Data science in practice

Being a successful Data Scientist is much more than just knowing models.

Being able to gather, join, and clean Data is a prerequisite to being able to train a model.

- the remarkable success of Large Langauge Models (e.g., ChatGPT) is in large part due to vast quantities of carefully curated data

Manipulating and transforming the Data into a form in which a successful model can be built is a skill that may not be truly appreciated.

- data is frequently transformed from its raw form to synthesized features and targets
- the relationship between features and target may not be present in raw form

It is this latter skill that we know focus on.

Recall the Fundamental Assumption of Machine Learning

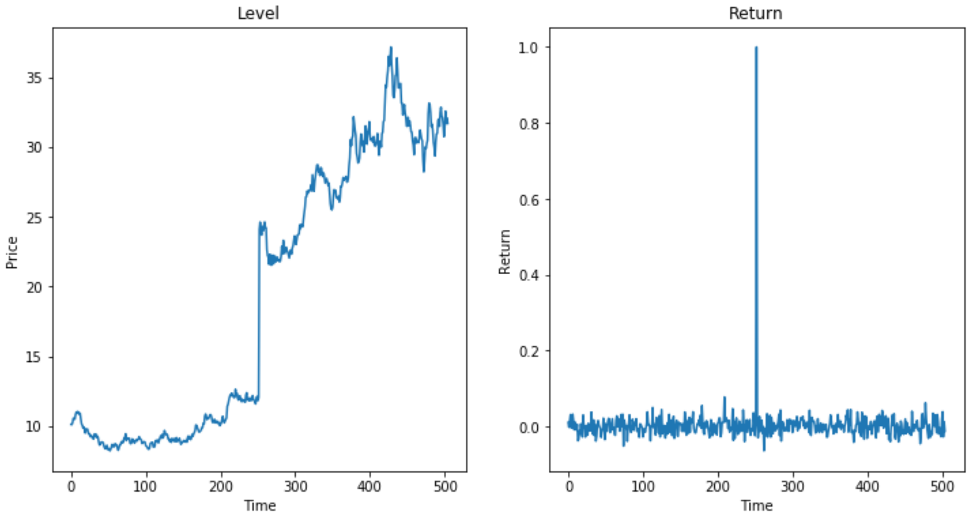
- That each training and test example $(\mathbf{x}, \mathbf{y})$
- is a sample drawn from p_{data}
- where p_{data} is the true (but unknown) distribution of examples

It is surprisingly easy to violate this assumption

Consider the timeseries of prices of an equity

In [7]:
```
fig_data
```

Out[7]:



One can easily see that there are two distributions here

- a low-mean (and low variance) distribution prior to the jump in Price
- a high-mean (and high variance) distribution after the jump

The test data (drawn post jump) is **not** from the same distribution as the training data (pre-jump).

In fact: if the training data also includes examples post-jump

- it would seem that the training data comes from the union of *two different* distributions

At first glance, this seems surprising.

After all, each Price is drawn from the distribution of a single equity.

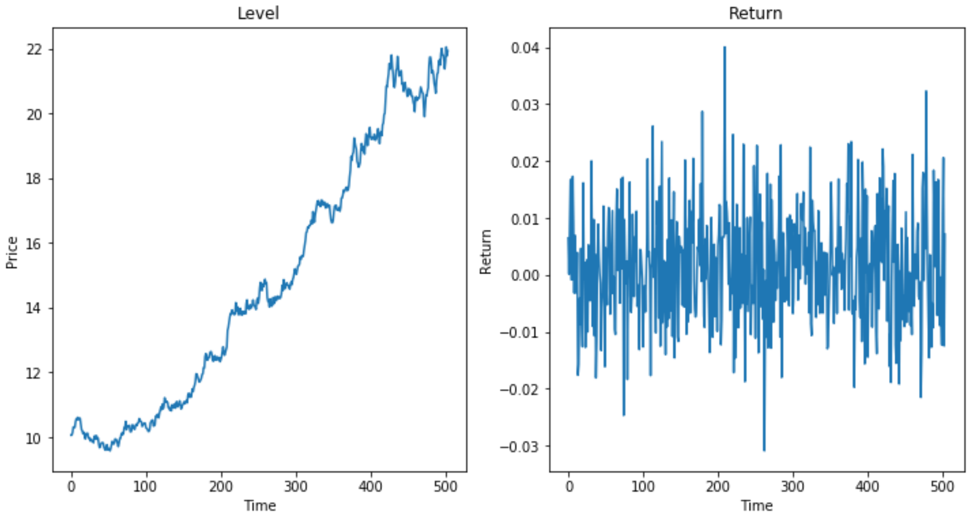
But, as is very typical in Finance, our distributions *vary with time*.

A more common example:

In [8]:

```
fig_sdrift
```

Out[8]:



So, in order to satisfy the Assumption

- we typically transform the Price distribution into a Return distribution
- which is often (but not always) more stable over time

Converting into percent price changes (from levels) results in

- near constant mean/variance across examples

A dataset which was previously heterogeneous (in Levels)

- has become homogeneous (in Returns)

Non-homogeneous data: examples

The purpose of this section is

- to **motivate** Transformations
- from raw features to synthetic features
- such that the model's predictions work better

We will study the Transformations is a subsequent module

Many Transformations are motivated by the presence of examples

- that are not homogeneous
    - seem to come from different distributions

We might have been surprised that Price (Level) data results in non-homogeneous examples.

- the ticker is the same
- but it's behavior over time has changed

There are many other variations of this problem.

Knowing how to recognize and correct them is important.

We will imagine that

- the available data comes from one or more *groups* each with its own distribution.

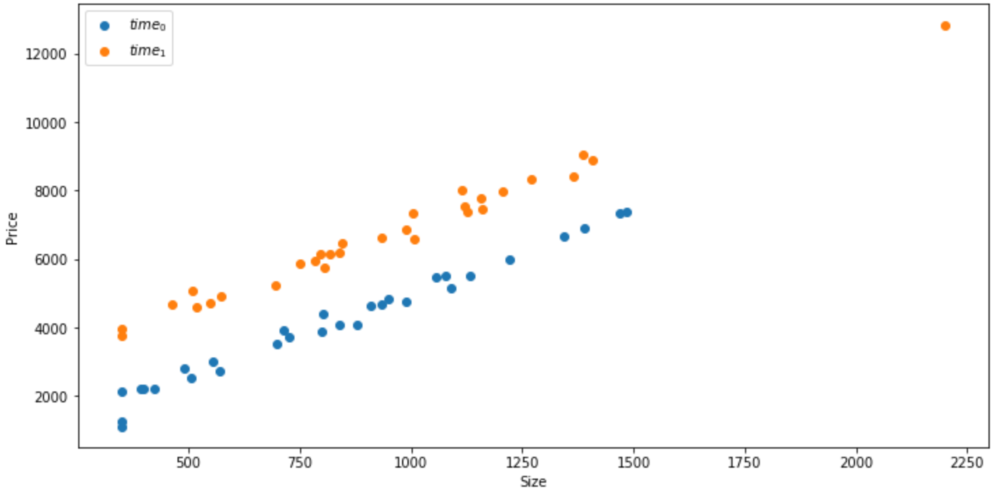
Let's go back to our original Regression problem

- predicting Price of an apartment from its Size

Consider the following graph illustrating the relationship between Price (target) and Size (features)

In [9]:
```
fig_sp
```

Out[9]:



It appears that the data in different groups is similar, but not identical.

- relationship is linear within group
- same "slope" in both groups
- different per-group intercept

Once we make that observation

- We might be tempted to fit a *separate model* for each group. This is not a great idea
  - Which model do we use for an out of sample example ?
  - The "goodness of fit" increases with sample size

But: how do we even **discover** the presence of two groups

- I added the colors for illustration
- In practice:
    - you are given a large dataset as a "cloud" of mixed examples
    - it might be up to **you** to discover that the single "cloud" of points represents two groups

There are multiple opportunities for a Data Scientist following the Recipe to uncover this

- Exploratory data analysis
    - slicing the data reveals different distributions
- Error Analysis
    - fitting a single line to all examples: the sign of the errors differs by group

Another reason for the presence of more than one group

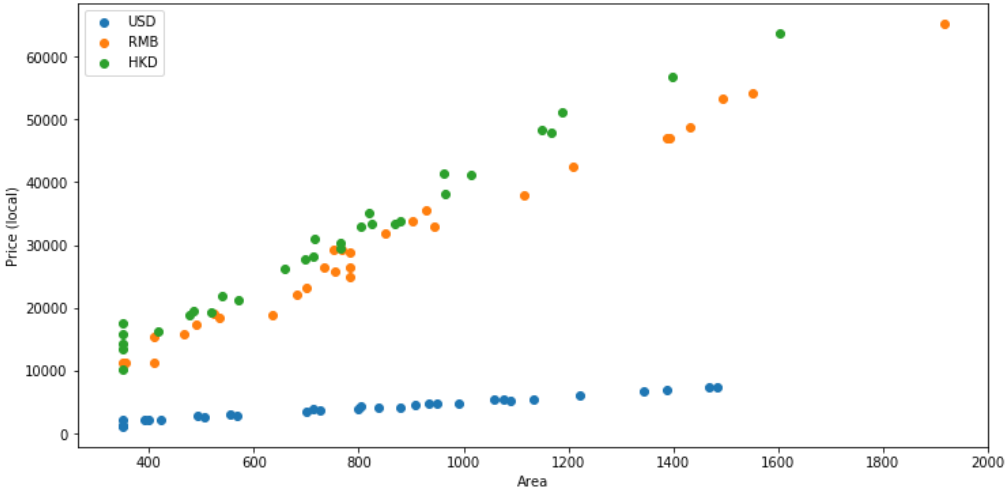
- differences in market/geography.

Consider the plot

In [10]:

```
fig_rp
```

Out[10]:



Again

- I have added the colors
- More typically: you have a cloud of points without any clear group distinction

Once you recognize the presence of multiple groups

- what can you do in this case ?

Each group seems to have a linear relationship between Price and Size.

Unlike the prior instance

- each group seems to have its own slope and intercept

Even though each group seems different **on the surface**

- there may be a *single* relationship between targets and features
- that is the *same* across groups
- at a **deeper** level

(We will reveal this common relationship after some time)

Being able to transform data to uncover the deeper level relationship between targets and features

- is a skill **critical to being able to construct good models**

Why do we have to deal with non-homogeneous data ?

What are the causes of heterogeneity ?

Some are structural

- a single equity ticker with a relationships that varies with time

But sometimes they arise out of necessity

- We need large amounts of training examples to fit models
    - more data yields better models

This sometimes requires us to **pool** examples from multiple groups

- not enough data for a single equity ticker in a single time window
    - pool across time
    - pool across equity tickers
- not enough data in a single geography
    - pool across markets

For example, a lot of Financial data is sampled at low frequency (e.g., daily)

- So not a lot of data points for a single ticker in a short time period
- Pool over time to increase number of examples
    - groups caused by time-varying distribution of single stock price/return
- Pool over many similar tickers to increase number of examples
    - distribution per ticker is different

Example: equity trading volume

It might be reasonable to hypothesize that

- the daily trading volume (or changes in volume) of an equity is a useful feature
- for predicting stock behavior
- stocks with high volume *appear* to be more liquid
- jumps in volume may be a signal

How should we measure the Volume of a ticker ?

The obvious unit of measurement is "number of shares".

If we do so

- we may not discover a signal *solely* related to the ticker
- when market volume is high
    - the volume of all tickers is high

Thus, an unusually high volume as measured in "number of shares" may not be a valid trading signal.

By re-denominating the ticker's Volume

- into units "fraction of total market volume"
- we remove the effect of changes in ticker volume solely related to changes in Market volume.

We have increased the number of examples (by increasing the time dimension)

- without introducing heterogeneity

A non-homogeneous measure of Volume has been made homogeneous !

This change of units is accomplished

- as a *Transformation* in our Recipe

Example: equity trading volume

Another re-denomination is possible

- into units "fraction of ticker's Market Cap"

That is: what *fraction* of the ticker's capitalization has traded.

This re-denomination might allow us to pool across two nearly-identical tickers

- same Market Cap
- but one with
    - twice the number of shares
        - and hence: likely twice the Volume when denominated in units o "number of shares"
    - but half the Price

This transformation of units also works for a single ticker

- that has had a Stock Split
    - doubling shares but halving price
- we can now pool across time

Again

- a Transformation into different units
- has allowed us to increase the number of training examples
- by pooling across multiple tickers
- without introducing heterogeneity

Example: equity trading volume

Another common re-denomination of Volume is

- into units of "Dollar Volume*

$$\text{Dollar Volume} = \text{Volume} * \text{Price}$$

Often: this is used as a threshold to limit which Tickers to pool

- include a ticker only if it's Dollar Volume each day is "signicant"

Example: modeling prepayments

Consider the following example.

Suppose our goal is to predict whether a borrower will prepay a mortgage.

As raw features: we have variables

- Interest rate (coupon) C on borrower's loan
- Rate r at a which a mortgage could be re-financed

The borrower has an incentive to prepay if $C > r$.

- there are many other reasons for pre-payment
- we focus here only on reasons caused by changing interest rates

What training examples should we use ?

Remember: we want the test examples (observed in the near future) to be similar to the training examples.

Should our examples only include instances where $r \approx r'$

- where r' is today's re-finance rate

We may not have a lot of *historical* data with a contemporaneous re-finance rate close to r'.

Consider a different approach to modeling:

- we use a *synthetic* feature I capturing *incentive* to prepay
- rather than just raw values C, r for borrower's current rate and contemporaneous re-finance rate

For example, we could create feature I in several possible ways

- $I = \max(0, C - r)$
    - incentive denominated in Percentage Points
- $I = \max(0, \frac{C}{r} - 1)$
    - incentive denominated as relative to re-finance rate"

Using the synthetic feature opens up the possibility of using a lot more historical data

- Many more historical episodes with a particular value of $\frac{C}{r}$ or $C - r$
- Compared to examples where the then-current re-finance rate equals today's rate r

Another advantage: It allows the model

- to work for examples with a range of values for I
- rather than the immediate need of examples where $r \approx r'$

In essence, re-denomination into Incentive I

- has allowed us to pool examples
    - over time
    - over different levels of re-finance rate

Moreover, we have probably uncovered

- the true *semantics/reason* of pre-payment
    - relationship between C and r
- rather than *syntax/surface* reasons
    - absolute level of re-finance rate r

A model based on deeper meaning of features

- hopefully is more likely to generalize to out of sample examples

Becoming a successful Data Scientist

The examples in this notebook illustrate the challenges that a Data Scientist encounters.

What distinguishes an "good" Data Scientist from a "great" one is the knowledge and skill to recognize and circumvent the challenges.

A key skill of a Data Scientist is the ability

- to strip away surface differences and reveal underlying commonality between groups
- being able to adapt models to deal with multiple groups

You can *recognize* the challenge

- before starting: by superior Exploratory Data Analysis
- after staring: by superior Error Analysis

Once you recognize the challenges, you may be able to *circumvent* them by

- Transforming the data
    - imposing homogeneity
    - adding features that distinguish between groups
- Find the correct functional form for the model
    - linear
    - polynomial
    - something new

The world before Black Scholes option pricing

The relationships between targets and features we have explored have usually been simple.

In the real world: the relationships are complex

- missing features
- interaction between features
- described by a new "theory" that is validated by data

Consider the goal of predicting the price of a call option from features of the option.

In the days before the Black Scholes pricing formula was discovered

- people proposed (partially successful) functional forms relating price to features
    - traders carried around tables of approximate prices
    - based on models that partially/conditionally explained prices

Even without a perfect model, an imaginative Data Scientist can come up with partially successful models with high utility.

A Data Scientist needs

- intuition
- imagination
- scientific rigor and the will to experiment

in order to be able to postulate/test theories relating target to features.

A couple of observations on the Black Scholes call price formula

$$C = \mathcal{N}(d_1)S_t + -\mathcal{N}(d_2)K * e^{-r*t}$$

where

$$\begin{aligned} d_1 &= \frac{\log_e(\frac{S_t}{K}) + (r + .5 * \sigma^2) * t}{\sigma * \sqrt{t}} \\ d_2 &= d_1 - \sigma * \sqrt{t} \end{aligned}$$

Notice some interesting "transformed features" in the d_1 term

- Price is denominated relative to the strike : $\frac{S_t}{K}$
- Volatility is scaled by time to maturity t: $\sigma * \sqrt{t}$

These non-obvious features derive, perhaps, from a need to "normalize" the feature.

Wrap-up

Obviously, these examples were contrived and overly simple.

In practice, more complex transformations are necessary to make groups more homogeneous.

The insight of the Data Scientist is key in guiding the process.

We will show transformations that address each of these examples (and more) in a separate module.

In [11]:
```
print("Done")
```

```
Done
```