laplace_breadth_function_code_rationale

March 23, 2020

[39]:

```
import pandas as pd
```

0.1 What is estimate breadth?

estimate breadth $= \frac{\text{estimate upgrades - estimate downgrades}}{\text{total number of estimates}}$

0.2 Why is it not sufficient on its own?

Consider the case of the following 3 Amazon review histories for three sellers: * 2/2 positive reviews. 100 % positive * 9/10 positive reviews. 90 % positive * 88/100 postiive reviews. 88% positive

Clearly not all of the review histories are equal, and you would probably feel that seller number 3 is going to give you the best experience. But how do we incorporate this in mathematically?

With a Bayesian prior obviously… With a uniform prior (i.e. one additional negative and positive review), we can retrieve a postierior that is closer to our real beliefs. Taking the first seller as an example:

$$\text{(Prior 1) Bin} \sim (\hat{p} = \frac{1}{2}, n = 2)$$

$$\text{(Prior 2) Bin} \sim (\hat{p} = \frac{2}{2}, n = 2)$$

$$\text{(Posterior) Beta} \sim (\hat{\alpha} = \frac{3}{4}, \hat{\beta} = \frac{1}{4})$$

$$\hat{\mathbb{E}}[\text{Success}_{\text{post}}] = \frac{\frac{3}{4} + \frac{1}{4}}{\frac{3}{4}} = 75\%$$

Whereas it is 87.2% for seller 3.

Thats pretty neat. The true optimal solution is actually the center of mass for the beta PDF… but we do not want to do any hectic maths in the calculation, so this quick and dirty alternative will work great for us.

0.3 Laplace breadth

I propose we call it Laplace breadth because although pastor Bayes discovered the formula, he was a layman, and his work gained no notice until Laplace dragged it kicking and screaming into 20th century maths along with the rest of probability theory. And Laplace even proposed a similar thought experiment in his own writings.

The revised estimate breadth formula is:

$$\text{laplace breadth} = \frac{(\text{estimate upgrades} + 1) - (\text{estimate downgrades} + 1)}{\text{total estimates} + 2}$$

$$\text{laplace breadth} = \frac{(\text{estimate upgrades}) - (\text{estimate downgrades})}{\text{total estimates} + 2}$$

```
[40]: est_data = pd.read_csv("C:/model_data/estimate_raw.csv")
```

```
[41]: type(est_data)
```

```
[41]: pandas.core.frame.DataFrame
```

```
[42]: est_data.head()
```

```
[42]:    Unnamed: 0  security_id   broker_id period_date estimate_date    value  \
      0           0  30064875557 -1676276586  2018-12-31    2019-03-21  0.49000
      1           1  30064875557 -1676276586  2019-12-31    2019-03-21  0.62000
      2           2  30064875557 -1676276586  2020-12-31    2019-03-21  0.79000
      3           3  30064799557  -705587338  2019-12-31    2019-03-21  0.52727
      4           4  30064799557  -705587338  2020-12-31    2019-03-21  0.57273

        currency source_id
      0      CNY      ibes
      1      CNY      ibes
      2      CNY      ibes
      3      HKD      ibes
      4      HKD      ibes
```

```
[43]: est_data.dtypes
```

```
[43]: Unnamed: 0         int64
      security_id        int64
      broker_id          int64
      period_date       object
      estimate_date     object
      value            float64
      currency          object
      source_id         object
      dtype: object
```

We need to change data columns to date types

```
[44]: col_names = list(est_data.columns)
      col_names
```

```
[44]: ['Unnamed: 0',
       'security_id',
       'broker_id',
       'period_date',
       'estimate_date',
       'value',
       'currency',
       'source_id']
```

```
[45]: est_data['period_date'] = pd.to_datetime(est_data['period_date'])
      est_data['estimate_date'] = pd.to_datetime(est_data['estimate_date'])
```

```
[46]: est_data.dtypes
```

```
[46]: Unnamed: 0                int64
      security_id               int64
      broker_id                 int64
      period_date      datetime64[ns]
      estimate_date    datetime64[ns]
      value                   float64
      currency                 object
      source_id                object
      dtype: object
```

Now we need to check the date range... The estimate date refers to the date when the estimate was made and not the forward earnings date that is is applied to (Which is the period date)

```
[47]: max(est_data['estimate_date'])
```

```
      ␣
 ↪---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call␣
 ↪last)

        <ipython-input-47-b7e989ae2e78> in <module>
    ----> 1 max(est_data['estimate_date'])


        TypeError: 'Series' object is not callable
```

[ ]:

```
min(est_data['estimate_date'])
```

So for this working example we have a years worth of data…

Now we can set the key to the four columns that together uniquely identify a row

Now we need a change in estimates, so we need a before date, and a after date.

Additionally we need a lag period - the minimum amount of time we are willing to tolerate between the start and the end of the revision.

Where there are no before and after dates for the estimates subject to the lag period then those data points need to be discarded

[48]:

```
max_date = (est_data.groupby(['security_id', 'broker_id', 'period_date'])
     ['estimate_date']
     .max()
)
min_date = (est_data.groupby(['security_id', 'broker_id', 'period_date'])
     ['estimate_date']
     .min()
)
lag_in = (max_date - min_date).dt.days > lag_tol
lag_in
```

[48]:

```
security_id  broker_id    period_date
30064771087  -2084193872  2019-08-31     False
                          2020-08-31      True
                          2021-08-31      True
                          2022-08-31      True
             -1951260275  2020-08-31     False
                                         …
30064878801  -1201652488  2020-12-31     False
                          2021-12-31     False
30064878802  -1898442150  2020-03-31     False
30064878803  -1898442150  2020-03-31     False
30064878804  -1898442150  2020-02-29     False
Name: estimate_date, Length: 452990, dtype: bool
```

[49]:

```
est_data = (est_data
               .join(
                    lag_in,
                    on = ['security_id', 'broker_id', 'period_date'],
                    rsuffix = '_diff'
               )
           )
est_data
```

```
[49]:          Unnamed: 0  security_id   broker_id period_date estimate_date  \
      0                 0  30064875557 -1676276586  2018-12-31    2019-03-21
      1                 1  30064875557 -1676276586  2019-12-31    2019-03-21
      2                 2  30064875557 -1676276586  2020-12-31    2019-03-21
      3                 3  30064799557  -705587338  2019-12-31    2019-03-21
      4                 4  30064799557  -705587338  2020-12-31    2019-03-21
      …               …          …         …        …          …
      1603343     1603343  30064798004 -1723677634  2020-12-31    2019-10-16
      1603344     1603344  30064798004 -1723677634  2021-12-31    2019-10-16
      1603345     1603345  30064777941   213788152  2019-12-31    2019-10-16
      1603346     1603346  30064777941   213788152  2020-12-31    2019-10-16
      1603347     1603347  30064777941   213788152  2021-12-31    2019-10-16

                     value currency source_id  estimate_date_diff
      0            0.49000      CNY      ibes               False
      1            0.62000      CNY      ibes                True
      2            0.79000      CNY      ibes                True
      3            0.52727      HKD      ibes                True
      4            0.57273      HKD      ibes                True
      …                …      …       …                 …
      1603343  18603.00000      KRW      ibes                True
      1603344  19998.00000      KRW      ibes                True
      1603345   1304.00000      KRW      ibes                True
      1603346   1317.00000      KRW      ibes                True
      1603347   1377.00000      KRW      ibes                True

      [1603348 rows x 9 columns]
```

```
[50]: est_data[est_data.estimate_date_diff == True]
```

```
[50]:          Unnamed: 0  security_id   broker_id period_date estimate_date  \
      1                 1  30064875557 -1676276586  2019-12-31    2019-03-21
      2                 2  30064875557 -1676276586  2020-12-31    2019-03-21
      3                 3  30064799557  -705587338  2019-12-31    2019-03-21
      4                 4  30064799557  -705587338  2020-12-31    2019-03-21
      5                 5  30064804463 -1653167482  2019-12-31    2019-03-21
      …               …          …         …        …          …
      1603343     1603343  30064798004 -1723677634  2020-12-31    2019-10-16
      1603344     1603344  30064798004 -1723677634  2021-12-31    2019-10-16
      1603345     1603345  30064777941   213788152  2019-12-31    2019-10-16
      1603346     1603346  30064777941   213788152  2020-12-31    2019-10-16
      1603347     1603347  30064777941   213788152  2021-12-31    2019-10-16

                     value currency source_id  estimate_date_diff
      1            0.62000      CNY      ibes                True
      2            0.79000      CNY      ibes                True
      3            0.52727      HKD      ibes                True
```

```
4             0.57273      HKD      ibes                True
5             5.28000      EUR      ibes                True
…              …      …       …                 …
1603343  18603.00000      KRW      ibes                True
1603344  19998.00000      KRW      ibes                True
1603345   1304.00000      KRW      ibes                True
1603346   1317.00000      KRW      ibes                True
1603347   1377.00000      KRW      ibes                True

[1392934 rows x 9 columns]
```

Now we need to change the estimate date column to a daily index that we can swim through to record the number of upward and downward revisions…

Be careful of memory issues when you do this…

[ ]: