Problem C: Trading Strategies

说明

- 此作品为原创作品，一切的更新仅在：https://mianbaoduo.com/o/bread/mbd-Ypebl55s
- 代码语言python，版本python3.8
- 下面的代码是用来说明的，具体代码见代码py文件

数据处理

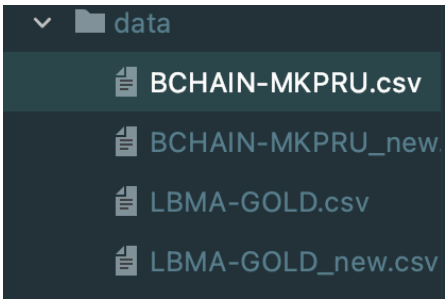
把2个csv原始数据放在data文件夹下：

- 首先我们知道，比特币可以每天交易，但黄金仅在开市日交易，所以黄金的csv文件里面肯定有些日期是空，所以先对空数据进行处理，
- 由于大概有10个空数据，所以不同的处理的方式对结果影响偏差不大，可以选择一下的处理方式之一即可
  - 用平均值填充
  - 删掉此行数据
  - 用此空数据行的上一行或者下一行填充
- 这边以用上一行的数据来填充为例，代码（第6行）如下：

```
1  def handle_gold(data_path):
2      data = pd.read_csv(data_path)
3      # 处理空缺值，设为平均
4      # data['USD (PM)'] = data['USD (PM)'].fillna(data['USD (PM)'].median())
5      # 用缺失值上面的值替换缺失值
6      data = data.fillna(axis=0, method='ffill')
7      data.to_csv('./data/LBMA-GOLD_new.csv')
```


- 如果用均值填充的，就是第4行，

对于比特币的也是如此，不过比特币每天都可以交易，所以没有空值，但是为了统一数据dateframe格式，这里一并处理了，具体见代码，处理后得到新的csv文件



问题一和问题二

建立模型

根据题意，我们可以把总的这个模型定义为：收益W = F(c, g, b)，当然模型的定义不唯一，这里仅提供一种可实现思路

从文件结构分析看，这是比较典型的回归模型组合问题

- 这里以多项式回归模型为例子

- 由题意：初始状态是[1000, 0, 0]，也就是说我们需要预测得到[C, G, B]的下一个状态或者某一个状态
- 我们先分别建立黄金和比特币这两个子模型，并求解第一个问题

黄金回归多项式模型

- 看代码

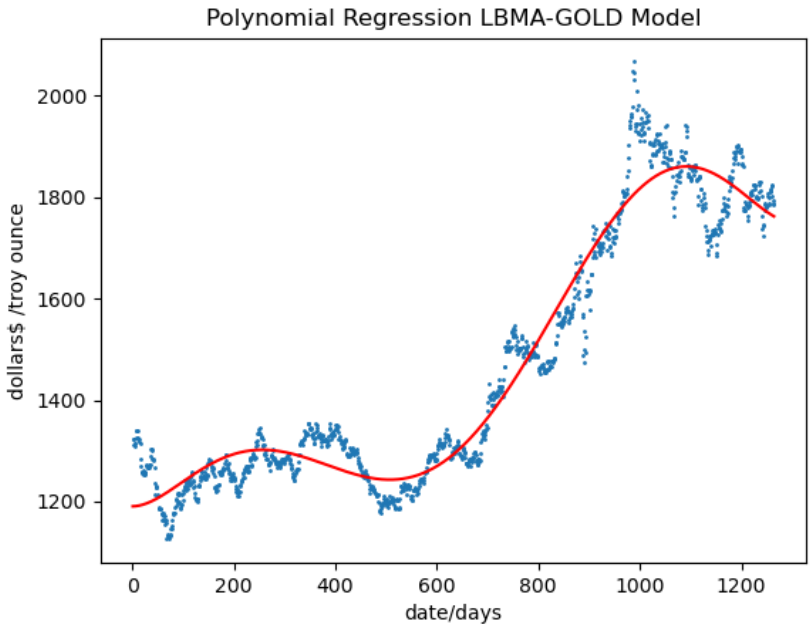
- 71行是处理数据的，如果你第一遍已经处理好，就可以把他注释掉了
- 72行展示数据量的
- 关键是得到38，39行，模型的系数和截距是我们要得到的模型参数，通过这些参数，我们就可以拟合预测出未来第n次交易的情况
- 你可以修改第31行，来调整模型，看哪个效果好几选哪个，注意degree不要选太大，太大的话容易过拟合，太小的话，就不准了


```
1  import pandas as pd
2  import numpy as np
3  from sklearn.linear_model import LinearRegression
4  from sklearn.preprocessing import PolynomialFeatures  # 生成多项式用的
5  import matplotlib.pyplot as plt
6
7
8  def handle_gold(data_path):
9      data = pd.read_csv(data_path)
10     # 处理空缺值，设为平均
11     # data['USD (PM)'] = data['USD (PM)'].fillna(data['USD (PM)'].median())
12     # 用缺失值上面的值替换缺失值
13     data = data.fillna(axis=0, method='ffill')
14     data.to_csv('./data/LBMA-GOLD_new.csv')
15
16
17 def show_gold(data_path):
18     data = pd.read_csv(data_path)
19     x_data = data.iloc[1:, 1]
20     y_data = data.iloc[1:, 2]
21     plt.scatter(x_data, y_data, s=1)
22     plt.show()
23
24 # 建立模型
25 def build_model(data_path):
26     data = pd.read_csv(data_path)
```

```
    x_data = data.iloc[1:, 0]
    y_data = data.iloc[1:, 2]
    # 转换为二维数据
    # degree = n, 相当于n次方拟合
    poly = PolynomialFeatures(degree=6)
    # 特征处理
    x_data = np.array(x_data).reshape((len(x_data), 1))
    x_poly = poly.fit_transform(x_data)
    model = LinearRegression()

    model.fit(x_poly, y_data)
    print('系数: ', model.coef_)
    print('截距: ', model.intercept_)

    # 画图
    plt.scatter(x_data, y_data, s=1)
    plt.plot(x_data, model.predict(x_poly), 'r')  # predict 传的是x_poly,是处理后的数
据
    plt.title('Polynomial Regression LBMA-GOLD Model')
    plt.xlabel('date/days')
    plt.ylabel('dollars$ /troy ounce')
    plt.show()

    return len(y_data), model.coef_[1:], model.intercept_


# 计算最初的 1000 美元投资价值
def prediction(init_money, count, coef: list, intercept, a=0.01):
    """
    :param init_money: 最初的投资
    :param days: 第几次交易
    :param coef: 模型系数
    :param intercept: 模型截距
    :param a: αgold = 1%
    :return: init_money在第days天的投资价值
    """
    res = intercept
    print('第几次交易: ', count)
    for i, c in enumerate(coef):
        res += c * pow(count, i + 1)
    print('每金衡盎司美元(dollars per troy ounce): ', res, '美元$')
    return init_money / res * (1 - a)


if __name__ == '__main__':
    # handle_gold('./data/LBMA-GOLD.csv')
    # show_gold('./data/LBMA-GOLD_new.csv')
    count, coef, intercept = build_model('./data/LBMA-GOLD_new.csv')
```

```
74      print('2021 年 9 月 10 日,最初的 1000 美元投资价值:', prediction(1000, count, coef,
        intercept), '盎司')
75
```

- 结果




```
系数:  [ 0.00000000e+00  5.30147327e-05  8.40271443e-03 -4.23485209e-05
  7.68679724e-08 -5.74630762e-11  1.51678800e-14]
截距:  1190.3827674894403
第几次交易:  1264
每金衡盎司美元(dollars per troy ounce):  1762.235394677591 美元$
2021 年 9 月 10 日,最初的 1000 美元投资价值: 0.5617864690438391 盎司

Process finished with exit code 0
```

- 结果第5行，也就是2021 年 9 月 10 日是第1264次交易，我们由模型可以得到当天1盎司黄金 = 1762.235394677591 美元，由此可以衡量出1000美金的投资价值
- 模型公式从系数和截距得到，比如上面结果得到的如下

```
系数：a1,a2,a3
截距：b
```

那么模型就是`y = a1*x^0 + a1*x^1 + a2*x^2 + b`

上面黄金模型得到的是6次多项式模型，所以最高项也就是6次

比特币的模型也是类似

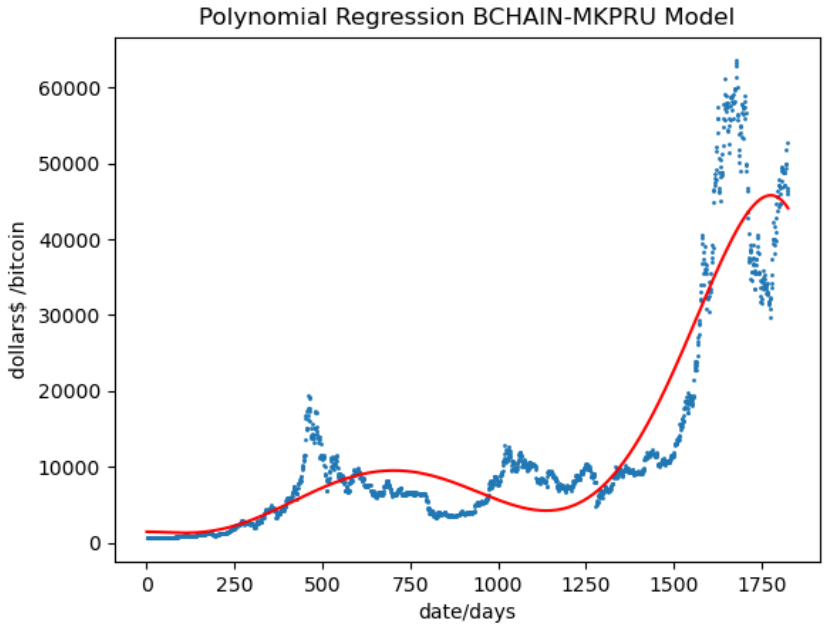
比特币多项式回归模型

- 需要改动的地方和黄金模型类似，不过个别地方需要注意，αbitcoin = 2%这些

```python
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures  # 生成多项式用的
import matplotlib.pyplot as plt


def handle_bitcoin(data_path):
    data = pd.read_csv(data_path)
    # 用缺失值上面的值替换缺失值
    data = data.fillna(axis=0, method='ffill')
    data.to_csv('./data/BCHAIN-MKPRU_new.csv')


def show_bitcoin(data_path):
    data = pd.read_csv(data_path)
    x_data = data.iloc[1:, 1]
    y_data = data.iloc[1:, 2]
    plt.scatter(x_data, y_data, s=1)
    plt.show()


def build_model(data_path):
    data = pd.read_csv(data_path)
    x_data = data.iloc[1:, 0]
    y_data = data.iloc[1:, 2]
    # 转换为二维数据
    # degree = n，相当于n次方拟合
    poly = PolynomialFeatures(degree=6)
    # 特征处理
    x_data = np.array(x_data).reshape((len(x_data), 1))
    x_poly = poly.fit_transform(x_data)
    model = LinearRegression()

    model.fit(x_poly, y_data)
    print('系数：', model.coef_)
    print('截距：', model.intercept_)

    # 画图
    plt.scatter(x_data, y_data, s=1)
    plt.plot(x_data, model.predict(x_poly), 'r')  # predict 传的是x_poly,是处理后的数据
    plt.title('Polynomial Regression BCHAIN-MKPRU Model')
    plt.xlabel('date/days')
    plt.ylabel('dollars$ /bitcoin')
```

```
    plt.show()

    return len(y_data), model.coef_[1:], model.intercept_


# 计算最初的 1000 美元投资价值
def prediction(init_money, count, coef: list, intercept, a=0.02):
    """
    :param init_money: 最初的投资
    :param count: 第几次交易
    :param coef: 模型系数
    :param intercept: 模型截距
    :param a: αbitcoin = 2%
    :return: init_money在第days天的投资价值
    """
    res = intercept
    print('第几次交易: ', count)
    for i, c in enumerate(coef):
        res += c * pow(count, 1 + i)
    print('比特币每日价格(dollars per bitcoin): ', res, '美元$')
    return init_money / res * (1 - a)


if __name__ == '__main__':
    # handle_bitcoin('./data/BCHAIN-MKPRU.csv')
    # show_bitcoin('./data/BCHAIN-MKPRU_new.csv')
    count, coef, intercept = build_model('./data/BCHAIN-MKPRU_new.csv')
    print('2021 年 9 月 10 日,最初的 1000 美元投资价值:', prediction(1000, count, coef,
intercept), '比特币')

```


- 结果




```
系数:  [ 0.00000000e+00 -5.52957703e-04 -4.65922448e-02  3.80701831e-04
 -6.85859284e-07  4.58595999e-10 -1.02636974e-13]
截距:  1433.8246533975998
第几次交易:  1825
比特币每日价格(dollars per bitcoin):  44117.196522225626 美元$
2021 年 9 月 10 日,最初的 1000 美元投资价值: 0.022213560181828185 比特币

Process finished with exit code 0
```


- 这里的分析类似黄金模型
- 最后得到，2021 年 9 月 10 日比特币每日价格(dollars per bitcoin): 44117.196522225626 美元$

求解

- 求解函数主要在prediction
- 所以我们得到：初始的[C, G, B]从[1000， 0， 0] 到 [1000, 1762.24, 44117.20]的结果
- 模型我们可以设为: 收益W = F(c, g, b)，而g和b就是我们上面得到的2个子模型，c就是当前的投资，限制c，g，b之间关系的就是策略条件，也就是哪些时候交易，交易价的最低点和最高点的时间点，以及汇率都有影响

策略

- 很明显，无论是比特币还是黄金，我们要交易的思路，肯定是在汇率最低的时候买入，在汇率最高的时候卖出，这样我们赚的才是最多的，思想有点类似股票模型，大家也可以去参考参考一下股票预测模型
- 接下来讨论W = F(c, g, b)
- 对于这样的组合模型，下面提供一种思路，题意并没有具体说到要预测后面多长时间的收益情况，所以这里我们的模型就是**基于多项式回归模型的组合改进模型**，如果有硬性说明要求后面长期的收益情况，那建议大家用LSTM这种长短时记忆模型，但是这里以短期模型为例，当然，短期的模型还有灰度预测模型等，不过无论是

用哪种经典的模型，你都不可能照搬过来，你肯定是要做改进的

我们一步步分析：

```
W = F(c, g, b) = F(c, g(t), b(t)) = F卖 - F买
t是交易时间点，我们这边用第几天/次交易来表示
很明显我们由上面的代码结果得到了：
            g(t) = a1*t^0 + a1*t^1 + a2*t^2 + ... + b1, 这里的an就是黄金回归模型对应的系数
            b(t) = a1*t^0 + a1*t^1 + a2*t^2 + ... + b2, 这里的an是比特币回归模型对应的系数
            c是我们最开始需要去低价买入的本金，也是最后总收益需要减掉的
    这里模型的an系数是根据代码里面的degree来的，主要看你如何设置次数以及前面数据处理的方式，都会对系数
    有影响。

定义下面几个变量：tbmin, tbmax, tgmin, tgmax
分别表示：比特币汇率最低时间点，最高时间点，黄金汇率最低时间点，最高时间点
则可以得到：
F卖 - F买 = F(c, g(tgmax), b(tbmax)) - F(c, g(tgmin), b(tbmin))

但是实际上，我们不能只考虑最值点，应该考虑的是多个较大或较小的点（包含最值点），因为要保证最大收益
所以上面的tbmin, tbmax, tgmin, tgmax分别代表着比特币汇率较低时间点(可以有多个，不同)，较高时间点，黄金汇率较低时间点，较高时间点，也就是不只是4个点，可以是多个点，我们待会通过代码来设置即可
```


- 下面代码需要自己改动的是82到87行，这里得根据你自己前面得到的子模型的数据去写
- 另外优化的参数是102,103行的r，不同的输入数据得到的结果会有偏差，这个r也是需要自己改的
- `F(c, g(tgmax), b(tbmax)) - F(c, g(tgmin), b(tbmin))`体现在第42行的max_w - min_w的累积计算

```python
import pandas as pd

max_best_days_g = []
max_best_days_b = []
min_best_days_g = []
min_best_days_b = []


# 获取模型当天的交易额
def prediction(count, coef: list, intercept, a):
    res = intercept
    # print('第几次交易：', count)
    for i, c in enumerate(coef):
        res += c * pow(count, 1 + i)
    return res * (1 - a)
```

```python


def get_point(file_path, coef: list, intercept, a, num=3):
    d = dict()
    data = pd.read_csv(file_path)
    y_data = data.iloc[1:, 2]
    for i in range(len(y_data)):
        d[i] = y_data[i + 1]
    d = sorted(d.items(), key=lambda x: x[1])
    # print('前几个较大的交易点', d[-num:])
    # print('前几个较小的交易点', d[: num])
    global max_best_days_g, min_best_days_g, max_best_days_b, min_best_days_b
    if 'GOLD' in file_path:
        max_best_days_g = d[-num:]
        min_best_days_g = d[:num]
    else:
        max_best_days_b = d[-num:]
        min_best_days_b = d[:num]

    # 把点传入模型进行计算,max_w卖出， min_w买入
    max_w, min_w = 0, 0
    for e in d[-num:]:
        max_w += prediction(e[0], coef, intercept, a)
    for e in d[:num]:
        min_w += prediction(e[0], coef, intercept, a)

    return max_w - min_w


# 找到最佳的交易时间
def find_trading_days(data_path_g, data_path_b):
    g_data = pd.read_csv(data_path_g).iloc[1:, 1]
    b_data = pd.read_csv(data_path_b).iloc[1:, 1]
    res_date_sale_g = []
    res_date_sale_b = []
    res_date_buy_g = []
    res_date_buy_b = []

    for i in max_best_days_g:
        res_date_sale_g.append(g_data[i[0]])
    for i in max_best_days_b:
        res_date_sale_b.append(b_data[i[0]])
    for i in min_best_days_g:
        res_date_buy_g.append(g_data[i[0]])
    for i in min_best_days_g:
        res_date_buy_b.append(b_data[i[0]])
    print('最佳黄金卖出时间', res_date_sale_g)
    print('最佳黄金买出时间', res_date_buy_g)
    print('最佳比特币卖出时间', res_date_sale_b)
```

```
    print('最佳比特币买出时间', res_date_buy_b)


def train(file_path, coef: list, intercept, a, r=11):
    dw = dict()
    for i in range(1, r):
        w = get_point(file_path, coef, intercept, a, i)
        dw[i] = w
    print(dw)
    dw = sorted(dw.items(), key=lambda x: x[1])
    best_num = max(dw)[0]
    best_w = max(dw)[1]
    print(best_num, best_w)
    return best_num, best_w


if __name__ == '__main__':
    bit_coef = [-5.52957703e-04, -4.65922448e-02, 3.80701831e-04,
                -6.85859284e-07, 4.58595999e-10, -1.02636974e-13]
    gold_coef = [5.30147327e-05, 8.40271443e-03, -4.23485209e-05,
                 7.68679724e-08, -5.74630762e-11, 1.51678800e-14]
    bit_intercept = 1433.8246533975998
    gold_intercept = 1190.3827674894403

    # 下面是找3个较大和3个较小的点的例子
    print('3个较大和3个较小的点的例子')
    gold_w = get_point('./data/LBMA-GOLD_new.csv', gold_coef, gold_intercept,
0.01, 3)
    print('黄金总收益:', gold_w)

    bitcoin_w = get_point('./data/BCHAIN-MKPRU_new.csv', bit_coef, bit_intercept,
0.02, 3)
    print('比特币总收益:', bitcoin_w)

    W = gold_w + bitcoin_w - 1000
    print('总收益:', W, 'dollars\n')

    # 下面是自动找多个较大和较小的点，控制范围是r，也就是点的个数，可以自己调整
    print('多个较大较小点的例子\n')
    best_num1, gold_w = train('./data/LBMA-GOLD_new.csv', gold_coef,
gold_intercept, 0.01, r=11)
    best_num2, bitcoin_w = train('./data/BCHAIN-MKPRU_new.csv', bit_coef,
bit_intercept, 0.02, r=11)
    print('黄金总收益:', gold_w)
    print('比特币总收益:', bitcoin_w)
    print('黄金最佳的选点方案的交易次数: ', best_num1)
    print('比特币最佳的选点方案的交易次数: ', best_num2)

    W = gold_w + bitcoin_w - 1000
```

```
    print('总收益:', W, 'dollars')

    find_trading_days('./data/LBMA-GOLD_new.csv', './data/BCHAIN-MKPRU_new.csv')

```

- 结果

```
3个较大和3个较小的点的例子
黄金总收益: 1738.540734107969
比特币总收益: 116112.68903681794
总收益: 116851.22977092591 dollars

多个较大较小点的例子

{1: 578.9666010475939, 2: 1158.815821809158, 3: 1738.540734107969, 4: 2316.5881348150824, 5: 2904.8606027092137, 6: 3492.16309321276, 7: 4066.4313804528065, 8: 4653.09039112217, 9: 5245.160707900817, 10: 5842.555674543442}
10 5842.555674543442
{1: 38621.72572312448, 2: 77409.08891509722, 3: 116112.68903681794, 4: 154986.30504686892, 5: 190810.3399985352, 6: 229769.42306144527, 7: 268223.6468409848, 8: 306766.575938994, 9: 345138.9909343007, 10: 381053.0646109808}
10 381053.0646109808
黄金总收益: 5842.555674543442
比特币总收益: 381053.0646109808
黄金最佳的选点方案的交易次数:  10
比特币最佳的选点方案的交易次数:  10
总收益: 385895.62028552423 dollars
最佳黄金卖出时间 ['9/9/20', '8/28/20', '8/14/20', '8/3/20', '8/18/20', '8/17/20', '8/6/20', '8/7/20', '8/4/20', '8/5/20']
最佳黄金买出时间 ['12/19/16', '12/14/16', '12/21/16', '12/22/16', '12/15/16', '12/20/16', '12/23/16', '12/16/16', '12/28/16', '12/29/16']
最佳比特币卖出时间 ['3/14/21', '4/10/21', '4/12/21', '4/11/21', '4/17/21', '3/13/21', '4/16/21', '4/14/21', '4/15/21', '4/13/21']
最佳比特币买出时间 ['11/20/16', '11/17/16', '11/22/16', '11/23/16', '11/18/16', '11/21/16', '11/24/16', '11/19/16', '11/25/16', '11/26/16']

Process finished with exit code 0


```


- 当你找到一个比较好的收益时，记下r这个值，然后把101-110行注释掉，把91行和94行的num改成你r范围内的最佳次数，然后再运行，就是我们要什么时候交易收益最大的方案了

问题三

- 确定策略对交易成本的敏感程度。 交易成本如何影响策略和结果

这里要用灵敏度分析的做法，做法的思路就是：固定其他变量不变来修改某一个变量，然后画出这个模型，看看对实际的影响大不大

- 比如这里的话可以用画图做出模型曲线来判断，也可以算出总收益，看看变化大不大；举个例子，你固定模型里面g的tmin,tmax，然后去改变b的tmin，tmax，其他的（包括代码的num，r）都不变，然后求出最大的总收益，看看这个总收益和原来的差距大不大，同样的道理，改变g，不改变b也算一下，如果你在你的模型W中又加入其他影响因素的话，做法也是类似，有点像控制变量法
- 那怎么判断结果呢

  如果这个变化大的话，那就说明你的模型灵敏度大，容易受到影响，波动大

  否则反之。

问题四

- 也就是写个2页的备忘录去描述你上面的解法，这里备忘录的格式大家去网上参考找规范的来，以前这里最后一问都是写封信或者写个日记，或者写个2页的教课书课文什么的，今年是备忘录，我也头一回见到备忘录，可能是翻译问题，建议大家去网上找规格的来写就行，基本就是总结的策略、模型和结果
- 网上有格式的，你们可以看看，我会把相关链接放在付费内容页的最下方

后续有更新会通知大家的

- 更新仅在：https://mianbaoduo.com/o/bread/mbd-Ypebl55s

注意

- 代码的注释如果到时候你们自己修改完要注意不要出现中文，否则最高不会超过S奖，甚至gg
- 如果你的论文还没放代码的话，就已经到达规定页码了，这个时候你就不要放代码到附录了，如果没超过或者超过一点的话，就自己压缩一下，然后把代码核心部分放上去即可，如果代码全放上去刚刚好的，那就都放上去