
Description

- 2016.03.01~2019.02.28 까지의 카드 거래 데이터를 이용해 2019.03.01~2019.05.31 까지의 **상점별 3개월 총 매출**을 예측하자.

Data field

- store_id : 상점의 고유 아이디
- card_id : 사용한 카드의 고유 아이디
- card_company : 비식별화된 카드 회사
- trasacted_date : 거래 날짜
- transacted_time : 거래 시간(시:분)
- installment_term : 할부 개월 수(포인트 사용 시 할부 개월수 = 60개월 + 실제할부개월)
- region : 상점의 지역
- type_of_business : 상점의 업종
- amount : 거래액(단위는 원이 아니다)

Evaluate Metric

- MAE : 절댓값 오차의 평균 = $|실제값 - 예측값|$ 의 평균
 - 장점
 1. 매우 직관적인 지표라 성능지표
 - ex) 자전거 대여 개수 예측 모델에서 MAE가 3이 나왔다면, 평균적으로 자전거 3대 정도를 잘못 예측
 - 단점
 1. 잔차에 절댓값 씌우기에 실제 값에 대해 underestimates인지 overestimates인지 파악하기 힘들다
 2. 스케일에 의존적이다
 - ex) 삼성전자의 주가가 10000원이고, 네이버가 5000원 일때, 두 주가를 예측하는 각각 모델의 MSE가 1000이 나왔다면, 동일한 에러율이 아님에도 MAE 숫자는 동일하게 보여진다

Index

Issue

- 예측 날짜는 2019-03~2019-05로 동일하나, 제공 데이터의 마지막 날짜는 차이가 있다.
 - 마지막 날짜부터 3개월만 예측하여 제출한다.(ex: store_id 111의 마지막 날짜는 2018-09월로 뒤 3개월인 2018-10~2018-12만 예측하여 제출)
 - 예측 기간이 길어질수록 오차가 크게 발생하여 바로 뒤 3개월만 예측하는 것이 정확도가 높았음

step1. Data Load & Resampling

전체적인 데이터, 결측치 확인

- col, num_of_unique, num_of_nan, dtype, front5_value 통해 데이터 확인
- 1) 9개 변수 중 2개의 변수(region, type_of_business)에서 결측치 확인
 => 결측치 비율이 높아서, region, type_of_business 두 변수 제거
- 2) date 정보를 object 에서 datetime 타입으로 변경

변수 선택, 파생 변수 생성 (일자 별 데이터) 등

- store_id 선택
- resampling 을 통한 파생 변수 생성

Day 단위로 Resampling	Month 단위로 Resampling
day_of_week 변수 생성 월 0 ~ 일 6	real_tot_day 변수 생성 한달 전체 거래 일 수 1) 월 단위 day_of_week 행 count
business_day 변수 생성 월~금 1, 토~일 0	real_business_day 변수 생성 한달 거래 중 business 일 수 1) 월 단위 business_day 행 sum
num_of_pay 변수 생성 일일 store_id 별 결제 건수 1) store_id별 하루 몇 번 결제되는지 count 2) *resample('1D') 시계열 메소드 활용	num_of_pay 변수 가공 월 store_id 별 결제 건수
num_of_revisit 변수 생성 일일 store_id 별 단골의 결제 건수 1) *resample('1D') 시계열 메소드 활용	num_of_revisit 변수 가공 월 store_id 별 단골의 결제 건수

1) store_id별 card_id가 3회 이상이라면 단골이라고 판단 2) store_id별 단골 card_id가 하루 몇 번 결제되는지 count	
installment_term 변수 가공 일 총 할부 개월 수 <u>* 할부를 이용해서 일일 포인트 횟수를 count한 변수 생성하면 좋을 것 같다</u>	installmen_term 변수 가공 월 총 할부 개월 수
amount 변수 가공 일 매출 액	amount 변수 가공 월 매출 액

* resample 메소드의 시간 단위 구간 설정

10분 단위 구간 : resampel('10T')

1시간 단위 구간 : resampel('1H')

1일 단위 구간 : resampel('1D')

1주일 단위 구간 : resampel('1W')

step2. EDA

- target 데이터 분포 확인

왜도 = 분포의 비대칭도를 나타내는 통계량

*왜도

왜도 = 0 : 대칭인 분포

왜도 > 0 : 오른쪽으로 긴 꼬리

왜도 < 0 : 왼쪽으로 긴 꼬리

- store_id 별 target 변수 특성과 분포가 각각 다르다
- 설명 변수의 분포 확인
- 다른 변수들이 'amount'와 같은 pattern을 가지는지 확인

step3. Modeling - Time Series

이동평균법 (단순 이동 평균, 지수 이동 평균)

일정기간별 이동평균을 계산하고 이들의 추세를 파악하여 다음 기간을 예측한다
시계열 자료에서 계절 변동과 불규칙 변동을 제거하여, 추세 변동과 순환 변동만 가진 시계열로 반환

지수평활법 (단순 지수 평활법, 홀트의 지수 평활법, 홀트-윈터스의 지수 평활법)

일정기간별이 아닌 모든 시계열 자료를 사용하여 평균을 구하며, 시간 흐름에 따라 최근 시계열에 더 많은 가중치를 부여해 미래를 예측한다

1. 이동 평균법

1-1. Simple Moving Average (단순 이동 평균)

Simple Moving Average

- 단순이동평균은 특정 기간 동안의 data를 단순 평균하여 계산한다. 따라서 그 기간 동안의 data를 대표하는 값이 이동평균 안에는 그 동안의 data 움직임을 포함하고 있다.
- 이동평균의 특징인 지연(lag)이 발생하며 수학적으로 $n/2$ 시간 만큼의 지연이 발생한다.
- 단순이동평균은 모든 데이터의 중요도를 동일하다고 간주한다.

$$SMA_t = \frac{D_{t-(n-1)} + D_{t-(n-2)} + \cdots + D_{t-1} + D_t}{n}$$

- `df.rolling(갯수).mean()`
- `window=3`을 예로 들면, `rolling`의 경우 t 시점에 대한 값으로 t 시점, $t-1$ 시점, $t-2$ 시점의 평균을 사용한다.
- 하지만 우리가 필요한 건 t 시점에 대한 값으로 $t-1$ 시점, $t-2$ 시점, $t-3$ 시점의 평균을 구하는 것이다.

1-2. Exponential Moving Average (지수 이동 평균)

Exponential Moving Average

- 지수이동평균은 가중이동평균 중의 하나로 단순이동평균보다 최근의 데이터에 높은 가중치를 부여하는 방법이다.

$$EMA_t = D_t \times \frac{2}{N+1} + EMA_{t-1} \times \left(1 - \frac{2}{N+1}\right)$$

- `df.ewm(span=갯수).mean()`

2. Exponential Smoothing (지수 평활법)

2-1. Simple Exponential Smoothing (단순 지수 평활법)

1) Simple Exponential Smoothing

- trend나 seasonality 반영을 하지 못함
- level 정도만 수평선으로 나옴

$$F_t = F_{t-1} + \alpha(D_{t-1} - F_{t-1})$$

$$F_t = (1 - \alpha)F_{t-1} + \alpha D_{t-1}$$

- F_t : 현재 시점의 예측 값
- F_{t-1} : 이전 시점의 예측 값
- D_{t-1} : 이전 시점의 실제 값
- α : smoothing 요소, $0 < \alpha < 1$

- 추세나 계절성 패턴이 없는 데이터를 예측할 때 쓰기 좋다

- `ses_model = SimpleExpSmoothing(df)`
- `ses_result = ses_model.fit()`
- `ses_pred = ses_result.forecast(3)`

2-2. Holt's Exponential Smoothing (홀트 지수 평활법)

2) Holt's Exponential Smoothing

- trend로 데이터를 예측하기 위해 Simple Exponential Smoothing에서 확장한 것이다.
- 예측을 위한 식 외에 level smoothing을 위한 식과 trend smoothing을 위한 식이 포함된다
- 생성된 예측은 선형적으로 나타나기 때문에 예측 범위가 멀어질 수록 over-forecast 되는 경향이 있다.

- Forecast equation :

$$\hat{y}_{t+h|t} = l_t + hb_t$$

- Level equation :

$$l_t = \alpha y_t + (1 - \alpha)(l_{t-1} + b_{t-1})$$

- Trend Equation :

$$b_t = \beta^*(l_t - l_{t-1}) + (1 - \beta^*)b_{t-1}$$

- l_t : t 시점에서의 level(수준)의 추정을 나타낸다.
- b_t : t 시점에서의 추세(경사)의 추정을 나타낸다.
- α : level(수준)에 대한 smoothing parameter이고 다음과 같은 범위를 갖는다. $0 < \alpha < 1$
- β^* : trend(추세)에 대한 smoothing parameter이고 다음과 같은 범위를 갖는다. $0 < \beta^* < 1$
- level에 대한 식 l_t 는 t 시간에 대한 관측치 y_t 와 훈련 예측 $l_{t-1} + b_{t-1}$ 의 가중 평균을 나타낸다.
- trend에 대한 식 b_t 는 $(l_t - l_{t-1})$ 에 근거한 t시간에 대한 추정치와 이전 추정치인 b_{t-1} 의 가중 평균을 나타낸다.

- SES + trend(지속적인 추세)

- holt_model = Holt()
- holt_result = holt_model.fit()
- holt_pred = holt_result.forecast()

2-3. Holt-Winter's Exponential Smoothing (홀트-윈터스의 지수 평활법)

3) Holt-Winter's Exponential Smoothing

- Holt-Winter's 방법은 seasonality를 반영하기 위해 Holt's 방법에서 확장된 것이다.
- Holt-Winter's 방법은 예측식과 3개의 smoothing 식으로 구성되어 있다.
 - level에 대한 식 l_t
 - trend에 대한 식 $b - t$
 - seasonal에 대한 요소 s_t
 - smoothing parameter에 해당하는 α, β^*, γ
 - seasonality의 빈도를 나타내기 위한 m
- seasonal이 변화하는 형태에 따라 두 가지 방법이 있다.
 - additive : seasonal의 변화가 일정하게 지속될 때
 - multiplicative : seasonal의 변화가 level에 비례적일 때

- Forecast equation

$$\hat{y}_{t+h|t} = l_t + hb_t + s_{t+h-m(k+1)}$$

- Level equation

$$l_t = \alpha(y_t - s_{t-m}) + (1 - \alpha)(l_{t-1} + b_{t-1})$$

- Trend equation

$$b_t = \beta^*(l_t - l_{t-1}) + (1 - \beta^*)b_{t-1}$$

- Seasonal equation

$$s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1 - \gamma)s_{t-m}$$

- level에 대한 식은 t 시간에 대한 조정된 seasonally 관측치 $(y_t - s_{t-m})$ 와 non-seasonally 예측치 $(l_{t-1} + b_{t-1})$ 의 가중 평균을 나타낸다.
- trend에 대한 식은 Holt's 선형식에서와 동일하다.
- seasonal에 대한 식은 현재 seasonal 지수 $(y_t - l_{t-1} - b_{t-1})$ 와 이전 seasonal 지수(m 기간

전) 사이의 가중 평균을 나타낸다.

- SES + trend(지속적인 추세) + seasonality(계절성)

- 1) 가법 방법은 계절 변동이 계열 전체에서 대략적으로 일정 할 때 선호
- 2) 곱셈 방법은 계절 변동이 계열 수준에 비례하여 변할 때 선호