In [1]:

```
import numpy as np
import pandas as pd
import gc
from tqdm import tqdm_notebook as tqdm
import warnings
warnings.filterwarnings('ignore')
import lightgbm as lgb
import scipy.stats  as stats
import pickle
```

In [2]:

```
def final_pipeline_model(test_data):

    def data_list_fn():
        for j in range(1,5):
            for i in range(10):
                data_list.append(f'data_s{i}_w{j}')

    def model_list_fn():
        for j in range(1,5):
            for i in range(10):
                model_list.append(f'model_s{i}_w{j}')

    def ramu_data(data_list,data,data_parts):
        m = 0
        for j,k in zip(week_start,week_end):
            for i in range(10):
                data_parts[data_list[m]]=data[(data['store_id']==i) & ((data['tm_d']>=j)&(data['tm_
d']<=k))]
                m+=1


    def ramu_data2(data_list,data,data_parts):
        m = 10
        for i in range(10):
            data_parts[data_list[m]]=data[(data['store_id']==i) & ((data['tm_d']>=30) | (data['tm_d
']<=5))]
            m+=1


    def prediction(features_list,model_data_parts,model_list,data_list,test_data_parts):
        m=0
        for i in features_list:
            for k in range(10):
                model= model_data_parts.get(model_list[m])
                y_pred = model.predict(test_data_parts.get(data_list[m])[i], num_iteration=model.be
st_iteration)
                test_data_parts.get(data_list[m])[['unit_sales']] = y_pred
                m+=1

    def quantile_coefficients(q):
        return ratios.loc[q].values

    def individual_aggregation(df, level):
        df = df.groupby(level)[cols].sum()
        q = np.repeat(qs, len(df))
        df = pd.concat([df]*9, axis=0, sort=False)
        df.reset_index(inplace=True)
        df[cols]*= quantile_coefficients(q)[:, None]
        if level != 'id':
            df['id'] = [f"{l}_X_{q:.3f}_evaluation" for l,q in zip(df[level].values, q)]
        else:
            df['id'] = [f"{l.replace('_evaluation','')}_{q:.3f}_evaluation" for l,q in zip(df[level
].values,q)]
        df = df[['id']+list(cols)]
        return df

    def grouped_aggregation(df,level1,level2):
        df = df.groupby([level1,level2])[cols].sum()
        q = np.repeat(qs,len(df))
```

```
    df = pd.concat([df]*9,axis=0,sort = False)
    df.reset_index(inplace=True)
    df[cols] *= quantile_coefficients(q)[:, None]
    df['id']= [f'{l1}_{l2}_{q:.3f}_evaluation' for l1,l2,q in
               zip(df[level1].values,df[level2].values,q)]
    df = df[['id']+list(cols)]
    return df

# Loading saved models
location2 = open("../input/training-40-models/models.sav","rb")
model_data_parts= pickle.load(location2)


features_w1 = ['item_id', 'dept_id', 'cat_id', 'store_id', 'state_id', 'wm_yr_wk',
            'event_name_1', 'event_type_1','event_name_2', 'event_type_2','snap_CA', 'snap_TX',
            'snap_WI','sell_price', 'price_max', 'price_min', 'price_std', 'price_mean',
            'price_nunique', 'item_nunique', 'encoded_id', 'lag_d_7', 'lag_d_8','lag_d_9',
            'r_std_d7', 'r_std_d14','r_std_d30', 'r_mean_d7', 'r_mean_d14',
            'r_mean_d30', 'tm_d', 'tm_w','tm_m', 'tm_y', 'tm_wm', 'tm_dw', 'tm_w_end']

features_w2 = ['item_id', 'dept_id', 'cat_id', 'store_id', 'state_id', 'wm_yr_wk',
            'event_name_1', 'event_type_1','event_name_2', 'event_type_2','snap_CA', 'snap_TX',
            'snap_WI','sell_price', 'price_max', 'price_min', 'price_std', 'price_mean',
            'price_nunique', 'item_nunique', 'encoded_id','lag_d_14', 'lag_d_15', 'lag_d_16',
            'r_std_d7', 'r_std_d14','r_std_d30', 'r_mean_d7', 'r_mean_d14',
            'r_mean_d30', 'tm_d', 'tm_w','tm_m', 'tm_y', 'tm_wm', 'tm_dw', 'tm_w_end']

features_w3 = ['item_id', 'dept_id', 'cat_id', 'store_id', 'state_id', 'wm_yr_wk',
            'event_name_1', 'event_type_1','event_name_2', 'event_type_2','snap_CA', 'snap_TX',
            'snap_WI','sell_price', 'price_max', 'price_min', 'price_std', 'price_mean',
            'price_nunique', 'item_nunique', 'encoded_id','lag_d_21', 'lag_d_22','lag_d_23',
            'r_std_d7', 'r_std_d14','r_std_d30', 'r_mean_d7', 'r_mean_d14',
            'r_mean_d30', 'tm_d', 'tm_w','tm_m', 'tm_y', 'tm_wm', 'tm_dw', 'tm_w_end']

features_w4 = ['item_id', 'dept_id', 'cat_id', 'store_id', 'state_id', 'wm_yr_wk',
            'event_name_1', 'event_type_1','event_name_2', 'event_type_2','snap_CA', 'snap_TX',
            'snap_WI','sell_price', 'price_max', 'price_min', 'price_std', 'price_mean',
            'price_nunique', 'item_nunique', 'encoded_id', 'lag_d_28','lag_d_29', 'lag_d_30',
            'r_std_d7', 'r_std_d14','r_std_d30', 'r_mean_d7', 'r_mean_d14',
            'r_mean_d30', 'tm_d', 'tm_w','tm_m', 'tm_y', 'tm_wm', 'tm_dw', 'tm_w_end']
features_list = [features_w1,features_w2,features_w3,features_w4]



data_list=[]
data_list_fn()

model_list = []
model_list_fn()

# Creating the required data for testing from test_data

test_data_parts={}
week_start = [23,30,6,13]
week_end = [29,5,12,19]

ramu_data(data_list,test_data,test_data_parts)
ramu_data2(data_list,test_data,test_data_parts)


prediction(features_list,model_data_parts,model_list,data_list,test_data_parts)


# Creating point forecast submission file - FORECASTING ACCURACY
concat_list = []
for i in range(40):
    concat_list.append(test_data_parts.get(data_list[i]))

final_test_data =  pd.concat(concat_list,axis=0)
final_test_data = final_test_data[['id','unit_sales','date']].pivot(index='id', columns='date')
day_test_columns = [f'F{row}' for row in range(1,29)]
final_test_data.columns = day_test_columns
final_test_data.reset_index(inplace=True)

sales = pd.read_csv('../input/m5-forecasting-accuracy/sales_train_evaluation.csv')
id2 = ['id']
val_columns = [f'd_{row}' for row in range(1914,1942)]
```

```
    id2.extend(val_columns)
    final_validation_data = sales[id2]
    id1 = ['id']
    day_val_columns = [f'F{row}' for row in range(1,29)]
    id1.extend(day_val_columns)
    final_validation_data.columns = id1
    final_submission_accuracy_data = pd.concat([final_validation_data,final_test_data],axis=0)

    # Creating point to uncertainty submission file - FORECASTING UNCERTAINTY

    sales = pd.read_csv("../input/m5-forecasting-accuracy/sales_train_evaluation.csv")
    sub_val = final_validation_data.merge(sales[["id", "item_id", "dept_id", "cat_id", "store_id",
"state_id"]], on = "id")
    sub_val["_all_"] = "Total"
    sub_eval = final_test_data.merge(sales[["id", "item_id", "dept_id", "cat_id", "store_id", "stat
e_id"]], on = "id")
    sub_eval["_all_"] = "Total"

    qs = np.array([0.005,0.025,0.165,0.25,0.5,0.75,0.835,0.975,0.995])  # 9 quantile values

    qs2 = np.log(qs/(1-qs))*0.07
    ratios = stats.norm.cdf(qs2)

    ratios/=ratios[4]  # division based on the middle(center) value.
    ratios = pd.Series(ratios, index=qs)
    ratios.round(3)   # rounding off to 3 decimal values



    individual = ['id','item_id','dept_id','cat_id','store_id','state_id','_all_']
    grouped = [('state_id','item_id'),('state_id','dept_id'),('store_id','dept_id'),
               ('state_id','cat_id'),('store_id','cat_id')]
    cols = [f'F{i}' for i in range(1,29)]


    df_v = []
    for level in individual :
        df_v.append(individual_aggregation(sub_val, level))
    for level1,level2 in grouped:
        df_v.append(grouped_aggregation(sub_val, level1, level2))
    df_v = pd.concat(df_v, axis=0, sort=False)
    df_v.reset_index(drop=True, inplace=True)

    df_t = []
    for level in individual :
        df_t.append(individual_aggregation(sub_eval, level))
    for level1,level2 in grouped:
        df_t.append(grouped_aggregation(sub_eval, level1, level2))
    df_t = pd.concat(df_t, axis=0, sort=False)
    df_t.reset_index(drop=True, inplace=True)

    df = pd.concat([df_v,df_t] , axis=0, sort=False)
    df.reset_index(drop=True, inplace=True)
    df.loc[df.index < len(df.index)//2, "id"] = df.loc[df.index < len(df.index)//2,
"id"].str.replace("_evaluation","_validation")
    return df


location1 = open('/kaggle/input/testing-40-models/test_data.csv',"rb")
test_data= pickle.load(location1)
final_data = final_pipeline_model(test_data)
print('The final output of the test_data is')
final_data
```

```
The final output of the test_data is
```

Out[2]:

	id	F1	F2	F3	F4	F5	
0	FOODS_1_001_CA_1_0.005_validation	1.421973	0.000000	0.000000	0.000000	0.000000	0.710987
1	FOODS_1_001_CA_2_0.005_validation	0.000000	2.132960	0.000000	0.000000	0.000000	0.710987
2	FOODS_1_001_CA_3_0.005_validation	0.710987	0.000000	0.710987	0.000000	5.687893	0.710987

	id	F1	F2	F3	F4	F5	
3	FOODS_1_001_CA_4_0.005_validation	0.000000	0.710987	0.000000	0.000000	0.000000	0.000000
4	FOODS_1_001_TX_1_0.005_validation	0.000000	0.000000	0.710987	0.000000	0.710987	0.000000
...	...	...	...	...	...	...	...
771115	WI_2_HOBBIES_0.995_evaluation	337.146160	332.953129	352.448870	351.904768	381.002237	395.4848
771116	WI_2_HOUSEHOLD_0.995_evaluation	1191.187797	1216.836419	1303.325115	1345.803352	1580.246246	1689.476
771117	WI_3_FOODS_0.995_evaluation	2991.460493	2905.064714	3003.488416	2887.731353	3188.431701	3906.876
771118	WI_3_HOBBIES_0.995_evaluation	327.739648	320.267303	353.105532	333.209548	357.736747	423.9969
771119	WI_3_HOUSEHOLD_0.995_evaluation	929.067587	919.164293	926.483778	905.448392	1042.049093	1299.959

771120 rows × 29 columns