招行Fintech数据GB_Blend方法

数据分两部分，train_data与wkd，分别为训练数据与节假日数据

train_data:

列名	格式
biz_type	A1~A13 B1
date	YYYY-MM-DD
period	1~48 (00:00:00~23:30:00)
amount	数据

wkd:

列名	格式
date	YYYY-MM-DD
WKD_TYPE_CD	WN: 正常工作日 SN: 周末 NH: 法定假日 SS: 工作日调休 WS: 假期调为工作日

思路为，对A1~A13、B1业务分别建模，提取时间特征与周度特征，利用GB类算法进行训练与预测，最终结果取四个模型的加权和（日预测）或两个模型的加权和（时间段预测）。

---

1. 预处理

预处理为两步，首先合并wkd与train_data表，其次分割不同业务数据，数据集分为A1~A13、B1

预处理代码运行时间较长，建议直接运行task1与task2代码

```
import pandas as pd
import numpy as np

from zhprepocess.concat_all import concat_all
from zhprepocess.split_type import split_type

# ==============================================================================
# 1. 合并wkd与train_data表
# ==============================================================================

print('这一步很慢，建议直接运行task1与task2，预处理后数据已经在压缩包中给出')

# 读取数据
```

```
train_data = pd.read_csv(r"./train_v1.csv")
wkd = pd.read_csv(r"./wkd_v1.csv",sep=",",encoding="UTF-8")

# date列转换为datetime格式
train_data['date']=pd.to_datetime(train_data['date'])

# 节假日类型数字化表示
wkd = wkd.replace(['WN','SN','NH','SS','WS'],[0,1,2,3,4])
wkd['ORIG_DT']=pd.to_datetime(wkd['ORIG_DT'])
train_data['wkd'] = ''

# 合并表
train_data = concat_all(wkd, train_data)



# ================================================================================
# 2. 分割不同业务数据
# ================================================================================

type_data = split_type(train_data)
```

concat_al合并函数:

```
# ================================================================================
# 合并节假日与原数据矩阵
# ================================================================================
def concat_all(wkd, train_data, train=True):
    train_data_len = len(train_data)
    if train:
        for i in range(len(wkd)):
            w_date = wkd.loc[i, 'ORIG_DT']
            wkd_num = wkd.loc[i, 'WKD_TYP_CD']
            j = 0
            while j < train_data_len:
                t_date = train_data.loc[j, 'date']
                if t_date == w_date:
                    train_data.loc[j:j + 47, 'wkd'] = wkd_num
                j += 48
            # print(i)
    else:
        for i in range(len(wkd)):
            w_date = wkd.loc[i, 'ORIG_DT']
            wkd_num = wkd.loc[i, 'WKD_TYP_CD']
            j = 0
            while j < train_data_len:
                t_date = train_data.loc[j, 'date']
                if t_date == w_date:
                    train_data.loc[j:j + 2, 'wkd'] = wkd_num
                j += 2
            # print(i)
    return train_data
```

split_type拼接函数：

```
# ==============================================================================
# 将每种业务数据分离
# ==============================================================================
def split_type(train_data):
    A1 = train_data[(train_data['biz_type'] == 'A1')]
    A2 = train_data[(train_data['biz_type'] == 'A2')]
    A3 = train_data[(train_data['biz_type'] == 'A3')]
    A4 = train_data[(train_data['biz_type'] == 'A4')]
    A5 = train_data[(train_data['biz_type'] == 'A5')]
    A6 = train_data[(train_data['biz_type'] == 'A6')]
    A7 = train_data[(train_data['biz_type'] == 'A7')]
    A8 = train_data[(train_data['biz_type'] == 'A8')]
    A9 = train_data[(train_data['biz_type'] == 'A9')]
    A10 = train_data[(train_data['biz_type'] == 'A10')]
    A11 = train_data[(train_data['biz_type'] == 'A11')]
    A12 = train_data[(train_data['biz_type'] == 'A12')]
    A13 = train_data[(train_data['biz_type'] == 'A13')]
    B1 = train_data[(train_data['biz_type'] == 'B1')]

    A1.to_csv(r"./pocessed_data/A1.csv", index = 0)
    A2.to_csv('r"./pocessed_data/A2.csv', index = 0)
    A3.to_csv('r"./pocessed_data/A3.csv', index = 0)
    A4.to_csv('r"./pocessed_data/A4.csv', index = 0)
    A5.to_csv('r"./pocessed_data/A5.csv', index = 0)
    A6.to_csv('r"./pocessed_data/A6.csv', index = 0)
    A7.to_csv('r"./pocessed_data/A7.csv', index = 0)
    A8.to_csv('r"./pocessed_data/A8.csv', index = 0)
    A9.to_csv('r"./pocessed_data/A9.csv', index = 0)
    A10.to_csv('r"./pocessed_data/A10.csv', index = 0)
    A11.to_csv('r"./pocessed_data/A11.csv', index = 0)
    A12.to_csv('r"./pocessed_data/A12.csv', index = 0)
    A13.to_csv('r"./pocessed_data/A13.csv', index = 0)
    B1.to_csv('r"./pocessed_data/B1.csv', index = 0)
    return
```

2. 特征工程

2.1 提取月度特征（方案废除，可以不看，只是提供思路历程）

提取特征，分别提取当天对应week，当天对应day，当天对应季度

之后按月度颗粒大小，提取前一个月的mean，前一个月的std，前一个月的max等等

对task1训练数据进行特征工程

```
def to_train_data(data, plus=False):
    data['date']=pd.to_datetime(data['date'])
    A_train = data
    # 30648 -> 十月
    start = 35040
    end = 49680
    gap = 1464
    A_train['week'] = A_train['date'].dt.week
    A_train['quarter'] = A_train['date'].dt.quarter
```

```
    A_train['day'] = A_train['date'].dt.day
    for i in range(12):
        # 默认提取前三月特征
        for j in range(3):
            A_train.loc[start + i * gap : start + (i + 1) * gap, ('last_'+str(j
+ 1)+'_month_mean')] = A_train.loc[start  + (i - j - 1) * gap : start + i * gap,
( 'amount')].mean()
            A_train.loc[start + i * gap : start + (i + 1) * gap, ('last_'+str(j
+ 1)+'_month_std')] = A_train.loc[start  + (i - j - 1) * gap : start + i * gap,
( 'amount')].std(ddof=0)
            A_train.loc[start + i * gap : start + (i + 1) * gap, ('last_'+str(j
+ 1)+'_month_max')] = A_train.loc[start  + (i - j - 1) * gap : start + i * gap,
( 'amount')].max()
    A_train = A_train[start:end]
    A_train=A_train.groupby(A_train['date']).apply(concat_func1).reset_index()
    # A_train.columns = ['date','day_of_week','month',  'wkd', 'week', 'quarter',
'amount','last_one_month_mean', 'last_two_month_mean', 'last_one_month_std',
'last_two_month_std', 'last_one_month_max', 'last_two_month_max',
'last_three_month_max', 'last_three_month_mean',
'last_three_month_std','last_one_month_skew','last_two_month_skew','last_three_m
onth_skew']
    # X_train = pd.DataFrame(A_train, columns = ['day_of_week', 'month',  'wkd',
'week', 'quarter', 'last_one_month_mean', 'last_two_month_mean',
'last_one_month_std', 'last_two_month_std', 'last_one_month_max',
'last_two_month_max', 'last_three_month_max', 'last_three_month_mean',
'last_three_month_std','last_one_month_skew','last_two_month_skew','last_three_m
onth_skew'])


    # 没加date以前是0.07
    A_train.columns = ['date',
                       'day',
                       'day_of_week',
                       'month',
                       'wkd',
                       'week',
                       'quarter',
                       'amount',
                        'last_1_month_mean',
                        'last_2_month_mean',
                       'last_1_month_std',
                       'last_2_month_std',
                        # 'last_3_month_mean',
                        # 'last_3_month_std',
                        # 'last_1_month_max',
                        # 'last_2_month_max',
                        # 'last_3_month_max'
                       ]
    X_train = pd.DataFrame(A_train, columns = ['day',
                                               'day_of_week',
                                               'month',
                                               'wkd',
                                               'week',
                                               'quarter',
                                                'last_1_month_mean',
                                                'last_2_month_mean',
                                               'last_1_month_std',
                                               'last_2_month_std',
```

```
                                                # 'last_3_month_mean',
                                                # 'last_3_month_std',
                                                # 'last_1_month_max',
                                                # 'last_2_month_max',
                                                # 'last_3_month_max'
                                               ])

    y_train = pd.DataFrame(A_train, columns = ['amount'])
    data_list = []
    data_list.append(X_train)
    data_list.append(y_train)


    return data_list

def concat_func1(x):
    return pd.Series([
        x['day'].mean(),
        x['day_of_week'].mean(),
        x['month'].mean(),
        x['wkd'].mean(),
        x['week'].mean(),
        x['quarter'].mean(),
        x['amount'].sum(),
        x['last_1_month_mean'].mean(),
        x['last_2_month_mean'].mean(),
        x['last_1_month_std'].mean(),
        x['last_2_month_std'].mean(),
        # x['last_3_month_mean'].mean(),
        # x['last_3_month_std'].mean(),
        # x['last_1_month_max'].mean(),
        # x['last_2_month_max'].mean(),
        # x['last_3_month_max'].mean(),
        # x['last_one_month_skew'].mean(),
        # x['last_two_month_skew'].mean(),
        # x['last_three_month_skew'].mean(),
        ]
    )
```

对task1的测试数据进行特征工程

```
def to_day_test_data(data, last):
    data['date']=pd.to_datetime(data['date'])
    # A=data.groupby(data['date']).apply(concat_func1).reset_index()
    A_test = data
    A_test['week'] = A_test['date'].dt.week
    A_test['quarter'] = A_test['date'].dt.quarter
    A_test['day'] = A_test['date'].dt.day
    end = 49680
    gap = 1464
    for j in range(3):
        A_test.loc[0 : 60, ('last_'+str(j + 1)+'_month_mean')] = last.loc[end -
(j + 1)*gap : end, ( 'amount')].mean()
        A_test.loc[0 : 60, ('last_'+str(j + 1)+'_month_std')] = last.loc[end -
(j + 1)*gap : end, ( 'amount')].std(ddof=0)
        A_test.loc[0 : 60, ('last_'+str(j + 1)+'_month_max')] = last.loc[end -
(j + 1)*gap : end, ( 'amount')].max()
```

```
    X_test = pd.DataFrame(A_test, columns =  ['day',
                                              'day_of_week',
                                              'month',
                                              'wkd',
                                              'week',
                                              'quarter',
                                               'last_1_month_mean',
                                                'last_2_month_mean',
                                              'last_1_month_std',
                                              'last_2_month_std',
                                               # 'last_3_month_mean',
                                               # 'last_3_month_std',
                                               # 'last_1_month_max',
                                               # 'last_2_month_max',
                                               # 'last_3_month_max'
                                              ])
    return X_test
```

对task2的训练数据特征工程：

```
def to_periods_train_data(data):
    data['date']=pd.to_datetime(data['date'])
    A_train = data
    # 30648 -> 十月
    start = 43824
    end = 49680
    gap = 1464
    A_train['week'] = A_train['date'].dt.week
    A_train['quarter'] = A_train['date'].dt.quarter
    A_train['day'] = A_train['date'].dt.day
    for i in range(12):
        # 默认提取前三月特征
        for j in range(3):
            A_train.loc[start + i * gap : start + (i + 1) * gap, ('last_'+str(j
+ 1)+'_month_mean')] = A_train.loc[start  + (i - j - 1) * gap : start + i * gap,
( 'amount')].mean()
            A_train.loc[start + i * gap : start + (i + 1) * gap, ('last_'+str(j
+ 1)+'_month_std')] = A_train.loc[start  + (i - j - 1) * gap : start + i * gap,
( 'amount')].std(ddof=0)
            A_train.loc[start + i * gap : start + (i + 1) * gap, ('last_'+str(j
+ 1)+'_month_max')] = A_train.loc[start  + (i - j - 1) * gap : start + i * gap,
( 'amount')].max()
    A_train = A_train[start:end]
    # X_train = pd.DataFrame(A_train, columns = ['day','day_of_week', 'month',
'periods',  'wkd', 'week', 'quarter'])
    X_train = pd.DataFrame(A_train, columns = ['day','day_of_week', 'month',
'periods',  'wkd', 'week', 'quarter', 'last_1_month_mean', 'last_2_month_mean',
'last_1_month_std', 'last_2_month_std'])
    y_train = pd.DataFrame(A_train, columns = ['amount'])
    data_list = []
    data_list.append(X_train)
    data_list.append(y_train)
    return data_list
```

对task2的测试数据进行特征工程：

```
def to_periods_test_data(data, last):
    data['date']=pd.to_datetime(data['date'])
    # A=data.groupby(data['date']).apply(concat_func1).reset_index()
    A_test = data
    A_test['week'] = A_test['date'].dt.week
    A_test['quarter'] = A_test['date'].dt.quarter
    A_test['day'] = A_test['date'].dt.day
    end = 49680
    gap = 1464
    for j in range(3):
        A_test.loc[0 : 2880, ('last_'+str(j + 1)+'_month_mean')] = last.loc[end
- (j + 1)*gap : end, ( 'amount')].mean()
        A_test.loc[0 : 2880, ('last_'+str(j + 1)+'_month_std')] = last.loc[end -
(j + 1)*gap : end, ( 'amount')].std(ddof=0)
        A_test.loc[0 : 2880, ('last_'+str(j + 1)+'_month_max')] = last.loc[end -
(j + 1)*gap : end, ( 'amount')].max()
    # X_test = pd.DataFrame(A_test, columns =  ['day','day_of_week', 'month',
'periods',  'wkd', 'week', 'quarter', 'last_1_month_mean', 'last_2_month_mean',
'last_1_month_std', 'last_2_month_std', 'last_1_month_max', 'last_2_month_max'])
    X_test = pd.DataFrame(A_test, columns =  ['day','day_of_week', 'month',
'periods',  'wkd', 'week', 'quarter'])
    # X_test = pd.DataFrame(A_test, columns =  ['day_of_week', 'month',
'periods',  'wkd', 'week', 'quarter', 'last_1_month_mean', 'last_2_month_mean',
'last_1_month_std', 'last_2_month_std', 'last_1_month_max', 'last_2_month_max',
'last_3_month_max', 'last_3_month_mean', 'last_3_month_std'])
    # X_test = pd.DataFrame(A_test, columns =  ['day_of_week', 'month',
'periods',  'wkd', 'week', 'quarter', 'last_1_month_mean', 'last_2_month_mean',
'last_1_month_std', 'last_2_month_std'])
    # X_test = pd.DataFrame(A_test, columns =  ['day_of_week', 'month',
'periods',  'wkd', 'week', 'quarter', 'last_1_month_mean',  'last_1_month_std',
'last_1_month_max'])
    return X_test
```

2.2 提取周度特征（现使用方案）

使用周度特征，添加之前第n周的'amount'均值与标准差特征。除此之外，还添加day_of_week，month，quarter，monthday等特征。

对task1训练数据进行特征工程：

```
import pandas as pd

def to_train_data(data, plus=False):
    data['date']=pd.to_datetime(data['date'])
    A_train = data
    A_train['day_of_week'] = A_train['date'].dt.weekday + 1
    A_train['month'] = A_train['date'].dt.month
    A_train['week'] = A_train['date'].dt.isocalendar().week
    A_train['quarter'] = A_train['date'].dt.quarter
    A_train['day'] = A_train['date'].dt.day


    start = 47760
    end = 51120
    gap = 336
    for i in range(30):
     # 默认提取前6~12周特征
```

```
        for j in range(10):
            A_train.loc[start + i * gap : start + (i + 1) * gap, ('last_'+str(j
+ 1)+'_week_mean')] = A_train.loc[start  + (i - j - 6) * gap : start + (i-j-5) *
gap, ( 'amount')].mean()
            A_train.loc[start + i * gap : start + (i + 1) * gap, ('last_'+str(j
+ 1)+'_week_std')] = A_train.loc[start  + (i - j - 6) * gap : start + (i-j-5) *
gap, ( 'amount')].std(ddof=0)
            A_train.loc[start + i * gap : start + (i + 1) * gap, ('last_'+str(j
+ 1)+'_week_max')] = A_train.loc[start  + (i - j - 6) * gap : start + (i-j-5) *
gap, ( 'amount')].max()
            A_train.loc[start + i * gap : start + (i + 1) * gap, ('last_'+str(j
+ 1)+'_week_skew')] = A_train.loc[start  + (i - j - 6) * gap : start + (i-j-5) *
gap, ( 'amount')].skew()
    A_train = A_train[start:end]
    A_train=A_train.groupby(A_train['date']).apply(concat_func1).reset_index()

    A_train.columns = ['date',
                       'day',
                        'day_of_week',
                        'month',
                        'wkd',
                        'week',
                        'quarter',
                        'amount',
                        'last_1_week_mean',
                        'last_1_week_std',
                        'last_2_week_mean',
                        'last_2_week_std',
                        'last_3_week_mean',
                        'last_3_week_std',
                        'last_4_week_mean',
                        'last_4_week_std',
                        'last_5_week_mean',
                        'last_5_week_std',
                        'last_6_week_mean',
                        'last_6_week_std',
                        'last_7_week_mean',
                        'last_7_week_std',
                        ]
    X_train = pd.DataFrame(A_train, columns = ['day',
                                               'day_of_week',
                                               'month',
                                               'wkd',
                                               'week',
                                               'quarter',
                                                'last_1_week_mean',
                                                'last_1_week_std',
                                                'last_2_week_mean',
                                                'last_2_week_std',
                                                'last_3_week_mean',
                                                'last_3_week_std',

                                                'last_4_week_mean',
                                                'last_4_week_std',
                                                 'last_5_week_mean',
                                                 'last_5_week_std',
                                                 'last_6_week_mean',
                                                 'last_6_week_std',
```

```
                                                'last_7_week_mean',
                                                'last_7_week_std',
                                            ])

    y_train = pd.DataFrame(A_train, columns = ['amount'])
    data_list = []
    data_list.append(X_train)
    data_list.append(y_train)


    return data_list

def concat_func1(x):
    return pd.Series([
        x['day'].mean(),
        x['day_of_week'].mean(),
        x['month'].mean(),
        x['wkd'].mean(),
        x['week'].mean(),
        x['quarter'].mean(),
        x['amount'].sum(),
        x['last_1_week_mean'].mean(),
        x['last_1_week_std'].mean(),
        x['last_2_week_mean'].mean(),
        x['last_2_week_std'].mean(),
        x['last_3_week_mean'].mean(),
        x['last_3_week_std'].mean(),
        x['last_4_week_mean'].mean(),
        x['last_4_week_std'].mean(),
        x['last_5_week_mean'].mean(),
        x['last_5_week_std'].mean(),
        x['last_6_week_mean'].mean(),
        x['last_6_week_std'].mean(),
        x['last_7_week_mean'].mean(),
        x['last_7_week_std'].mean(),

        ]
    )
```

3. GB类模型训练+blending融合

task1

task1由于数据较少，为了提高鲁棒性与泛化性，使用XGBRegressor、LGBMRegressor、AdaBoostRegressor、GradientBoostingRegressor四个GB类算法进行Blend权重加和计算，**由于AdaBoost与GBDT的列采样等随机性，每次运行结果会稍有不同，但影响不大，如果去除随机性将降低泛化性与鲁棒性，故不建议这么做。**

另，12月为年末，对比18、19年数据，并查阅相关资料后得出结论，即年末12月的业务量要比11月上升些许，故在最终结果前加入一个权重增加环节，作为对月度趋势的整体预测，经测试发现A业务权重为1.14倍时、B业务权重为1.12倍时效果最好。

```
# ========================================================================
```

```
# Training&Predicting_stage
# ================================================================================

A_amount = []
B_amount = []

for i in range(len(day_data)):
    reg1 = XGBRegressor(tree_method='gpu_hist', gpu_id=0)
    reg2 = LGBMRegressor()
    reg3 = AdaBoostRegressor()
    reg4 = GradientBoostingRegressor()
    X_train = day_data[i][0]
    X_train['wkd'] = X_train['wkd'].astype(int)
    X_train = X_train.astype(float)
    y_train = np.array(day_data[i][1])
    y_train = y_train.ravel()
    X_test = day_test_data[i]
    X_test = X_test.astype(float)


    reg1.fit(X_train, y_train)
    reg2.fit(X_train, y_train)
    reg3.fit(X_train, y_train)
    reg4.fit(X_train, y_train)

    reg1_result = reg1.predict(X_test)
    reg2_result = reg2.predict(X_test)
    reg3_result = reg3.predict(X_test)
    reg4_result = reg4.predict(X_test)

    #Blend操作
    forecast_data = 0.4 * reg1_result + 0.2 * reg2_result + 0.2 * reg3_result +
0.2 * reg4_result
    forecast_data = forecast_data.astype(int)
    if i <= 12:
        A_amount.append(forecast_data)
    else:
        B_amount.append(forecast_data)

A_result = A_amount[0]
B_result = B_amount[0]
final_result = []

for i in range(len(A_amount) - 1):
    A_result = np.array(A_amount[i]) + np.array(A_result)

#加入权重预测
A_result = (A_result * 1.14).astype(int)
B_result = (B_result * 1.12).astype(int)

for i in range(62):
    if i % 2 == 0:
        final_result.append(A_result[i])
    else:
        final_result.append(B_result[i])


final_result = np.array(final_result)
```

```
for i in range(len(final_result)):
    if final_result[i] < 1000:
        final_result[i] = 0
```

task2

task2使用XGBRegressor、LGBMRegressor两个GB类算法进行Blend权重加和计算。

另，12月为年末，对比18、19年数据，并查阅相关资料后得出结论，即年末12月的业务量要比11月上升些许，故在最终结果前加入一个权重增加环节，作为对月度趋势的整体预测，经测试发现A业务权重为1.14倍时、B业务权重为1.0倍时效果最好。

```
# ==============================================================================
# Training&Predicting_stage
# ==============================================================================
A_periods_amount = []
B_periods_amount = []


for i in range(len(periods_data)):

    reg1 = XGBRegressor(tree_method='gpu_hist', gpu_id=0)
    reg2 = LGBMRegressor()

    X_train = periods_data[i][0]
    X_train = X_train.astype(float)
    y_train = np.array(periods_data[i][1])
    y_train = y_train.ravel()
    X_test = periods_test_data[i]
    X_test = X_test.astype(float)

    reg1.fit(X_train, y_train)
    reg2.fit(X_train, y_train)


    reg1_result = reg1.predict(X_test)
    reg2_result = reg2.predict(X_test)

    #Blend操作
    forecast_data = 0.5 * reg1_result + 0.5 * reg2_result

    forecast_data = forecast_data.astype(int)

    if i <= 12:
        A_periods_amount.append(forecast_data)
    else:
        B_periods_amount.append(forecast_data)

A_result = A_periods_amount[0]
B_result = B_periods_amount[0]
final_result = []

for i in range(len(A_periods_amount) - 1):
    A_result = np.array(A_periods_amount[i]) + np.array(A_result)
```

```
for i in range(62):
    if i % 2 == 0:
        #加入权重预测
        final_result.append(A_result[i * 48 : i * 48 + 48] * 1.14)
    else:
        final_result.append(B_result[i * 48 : i * 48 + 48] )

a = final_result[0]
for i in range(61):
    a = np.hstack((a, final_result[i + 1]))

for i in range(len(a)):
    if a[i] < 0:
        a[i] = 0

for i in range(60):
    a[0 + i*48 : 17 + i*48] = 0
    a[38 + i*48 : 48 + i*48] = 0
a = a.astype(int)

result = test_data.loc[:,['day_of_week','post_id','periods']]
result.loc[:,'amount'] = pd.DataFrame(a).loc[:, 0]
b = result[((result['day_of_week'] == 6) | (result['day_of_week'] == 7)) &
(result['post_id'] == 'B')| (result['periods'] < 17) | (result['periods'] >=
37)].index
result.loc[b,'amount'] = 0
result = np.array(result)
```

4. 再处理

注意到B业务周末一定不开放，故将B业务的预测结果中，周末的值设为0，并将非工作时间业务量设为0

```
for i in range(60):
    a[0 + i*48 : 17 + i*48] = 0
    a[38 + i*48 : 48 + i*48] = 0
a = a.astype(int)

result = test_data.loc[:,['day_of_week','post_id','periods']]
result.loc[:,'amount'] = pd.DataFrame(a).loc[:, 0]
b = result[((result['day_of_week'] == 6) | (result['day_of_week'] == 7)) &
(result['post_id'] == 'B')| (result['periods'] < 17) | (result['periods'] >=
37)].index
result.loc[b,'amount'] = 0
result = np.array(result)
```