Real Estate Price Prediction

106306029 資管四甲 羅恩力 En-Li, Luo

Outline

- Web Crawler
- Grey-Markov Model

Open Government Data

1 2 3 4 5 6 7 8 9 10 ... 最末頁

加入清單	行政區	土地位置或建物門牌	交易日期	交易總價(萬元)註1	交易單價(萬元/坪)註2	單價是否含車位	建物移轉面積(坪)	土地移轉面積(坪)	建物型態註3	屋齡註4	樓層別/總樓層	交易種類	備註事項	歷次移轉	詳細資訊
選取	松山區	八德路二段346-350號(雙號)	110/04/09	1,008	112.63	否	8.95	0.79	大樓(11層含以上有電梯)	12	010/014	房地	有	有	More
選取	松山區	光復北路100巷16-20號(雙號)	110/04/06	2,620	65.11	否	40.24	14.22	公寓(5樓含以下無電梯)		004/004	房地	無	無	More
選取	松山區	八德路四段656-660號(雙號)	110/04/03	1,200	55.63	否	21.57	3.77	華廈(10層含以下有電梯)	43	005/008	房地	有	有	More
選取	松山區	八德路四段781-785號(單號)	110/04/10	3,188	90.54	否	35.21	3.44	大樓(11層含以上有電梯)	14	005/015	房地	有	有	More
選取	松山區	延吉街22巷6-10號(單號)	110/04/13	1,530	66.21	否	23.11	4.57	華廈(10層含以下有電梯)	38	003/007	房地	有	無	More
選取	松山區	南京東路四段186-190號(雙號)	110/04/09	2,303	72.58	否	31.73	2.7	大樓(11層含以上有電梯)	37	008/014	房地	有	有	More
選取	松山區	民生東路五段260巷21-25號(單號)	110/04/13	3,300	58.82	否	56.1	11.02	華廈(10層含以下有電梯)	40	004/007	房地	有	無	More
選取	松山區	八德路四段396-400號(單號)	110/04/13	1,245	58.78	否	21.18	6.88	公寓(5樓含以下無電梯)		004/004	房地	無	無	More
選取	松山區	八德路三段12巷20弄11-15號(單號)	110/04/07	3,400	78.59	否	43.26	10.49	華廈(10層含以下有電梯)	48	002/006	房地	有	無	More
選取	松山區	三民路29巷6-10號(單號)	110/04/02	4,868	77.43	是	62.87	13.38	大樓(11層含以上有電梯)	16	013/013	房地車	無	無	More

https://cloud.land.gov.taipei/ImmPrice/TruePriceA.aspx

Scrapy

```python
def parse(self, response, **kwargs):
    driver = self.driver
    driver.get(response.url)

    driver.execute_script("document.getElementById('tab1').style.display = 'none';")
    driver.execute_script("document.getElementById('tab2').style.display = 'block';")

    for year in range(self.start_year, self.end_year + 1):  # every year
        fr = self.start_month if year == self.start_year else 1
        to = self.end_month if year == self.end_year else 12
        for month in range(fr, to + 1):  # every month
            for district_no in range(self.start_district_no, self.end_district_no + 1):  # every district
                driver.execute_script(f"document.getElementById('{choose_district_id}{district_no}').checked = true;")
                self._select_time_range(year, month)
                self.driver.execute_script('arguments[0].click();', self.wait.until(EC.element_to_be_clickable((By.ID, search_btn_id))))

                self.wait.until(EC.visibility_of_element_located((By.ID, table_id)))
                district_name = district_list[district_no] + '區'
                self.wait.until(EC.text_to_be_present_in_element((By.CSS_SELECTOR, f"#{table_id} > tbody > tr:nth-child(3) > td:nth-child(2)"), district_name))

                total_rows = int(driver.find_element(By.ID, total_rows_id).text)
                for page in range(1, math.ceil(total_rows / 10) + 1):  # every page
                    row_count = len(driver.find_elements(By.CSS_SELECTOR, f"#{table_id} > tbody > tr"))

                    for row_no in range(3, row_count + 1):  # every row in the table
                        if (year, month, district_name, page, row_no - 2) not in ignore_list:    # click more info might trigger error
                            table_td_list = driver.find_elements(By.CSS_SELECTOR, f"#{table_id} > tbody > tr:nth-child({row_no}) td")
                            yield self._list_to_item(table_td_list, district_name)

                    if page != math.ceil(total_rows / 10):
                        self._swap_table_page('...' if page % 10 == 0 else page + 1)

                if total_rows > 10:    # swap to first page before starting new district
                    self._swap_table_page('第一頁' if total_rows > 100 else 1)
                driver.execute_script(f"document.getElementById('{choose_district_id}{district_no}').checked = false;")
```

Selenium

加入清單	行政區	土地位置或建物門牌	交易日期	交易總價(萬元)註1	交易單價(萬元/坪)註2	含車位	面積(坪)	面積(坪)	建物型態註3	屋齡註4	樓層別/總樓層	交易種類	備註事項	歷次移轉	詳細資訊
選取	松山區	八德路二段346-350號(雙號)	110/04/09	1,008	112.63	否	8.95	0.79	大樓(11層含以上有電梯)	12	010/014	房地	有	有	More
選取	松山區	光復北路100巷16-20號(雙號)	110/04/06	2,620	65.11	否	40.24	14.22	公寓(5樓含以下無電梯)		004/004	房地	無	無	More
選取	松山區	八德路四段656-660號(雙號)	110/04/03	1,200	55.63	否	21.57	3.77	華廈(10層含以下有電梯)	43	005/008	房地		有	More
選取	松山區	八德路四段781-785號(單號)	110/04/10	3,188	90.54	否	35.21	3.44	大樓(11層含以上有電梯)	14	005/015	房地		有	More
選取	松山區	延吉街22巷6-10號(單號)	110/04/13	1,530	66.21	否	23.11	4.57	華廈(10層含以下有電梯)	38	003/007	房地		無	More
選取	松山區	南京東路四段186-190號(雙號)	110/04/09	2,303	72.58	否	31.73	2.7	大樓(11層含以上有電梯)	37	008/014	房地		有	More
選取	松山區	民生東路五段260巷21-25號(單號)	110/04/13	3,300	58.82	否	56.1	11.02	華廈(10層含以下有電梯)	40	004/007	房地		無	More
選取	松山區	八德路四段396-400號(單號)	110/04/13	1,245	58.78	否	21.18	6.88	公寓(5樓含以下無電梯)		004/004	房地		無	More
選取	松山區	八德路三段12巷20弄11-15號(單號)	110/04/07	3,400	78.59	否	43.26	10.49	華廈(10層含以下有電梯)	48	002/006	房地	有	無	More
選取	松山區	三民路29巷6-10號(單號)	110/04/02	4,868	77.43	是	62.87	13.38	大樓(11層含以上有電梯)	16	013/013	房地車	無		More

https://cloud.land.gov.taipei/ImmPrice/TruePriceA.aspx

Distance to Nearest MRT Exit

```python
@staticmethod
def _trans_addr_to_coord(addr: str) -> Tuple[float, float]:
    chrome_options = webdriver.ChromeOptions()
    chrome_options.add_argument('--headless')
    chrome_options.add_argument('--disable-gpu')
    driver = webdriver.Chrome(chrome_options=chrome_options)
    url = requote_uri(f"https://map.tgos.tw/TGOSimpleViewer/Web/Map/TGOSimpleViewer_Map.aspx?addr={addr}")
    driver.get(url)

    try:
        WebDriverWait(driver, 30, 0.01).until(lambda url_change: driver.current_url != url)
        info = driver.current_url
        building_lon = float(info[info.index('CX=') + 3:info.index('CY=') - 1])
        building_lat = float(info[info.index('CY=') + 3:info.index('L=') - 1])
    except (TimeoutException, ValueError):
        driver.close()
        return -1.0, -1.0

    driver.close()
    return building_lon, building_lat

def _get_dist_to_mrt(self, lon: float, lat: float) -> Tuple[float, float]:
    shortest_distance = float('inf')
    shortest_exit_id = -1
    for row in self.mrt_exit_coordinate:
        lon1, lat1, lon2, lat2 = map(radians, [lon, lat, row[1], row[2]])
        a = sin((lat2 - lat1) / 2) ** 2 + cos(lat1) * cos(lat2) * sin((lon2 - lon1) / 2) ** 2
        c = 2 * asin(sqrt(a))
        r = 6371
        distance = c * r * 1000
        if distance < shortest_distance:
            shortest_exit_id = row[0]
            shortest_distance = distance
    return shortest_exit_id, shortest_distance
```

data from Taipei MRT

	exit_id	exit_name	exit_no	longitude	latitude
1	1	頂埔站出口1	1	121.418	24.9593
2	2	頂埔站出口2	2	121.419	24.9593
3	3	頂埔站出口3	3	121.42	24.9596
4	4	頂埔站出口4	4	121.42	24.9604
5	5	松山機場站出口1	1	121.552	25.0637
6	6	松山機場站出口2	2	121.551	25.0637
7	7	松山機場站出口3	3	121.552	25.0629
8	8	中山國中站出口	0	121.544	25.0609
9	9	忠孝復興站出口1	1	121.543	25.0418
10	10	忠孝復興站出口2	2	121.543	25.0414
11	11	忠孝復興站出口3	3	121.545	25.0415
12	12	忠孝復興站出口4	4	121.545	25.0418
13	13	忠孝復興站出口5	5	121.544	25.042
14	14	大安站出口1	1	121.542	25.0335
15	15	大安站出口2	2	121.542	25.0335
16	16	大安站出口3	3	121.542	25.0332
17	17	大安站出口4	4	121.544	25.0331
18	18	大安站出口5	5	121.544	25.0329
19	19	大安站出口6	6	121.544	25.034
20	20	科技大樓站出口	0	121.544	25.0262
21	21	六張犁站出口	0	121.553	25.0239
22	22	麟光站出口	0	121.559	25.0186
23	23	辛亥站出口	0	121.557	25.0051
24	24	萬芳醫院站出口	0	121.558	24.9994
25	25	萬芳社區站出口	0	121.568	24.9986
26	26	木柵站出口	0	121.573	24.9982
27	27	動物園站出口1	1	121.58	24.9982
28	28	動物園站出口2	2	121.579	24.9979
29	29	大直站出口1	1	121.547	25.0801
30	30	大直站出口2	2	121.547	25.0796
31	31	大直站出口3	3	121.547	25.0793
32	32	劍南路站出口1	1	121.555	25.0852

Data Preview

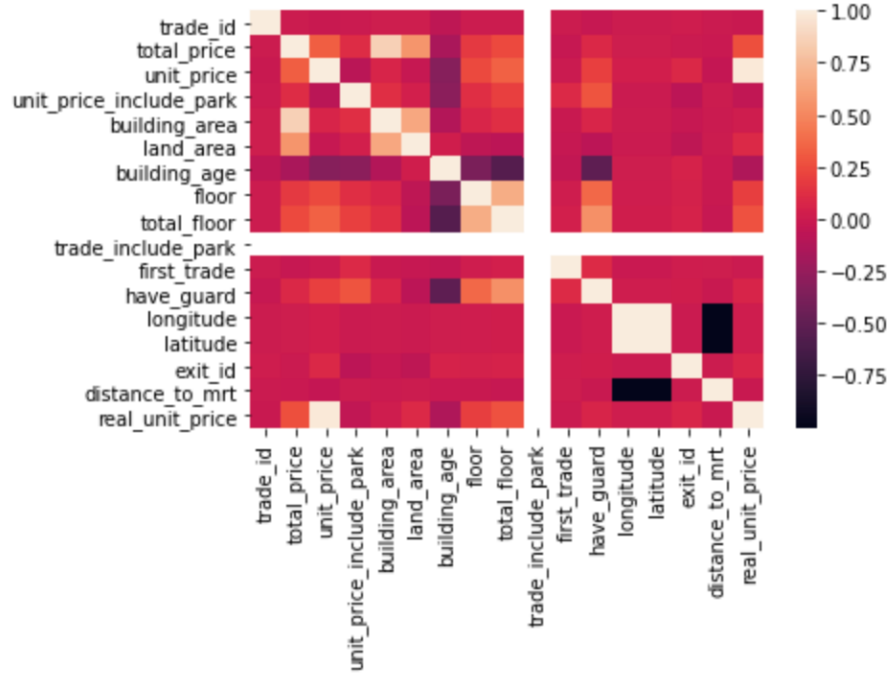
- 2018.01.01 – 2020.12.31
- 60368 rows

	51838	51839	51840	51841	51842	51843	51844	51845
trade_id	51874	51875	51876	51877	51878	51879	51880	51881
district	中山區	中山區	中山區	中山區	中山區	中山區	中山區	中山區
address	南京東路三段194巷1	遼寧街105巷3:	新生北路三段88巷	林森北路415號	南京東路三段91號	新生北路一段106號	復興北路446號	松江路184巷26弄8
trade_date	2018-07-25	2018-07-25	2018-07-21	2018-07-30	2018-07-23	2018-07-30	2018-07-24	2018-07-23
total_price	79030000	68500000	10860000	6150000	24000000	8300000	10250000	23000000
unit_price	<null>	<null>	453100	371800	600900	537900	761500	677900
unit_price_include_park	0	0	0	0	0	0	1	1
building_area	37.54	34.92	23.97	16.54	39.94	15.43	13.46	33.93
land_area	42.09	30.55	8.92	1.55	4.35	2.14	2.27	4.6
building_type	透天厝	透天厝	公寓(5樓含以下無	大樓(11層含以上有	大樓(11層含以上有	大樓(11層含以上有	大樓(11層含以上有	華廈(10層含以下有
building_age	<null>	<null>	52	34	46	41	10	10
floor	<null>	<null>	2	9	4	11	3	7
total_floor	<null>	<null>	4	13	11	12	12	9
trade_include_park	0	0	0	0	0	0	0	0
remark	<null>	<null>	陽台外推	<null>	<null>	含增建或未登記建物	含增建或未登記…	含增建或未登記建物
first_trade	0	0	1	1	0	1	1	1
building_layout	6房2廳2衛 有隔間	6房2廳2衛 有隔	2房1廳1衛 有隔間	1房1廳1衛 有隔間	0房0廳0衛 無隔間	1房1廳1衛 有隔間	1房1廳1衛 有隔間	1房1廳1衛 有隔間
have_guard	0	0	0	0	1	1	1	1
land_type	第三種住宅區。	第三種住宅區。	第肆種商業區(…	第肆種商業區(依…	第三種住宅區。	第肆種商業區(依…	第三之一種住宅區。	第貳種商業區(依…
building_material	加強磚造	加強磚造	鋼筋混凝土造	鋼筋混凝土造	鋼筋混凝土造	鋼筋混凝土造	鋼筋混凝土造	鋼筋混凝土造
longitude	121.543	121.543	121.527	121.526	121.539	121.528	121.544	121.531
latitude	25.0513	25.0512	25.0678	25.0598	25.0522	25.0494	25.064	25.0566
exit_id	329	329	299	297	327	347	8	300
distance_to_mrt	40.3683	58.0452	557.765	304.493	309.624	452.804	342.174	304.735
create_time	2021-06-21 03…	2021-06-2…	2021-06-21 …	2021-06-21 0…	2021-06-21 0…	2021-06-21 0…	2021-06-21 0…	2021-06-21 0…

Data Preview

In [15]:
```
import seaborn as sns

corr = df.corr()
sns.heatmap(corr)
plt.show()
```

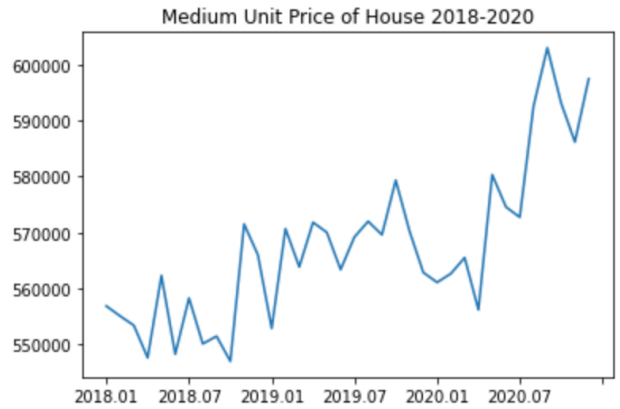


Medium Price for Every Month

In [2]:
```
db = MySQLdb.connect(host=                    , user=        , passwd=                    , db=                )
cursor = db.cursor()
start = '2018'
end = '2020'
df = pd.read_sql(f"SELECT * FROM trade_price_raw where trade_date between '{start}-01-01' and '{end}-12-31'", con=db)
time_list = [dt.datetime.strftime(x, '%Y.%m') for x in pd.date_range(start=start+'0101', end=end+'1231', freq='MS')]
```

In [4]:
```
df['trade_date'] = pd.to_datetime(df['trade_date'])
df['real_unit_price'] = df['total_price'] / df['building_area']
gb = df.groupby(df.trade_date.dt.to_period('M'))
raw = list(gb['real_unit_price'].median())

plt.plot(time_list, raw)
plt.gca().xaxis.set_major_locator(mticker.MultipleLocator(6))
plt.title(f'Medium Unit Price of House {start}-{end}')
plt.show()
```



Grey Model(GM) for time series prediction

Ex: Have a data series: [1, 2, 3]

$y = [1, 3, 6]^T$

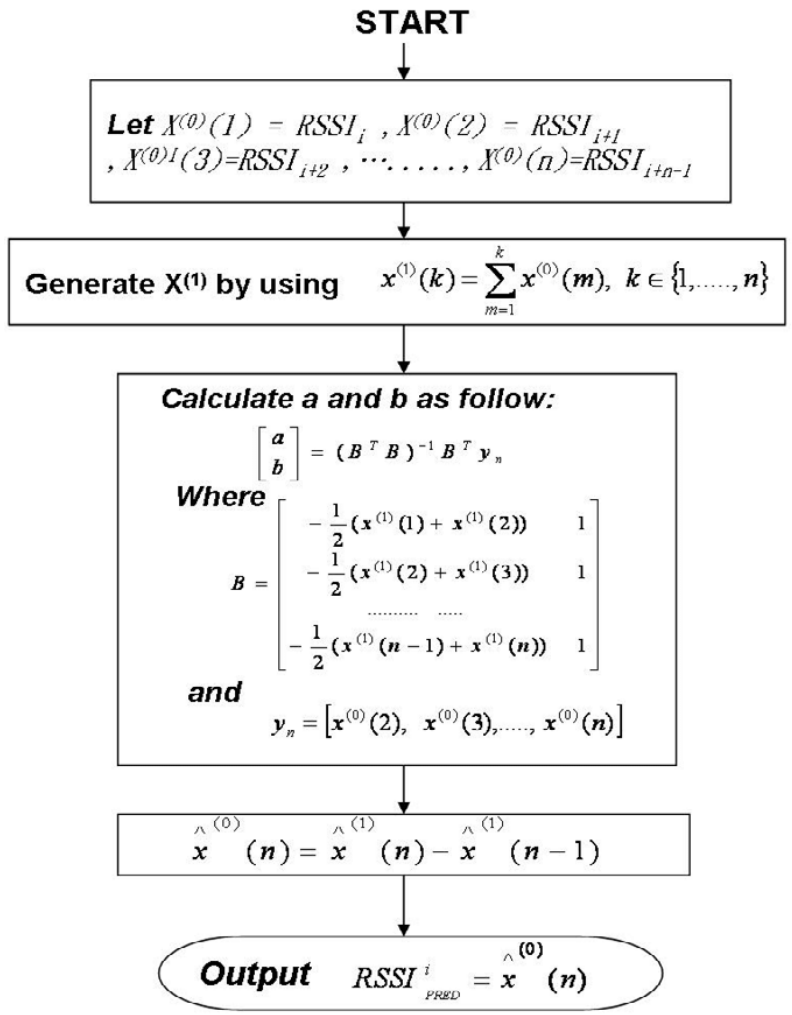
$-\frac{1}{2}(1+3)$ $-\frac{1}{2}(3+6)$

$$B = \begin{bmatrix} -2 & 1 \\ 4.5 & 1 \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \end{bmatrix}^T = (B^TB)^{-1}B^Ty$$

a, b

Finally, get $X^{(1)}(k) = (X^{(0)}(1) - \frac{b}{a})e^{-a(k-1)} + \frac{b}{a}$



Grey Model(GM) for time series prediction

Ex: Have a data series: [1, 2, 3]

```
21      y_cum_sum = np.cumsum(raw_data)
```

$y = [1, 3, 6]^T$

```
23      for i in range(len(raw_data) - 1):  # (-1/2)(X(1)(n-1) + X(1)(n))
24          b.append([-0.5 * (y_cum_sum[i] + y_cum_sum[i + 1]), 1])
```

$$B = \begin{bmatrix} -2 & 1 \\ 4.5 & 1 \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \end{bmatrix}^T = (B^T B)^{-1} B^T y$$

```
26      # ((B^T)B)^(-1) * (B^T) * y
27      b = np.array(b)
28      b_t = np.transpose(b)
29      output = np.linalg.inv(np.dot(b_t, b))
30      output = np.dot(output, b_t)
31      y = np.transpose(raw_data[1:])
32      output = np.dot(output, y)
33      output = np.transpose(output)
34
35      return output[0], output[1]
```

Finally, get $X^{(1)}(k) = (X^{(0)}(1) - \frac{b}{a})e^{-a(k-1)} + \frac{b}{a}$

```
6   def fit(raw_data) -> List[float]:
7       a, b = _get_coefficient(raw_data)
8
9       output_list = []
10      for i in range(len(raw_data)):    # X(1)(k) = (X(0)(1) - b/a) * e^(-a(k-1)) + (b/a)
11          output_list.append((raw_data[0] - b / a) * np.exp(-a * i) + b / a)
12
13      output_list = [el - output_list[i - 1] if i > 0 else el for i, el in enumerate(output_list)]    # inverse of cumulative sum
14      return output_list
```

Grey Model(GM) for time series prediction

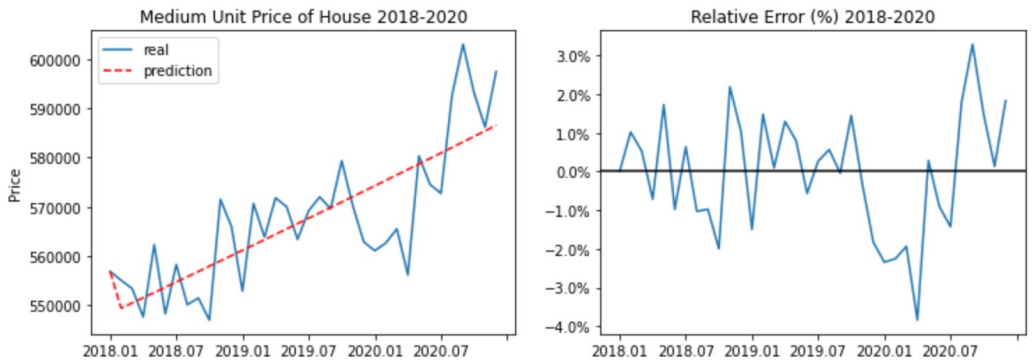
In [5]:
```
pred = grey_model.fit(raw)
relative_err_list = [(i - j) / i for i, j in zip(raw, pred)]

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(time_list, raw)
plt.plot(time_list, pred, color = 'red', linestyle = '--')
plt.ylabel('Price')
plt.gca().xaxis.set_major_locator(mticker.MultipleLocator(6))
plt.title(f'Medium Unit Price of House {start}-{end}')
plt.legend(['real', 'prediction'])

plt.subplot(1, 2, 2)
plt.plot(time_list, relative_err_list)
plt.axhline(y=0, ls='-', color='black')
plt.gca().xaxis.set_major_locator(mticker.MultipleLocator(6))
plt.gca().yaxis.set_major_formatter(mticker.PercentFormatter(xmax=1, decimals=1))
plt.title(f'Relative Error (%) {start}-{end}')

plt.show()
```



Validate My Grey Model(GM)

```python
def get_precision(raw_data: List[float], pred_data: List[float]) -> float:
    raw_corr = 0
    for i in range(1, len(raw_data) - 1):
        raw_corr += raw_data[i] - raw_data[0]
    raw_corr += (raw_data[-1] - raw_data[0]) / 2
    raw_corr = np.abs(raw_corr)

    pred_corr = 0
    for i in range(1, len(pred_data) - 1, 1):
        pred_corr += pred_data[i] - pred_data[0]
    pred_corr += (pred_data[-1] - pred_data[0]) / 2
    pred_corr = np.abs(pred_corr)

    sub_corr = 0
    for i in range(1, len(pred_data) - 1, 1):
        sub_corr += (raw_data[i] - raw_data[0]) - (pred_data[i] - pred_data[0])
    sub_corr += ((raw_data[-1] - raw_data[0]) - (pred_data[-1] - pred_data[0])) / 2
    sub_corr = np.abs(sub_corr)

    return (1 + raw_corr + pred_corr) / (1 + raw_corr + pred_corr + sub_corr)
```

$$|S| = |\sum_{k=2}^{n-1}(x(k) - x(1) + \frac{1}{2}(x(n) - x(1))|$$

$$|\hat{s}| = |\sum_{k=2}^{n-1}(\hat{x}(k) - \hat{x}(1) + \frac{1}{2}(\hat{x}(n) - \hat{x}(1))|$$

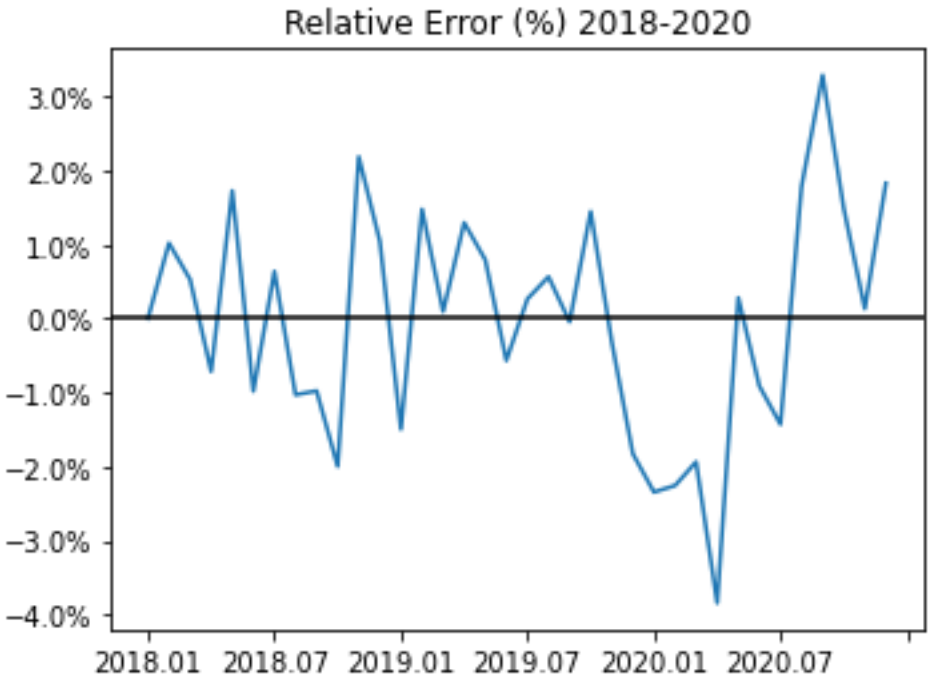
$$|\hat{s} - s| = |\sum_{k=2}^{n-1}[\big(x(k) - x(1)\big) - ((\hat{x}(k) - \hat{x}(1))] + \frac{1}{2}[(x(n) - x(1)) - \big(\hat{x}(n) - \hat{x}(1)\big)]|$$

```
In [5]: grey_model.get_precision(raw, pred)
Out[5]: 0.9926133493297309
```

$$\varepsilon = \frac{1 + |S| + |\hat{S}|}{1 + |S| + |\hat{S}| + |\hat{S} - S|}$$

1(best)	> 0.9
2	> 0.8
3	> 0.7
4	> 0.6

Markov Chain for Status Transform Prediction



```
relative_err_list
```

```
[4.411080748789029e-14,
 0.010189830765070271,
 0.005272155327967224,
 -0.007148685226892636,
 0.01726793203056345,
 -0.009827616273494664,
 0.006363737904553596 5,
 -0.010312650533956894,
 -0.00978130742130919,
 -0.019991591186914154,
 0.021861328604199134,
 0.010439853727232697,
 -0.014971622996744918,
 0.014776632287169633,
 0.0009404103565779892,
 0.012943907832977405,
 0.007940105410239607,
 -0.005677805229021457,
 0.0026566345010391574,
 0.005644730223874989 4,
 -0.00045300852457860107
 0.014461054628911563,
 -0.0031504435686536556,
 -0.0182880057706567,
 -0.023462355530152627,
 -0.022582370697113965,
 -0.019384608076519334,
 -0.03845239113679737,
 0.002810579079834714,
 -0.009127560724893017,
 -0.014304775597038773,
 0.017824656826236452,
 0.032900256808440234,
 0.014914457774912094,
 0.0013213706187519137,
 0.0182795438506177]
```

Markov Chain for Status Transform Prediction

Min: -3.85%
Max: 3.29%

```
relative_err_list
```

```
[4.411080748789029e-14,
 0.010189830765070271,
 0.005272155327967224,
 -0.007148685226892636,
 0.01726793203056345,
 -0.009827616273494664,
 0.006363737904535965,
 -0.010312650533956894,
 -0.00978130742130919,
 -0.019991591186914154,
 0.021861328604199134,
 0.010439853727232697,
 -0.014971622996744918,
 0.014776632287169633,
 0.0009404103565779892,
 0.012943907832977405,
 0.007940105410239607,
 -0.005677805229021457,
 0.0026566345010391574,
 0.0056447302238749894,
 -0.00045300852457860107
 0.014461054628911563,
 -0.0031504435686536556,
 -0.0182880057706567,
 -0.023462355530152627,
 -0.022582370697113965,
 -0.019384608076519334,
 -0.03845239113679737,
 0.002810579079834714,
 -0.009127560724893017,
 -0.014304775597038773,
 0.017824656826236452,
 0.032900256808440234,
 0.014914457774912094,
 0.0013213706187519137,
 0.0182795438506177]
```

Slice into 7 intervals:

Status1: -3.85% ~ -2.98%
Status2: -2.98% ~ -2.12%
Status3: -2.12% ~ -1.26%
Status4: -1.26% ~ -0.04%
Status5: -0.04% ~ 0.05%
Status6: 0.05% ~ 1.32%
Status7: 1.32% ~ 2.19%

```
In [11]: model.interval_list

Out[11]: [Interval(-0.03845239113679737, -0.029836145459512152, lower_closed=True, upper_closed=True),
          Interval(-0.029836145459512152, -0.02121989978222694, lower_closed=True, upper_closed=True),
          Interval(-0.02121989978222694, -0.012603654104941728, lower_closed=True, upper_closed=True),
          Interval(-0.012603654104941728, -0.003987408427656512, lower_closed=True, upper_closed=True),
          Interval(-0.003987408427656512, 0.004628837249628703, lower_closed=True, upper_closed=True),
          Interval(0.004628837249628703, 0.013245082926913912, lower_closed=True, upper_closed=True),
          Interval(0.013245082926913912, 0.021861328604199134, lower_closed=True, upper_closed=True)]
```

Markov Chain for Status Transform Prediction

	Status1	Status2	Status3	Status4	Status5	Status6	Status7
Status1	0	0	0	0	0	0	0
Status2	0	0	0	0	0	0	0
Status3	0	0	0	0	0	0	0
Status4	0	0	0	0	0	0	0
Status5	0	0	0	0	0	0	0
Status6	0	0	0	0	0	0	0
Status7	0	0	0	0	0	0	0

Markov Chain for Status Transform Prediction

```
relative_err_list
```

```
[4.411080748789029e-14,
 0.010189830765070271,
 0.005272155327967224,
 -0.007148685226892636,
 0.01726793203056345,
 -0.009827616273494664,
 0.0063637379045535965,
 -0.010312650533956894,
 -0.00978130742130919,
 -0.019991591186914154,
 0.021861328604199134,
 0.010439853727232697,
 -0.014971622996744918,
 0.014776632287169633,
 0.0009404103565779892,
 0.012943907832977405,
 0.007940105410239607,
 -0.005677805229021457,
 0.0026566345010391574,
 0.0056447302238749894,
 -0.00045300852457860107
 0.014461054628911563,
 -0.0031504435686536556,
 -0.0182880057706567,
 -0.023462355530152627,
 -0.022582370697113965,
 -0.019384608076519334,
 -0.03845239113679737,
 0.002810579079834714,
 -0.009127560724893017,
 -0.014304775597038773,
 0.017824656826236452,
 0.032900256808440234,
 0.014914457774912094,
 0.0013213706187519137,
 0.0182795438506177]
```

Status6: vector = [0 0 0 0 0 1 0]
Status7: vector = [0 0 0 0 0 0 1]

Status6 to Status7

Status7 to …

```
In [12]: model.trans_prob

Out[12]: array([[0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        ],
                [0.        , 0.5       , 0.5       , 0.        , 0.        ,
                 0.        , 0.        ],
                [0.25      , 0.25      , 0.        , 0.        , 0.        ,
                 0.        , 0.5       ],
                [0.        , 0.        , 0.2       , 0.2       , 0.2       ,
                 0.2       , 0.2       ],
                [0.        , 0.        , 0.16666667, 0.16666667, 0.        ,
                 0.5       , 0.16666667],
                [0.        , 0.        , 0.14285714, 0.42857143, 0.14285714,
                 0.28571429, 0.        ],
                [0.        , 0.        , 0.        , 0.25      , 0.5       ,
                 0.25      , 0.        ]])
```

Slice into 7 intervals:

Status1: -3.85% ~ -2.98%
Status2: -2.98% ~ -2.12%
Status3: -2.12% ~ -1.26%
Status4: -1.26% ~ -0.04%
Status5: -0.04% ~ 0.05%
Status6: 0.05% ~ 1.32%
Status7: 1.32% ~ 2.19%

Markov Chain for Status Transform Prediction

2020.06	[0 0 0 1 0 0 0]	4 steps to 2020.10	[0 0 0 1 0 0 0] * $P^{(4)}$
2020.07	[0 0 1 0 0 0 0]	3 steps to 2020.10	[0 0 1 0 0 0 0] * $P^{(3)}$
2020.08	[0 0 0 0 0 0 1]	2 steps to 2020.10	[0 0 0 0 0 0 1] * $P^{(2)}$
2020.09	[0 0 0 0 0 0 1]	1 steps to 2020.10	[0 0 0 0 0 0 1] * $P^{(1)}$

P =

```
model.trans_prob
```

```
array([[0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        ],
       [0.        , 0.5       , 0.5       , 0.        , 0.        ,
        0.        , 0.        ],
       [0.25      , 0.25      , 0.        , 0.        , 0.        ,
        0.        , 0.5       ],
       [0.        , 0.        , 0.2       , 0.2       , 0.2       ,
        0.2       , 0.2       ],
       [0.        , 0.        , 0.16666667, 0.16666667, 0.        ,
        0.5       , 0.16666667],
       [0.        , 0.        , 0.14285714, 0.42857143, 0.14285714,
        0.28571429, 0.        ],
       [0.        , 0.        , 0.        , 0.25      , 0.5       ,
        0.25      , 0.        ]])
```

Plus these 4 vector and calculate the most possible status.

```
50      def validate_test_data(self):
51          prob_vector_list = np.zeros((1, self.count))
52          p = self.trans_prob
53          for i in range(4):
54              prob_dist = np.dot(self.__to_vector(self.relative_err_list[-i-1]), p)
55              prob_vector_list += prob_dist
56              p = np.dot(p, self.trans_prob)
57          return prob_vector_list / prob_vector_list.sum(axis=1) * 100
58
```

Markov Chain for Status Transform Prediction

```python
class GreyMarkovModel:
    def __init__(self, relative_err_list: List[float], count: int):
        self.relative_err_list = relative_err_list
        self.max_relative_err = max(relative_err_list)
        self.min_relative_err = min(relative_err_list)
        self.count = count
        self.interval_val = (self.max_relative_err - self.min_relative_err) / self.count
        self.interval_list = []

        self.expected_val_list = []
        for i in range(count):
            self.expected_val_list.append(min(relative_err_list) + (0.5 + i) * self.interval_val)

        self.trans_prob = np.zeros((self.count, self.count))
        self.__init_status_trans_prob()

    def __init_status_trans_prob(self) -> None:
        for i in range(self.count):
            if i != self.count - 1:
                self.interval_list.append(Interval(self.min_relative_err + i * self.interval_val, self.min_relative_err + (i + 1) * self.interval_val))
            else:
                self.interval_list.append(Interval(self.min_relative_err + i * self.interval_val, self.max_relative_err))

        prev = None
        for i, err_val in enumerate(self.relative_err_list):
            matrix = self.__to_vector(err_val)
            if prev:
                self.trans_prob[prev] += matrix
            prev = matrix.index(1)

        sum_list = np.sum(self.trans_prob, axis=1)
        for row_i, row in enumerate(self.trans_prob):
            if sum_list[row_i] != 0:
                for col_i, val in enumerate(row):
                    self.trans_prob[row_i, col_i] /= sum_list[row_i]
```

In [8]:

```python
sum = 0
for i in range(6):
    model = GreyMarkovModel(relative_err_list[:-i-1], 7)
    status_num = np.argmax(model.validate_test_data(), axis=1)[0]
    mc_pred = pred[-i-1] / (1 - model.expected_val_list[status_num])
    err = (raw[-i-1] - mc_pred) / raw[-i-1] * 100
    sum += err
    print(err)
print('mean: ' + str(sum / 6) + '%')
```

```
0.06811352398320765
-0.6141349057096572
0.7553313717076561
3.2589994478337347
0.027630102950403166
-0.5959804585459864
mean: 0.4833265137032263%
```

Thank You