

# Store Sales - Time Series Forecasting

---

team2

고나경, 김윤진, 김은호, 전소연, 정소영, 최윤서

kaggle 데이터 분석

# Store Sales

## - Time Series Forecasting

매장에서 판매되는 제품의 매출 예측

**Additional Notes**  
public 부분의 임금은 2주마다 15일과 매월 말일에 지급된다.  
2016년 4월 16일에 규모 7.8의 지진이 에콰도르를 강타했다  
. -구호 활동으로 물, 생필품 거래량 급증

결측치 존재 oil.csv 유가, transactions.csv 거래량

### train.CSV

- id                    거래 번호
- date                판매 날짜
- store\_nbr          판매 상점
- Family            제품 유형
- **sales**            **매출**
- onpromotion      판촉 제품 수

### test.CSV

train data에서 sales 제외 동일하게 포함  
-> sales 예측

- train : 2013-01-01 ~ 2017-08-15
- test : 2017-08-15 ~ 2017-08-31

### stores.CSV

- **store\_nbr**            **상점**
- city                도시
- state              주
- type                가게 유형
- cluster            유사 그룹

### oil.csv

- date                날짜
- **dcoilwtico**          **유가**

### transaction.CSV

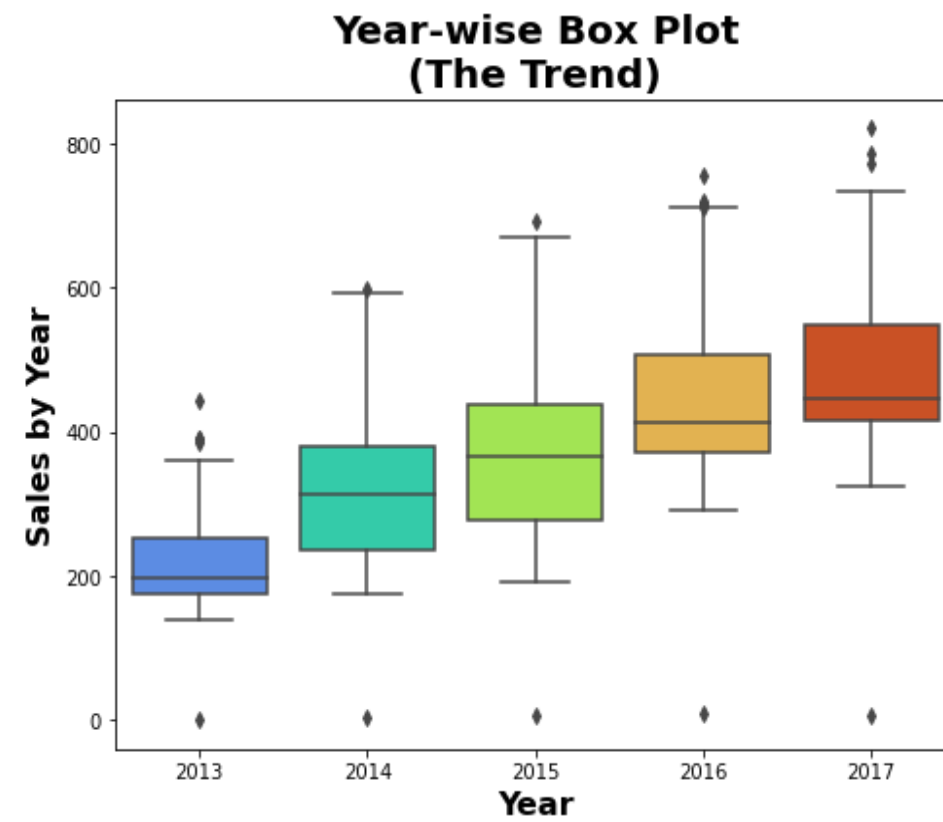
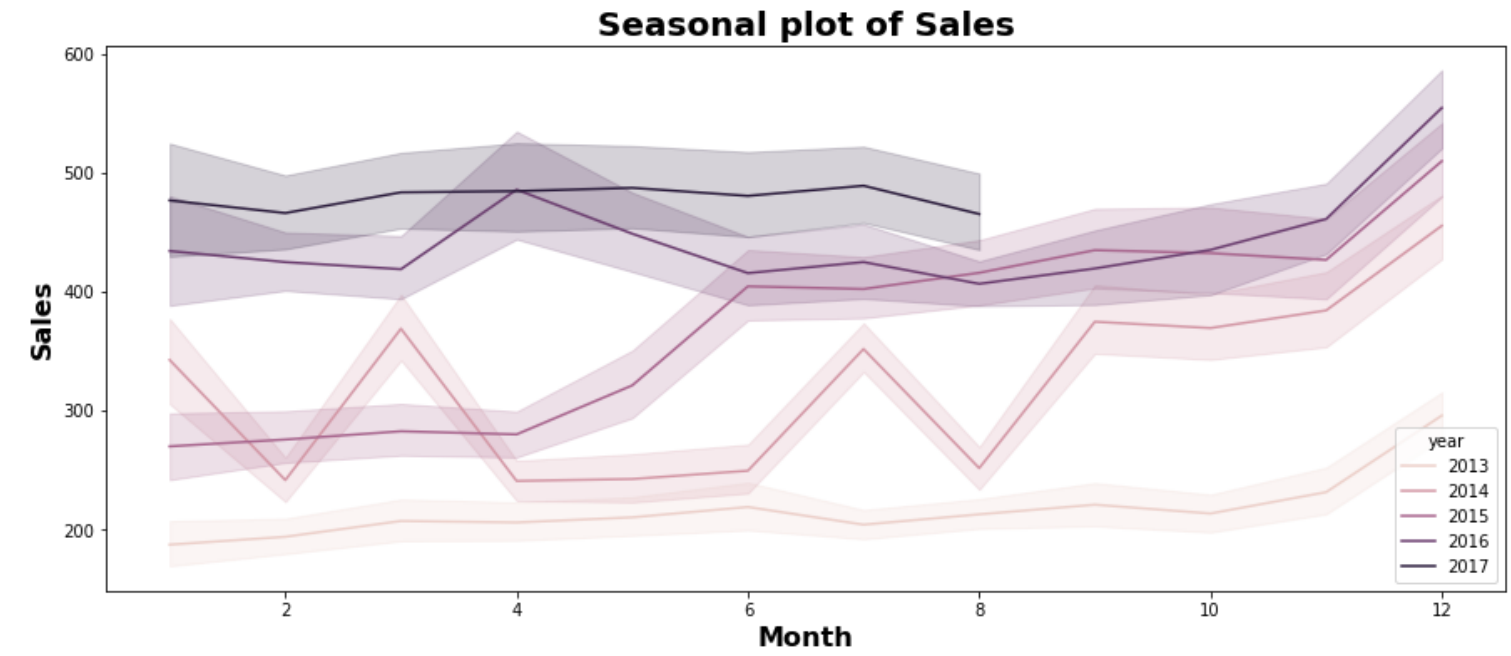
- Date                날짜
- Store\_nbr          상점
- **Transaction**          **일별 거래량**

### holiday events.CSV

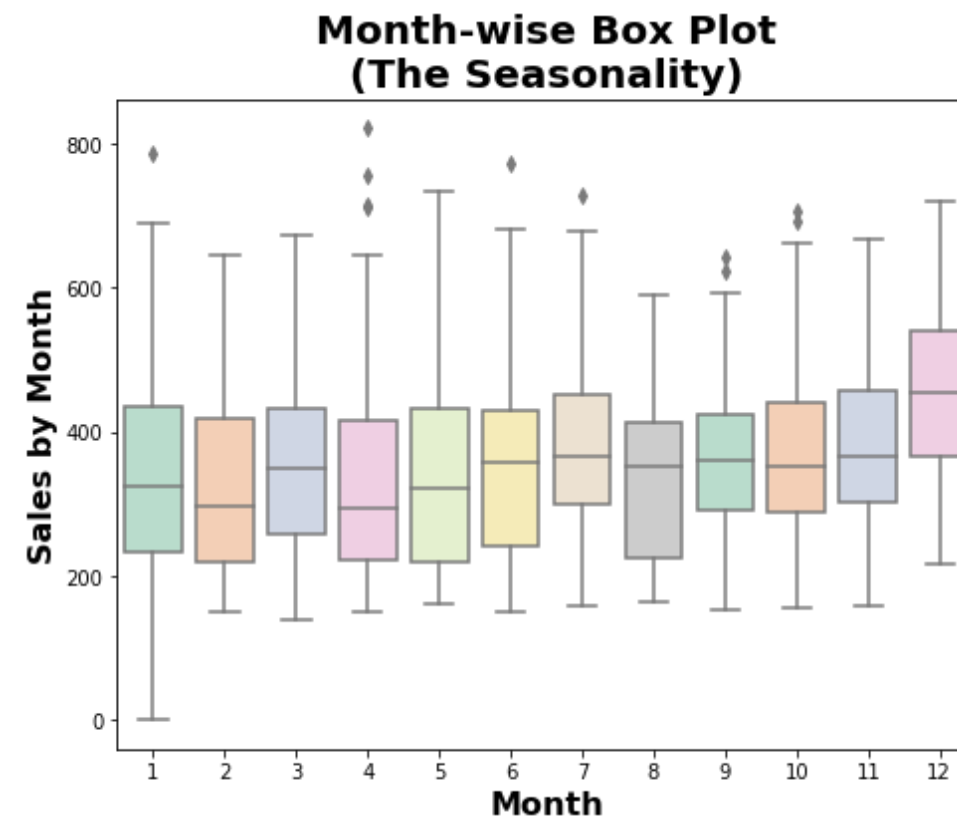
- Date                날짜
- **Type**                **휴일 종류**
- **Local**                **지역/전국 휴일 여부**
- Locale\_name        지역명
- Description        휴일/이벤트명
- Transferred        True(평일),False

EDA

## Sales의 시계열 요소



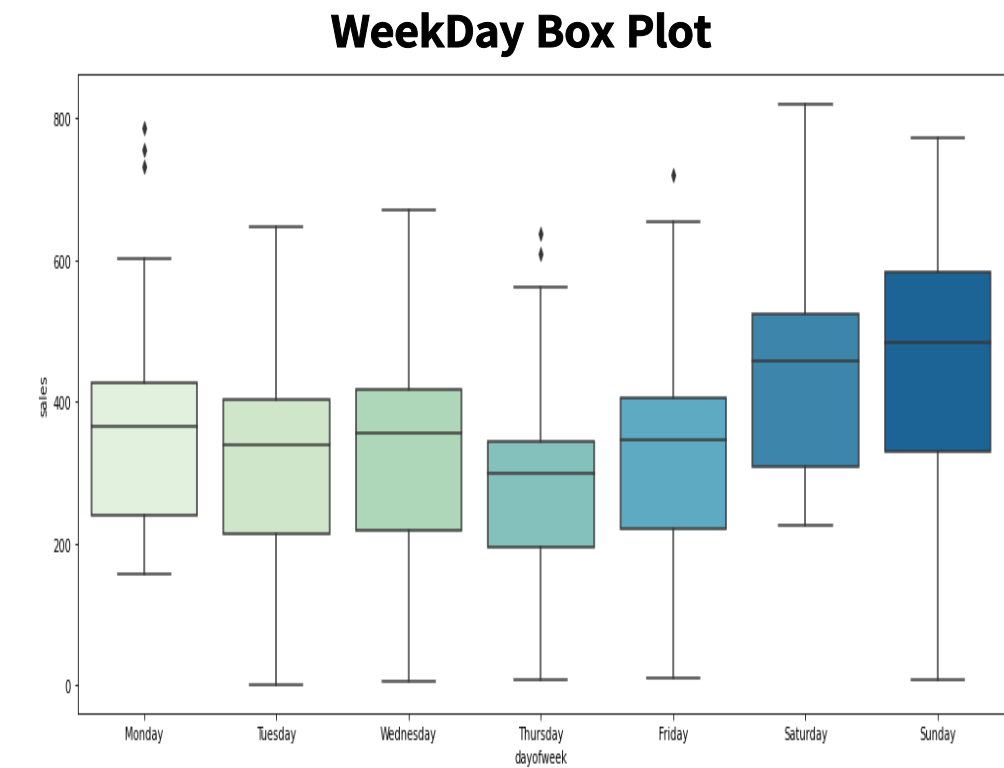
- 해마다 증가하는 추세



- 뚜렷한 차이는 보이지 않으나

몇몇 달에서 약간의 차이가 존재함

- 12월에 가장 높은 매출



- 요일마다 약간의 차이가 존재

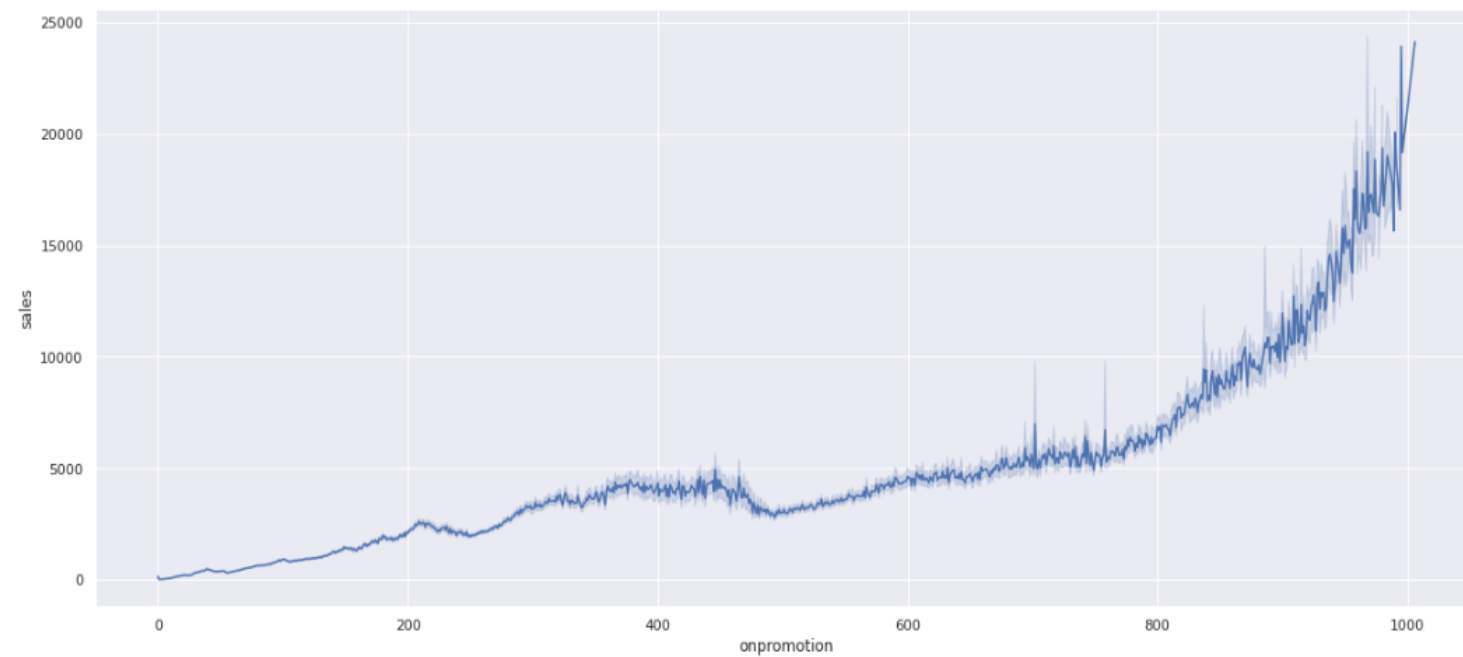
- 주말에 매출 증가

(일요일에 가장 높은 매출)

# EDA

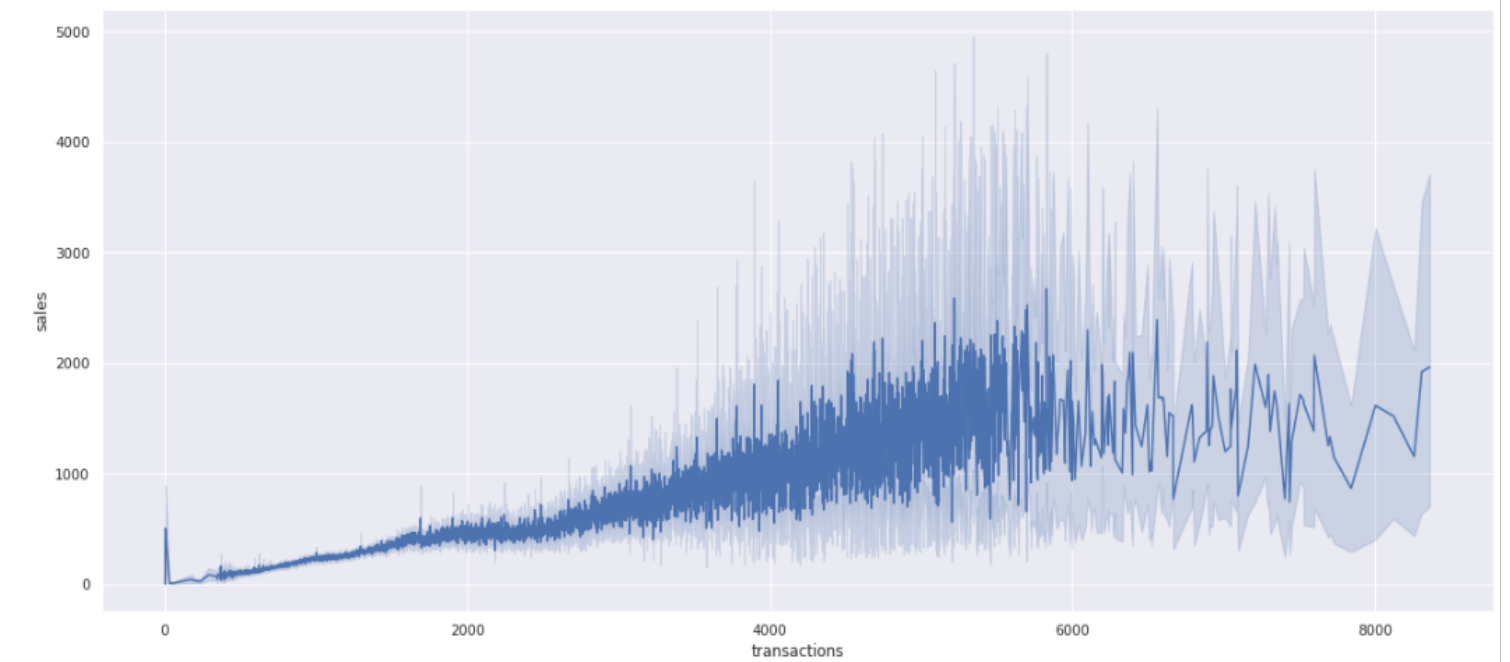
## 연속형 변수

### Average Sales by Onpromotion



- onpromotion이 증가함에 따라 평균 sales도 증가하는 양상
- 600~1000 구간에서 평균 sales가 급격하게 증가

### Average Sales by Transactions

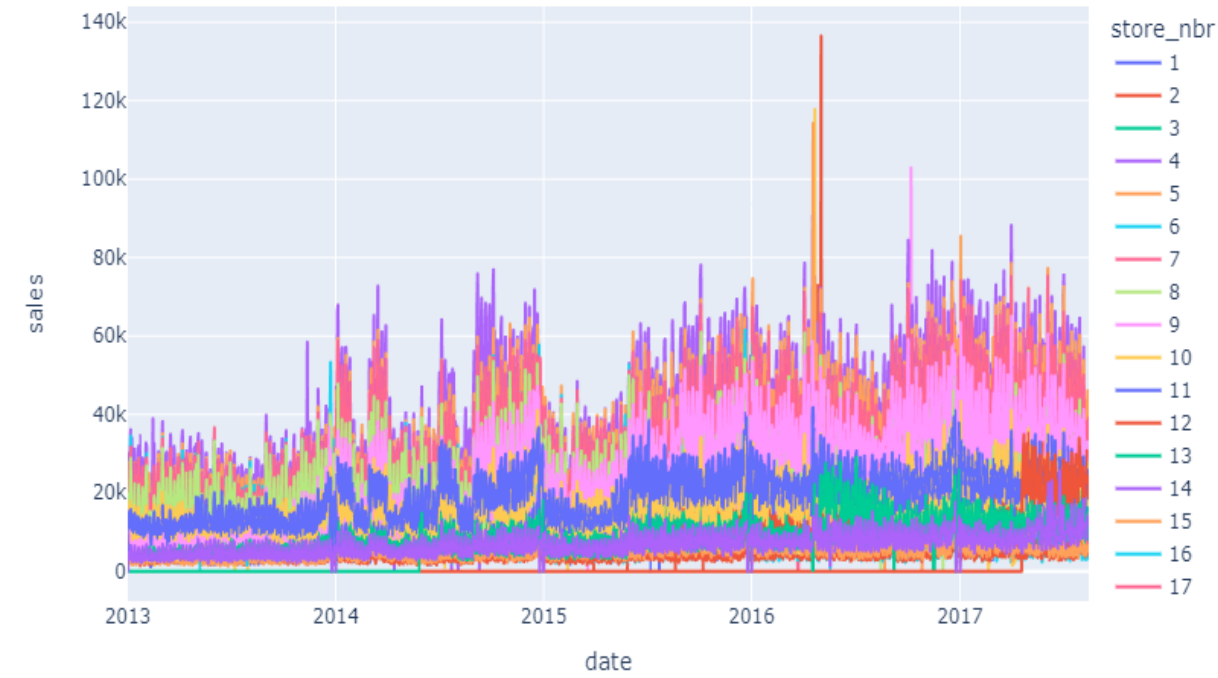


- transactions가 증가함에 따라 sales 평균값과 분산 증가
- transactions가 6000 이상인 구간에서는 레코드 수가 적음

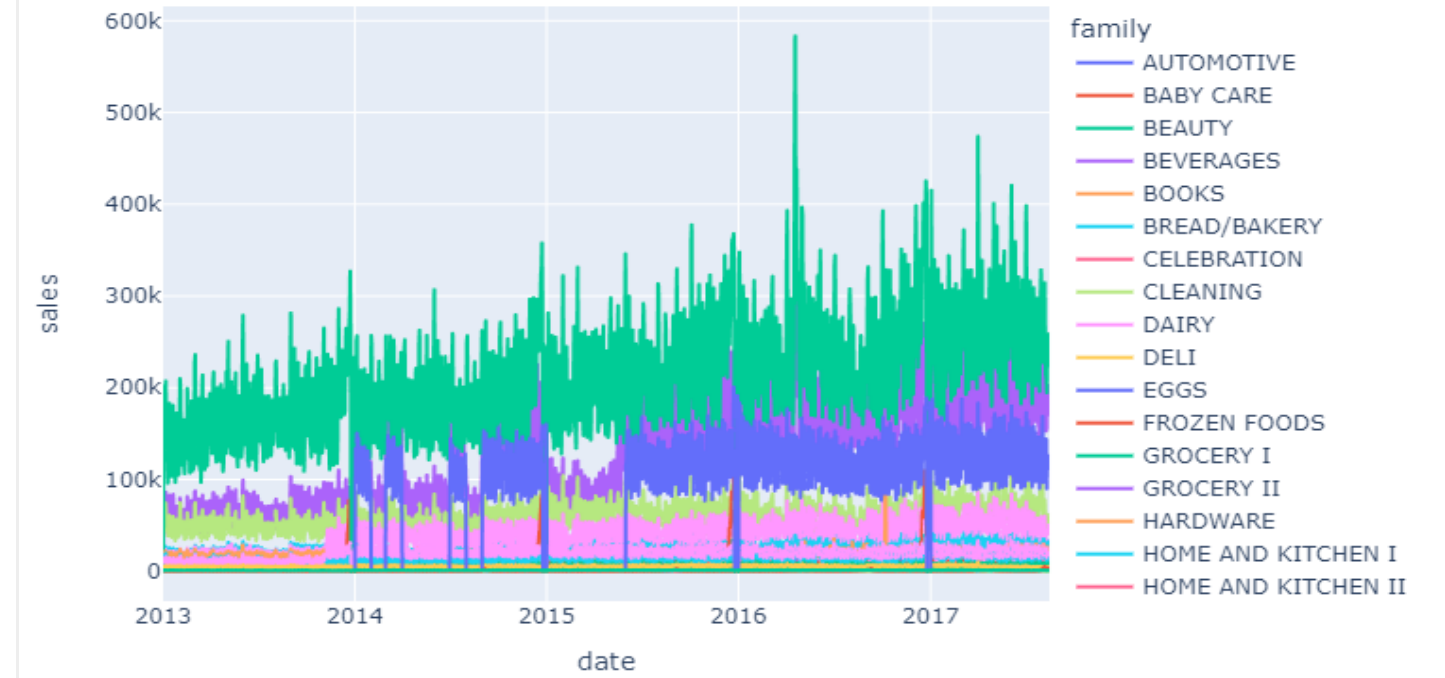
# EDA

## 범주형 변수

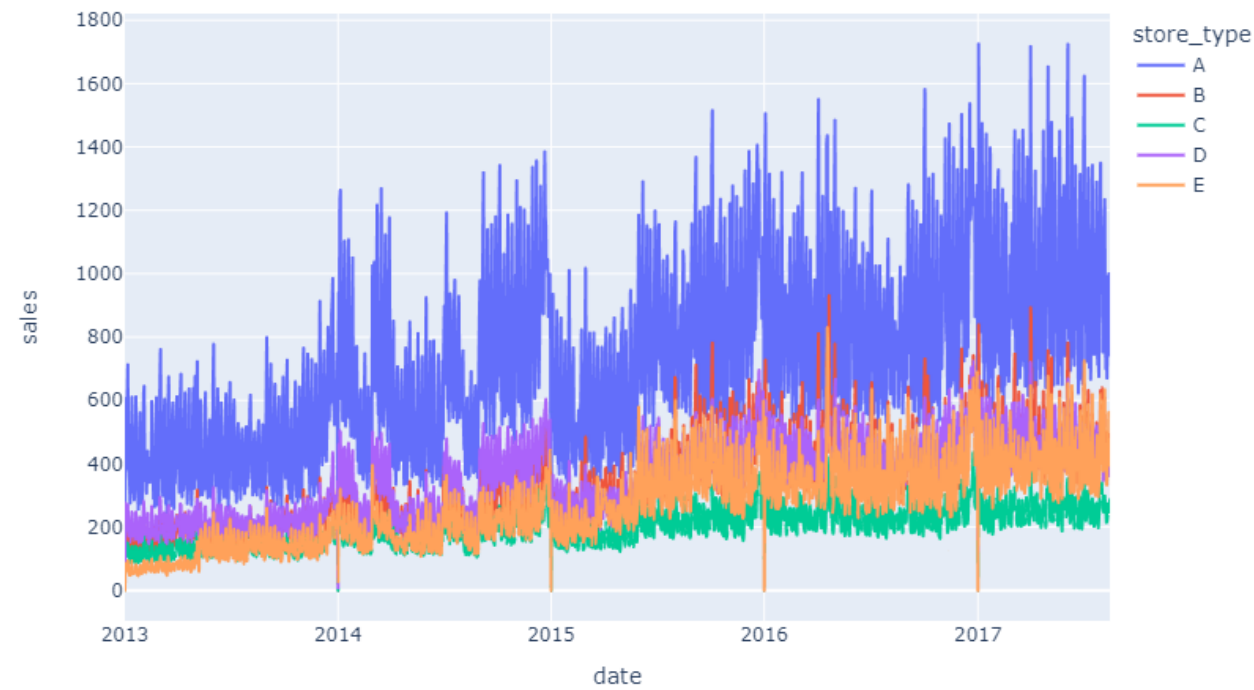
Daily total sales of the stores



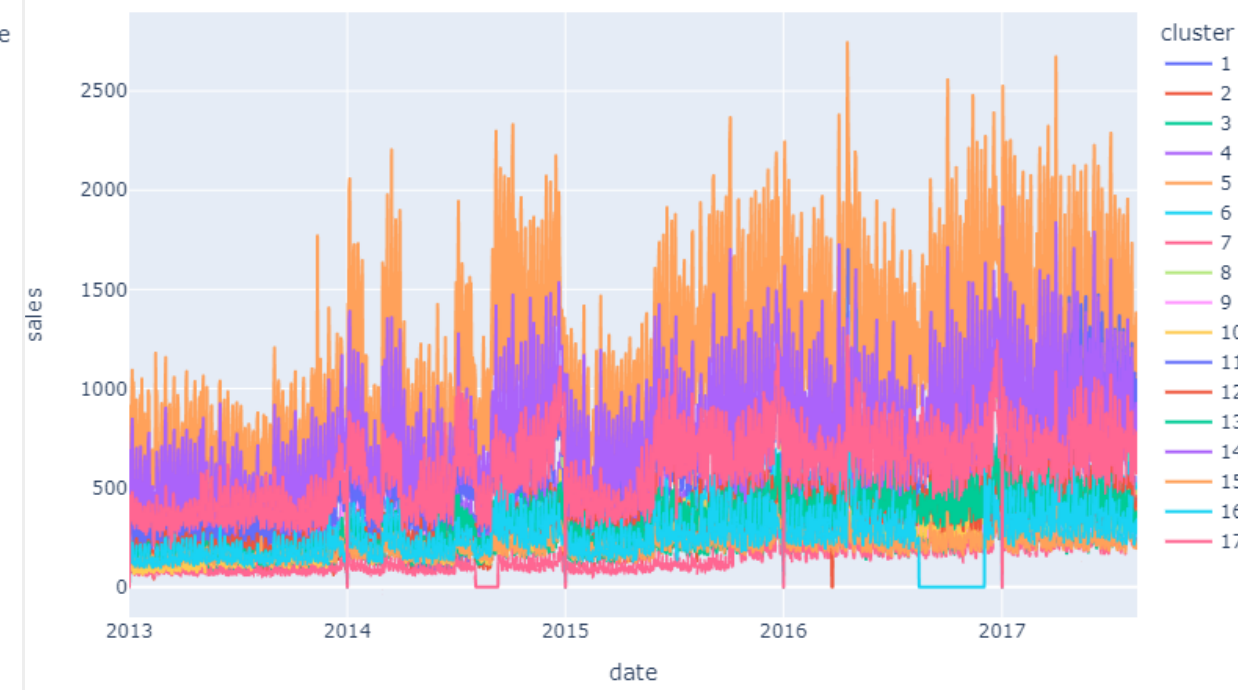
Daily total sales of the family



Average Sales by Date and Store Type



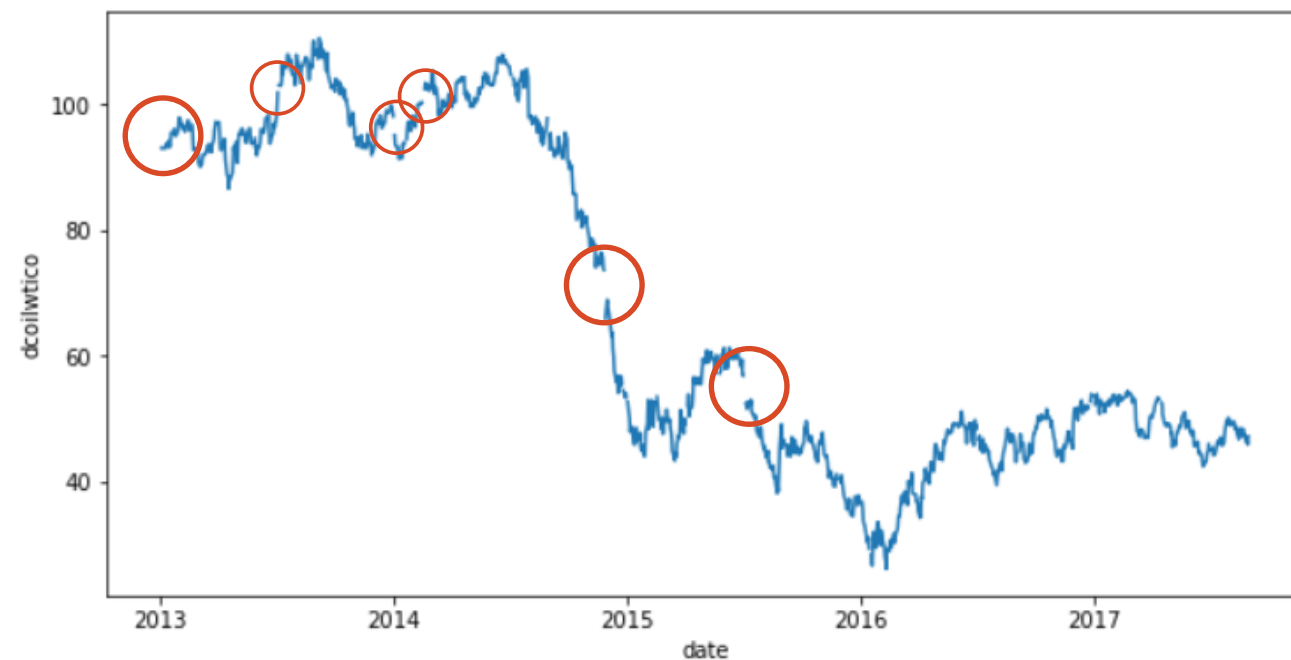
Average Sales by Date and Store Number



- store\_nbr, family, store\_type, cluster 에서 범주에 따른 sales에 차이를 보임
- 전체적으로 sales와 그 분산이 증가하는 추세
- 2016년 4월 지진으로 인해 몇몇 범주 (특히 GROCERY I)에서 매출이 급격히 증가
- 다수의 범주에서 연말에 매출이 증가하는 것을 확인할 수 있음

# 데이터 전처리 - 결측치 처리

## 1. dcoilwtico (oli price)



### - interpolation (보간)

실측값과 실측값 사이의 결측값을 그라데이션 기법으로 부드럽게 채워나가는 방법

### - backward

바로 뒤에 나오는 실측값으로 앞에 값을 채우는 방법

## 2. transactions (거래량)

locale_name	description	...	transactions	city	state	type_y
Empty	Empty	...	NaN	Quito	Pichincha	D
Empty	Empty	...	NaN	Quito	Pichincha	D
Empty	Empty	...	NaN	Quito	Pichincha	D
Empty	Empty	...	NaN	Quito	Pichincha	D
Empty	Empty	...	NaN	Quito	Pichincha	D

### - linear Regression model

- 결측이 아닌 데이터를 이용하여 선형회귀모형 적합

- 종속변수: transactions

- 예측변수: store\_nbr, state, type\_y, cluster, dcoilwtico, year, month, weekend, weekday, trend

# 데이터 전처리 - 파생변수 생성

## 1. 시계열 관련 변수

- Year (연도)
- Month (달)
- trend
- Weekend (주말)
- Weekday (요일)
- NewYearsDay (새해)

## 2. hoilday 관련 데이터 변형

type	locale	locale_name	description	transferred
Holiday	Local	Manta	Fundacion de Manta	False
Holiday	Regional	Cotopaxi	Provincializacion de Cotopaxi	False
Holiday	Local	Cuenca	Fundacion de Cuenca	False
Holiday	Local	Libertad	Cantonizacion de Libertad	False
Holiday	Local	Riobamba	Cantonizacion de Riobamba	False

### - 'no\_work\_day' feature 생성

- 실제로 유의미하고(transferred = 'False'), 국가적인 행사이며(locale = 'National'), 휴일인 날(type = 'Holiday', 'Additional', 'bridge')을 선택

- 위에서 선택한 날을 weekend 열과 OR 연산

### - 기념일 feature 생성

- description 정보를 활용하여 Earthquake, Football, Cyber Monday, Black Friday, Dia de la Madre(어머니날) feature 생성

# 데이터 전처리 - family 변수 그룹화

비슷한 특성을 가진 family 변수를 그룹화 한 후, 그룹화 한 것과 그룹화하지 않는 것의 rmsle 비교

Tools	AUTOMOTIVE, HARDWARE, LAWN AND GARDEN, PLAYERS AND ELECTRONICS
LifeStyle	BEAUTY, LINGERIE, LADIESWEAR, PERSONAL CARE, CELEBRATION, MAGAZINES, BOOKS, BABY CARE
Home	HOME APPLIANCES, HOME AND KITCHEN I, HOME AND KITCHEN II, HOME CARE, SCHOOL AND OFFICE SUPPLIES
Food	GROCERY II, PET SUPPLIES, SEAFOOD, LIQUOR,WINE,BEER
Daily	DELI, EGGS

## 그룹화 O 성능

training : 1.471248  
validation : 1.322420

## 그룹화 X 성능

training : 0.311052  
validation : 0.712528

그룹화하지 않는 것이 더 좋은 성능을 보임



# 데이터 전처리

## 1. onehotencoding

```
pd.get_dummies(drop_first = True)
```

store\_nbr(54개의 값들)

store_nbr	store_nbr_2	store_nbr_3	...	store_nbr_53	store_nbr_54
1	0	0		0	0
1	0	0		0	0
1	0	0		0	0
1	0	0		0	0
1	0	0		0	0
...					
9	...	...		...	...
9	0	0		0	0
	0	0		0	0
9	0	0		0	0
9	0	0		0	0
9	0	0		0	0
9	0	0		0	0

->

type\_y(5개의 값들)

type_y	type_y_B	type_y_C	type_y_D	type_y_E
D	0	0	1	0
D	0	0	1	0
D	0	0	1	0
D	0	0	1	0
D	0	0	1	0
...				
B	...	...	...	...
B	1	0	0	0
B	1	0	0	0
B	1	0	0	0
B	1	0	0	0
B	1	0	0	0
B	1	0	0	0

->

family(33개의 값들)

family	family_EGGS	family_FROZEN FOODS
AUTOMOTIVE	0	0
BABY CARE	0	0
BEAUTY	0	0
BEVERAGES	0	0
BOOKS	0	0
...		
POULTRY	...	...
PREPARED FOODS	0	0
	0	0
PRODUCE	0	0
SCHOOL AND OFFICE SUPPLIES	0	0
SEAFOOD	0	0

->

# 데이터 전처리

## 2. Min-Max Normalization

min-max normaliztion 전

onpromotion	dcoilwtico	transactions
0	93.140000	1420.876311
0	93.140000	1420.876311
0	93.140000	1420.876311
0	93.140000	1420.876311
0	93.140000	1420.876311
...	...	...
1	40.546667	2115.000000
1	40.546667	2115.000000
3	40.546667	2115.000000
0	40.546667	2115.000000
1	40.546667	2115.000000

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

min-max normaliztion 후

onpromotion	dcoilwtico	transactions
0.000000	0.792965	0.169385
0.000000	0.792965	0.169385
0.000000	0.792965	0.169385
0.000000	0.792965	0.169385
0.000000	0.792965	0.169385
...	...	...
0.001350	0.170042	0.252484
0.001350	0.170042	0.252484
0.004049	0.170042	0.252484
0.000000	0.170042	0.252484
0.001350	0.170042	0.252484

->

# 데이터 분할

## 1. train\_data

(date : 2013-01-01 ~ 2017-08-15)

### 1-1)sub\_train\_data

(date : 2013-01-01 ~ 2016-07-31)

### 1-2)valid\_data

(date : 2016-08-01 ~ 2017-08-15)

## 2. test\_data

(date : 2017-08-16 ~ 2017-08-31)

## 모델 적합

### - linear regression

$$y_i = \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i, \quad i = 1, \dots, n,$$

: 종속 변수  $y$ 와 예측 변수 사이의 **선형 관계**를 모델링

#### 선형 회귀 장단점

- 특성이 많은 데이터셋의 경우 성능이 우수하다. (장점)
- 모델의 복잡도를 제어할 방법이 없어서 과대적합이 되기 쉽다. (단점)

\* 모델의 정규화로 과대적합을 제어할 수 있다.

: 규제를 통해 정규화(Ridge, Lasso, Elastic net...)

training

2.4700

validation

2.3188

## 모델 적합 - Lasso

$$\sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j| = \text{RSS} + \lambda \sum_{j=1}^p |\beta_j|.$$

- MSE가 최소가 되게 하는 가중치와 편향 + 가중치들의 절댓값의 합 최소
- 가중치의 모든 원소가 0이 되거나 0에 가깝게 되어야 함

### 주요 파라미터

- **alpha** 알파 (L1 규제 계수)

### 성능

알파값 0.05 일때,  
training : 2.452747  
validation : 2.317404

## 모델 적합 - Ridge

$$\sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^p \beta_j^2.$$

- MSE가 최소가 되게 하는 가중치와 편향 + 가중치들의 제곱의 합 최소
- 가중치는 0에 가까워질 뿐 0이 되지는 않음

### 주요 파라미터

- **alpha** 알파 (L2 규제 계수)

### 성능

알파값 0.05 일때,  
training : 2.448911  
validation : 2.305556

(alpha값을 높이면 훈련 세트의 성능은 나빠지지만 일반화에는 도움이 됩니다)

## 모델 적합 - XGBoost

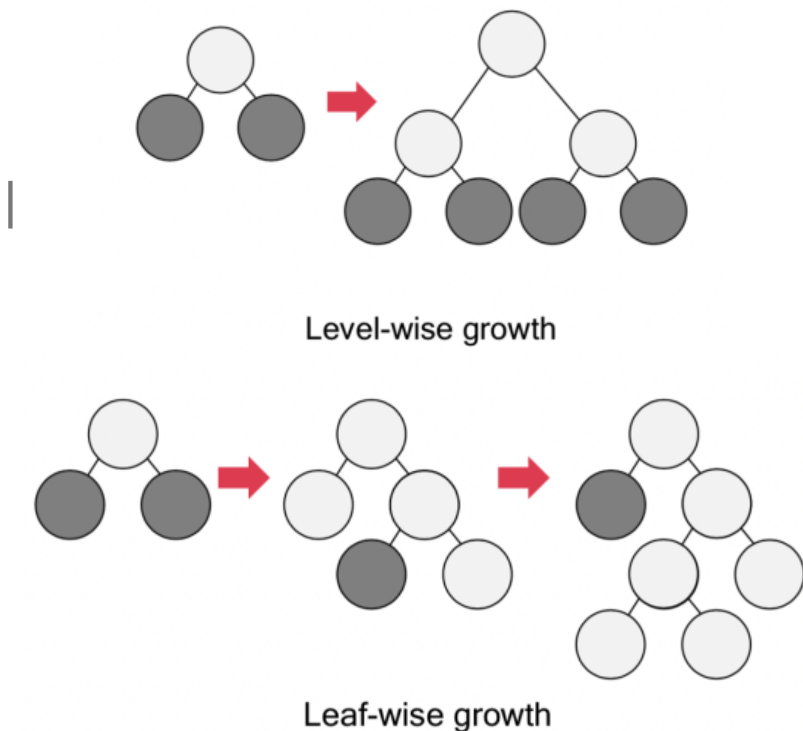


### Boosting

- 여러 개의 모델을 결합하는 앙상블 방식 중 하나
- 각 단계의 weak learner는 이전 단계의 weak learner의 잘못 예측된 데이터들에 가중치를 반영하면서 단계적으로 모델 구축

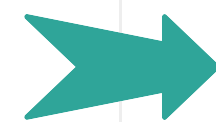
### XGBoost의 주요 특징

- level-wise 방식으로 분할 후 가지 치기
- 교차 검증 내장  
`xgboost.cv()`
- 과적합 규제
- 결측값 자체 처리



### 주요 부스터 파라미터

- eta : 학습률
- num\_boost\_around : 생성할 학습기의 수
- min\_child\_weight : 관측치에 대한 가중치 합의 최소
- gamma : 리프노드의 분할을 결정할 최소손실 감소값
- max\_depth : 최대 트리의 깊이
- subsample : 단계마다 사용할 샘플의 비율
- colsample\_bytree : 트리마다 사용할 feature의 비율
- lambda : L2 규제 파라미터
- alpha : L1 규제 파라미터



베이지안  
최적화

eta : 0.149  
num\_boost\_around : 2500  
min\_child\_weight : 0.581  
gamma : 5.388  
max\_depth : 12  
subsample : 0.946  
colsample\_bytree : 0.975  
lambda : 0.156  
alpha : 67.417

training

0.6783

validation

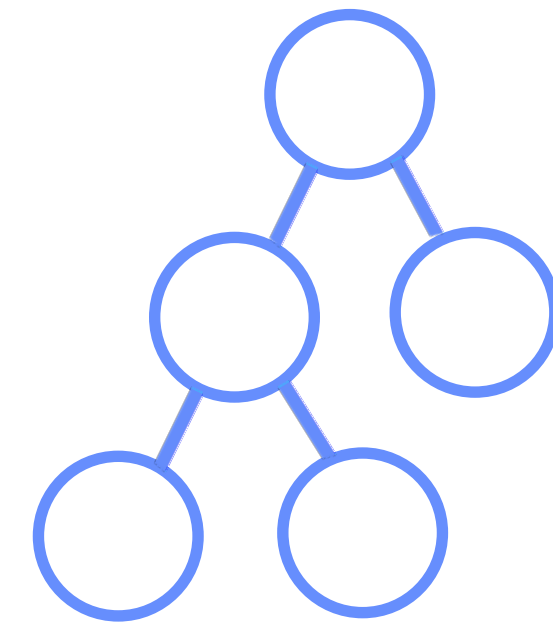
0.9676

# 모델 적합 - lightGBM

## - 주요 파라미터

- boosting : 부스팅 방식
- learning\_rate : 학습률
- max\_depth : 나무의 깊이
- num\_leaves : 최대 leaf node의 수
- feature\_fraction : feature sampling
- bagging\_fraction: data sampling

boosting : 'dart'(default), 'goss'		'goss'
learning_rate : (0.01, 0.1)		0.03
max_depth : (30, 100)	➡	43
num_leaves : (100, 10000)		8391
feature_fraction : (0.5, 0.9)		0.9
bagging_fraction : (0.4, 0.9)		0.5



leaf-wise

training

**0.8811**

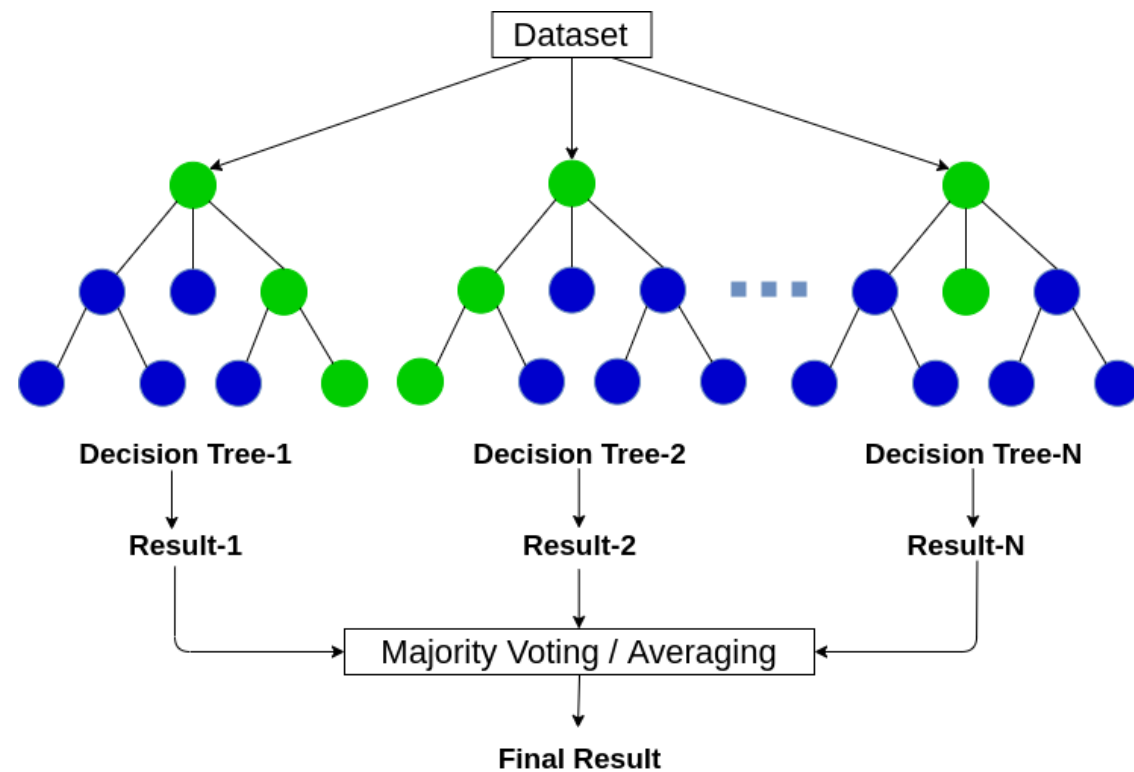
validation

**0.8991**

# 모델 적합 - Random Forest - 최종모델

## 주요 파라미터

- `n_estimators` 결정트리의 개수
- `max_depth` 트리의 최대 깊이
- `min_samples_split` 노드 분할에 필요한 최소한의 샘플 데이터 수
- `min_samples_leaf` 말단 노드에 필요한 최소한의 샘플 데이터 수
- `max_features` 최적의 분할을 위해 고려할 최대 feature 수
- `max_leaf_nodes` 말단 노드의 최대 개수



출처 : <https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/>

- 여러 개의 결정 트리가 각자의 샘플 데이터로 학습
- 최종적으로 투표나 평균을 통해 결과를 결정

- `n_estimators` 100
- `min_samples_leaf` 10
- `min_samples_split` 10

## 성능

training : 0.4867

validation : 0.6991

test: 0.52408

## 아쉬웠던 부분

### 1. lasso, lidge 상호작용 넣은 모델 (104 columns -> 5460 columns)

- 상호작용항 생성 : `PolynomialFeatures(degree=2,interaction_only=False,include_bias=False)`

`MemoryError`

`<ipython-input-39-ba6e52b85aa2>`

- 메모리 에러로 `sub_train_data` (100000행), `valid_data`(100000행)만 사용-

#### 1) Lidge(alpha = 30)

훈련데이터 rmsle: 1.586924924998741  
검증데이터 rmsle: 3.9853482596603054

#### 2) Lasso(alpha = 10)

훈련데이터 rmsle: 1.8023334342920907  
검증데이터 rmsle: 2.285426377422259

### 2. SVM

```
from sklearn.svm import SVR

model = SVR(kernel = 'rbf')
model.fit(X_train_data_trans, y_train)
```

- 클래스 결정경계를 찾아주는 방법
- 모델이 돌아가지 않아 결과 확인이 불가능했음
- 좀 더 덜 복잡한 kernel을 사용해보는 시도 (kernel = 'linear')