Transformations

It is often the case that the "raw" features given to us don't lead to an adequately successful model

- We may need to create "synthetic" features.
- Or filter out existing features
- This is called **feature engineering**.

Knowing how to perform feature engineering is a key skill of a Data Scientist.

As motivation, consider a Binary Classification task

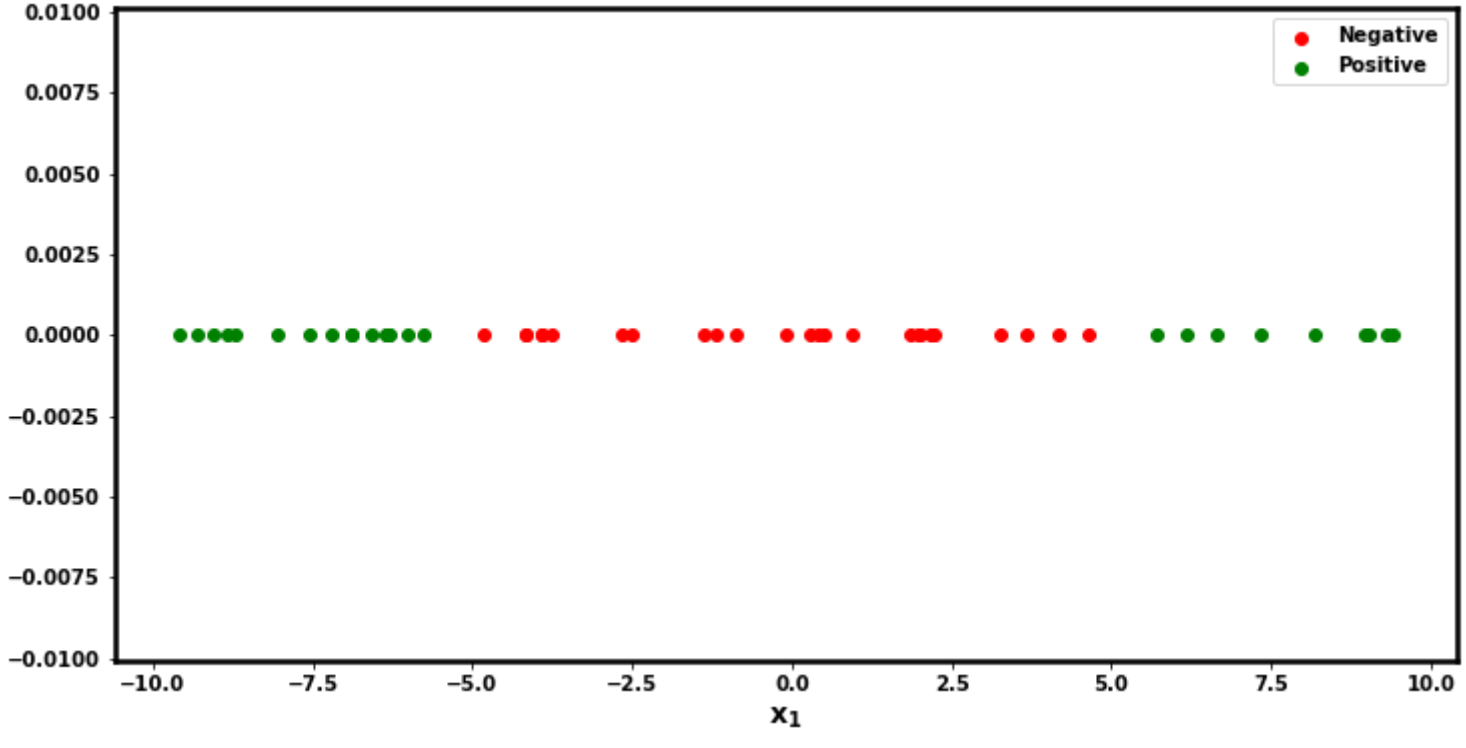
- Examples have a single numeric feature $\mathbf{x}_1$
- Target $\mathbf{y}$ in classes Positive and Negative
    - encoded as 1 and 0

It should be clear from the plot that the classes are not linearly separable.

In [16]:
```
th = transform_helper.Transformation_Helper()
fig_raw, ax_raw, fig_trans, ax_trans = th.LinearSeparate_1d_example(max_val=10,
num_examples=50, visible=False)

fig_raw
```

Out[16]:

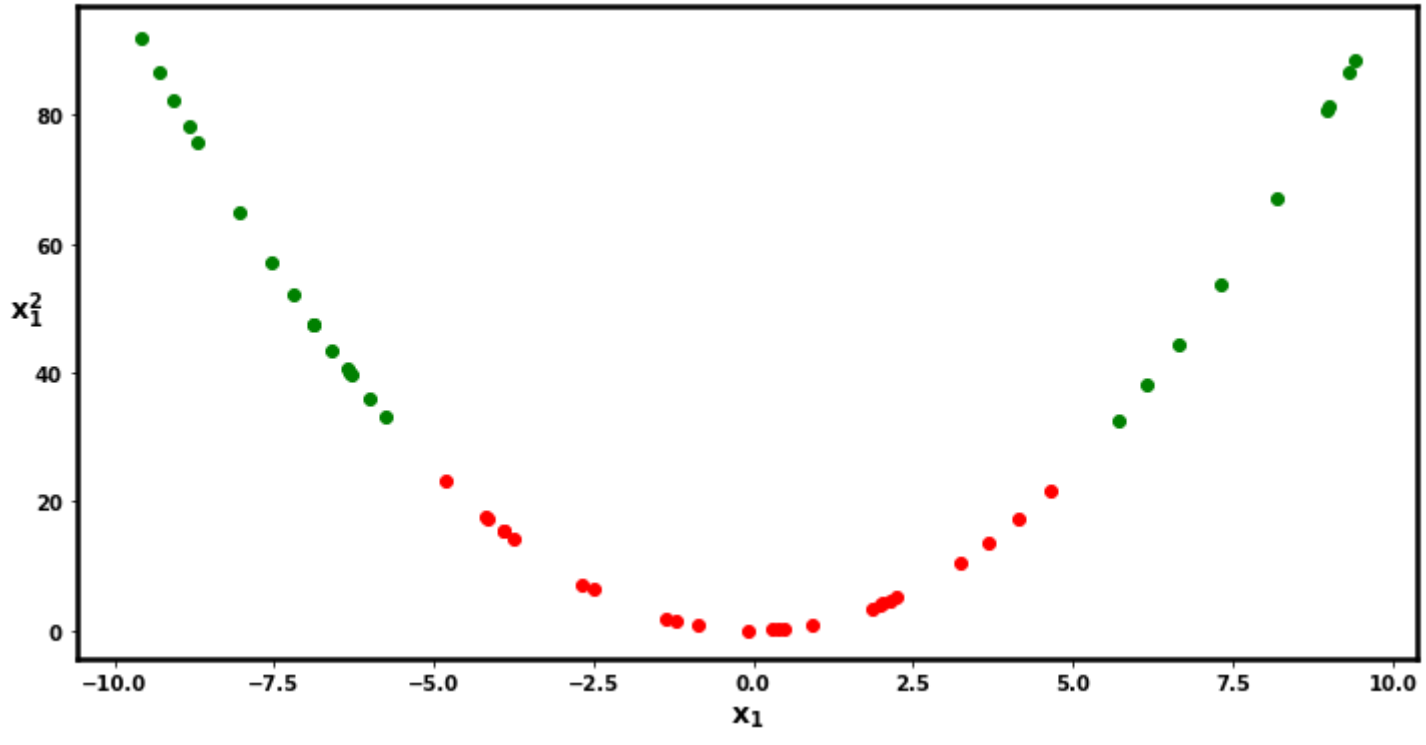


But consider the simple transformation that adds a second synthetic feature $\mathbf{x}_2$

- $\mathbf{x}_2 = \mathbf{x}_1^2$

In [17]:
```
fig_trans
```

Out[17]:



The transformed examples are now linearly separable.

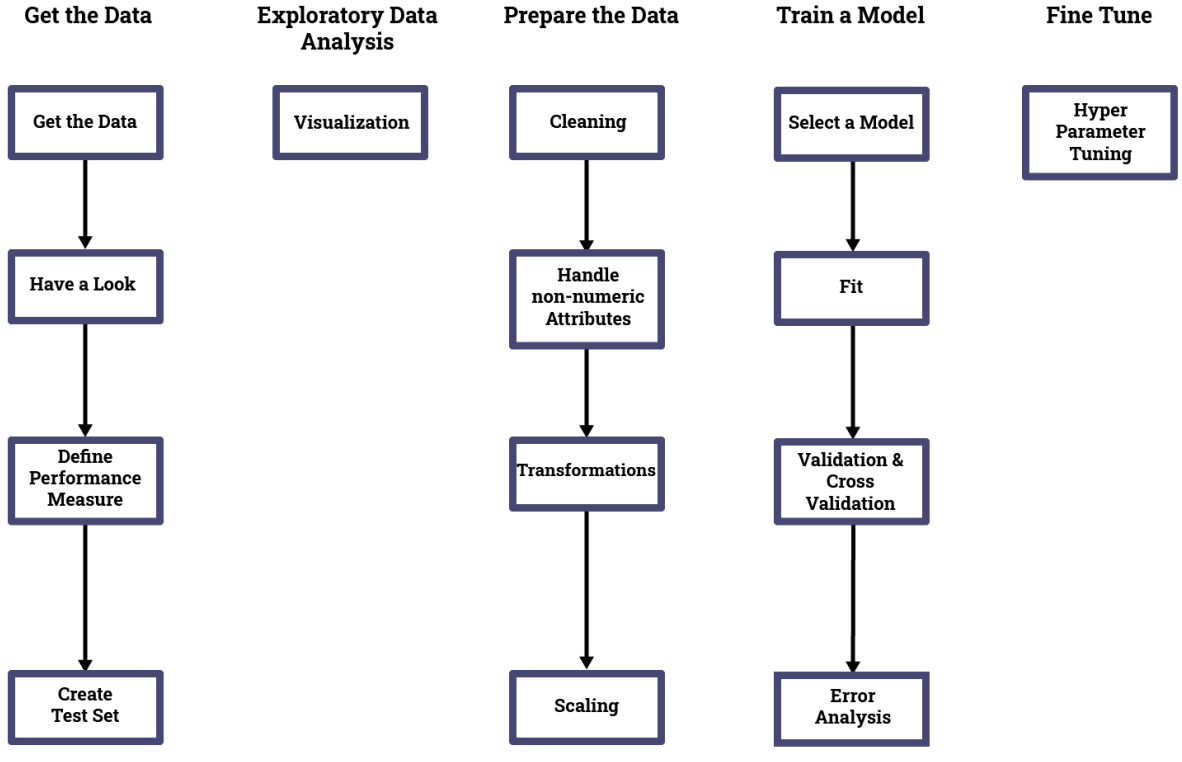
We focus on the process of transforming raw examples into a form that facilitates successful modeling.

We broaden the term *Transformation* to include

- Cleaning
- Numericalization of non-numeric features/targets
- Scaling

(These are distinct steps in the Recipe, but have similar API)

Recipe for Machine Learning



Our focus is mainly on the **why** rather than the **how**

Let's briefly review the *how*:

- Mechanics of transformations (Transformations_Mechanics.ipynb)

For a refresher on implementation in `sklearn`

- Revisit our previous module Coding transformations in `sklearn` (Transformations_Pipelines.ipynb))

The why's of transformations

We illustrate several useful types of transformations below.

This is organized more as a "case study" than a taxonomy.

- Many transformations have multiple uses, making a hard taxonomy difficult

Making the data fit your model's assumptions

Some models work under the assumption that there is a linear relationship between features and the target.

When this assumption doesn't hold, the "fit" is sub-optimal (prediction error)

Sometimes: transformations can induce linearity

- Add a squared term to make Linear Regression fit the "curvy" dataset better
- Changing the target to log odds to enable Linear Regression to work for Classification tasks

Let's visit the notebook section Inducing linearity (Transformations.ipynb#Linearity-inducing-transformations:-Making-data-fit-your-model)

Missing numeric feature

Sometimes, your examples have all the "information" you need, but in the wrong form.

Creating new "synthetic" features from raw features is one way of making this information available to the model.

Let's visit the notebook section Missing numeric feature (Transformations.ipynb#Transformation-to-add-a-%22missing%22numeric-feature)

Missing categorical feature: ”group” indicator

identical up to an additive constant

There is a more subtle case of a missing feature

- Examples that naturally partition into sub-groups

The sub-groups might be examples that come from similar geographies or points in time.

The key is that

- The relationship between target and features is *almost identical* between groups
- With the exception of a constant shift

Adding indicator/dummy variables as new features to indicate sub-group membership may be a solution

Let's visit the notebook section Missing Categorical Feature (Transformations.ipynb#Transformation-to-add-a-%22missing%22-categorical-feature)

Cross features

We witnessed the power of a simple indicator feature to isolate differences between groups of examples.

It is possible to

- Create multiple indicator features
- Create indicator features that are the *product* of other indicators

This is called a *cross feature* and is a powerful way to isolate complicated sub-groups of examples.

Let's visit the notebook section Cross features (Transformations.ipynb#Cross-features)

Scaling

There is a class of transformations that alter the *scale* (magnitude) of features or targets.

In the Recipe for ML, Scaling is treated as separate from the other transformations.

Perhaps this is because scaling is sometimes performed

- *Not* strictly because of the relationship between target and features
- But because of the mathematics of the *loss function*

Let's visit the notebook section Scale sensitive loss functions (http://localhost:8888/notebooks/NYU/Transformations.ipynb#Scaling)

Normalization

We will use the term "normalization" (non-standard terminology ?) to refer to a transformation that re-scales examples by different amounts

- As opposed to traditional "scaling" where all examples are scaled equally

Let's visit the notebook section Normalization (Transformations.ipynb#Normalization)

Other transformations

The continuation of the linked notebook describes more transformations, listed below.

We encourage you to study these.

We especially recommend the section Feature engineering example (Transformations.ipynb#Feature-engineering-example:-Geron-Housing-Data) which is a worked example from one of the recommend textbooks.

Other transformations

- Normality inducing transformaton (Transformations.ipynb#Normality-inducing-transformations)
- Categorical variable transformation (Transformations.ipynb#Categorical-transformation)
- Transformations to induce normality (Transformations.ipynb#Normality-inducing-transformations)
- Other transformation (Transformations.ipynb#Other-transformations)

In [6]:
```
print("Done")
```

```
Done
```