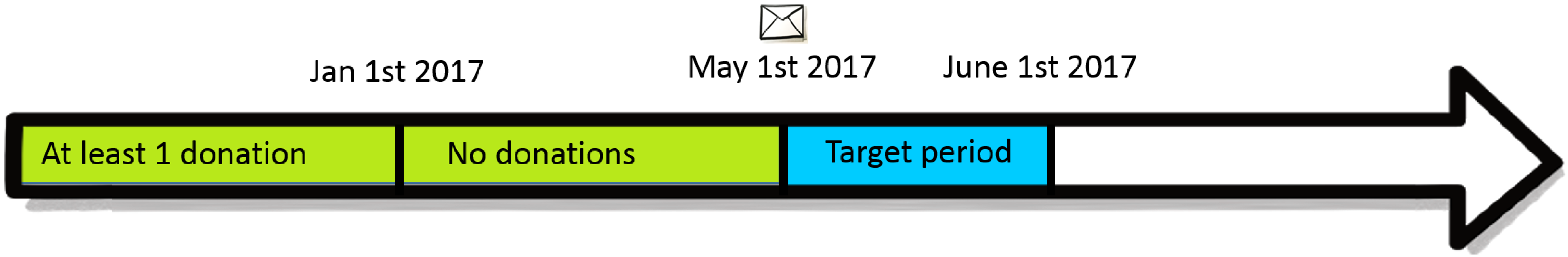


Timeline compliant population: donations (2)



Timeline compliant population: donations (3)





Population in python

```
donations_2016 = gifts[gifts["date"].dt.year==2016]
donors_include = set(donations_2016["id"])
print(donors_include)
```

```
{1002,3043,4934, ...}
```

```
donations_2017 = gifts[(gifts["date"].dt.year==2017)
                       & (gifts["date"].dt.month<5)]
donors_exclude = set(donations_2017["id"])
print(donors_exclude)
```

```
{2451,3047,4474, ...}
```

```
population = donors_include.difference(donors_exclude)
```

Let's practice!

INTERMEDIATE PREDICTIVE ANALYTICS IN PYTHON

The target

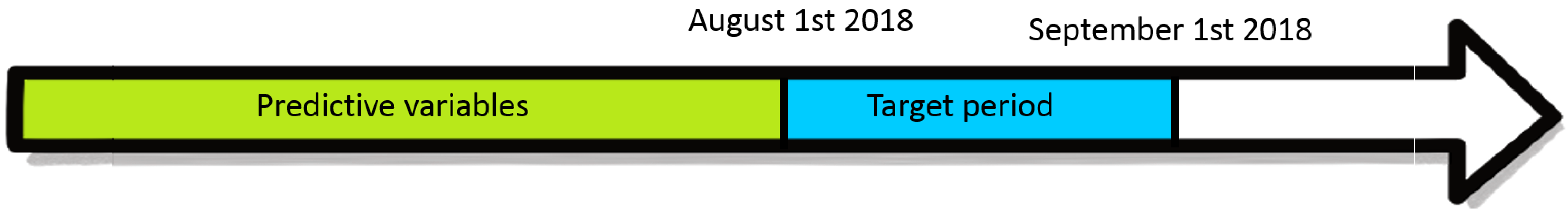
INTERMEDIATE PREDICTIVE ANALYTICS IN PYTHON

Nele Verbiest Ph. D.
Senior Data Scientist @PythonPredictions

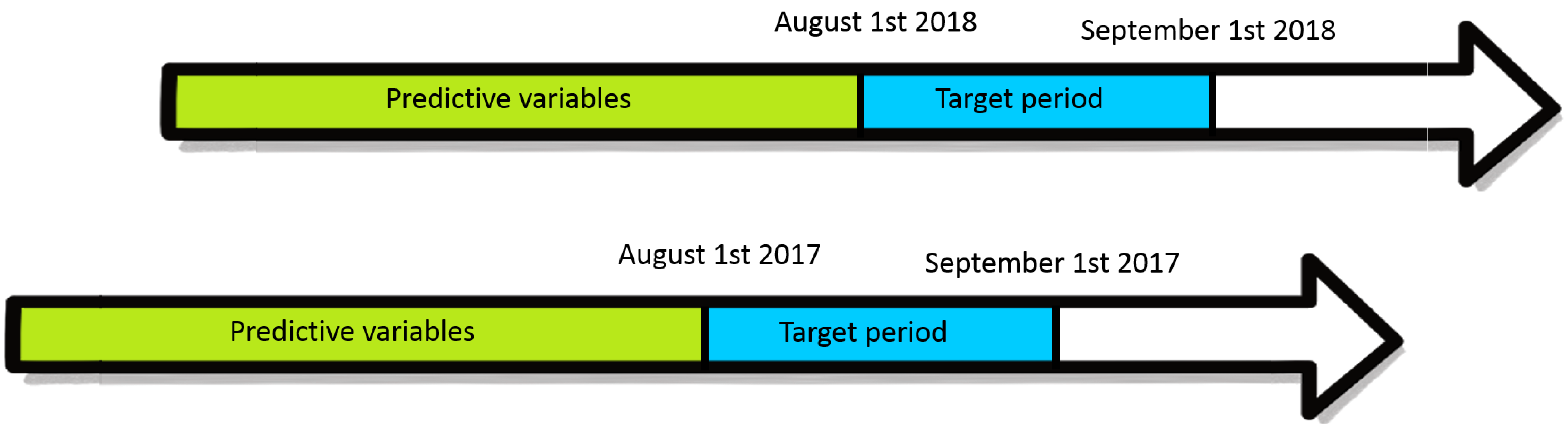
Target definition

Population

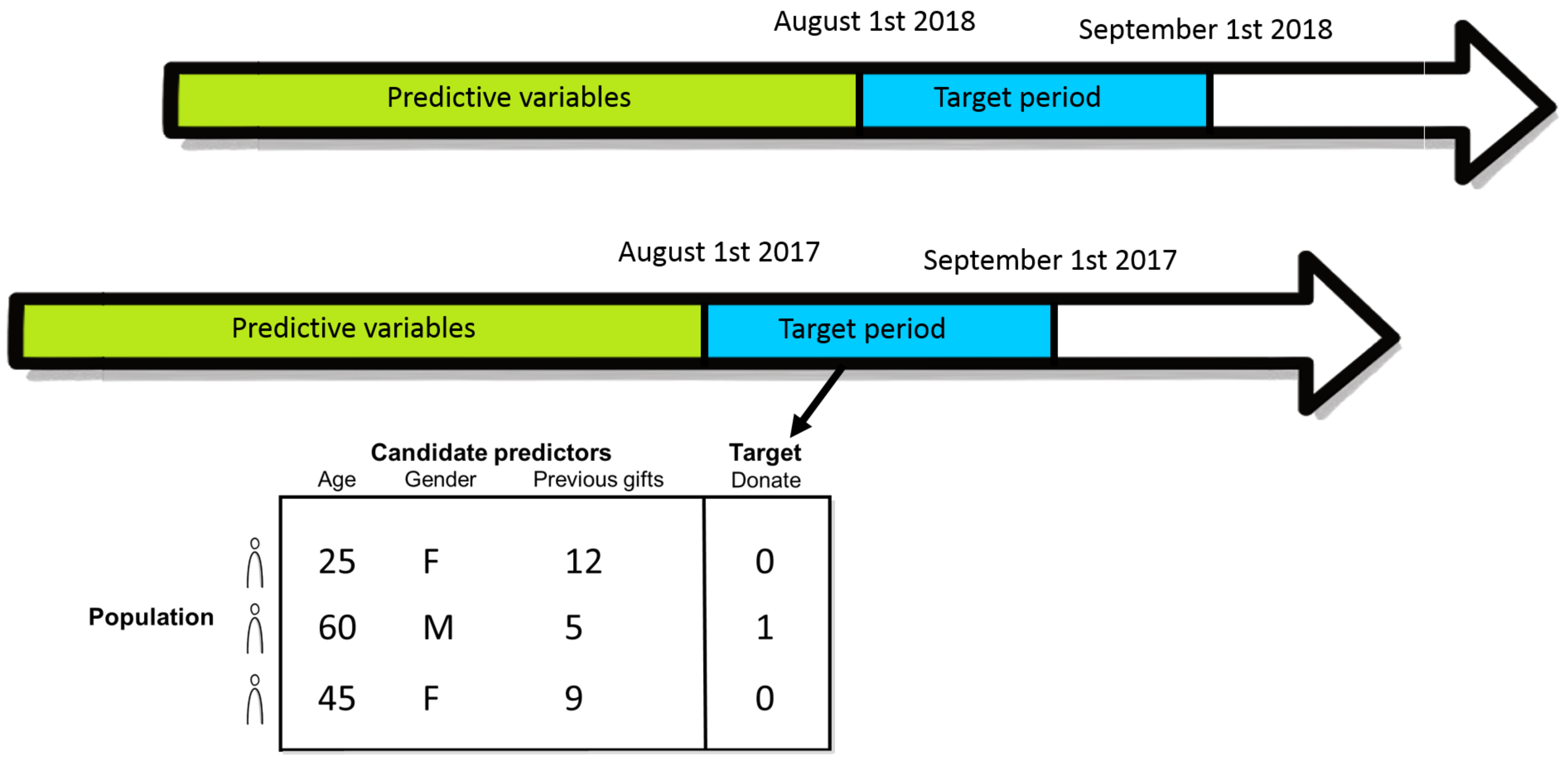
Target timeline (1)



Target timeline (2)



Target timeline (3)



	Candidate predictors			Target
	Age	Gender	Previous gifts	Donate
Population	25	F	12	0
	60	M	5	1
	45	F	9	0

Defining the target in Python

```
unsubscribe_2017[:5]
```

```
[90112, 65537, 24577, 8196, 73737]
```

```
baseteable.head()
```

```
   donor_id
0     65537
1     65538
2         4
3     98328
4     65564
```

```
basetable["target"] = pd.Series([1 if donor_id in unsubscribe_2017 else 0 \
for donor_id in basetable["donor_id"]])
```

Defining an aggregated target in Python

```
print(gifts.head(2))
```

```
   donor_id
0     65537
1     65538
```

```
# Target period
start_target = datetime(year = 2017, month = 1, day = 1)
end_target = datetime(year = 2018, month = 1, day = 1)
# Select target period donations
gifts_target = gifts[(gifts["date"]>=start_target) & (gifts["date"]<end_target)]
# Group and sum donations by donor
gifts_target_byid = gifts_target.groupby("id")["amount"].sum().reset_index()
# Derive targets and add to basetable
targets = list(gifts_target_byid["id"][gifts_target_byid["amount"]>500])
basetable["target"] = pd.Series([1 if donor_id in targets else 0 for donor_id in basetable["donor_id"]])
```

The basetable

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

```
   donor_id  target
0     65537  0
1     65538  1
2     65539  0
3     65540  1
4     65541  0
```

Let's practice

INTERMEDIATE PREDICTIVE ANALYTICS IN PYTHON