

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 [6]:

```
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']).mean().drop(columns=["id", "onpromotion"])
```

In [7]:

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

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

Out[7]:

	month	day	weekofyear	isweekend	startschool	daysinmonth	year_2014	year_2015	year_2016	year_2017	...	sin(1,freq=M)	cos(1,freq=M)
date													
2013-01-01	1	1	1	0	0	31	0	0	0	0	...	0.000000	1.000000
2013-01-02	1	2	1	0	0	31	0	0	0	0	...	0.201299	0.979949
2013-01-03	1	3	1	0	0	31	0	0	0	0	...	0.394356	0.918351
2013-01-04	1	4	1	0	0	31	0	0	0	0	...	0.571268	0.820796
2013-01-05	1	5	1	1	0	31	0	0	0	0	...	0.724793	0.688143
...
2017-08-27	8	27	34	1	1	31	0	0	0	1	...	-0.848644	0.528344
2017-08-28	8	28	35	0	1	31	0	0	0	1	...	-0.724793	0.688143
2017-08-29	8	29	35	0	1	31	0	0	0	1	...	-0.571268	0.820796
2017-08-30	8	30	35	0	1	31	0	0	0	1	...	-0.394356	0.918351
2017-08-31	8	31	35	0	1	31	0	0	0	1	...	-0.201299	0.979949

1704 rows × 41 columns

In [8]:

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

Out[8]:

	month	day	weekofyear	isweekend	startschool	daysinmonth	year_2014	year_2015	year_2016	year_2017	...	lagoil_4	lagoil_5	lagoil_6
date														
2013-12-31	12	31	1	0	0	31	0	0	0	0	...	99.94	99.18	99.18
2014-01-01	1	1	1	0	0	31	1	0	0	0	...	98.90	99.94	99.18
2014-01-02	1	2	1	0	0	31	1	0	0	0	...	98.90	98.90	99.18
2014-01-03	1	3	1	0	0	31	1	0	0	0	...	98.90	98.90	98.90
2014-01-04	1	4	1	1	0	31	1	0	0	0	...	98.17	98.90	98.90
...
2017-08-27	8	27	34	1	1	31	0	0	0	1	...	48.45	47.65	47.65
2017-08-28	8	28	35	0	1	31	0	0	0	1	...	47.24	48.45	47.65
2017-08-29	8	29	35	0	1	31	0	0	0	1	...	47.65	47.24	48.45
2017-08-30	8	30	35	0	1	31	0	0	0	1	...	46.40	47.65	47.65
2017-08-31	8	31	35	0	1	31	0	0	0	1	...	46.40	46.40	47.65

1340 rows × 61 columns

In [9]:

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

Out[9]:

sales		
family	date	
AUTOMOTIVE	2013-01-01	0.000000
	2013-01-02	4.722222
	2013-01-03	2.981481
	2013-01-04	3.129630
	2013-01-05	6.333333
...
SEAFOOD	2017-08-11	23.566963
	2017-08-12	19.037593
	2017-08-13	20.704574
	2017-08-14	17.975556
	2017-08-15	17.966241

55572 rows × 1 columns

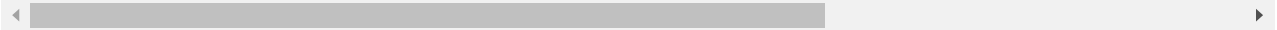
In [10]:

```
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[10]:

	date	sales	month	day	weekofyear	isweekend	startschool	daysinmonth	year_2014	year_2015	...	lagoil_4	lagoil_5	laç
family														
AUTOMOTIVE	2013-12-31	4.425926	12	31	1	0	0	31	0	0	...	99.94	99.18	
AUTOMOTIVE	2014-06-17	5.055556	6	17	25	0	0	30	1	0	...	107.49	107.20	1
AUTOMOTIVE	2016-05-11	5.833333	5	11	19	0	1	31	0	0	...	43.45	44.58	
AUTOMOTIVE	2015-04-06	5.314815	4	6	15	0	1	30	0	1	...	49.13	50.12	
AUTOMOTIVE	2016-05-12	5.055556	5	12	19	0	1	31	0	0	...	43.45	43.45	
...	
SEAFOOD	2017-04-25	14.884852	4	25	17	0	1	30	0	0	...	49.64	50.26	
SEAFOOD	2014-04-21	18.443093	4	21	17	0	1	30	1	0	...	104.33	103.71	1
SEAFOOD	2016-10-13	15.217407	10	13	41	0	0	31	0	0	...	49.76	49.76	
SEAFOOD	2015-01-07	24.158204	1	7	2	0	0	31	0	1	...	50.05	52.72	
SEAFOOD	2017-08-15	17.966241	8	15	33	0	1	31	0	0	...	48.81	48.54	

43593 rows × 63 columns



In [11]:

```
test_df = test.copy()
test_df["date"] = test_df["date"].dt.to_period('D')
test_df = test_df.drop(columns=["id", "store_nbr", "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"]).sort_index()
test_df
```

Out[11]:

		month	day	weekofyear	isweekend	startschool	daysinmonth	year_2014	year_2015	year_2016	year_2017	...	lagoil_4
family	date												
AUTOMOTIVE	2017-08-16	8	16	33	0	1	31	0	0	0	1	...	47.59
	2017-08-17	8	17	33	0	1	31	0	0	0	1	...	47.59
	2017-08-18	8	18	33	0	1	31	0	0	0	1	...	47.59
	2017-08-19	8	19	33	1	1	31	0	0	0	1	...	47.57
	2017-08-20	8	20	33	1	1	31	0	0	0	1	...	46.80
...
SEAFOOD	2017-08-27	8	27	34	1	1	31	0	0	0	1	...	48.45
	2017-08-28	8	28	35	0	1	31	0	0	0	1	...	47.24
	2017-08-29	8	29	35	0	1	31	0	0	0	1	...	47.65
	2017-08-30	8	30	35	0	1	31	0	0	0	1	...	46.40
	2017-08-31	8	31	35	0	1	31	0	0	0	1	...	46.40

528 rows × 61 columns

Modeling

In [12]:

```
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 [13]:

```
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 [14]:

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

In [15]:

```
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: 0.002836370285414381; val_perf: 0.01234325117701703
3%|█         | 1/33 [00:05<03:11, 5.99s/it]
train_perf: 0.001565459672415841; val_perf: 0.0031449816419343046
6%|██        | 2/33 [00:11<02:59, 5.79s/it]
train_perf: 0.003427353362530399; val_perf: 0.0027759407130513655
9%|███       | 3/33 [00:17<02:51, 5.71s/it]
train_perf: 0.000661105160317459; val_perf: 0.0029469596132325284
12%|████      | 4/33 [00:23<02:46, 5.74s/it]
train_perf: 0.000489830952152268; val_perf: 0.0006082945449292241
15%|█████     | 5/33 [00:25<02:06, 4.53s/it]
```

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

28512 rows × 5 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 × 2 columns

Submission

In [18]:

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