



FOUNDATIONS OF PREDICTIVE ANALYTICS IN PYTHON (PART 1)

Variable selection

Nele Verbiest, Ph.D

Data Scientist
Python Predictions



Candidate predictors

- `age`
- `max_gift`
- `income_low`
- `min_gift`, `mean_gift`, `median_gift`
- `country_USA`, `country_India`, `country_UK`
- `number_gift_min50`, `number_gift_min100`, `number_gift_min150`

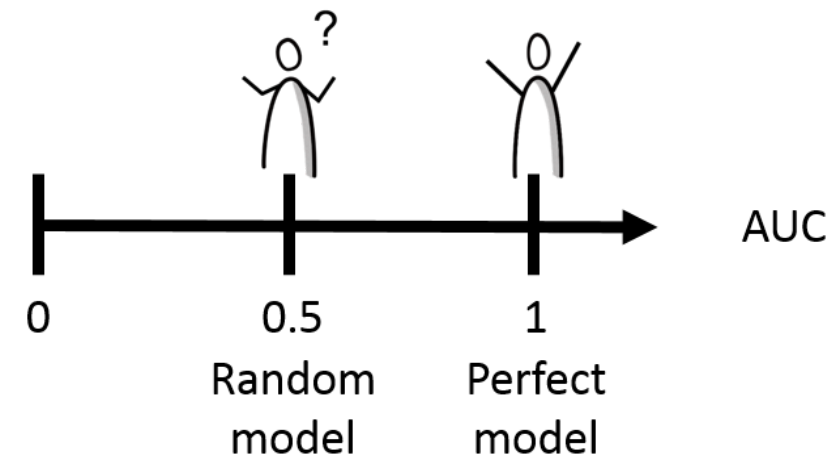


Variable selection: motivation

Drawbacks of models with many variables:

- Over-fitting
- Hard to maintain or implement
- Hard to interpret, multi-collinearity

Model evaluation: AUC



```
import numpy as np
from sklearn.metrics import roc_auc_score
roc_auc_score(true_target, prob_target)
```



FOUNDATIONS OF PREDICTIVE ANALYTICS IN PYTHON (PART 1)

Let's practice!



FOUNDATIONS OF PREDICTIVE ANALYTICS IN PYTHON (PART 1)

Forward stepwise variable selection

Nele Verbiest, Ph.D

Data Scientist
Python Predictions



The forward stepwise variable selection procedure

- Empty set
- Find best variable v_1
- Find best variable v_2 in combination with v_1
- Find best variable v_3 in combination with v_1, v_2
- ...

(Until all variables are added or until predefined number of variables is added)



Functions in Python

```
def function_sum(a,b):  
    s = a + b  
    return(s)  
  
print(function_sum(1,2))  
  
3
```




Implementation of the forward stepwise procedure

- Function `auc` that calculates AUC given a certain set of variables
- Function `best_next` that returns next best variable in combination with current variables
- Loop until desired number of variables



Implementation of the AUC function

```
from sklearn import linear_model
from sklearn.metrics import roc_auc_score

def auc(variables, target, basetable):

    X = basetable[variables]
    y = basetable[target]

    logreg = linear_model.LogisticRegression()
    logreg.fit(X, y)

    predictions = logreg.predict_proba(X)[:,1]
    auc = roc_auc_score(y, predictions)
    return(auc)

auc(["age", "gender_F"], ["target"], basetable)
print(round(auc, 2))

0.54
```

Calculating the next best variable

```
def next_best(current_variables, candidate_variables, target, basetable):  
  
    best_auc = -1  
    best_variable = None  
  
    for v in candidate_variables:  
        auc_v = auc(current_variables + [v], target, basetable)  
  
        if auc_v >= best_auc:  
            best_auc = auc_v  
            best_variable = v  
  
    return best_variable  
  
current_variables = ["age", "gender_F"]  
candidate_variables = ["min_gift", "max_gift", "mean_gift"]  
next_variable = next_best(current_variables, candidate_variables, basetable)  
print(next_variable)  
  
min_gift
```



The forward stepwise variable selection procedure

```
candidate_variables = ["mean_gift", "min_gift", "max_gift",  
"age", "gender_F", "country_USA", "income_low"]  
current_variables = []  
target = ["target"]  
  
max_number_variables = 5  
number_iterations = min(max_number_variables, len(candidate_variables))  
for i in range(0, number_iterations):  
    next_var = next_best(current_variables, candidate_variables, target, basetable)  
    current_variables = current_variables + [next_variable]  
    candidate_variables.remove(next_variable)  
  
print(current_variables)  
  
['max_gift', 'mean_gift', 'min_gift', 'age', 'gender_F']
```



FOUNDATIONS OF PREDICTIVE ANALYTICS IN PYTHON (PART 1)

Let's practice!



FOUNDATIONS OF PREDICTIVE ANALYTICS IN PYTHON (PART 1)

Deciding on the number of variables

Nele Verbiest, Ph.D

Data Scientist
Python Predictions



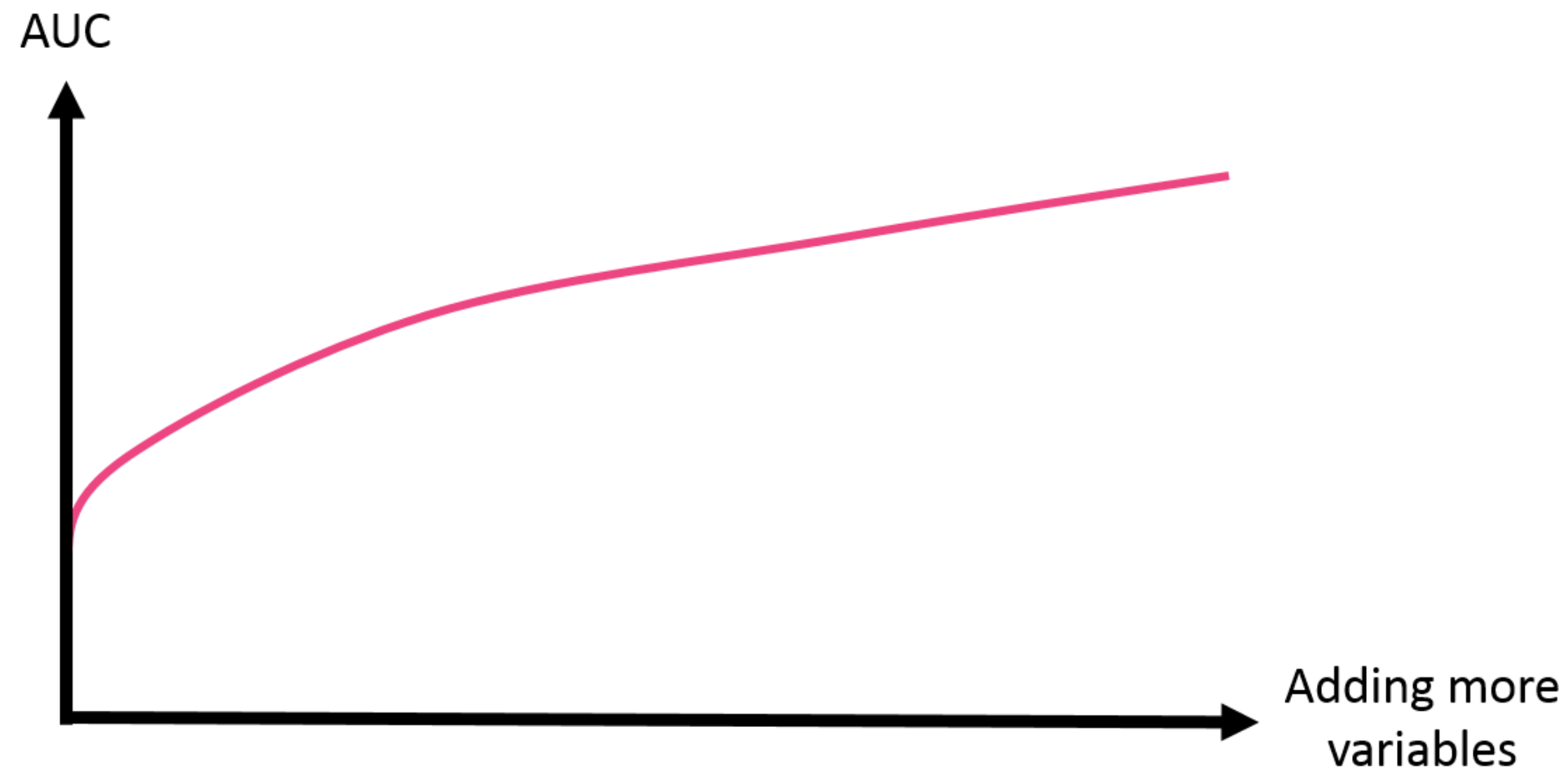
Evaluating the AUC

```
auc_values = []
variables_evaluate = []

for v in variables_forward:
    variables_evaluate.append(v)
    auc_value = auc(variables_evaluate, ["target"], basetable)
    auc_values.append(auc_value)
```

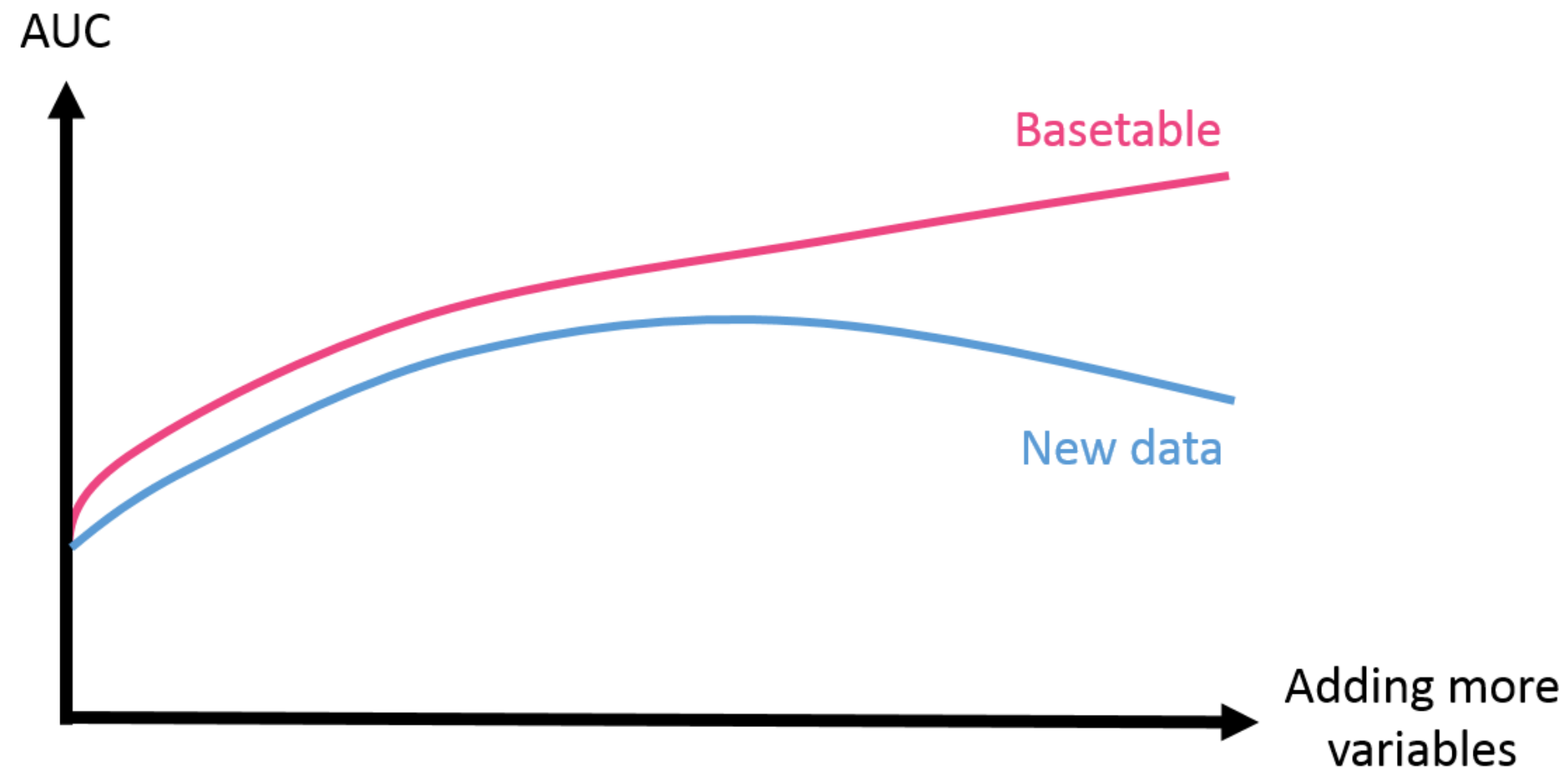


Evaluating the AUC



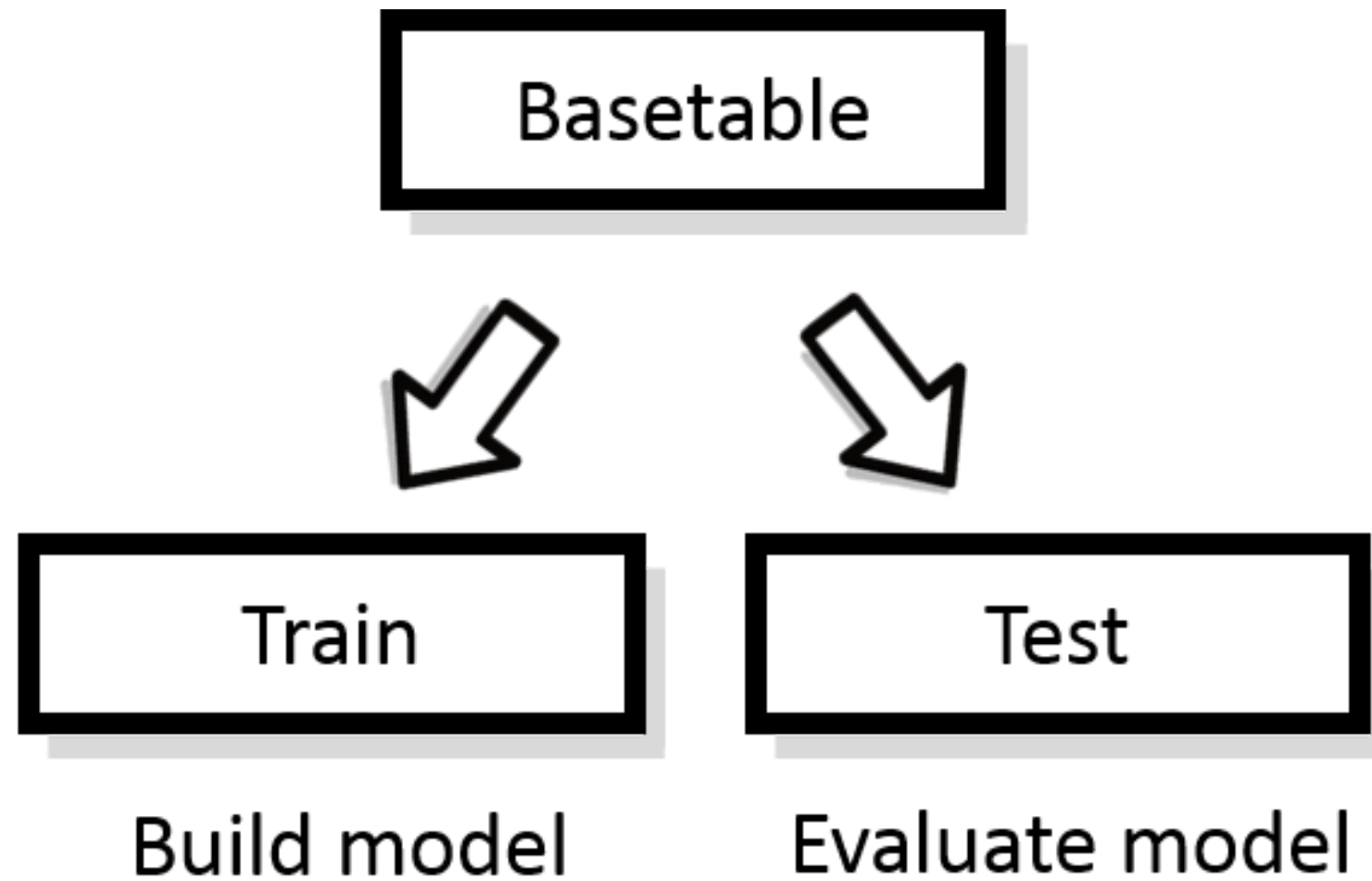


Over-fitting





Detecting over-fitting





Partitioning

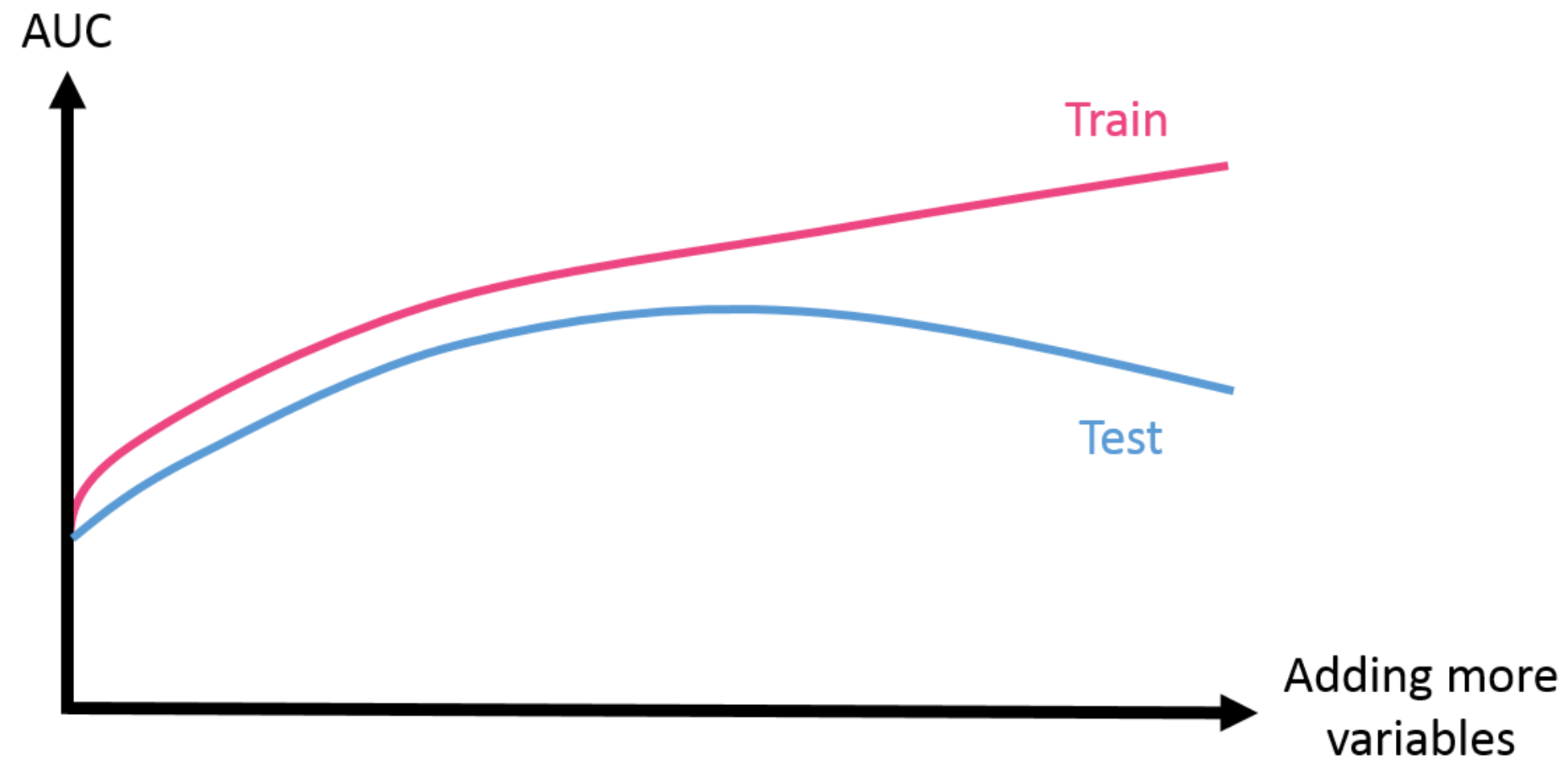
```
from sklearn.cross_validation import train_test_split

X = basetable.drop("target", 1)
y = basetable["target"]

X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size=0.4, stratify = Y)

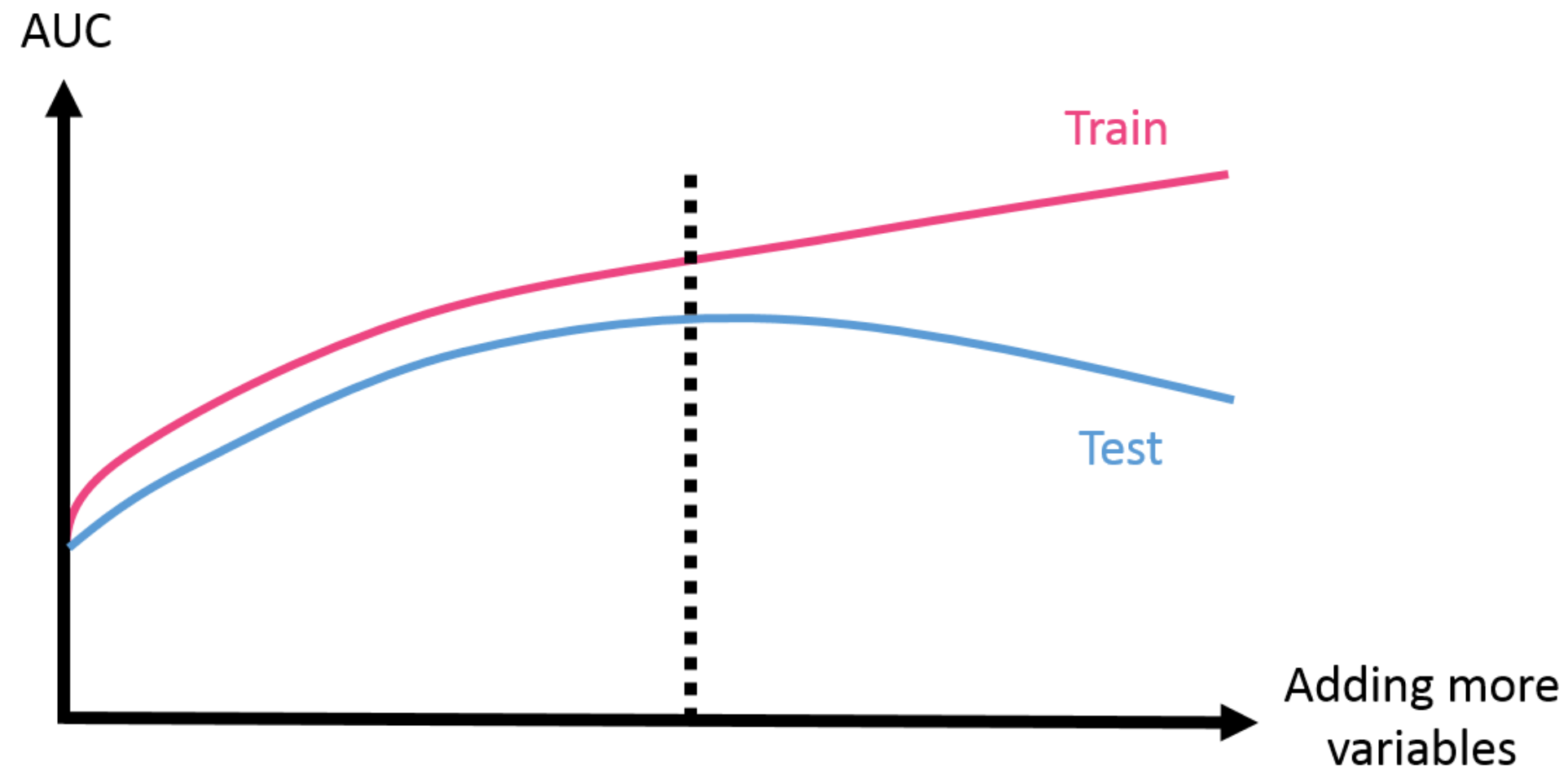
train = pd.concat([X_train, y_train], axis=1)
test = pd.concat([X_test, y_test], axis=1)
```

Deciding the cut-off



- High test AUC
- Low number of variables

Deciding the cut-off





FOUNDATIONS OF PREDICTIVE ANALYTICS IN PYTHON (PART 1)

Let's practice!