Recipe ”Get the data” step: subtleties

In describing the "Get the data" step

- we hinted at the *mechanical* difficulties of gathering examples

Beyond these difficulties, it is important that

- the examples used in training
    - and for evaluating the Performance Metric: test examples
- should be representative of the "out of sample" examples on which we want to predict in the future

We motivate this statement below.

Fundamental assumption of Machine Learning

Our goal is to learn (from training examples) to make a good prediction on a never before seen *test* example.

A necessary condition is that the training examples are representative of the future test examples we will encounter.

Let's imagine that there is some true (but unknown) distribution p_{data} of feature/label pairs $(\mathbf{x}, \mathbf{y})$.

In order to learn, we must assume

- The *training examples* are a sample drawn from p_{data}.
- Each "out of sample" example that will occur *post-training* is drawn from p_{data}

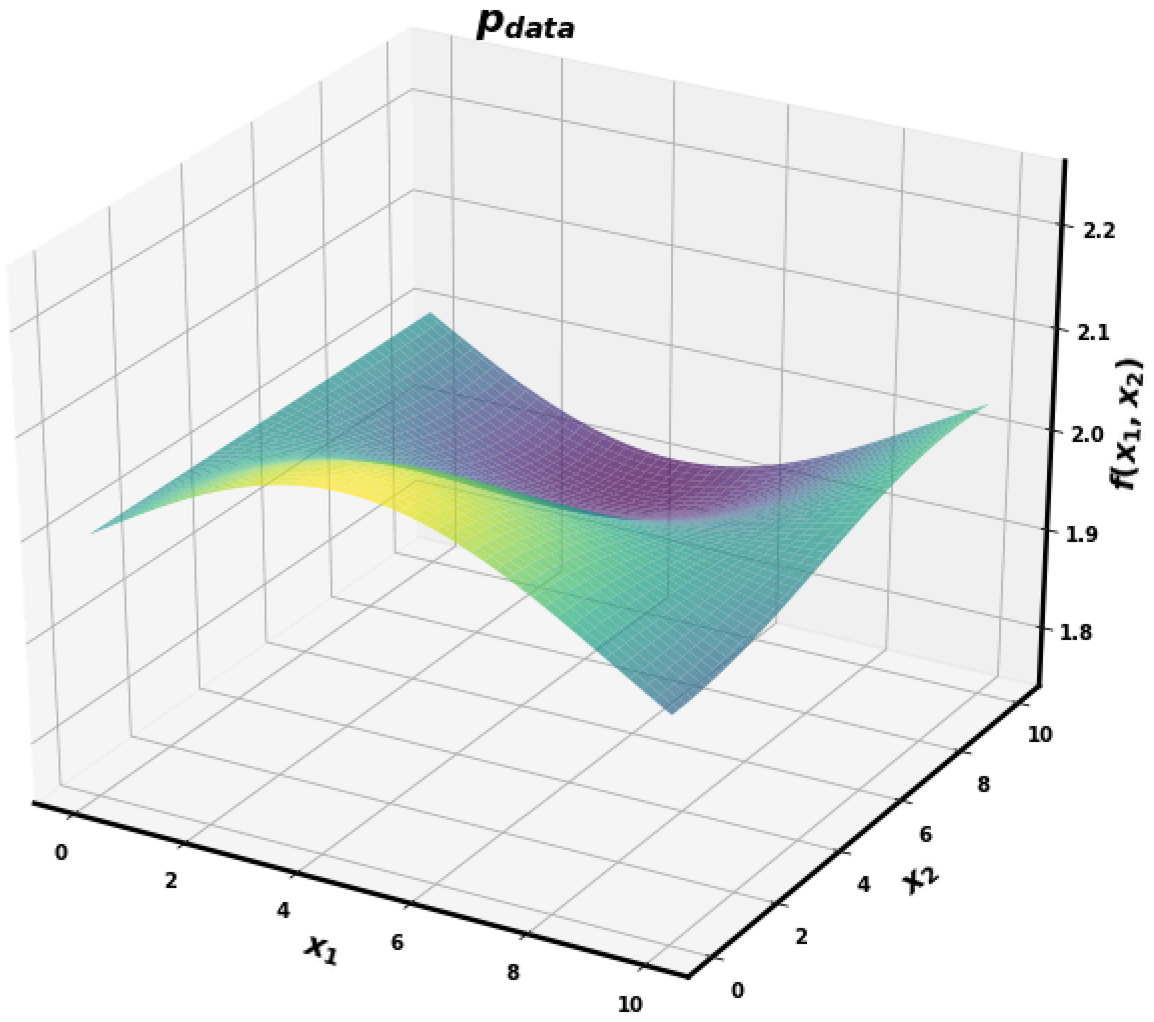
We sometimes call the training data an *empirical* distribution -- it is just a sample, not the "true" distribution.

Let's imagine a complex distribution where the target is a function of two features

$$\mathbf{y} = f(\mathbf{x}_1, \mathbf{x}_2)$$

In [6]:

```
fig, ax, X, Y, Z = draw_surface(title='$p_{data}$')
```



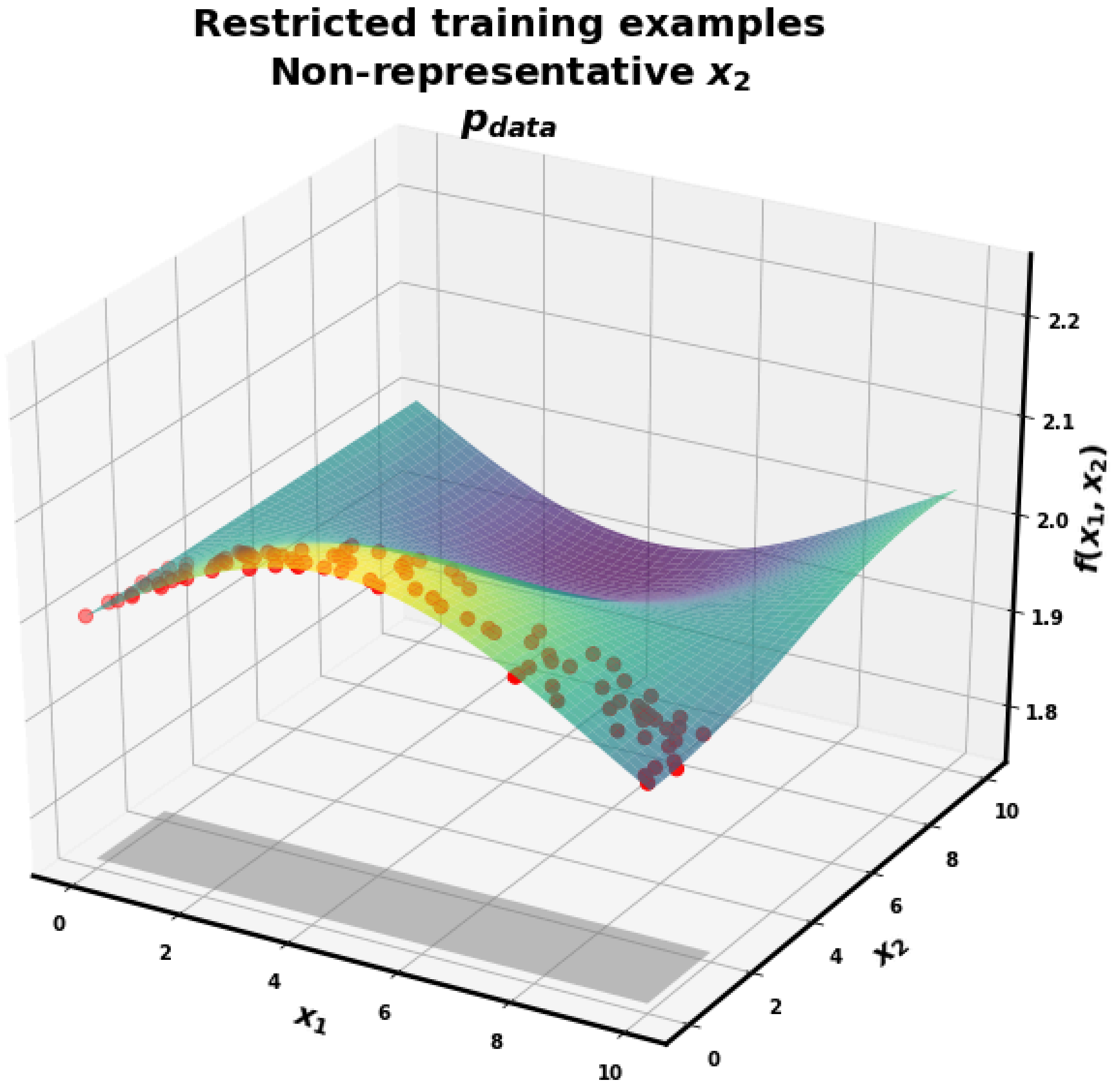
But suppose our training examples where drawn *only from a small subset* of $\mathbb{R} \times \mathbb{R}$

We would

- might learn to successfully predict *only* from the subset represented in the training dataset.
- have no guarantee as to the values predicted for feature vectors $\mathbf{x}$ very different than those in the training dataset

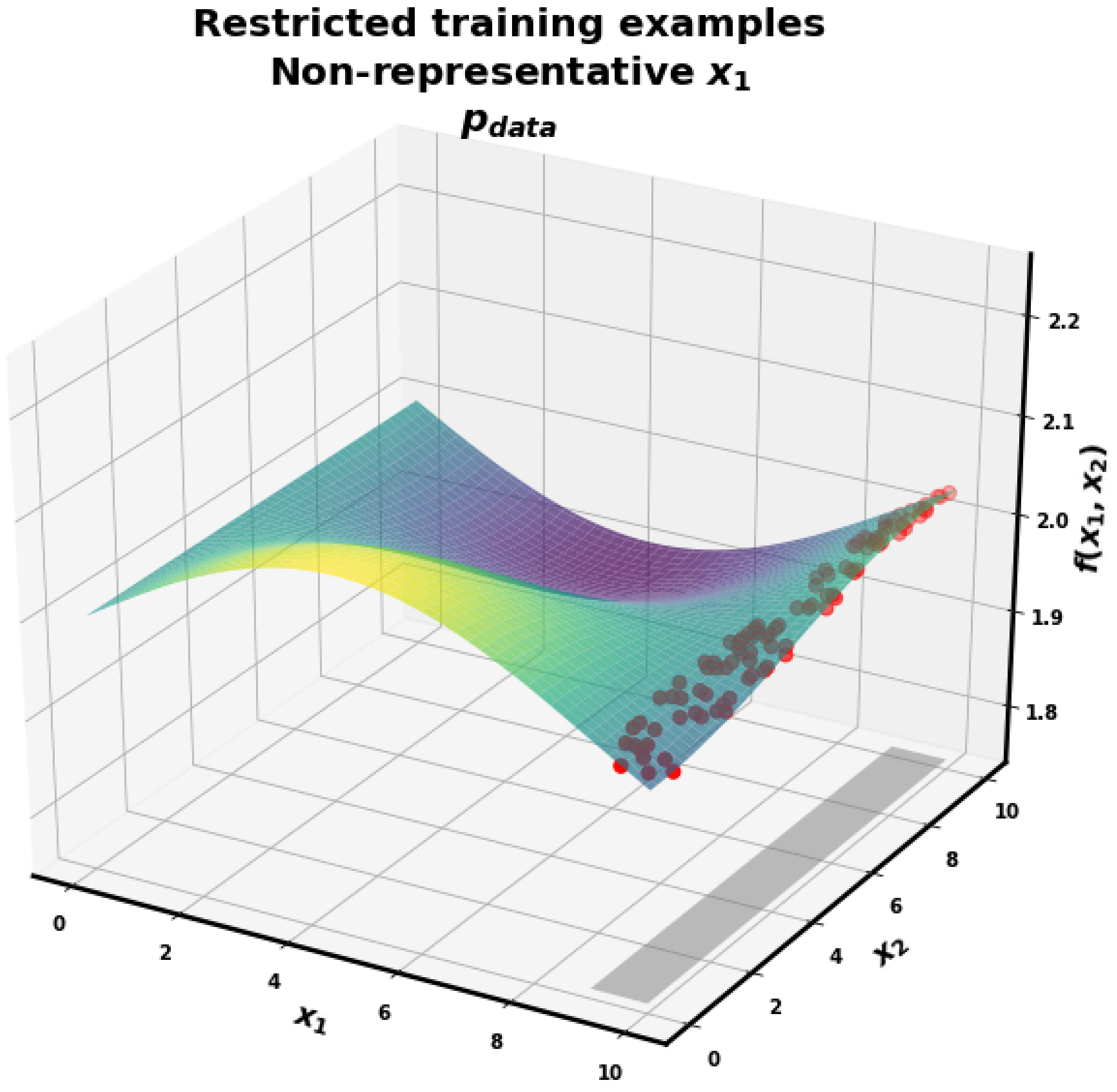
In [7]:

```
fig, ax, X, Y, Z = draw_surface(title='Restricted training examples\nNon-representative $x_2$\n$p_{data}$')

_, _ = add_random(fig, ax, X, Y, 0, X.max(), 0, 2)
_, _ = add_shaded(fig, ax, 0, X.max(), 0, 2)
```


Restricted training examples
Non-representative x_2
p_{data}
$f(x_1, x_2)$
2.2
2.1
2.0
1.9
1.8
x_1
0
2
4
6
8
10
x_2
0
2
4
6
8
10

In [8]:

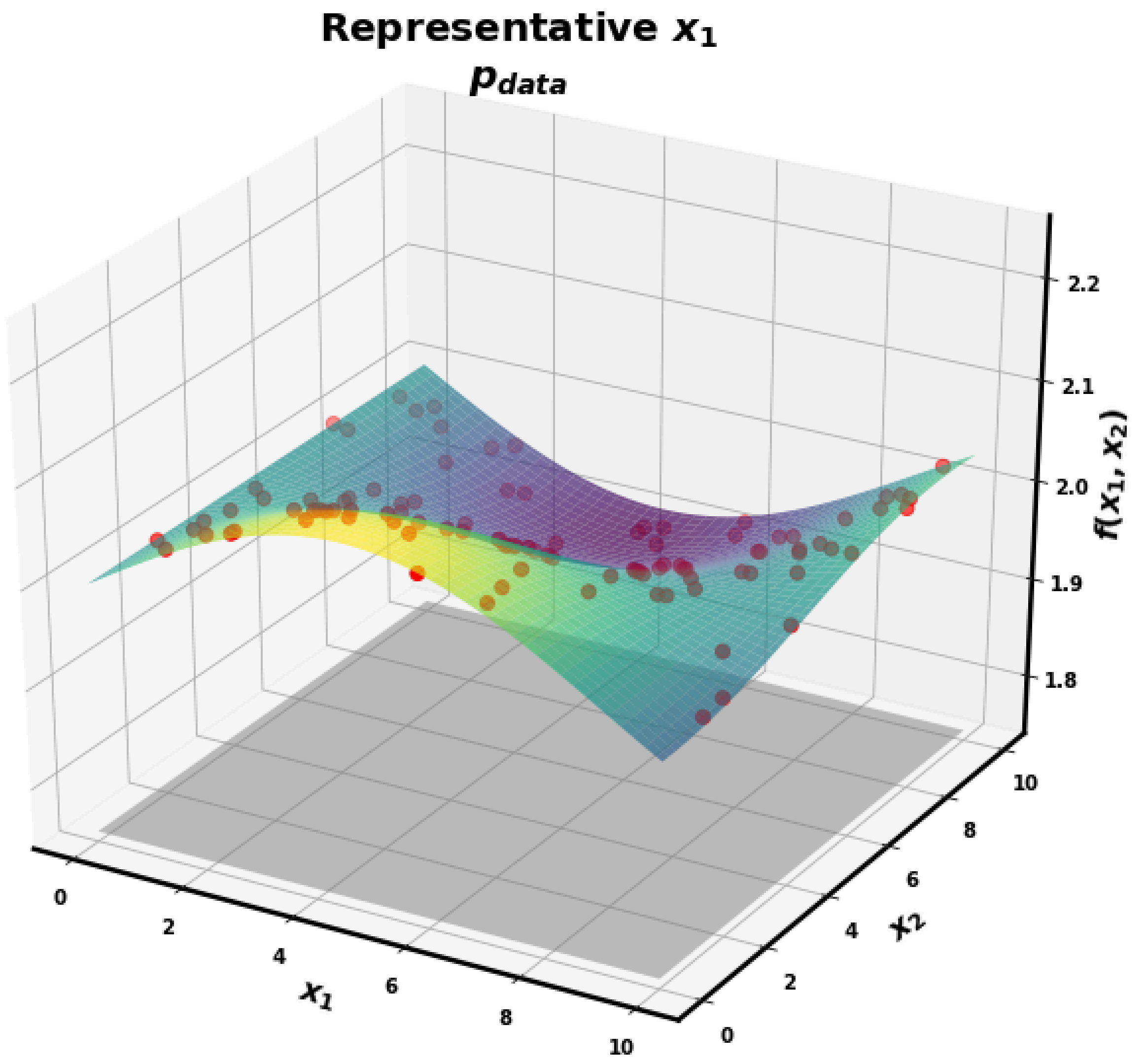
```
fig, ax, X, Y, Z = draw_surface(title='Restricted training examples\nNon-representative $x_1$\n$p_{data}$')

_, _ = add_random(fig, ax, X, Y, 9, X.max(), 0, Y.max())
_, _ = add_shaded(fig, ax, 9, X.max(), 0, Y.max())
```


Restricted training examples
Non-representative x_1
p_{data}
$f(x_1, x_2)$
2.2
2.1
2.0
1.9
1.8
x_1
0
2
4
6
8
10
x_2
0
2
4
6
8
10

In order to successfully predict over a representative range of $\mathbb{R} \times \mathbb{R}$

- the training examples should be sufficiently diverse

In [9]:
```
fig, ax, X, Y, Z = draw_surface(title='Representative $x_1$\n$p_{data}$')

_, _ = add_random(fig, ax, X, Y, X.min(), X.max(), Y.min(), Y.max())
_, _ = add_shaded(fig, ax, X.min(), X.max(), Y.min(), Y.max())
```



That is: our model can only generalize based on training examples

- The training examples need to be representative of unseen examples in the wild in order to generalize well
- Larger training sets are preferred as they may be more representative of the true p_{data}
    - They should also be diverse

If the test example $\mathbf{x}$ is *not* from p_{data}, the model is unconstrained in its prediction.,

Fundamental Assumption: Finance

You should not take for granted the satisfaction of this assumption !

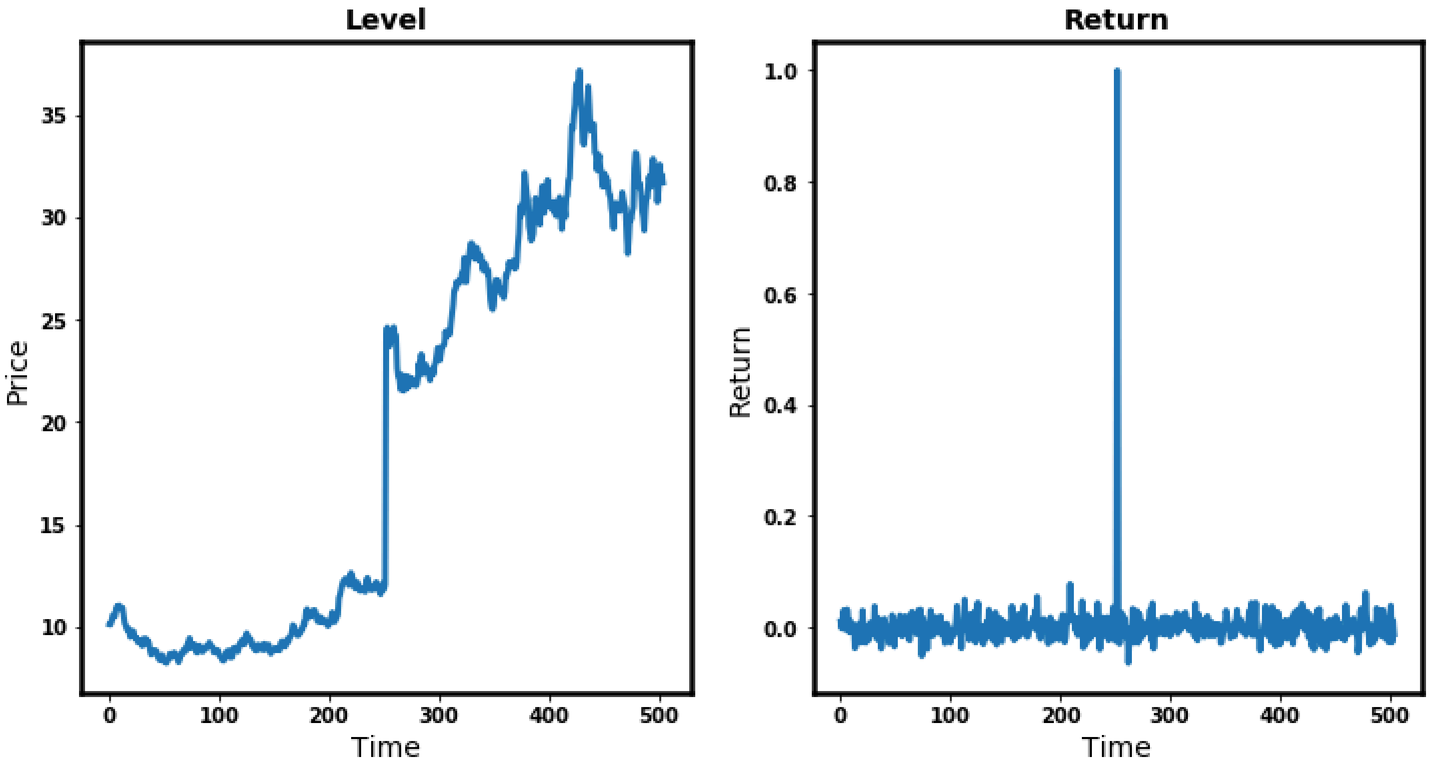
- It is very easy in Finance
- to inadvertently collect *non-representative* examples

Suppose we want to predict the future price of a stock using only past prices.

Consider the following price series:

In [10]:
```
fig_data
```

Out[10]:



From the Price Levels, you can see that there is a one-time jump in prices.

If your training examples were from the time before the jump and your "future" test examples were from after the jump

- the examples don't come from the same Price distribution
    - at a minimum: the mean prices are different
- using Price/Level as features/targets
    - can easily lead to violations of the Fundamental Assumption

Fortunately

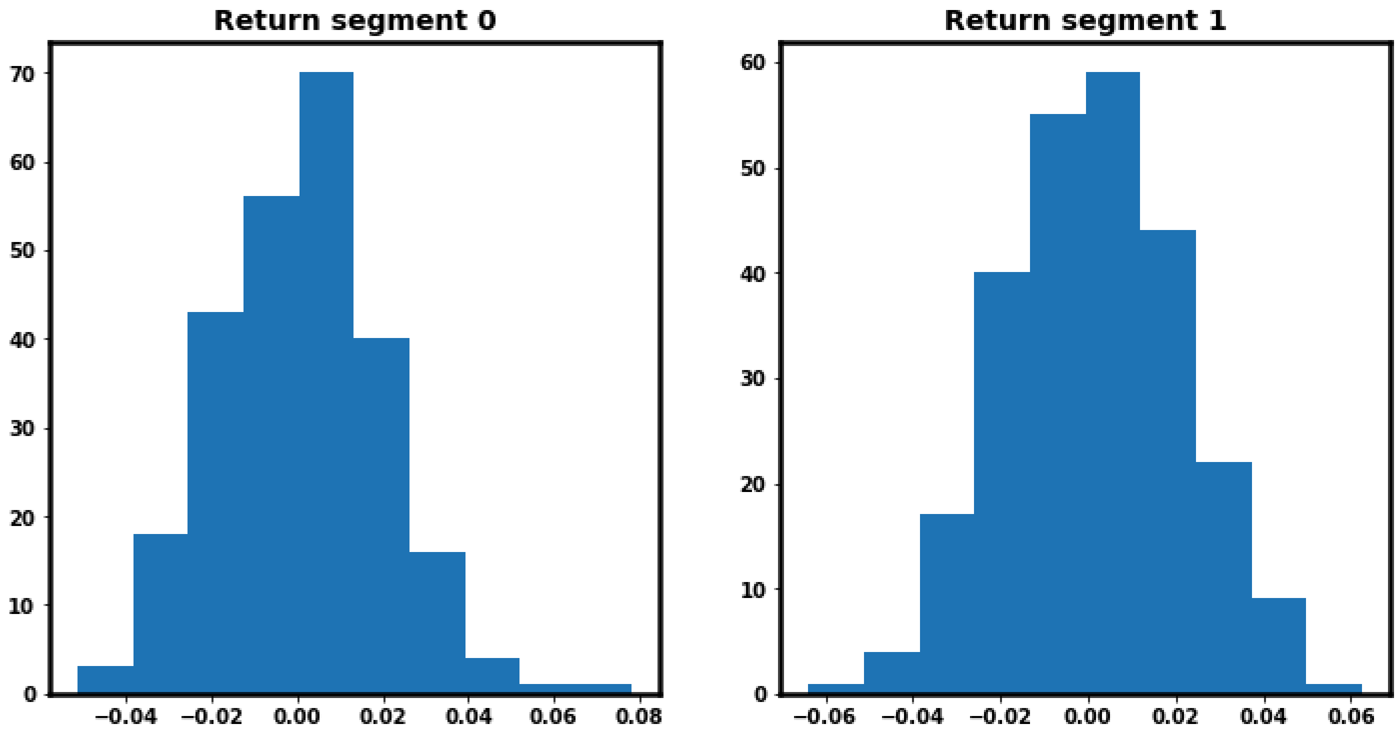
- there is often a simple *transformation* of raw features/targets
- into synthetic features/targets
- that *will* satisfy the Fundamental Assumption.

Let's examine

- the distribution of *returns*
- before and after the price jump

```
In [11]: fig_segs
```

Out[11]:



Very well could be the same (we can test for equality of moments to be sure).

So training a model to predict future returns from past returns

- Would satisfy the assumption
- We can readily convert from returns back to levels

Aside

Price jumps happen for many reasons.

- New product introduction
- New business model
- Dividend payout
- Company is the target of a take-over offer

A more frequent scenario is that data drifts over time rather than jumping suddenly.

Whatever the cause, we need to induce some stability over the data.

- returns are more stable than prices

Recipe ”Prepare the data/Transformations” step

This example illustrates another point.

Sometimes a *raw feature/target* (e.g., Level) need to be *transformed* into a synthetic feature/target (e.g. Return).

A successful Data Scientist needs to master the process of transforming data

- The "secrets" that need to be uncovered might not lie at the surface

The Recipe's "Transformation" step (sub-step of "Prepare the Data")

- is where raw features/targets
- are *transformed* into synthetic features/targets
- which may have more desirable properties for prediction
    - e.g., satisfy Fundamental Assumption

As we will see

- this step can also add new synthetic features
- drop features

Transformations are a very important part of Data Science

- to be covered in a future module

In [12]:
```
print("Done")
```

```
Done
```