Importing Libraries

In [1]:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_log_error
from statsmodels.tsa.deterministic import CalendarFourier, DeterministicProcess
from sklearn.preprocessing import MinMaxScaler
from tqdm import tqdm
import matplotlib.pyplot as plt
from collections import defaultdict
from xgboost import XGBRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
```

Loading Dataset

In [2]:

```
holidays_events = pd.read_csv("/kaggle/input/store-sales-time-series-forecasting/holidays_events.csv", parse_dates=['date'])
oil = pd.read_csv("/kaggle/input/store-sales-time-series-forecasting/oil.csv", parse_dates=['date'])
stores = pd.read_csv("/kaggle/input/store-sales-time-series-forecasting/stores.csv")
transactions = pd.read_csv("/kaggle/input/store-sales-time-series-forecasting/transactions.csv", parse_dates=['date'])

test = pd.read_csv("/kaggle/input/store-sales-time-series-forecasting/test.csv", parse_dates=['date'])
train = pd.read_csv("/kaggle/input/store-sales-time-series-forecasting/train.csv", parse_dates=['date'])
```

Preparing Dataset

In [3]:

```
families = train["family"].unique()
print(f"Unique families: {families}")
```

```
Unique families: ['AUTOMOTIVE' 'BABY CARE' 'BEAUTY' 'BEVERAGES' 'BOOKS' 'BREAD/BAKERY'
 'CELEBRATION' 'CLEANING' 'DAIRY' 'DELI' 'EGGS' 'FROZEN FOODS' 'GROCERY I'
 'GROCERY II' 'HARDWARE' 'HOME AND KITCHEN I' 'HOME AND KITCHEN II'
 'HOME APPLIANCES' 'HOME CARE' 'LADIESWEAR' 'LAWN AND GARDEN' 'LINGERIE'
 'LIQUOR,WINE,BEER' 'MAGAZINES' 'MEATS' 'PERSONAL CARE' 'PET SUPPLIES'
 'PLAYERS AND ELECTRONICS' 'POULTRY' 'PREPARED FOODS' 'PRODUCE'
 'SCHOOL AND OFFICE SUPPLIES' 'SEAFOOD']
```

In [4]:

```
def get_time_features():
    calendar = pd.DataFrame(index=pd.date_range('2013-01-01', '2017-08-31'))
    calendar['year'] = calendar.index.year.astype('int')
    calendar['quarter'] = calendar.index.quarter.astype('int')
    calendar['month'] = calendar.index.month.astype('int')
    calendar['day'] = calendar.index.day.astype('int')
    calendar['dayofweek'] = calendar.index.day_of_week.astype('int')
    calendar['weekofyear'] = calendar.index.week.astype('int')
    calendar['isweekend'] = calendar.dayofweek.apply(lambda x: 1 if x in (5,6) else 0)
    calendar['startschool'] = calendar.month.apply(lambda x: 1 if x in (4,5,8,9) else 0)

    calendar['daysinmonth'] = calendar.index.days_in_month.astype('int')

    calendar.index.rename("date", inplace=True)
    calendar = pd.get_dummies(calendar, columns=['year'], drop_first=True)
    calendar = pd.get_dummies(calendar, columns=['quarter'], drop_first=True)
    calendar = pd.get_dummies(calendar, columns=['dayofweek'], drop_first=True)

    fourierA = CalendarFourier(freq='A', order=5)
    fourierM = CalendarFourier(freq='M', order=2)
    fourierW = CalendarFourier(freq='W', order=4)

    dp = DeterministicProcess(index=calendar.index,
                              order=1,
                              seasonal=True,
                              constant=False,
                              additional_terms=[fourierA, fourierM, fourierW],
                              drop=True)
    dp_df = dp.in_sample()
    calendar = pd.concat([calendar, dp_df], axis=1)
    return calendar
```

In [5]:

```
def get_oil_features(calendar):
    oil_df = oil.copy()
    oil_df = pd.merge(calendar.reset_index(), oil_df, left_on='date', how='left', right_on='date')
    oil_df.fillna(method='bfill', inplace=True)

    moving_average_periods = [7, 14, 30, 120, 180, 365]
    for mv in moving_average_periods:
        oil_df[f'mavg_oil_{mv}'] = oil_df['dcoilwtico'].rolling(mv).mean()

    for i in [1, 2, 3, 4, 5, 6, 7, 10, 14, 21, 30, 60, 90]:
        oil_df['lagoil_' + str(i) ] = oil_df['dcoilwtico'].shift(i)
    oil_df.dropna(inplace=True)
    oil_df["date"] = oil_df["date"].dt.to_period('D')
    oil_df = oil_df.set_index("date")
    return oil_df
```

In [9]:

```
def get_family_sales(dataframe):
    df = dataframe.copy()
    df['date'] = df.date.dt.to_period('D')
    df = df.set_index(['store_nbr', 'family', 'date']).sort_index()
    return df.groupby(['family', 'date', 'store_nbr']).mean().drop(columns=["id", "onpromotion"])
```

In [10]:

```
calendar = get_time_features()
calendar
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8: FutureWarning: weekofyear and week have been deprec
ated, please use DatetimeIndex.isocalendar().week instead, which returns a Series.  To exactly reproduce the behavi
or of week and weekofyear and return an Index, you may call pd.Int64Index(idx.isocalendar().week)
```

Out[10]:

date	month	day	weekofyear	isweekend	startschool	daysinmonth	year_2014	year_2015	year_2016	year_2017	...	sin(1,freq=M)	cos(1,freq=
2013-01-01	1	1	1	0	0	31	0	0	0	0	...	0.000000	1.000(
2013-01-02	1	2	1	0	0	31	0	0	0	0	...	0.201299	0.979
2013-01-03	1	3	1	0	0	31	0	0	0	0	...	0.394356	0.918
2013-01-04	1	4	1	0	0	31	0	0	0	0	...	0.571268	0.820
2013-01-05	1	5	1	1	0	31	0	0	0	0	...	0.724793	0.688
...	...	...	...	...	...	...	...	...	...	...	...	...	
2017-08-27	8	27	34	1	1	31	0	0	0	1	...	-0.848644	0.528
2017-08-28	8	28	35	0	1	31	0	0	0	1	...	-0.724793	0.688
2017-08-29	8	29	35	0	1	31	0	0	0	1	...	-0.571268	0.820
2017-08-30	8	30	35	0	1	31	0	0	0	1	...	-0.394356	0.918
2017-08-31	8	31	35	0	1	31	0	0	0	1	...	-0.201299	0.979

1704 rows × 41 columns

In [11]:

```
oil_df = get_oil_features(calendar)
oil_df
```

Out[11]:

date	month	day	weekofyear	isweekend	startschool	daysinmonth	year_2014	year_2015	year_2016	year_2017	...	lagoil_4	lagoil_5	lagoil_
2013-12-31	12	31	1	0	0	31	0	0	0	0	...	99.94	99.18	99.1
2014-01-01	1	1	1	0	0	31	1	0	0	0	...	98.90	99.94	99.1
2014-01-02	1	2	1	0	0	31	1	0	0	0	...	98.90	98.90	99.9
2014-01-03	1	3	1	0	0	31	1	0	0	0	...	98.90	98.90	98.9
2014-01-04	1	4	1	1	0	31	1	0	0	0	...	98.17	98.90	98.9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2017-08-27	8	27	34	1	1	31	0	0	0	1	...	48.45	47.65	47.3
2017-08-28	8	28	35	0	1	31	0	0	0	1	...	47.24	48.45	47.6
2017-08-29	8	29	35	0	1	31	0	0	0	1	...	47.65	47.24	48.4
2017-08-30	8	30	35	0	1	31	0	0	0	1	...	46.40	47.65	47.2
2017-08-31	8	31	35	0	1	31	0	0	0	1	...	46.40	46.40	47.6

1340 rows × 61 columns

In [12]:

```
family_sales = get_family_sales(train)
family_sales
```

Out[12]:

			sales
family	date	store_nbr	
		1	0.000
		2	0.000
AUTOMOTIVE	2013-01-01	3	0.000
		4	0.000
		5	0.000
...	...	...	...
		50	15.314
		51	52.876
SEAFOOD	2017-08-15	52	7.000
		53	5.000
		54	3.000

3000888 rows × 1 columns

In [13]:

```
merged_df = family_sales.reset_index().merge(oil_df.reset_index(), left_on='date', right_on='date')
merged_df = merged_df.set_index("family").sort_index()
merged_df
```

Out[13]:

	date	store_nbr	sales	month	day	weekofyear	isweekend	startschool	daysinmonth	year_2014	...	lagoil_4	lagoil_5	lagoil_6
family														
AUTOMOTIVE	2013-12-31	1	2.0	12	31	1	0	0	31	0	...	99.94	99.18	99.18
AUTOMOTIVE	2017-06-13	15	5.0	6	13	24	0	0	30	0	...	45.82	45.68	45.80
AUTOMOTIVE	2017-06-13	16	0.0	6	13	24	0	0	30	0	...	45.82	45.68	45.80
AUTOMOTIVE	2017-06-13	17	5.0	6	13	24	0	0	30	0	...	45.82	45.68	45.80
AUTOMOTIVE	2017-06-13	18	2.0	6	13	24	0	0	30	0	...	45.82	45.68	45.80
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
SEAFOOD	2015-03-11	15	2.0	3	11	11	0	0	31	0	...	49.95	49.61	50.76
SEAFOOD	2015-03-11	14	1.0	3	11	11	0	0	31	0	...	49.95	49.61	50.76
SEAFOOD	2015-03-11	13	0.0	3	11	11	0	0	31	0	...	49.95	49.61	50.76
SEAFOOD	2015-03-11	21	0.0	3	11	11	0	0	31	0	...	49.95	49.61	50.76
SEAFOOD	2017-08-15	54	3.0	8	15	33	0	1	31	0	...	48.81	48.54	49.59

2354022 rows × 64 columns

In [14]:

```
test_df = test.copy()
test_df["date"] = test_df["date"].dt.to_period('D')
test_df = test_df.drop(columns=["id", "onpromotion"])
test_df = test_df.drop_duplicates()
test_df = test_df.merge(oil_df.reset_index(), left_on='date', right_on='date').set_index(["family", "date", "store_nbr"]).sort_in
test_df
```

Out[14]:

			month	day	weekofyear	isweekend	startschool	daysinmonth	year_2014	year_2015	year_2016	year_2017	...
family	date	store_nbr											
AUTOMOTIVE	2017-08-16	1	8	16	33	0	1	31	0	0	0	1	...
		2	8	16	33	0	1	31	0	0	0	1	...
		3	8	16	33	0	1	31	0	0	0	1	...
		4	8	16	33	0	1	31	0	0	0	1	...
		5	8	16	33	0	1	31	0	0	0	1	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...
SEAFOOD	2017-08-31	50	8	31	35	0	1	31	0	0	0	1	...
		51	8	31	35	0	1	31	0	0	0	1	...
		52	8	31	35	0	1	31	0	0	0	1	...
		53	8	31	35	0	1	31	0	0	0	1	...
		54	8	31	35	0	1	31	0	0	0	1	...

28512 rows × 61 columns

Modeling

In [15]:

```
RF_param = {
    'criterion': 'squared_error',
    'bootstrap': 'False',
    'max_depth': 9733,
    'max_features': 'auto',
    'max_leaf_nodes': 4730,
    'n_estimators': 159,
    'min_samples_split': 3,
    'min_samples_leaf': 8
}
default_models = {
    "linear_reg": LinearRegression(),
    "random_forest": RandomForestRegressor(**RF_param, random_state=0),
    "xgb": XGBRegressor(n_estimators=500)
}

def get_model(name):
    return default_models.get(name, default_models.get("random_forest"))
```

In [16]:

```
def train_model(X, y, model):
    X_train, X_val = train_test_split(X, test_size=0.1, shuffle=False)
    y_train, y_val = train_test_split(y, test_size=0.1, shuffle=False)

    model.fit(X_train, y_train)
    pred_train = model.predict(X_train).clip(0.0)
    pred_val = model.predict(X_val).clip(0.0)
    train_perf = mean_squared_log_error(y_train, pred_train)
    val_perf = mean_squared_log_error(y_val, pred_val)
    print(f"train_perf: {train_perf}; val_perf: {val_perf}")

    model.fit(X, y)
    return model


def get_prediction(X, model):
    preds = pd.DataFrame()
    pred = model.predict(X).clip(0.0)
    return np.exp(pred) - 1
```

In [17]:

```
import warnings; warnings.simplefilter('ignore')
```

In [ ]:

```
all_preds = {}
for family in tqdm(families):
    X = merged_df.loc[family].reset_index().drop(columns=['family', 'date'])
    X_test = test_df.loc[family].reset_index().drop(columns=['date'])
    y = np.log1p(X['sales'])
    X = X.drop(columns=['sales'])
    model = get_model("random_forest")
    train_model(X, y, model)
    all_preds[family] = get_prediction(X_test, model)
```

```
  0%|          | 0/33 [00:00<?, ?it/s]
train_perf: 0.036054773815855146; val_perf: 0.05823386194430574
  3%|▌          | 1/33 [08:05<4:18:55, 485.47s/it]
train_perf: 0.013086359329415546; val_perf: 0.020973826730213255
  6%|▌          | 2/33 [12:36<3:05:30, 359.06s/it]
train_perf: 0.0411909856744075; val_perf: 0.0696792583866669
  9%|▉          | 3/33 [20:41<3:28:29, 416.99s/it]
train_perf: 0.004376033205534012; val_perf: 0.005385184741154299
 12%|▉          | 4/33 [29:04<3:37:53, 450.81s/it]
train_perf: 0.004662805135679264; val_perf: 0.008309525964024274
 15%|▉         | 5/33 [30:29<2:28:46, 318.81s/it]
train_perf: 0.0039370425774740625; val_perf: 0.017270000582207422
 18%|▊         | 6/33 [39:42<2:59:14, 398.33s/it]
train_perf: 0.02331632798954872; val_perf: 0.03676244054566049
 21%|▊         | 7/33 [45:42<2:47:13, 385.92s/it]
train_perf: 0.002987693486861668; val_perf: 0.00987409292441045
 24%|▉         | 8/33 [54:40<3:01:00, 434.40s/it]
train_perf: 0.003816524558856149; val_perf: 0.00837646397371838
 27%|▊        | 9/33 [1:03:17<3:04:00, 460.03s/it]
train_perf: 0.0025678151755477393; val_perf: 0.0019602409492650943
 30%|▉        | 10/33 [1:12:19<3:06:06, 485.51s/it]
train_perf: 0.0044699083566319235; val_perf: 0.0046678746424454725
```

In [16]:

```
test_pred = test.copy()
test_pred = test_pred.set_index(['store_nbr', 'family'])

test_pred
```

Out[16]:

		id	date	onpromotion
store_nbr	**family**			
	AUTOMOTIVE	3000888	2017-08-16	0
	BABY CARE	3000889	2017-08-16	0
1	**BEAUTY**	3000890	2017-08-16	2
	BEVERAGES	3000891	2017-08-16	20
	BOOKS	3000892	2017-08-16	0
...	**...**	...	...	...
	POULTRY	3029395	2017-08-31	1
	PREPARED FOODS	3029396	2017-08-31	0
9	**PRODUCE**	3029397	2017-08-31	1
	SCHOOL AND OFFICE SUPPLIES	3029398	2017-08-31	9
	SEAFOOD	3029399	2017-08-31	0

28512 rows × 3 columns

In [17]:

```
predictions = pd.DataFrame()

for index in tqdm(test_pred.index.unique()):
    df = test_pred.loc[index].reset_index().drop(columns=['date','onpromotion', 'family', 'store_nbr'])
    df = df.set_index('id')
    df['sales'] = all_preds[index[1]]
    predictions = pd.concat([predictions, df], axis=0)

predictions = predictions.sort_index()
predictions
```

```
100%|██████████| 1782/1782 [00:05<00:00, 310.89it/s]
```

Out[17]:

	sales
id	
3000888	6.099217
3000889	0.155762
3000890	5.712249
3000891	3095.881590
3000892	0.017292
...	...
3029395	273.785656
3029396	81.532507
3029397	1660.206415
3029398	39.290185
3029399	15.877486

28512 rows × 1 columns

Submission

In [18]:

```
predictions.reset_index().to_csv('/kaggle/working/rf_per_fam_avg.csv', index=False)
```