# 6 Powerful Feature Engineering Techniques For Time Series Data (using Python)

**개요**

Feature engineering은 시계열 분석의 경우 특히 중요하다.

6가지 강력한 엔지니어링 기술을 살펴본다.

**소개**

시간은 비즈니스에서 가장 필수적인 요소이다. 우리는 판매량, 수익, 최저값, 성장률, 그리고 심지어는 미래에 대한 예측까지 시간적 요소와 매핑하곤 한다.

그러나 시간 요소는 분석 초보자가 이해하기 어려운 주제이다. 우리가 시간에 민감한 데이터셋을작업할 때 반드시 고려해야하는 많은 뉘앙스가 존재한다.

기존의 시계열 예측 모형들은 대체적으로 잘 작동하지만, 어떤 제약을 가지고 있다. 많은 데이터 사이언티스트들은 시간 성분과 종속변수만 주어진 데이터를 매핑하기 어려워한다. 이 작업은 까다로운 작업이지만 불가능한 도전은 아니다.

앞으로 다룰 접근법이 모든 경우에 들어맞는 것은 아니다. 상황에 맞는 방법을 적용해야 한다. 또한 앞으로 다룰 접근법이 ARIMA같은 전통적인 시계열 분석에 꼭 맞춰질 필요는 없다. 수요 예측과 같은 프로젝트는 지도학습이 필요할 수 있다.

시계열 데이터에 특성공학을 적용하여 훌륭한 예측 모형을 만들어보자.

이 문서에서는 날짜, 시간 변수를 사용하여 유용한 정보를 추출하기 위한 다양한 특성공학을 살펴볼 것이다.

**목차**

1. 시계열에 대한 간단한 소개
2. 시계열 데이터를 위한 문제 정의
3. 날짜와 관련된 특징
4. 시간과 관련된 특징
5. Lag 특징
6. 롤링 윈도우
7. 윈도우 확장
8. 도메인 지식

**시계열에 대한 간단한 소개**

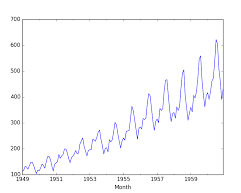
우리는 이 문서에서 시계열을 다룰 것이기 때문에 이곳에서 먼저 시계열에 대한 지식을 정리할 필요가 있다. 특성공학을 살펴보기 전, 시계열의 기본적인 컨셉에 대해 훑어보자.

시계열의 어떤 요소가 전통적인 머신러닝 문제를 어렵게 만들까?

시계열에는 데이터가 동일한 간격마다 기록되고 각각의 연속적인 포인트들은 과거의 포인트들에 의존한다.

이것을 위한 간단한 예시를 살펴보자. 만약 우리가 오늘 어떤 회사에 대한 시가를 예측한다고 해보자. 해당 회사의 전날 종가에 대한 정보를 아는 것이 당연히 큰 도움이 될 것이다. 유사하게, 웹사이트의 트래픽을 예측하는 것은 지난 몇 달 또는 몇 년 간의 데이터를 가지고 수행할 것이다.

여기에는 우리가 이해할 필요가 있는 또 다른 요소가 숨어있다. 시계열 데이터는 추세나 계절성을 가진다. 아래의 그림은 항공권의 예매 횟수에 대한 그래프이다.



우리는 위 그래프를 통해 추세가 증가한다는 사실을 알 수 있다. 이러한 정보는 예측 정확도를 향상시킬 수 있다. 이제 날짜와 시간을 가진 데이터셋을 보고 특성공학을 공부해보자.

**시계열 데이터의 문제 정의**

우리는 시계열을 위한 특성공학 기술들을 배울 것이다.

여기서 사용할 데이터는 대중교통의 일종인 ‘제트레일’의 교통량 데이터셋이다. 최근 제트레일의 사용량은 증가하고 있으며, 과거 데이터를 바탕으로 향후 7개월 동안의 제트레일의 교통량을 예측해야 한다.

**날짜와 관련된 특징**

특정 제품의 판매량을 예측하는 작업은 가장 일반적인 시계열 분석 문제이다. 실적 자료를 바탕으로 평일과 주말의 판매 패턴을 알 수 있다. 따라서 일, 월, 연도 등에 대한 정보를 가지고 있다면 값을 예측하는 데 유용할 수 있다.

우리는 앞으로 7개월 간의 제트레일의 교통량을 예측해야 한다. 이 수치는 평일에 더 높을 수 있고 주말이나 축제 기간에는 더 낮을 수 있다. 따라서 요일(주말 또는 평일) 또는 월이 중요한 요소가 될 수 있다.

테이블이(가) 표시된 사진

자동 생성된 설명

**시간과 관련된 특징**

마찬가지로 시간 변수가 있다면 더 세분화된 특징을 추출할 수 있다. 예를 들어 데이터가 기록된 시간 또는 분을 결정하고 영업 시간과 영업 외 시간 간의 추세를 비교할 수 있다.

우리는 제트레일의 교통량이 아침, 오후, 저녁 시간대에 더 높은 지 알 수 있다. 또는 이 값을 사용하여 한 주 동안 평균 시간 당 트래픽을 확인할 수 있다. 즉 오전 9시 ~ 10시, 오전 10시 ~ 11시 사이에 제트레일을 사용한 사람의 수를 확인할 수도 있다.

테이블이(가) 표시된 사진

자동 생성된 설명

다음은 날짜, 시간을 통해 추출할 수 있는 변수 목록이다.

테이블이(가) 표시된 사진

자동 생성된 설명

**Lag 특징**

어떤 회사의 주가를 예측하고 있다면, 전날 주가는 중요한 정보일 것이다. 왜냐하면 t시점의 값은 t-1시점의 값에 큰 영향을 받기 때문이다. 과거 값은 시차라고 하며, t-1은 시차1, t-2는 시차2라고 한다.

테이블이(가) 표시된 사진

자동 생성된 설명

그렇다면 lag1 말고 lag5나 lag7도 가능할까?

Lag의 값은 각 시계열 포인트가 과거의 값들과 얼마나 상관성을 갖는지에 의존한다. 만약 시계열이 일주일의 추세를 갖는다면, 지난 주 월요일 데이터가 이번 주 월요일 데이터를 예측하는 데 사용될 수 있음을 의미한다. 이런 경우에 lag7을 사용해야 한다.

Lag 변수는 여러 개를 만들 수도 있다. 1 ~ 7의 lag를 만든 데이터는 다음과 같다.

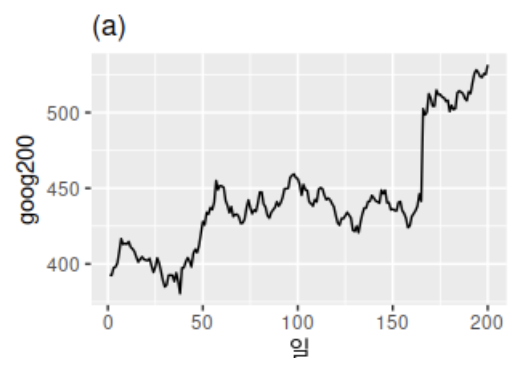
|  |
| --- |
|  |
| data['lag\_1'] = data['Count'].shift(1) |
| data['lag\_2'] = data['Count'].shift(2) |
| data['lag\_3'] = data['Count'].shift(3) |
| data['lag\_4'] = data['Count'].shift(4) |
| data['lag\_5'] = data['Count'].shift(5) |
| data['lag\_6'] = data['Count'].shift(6) |
| data['lag\_7'] = data['Count'].shift(7) |
|  |  |

테이블이(가) 표시된 사진

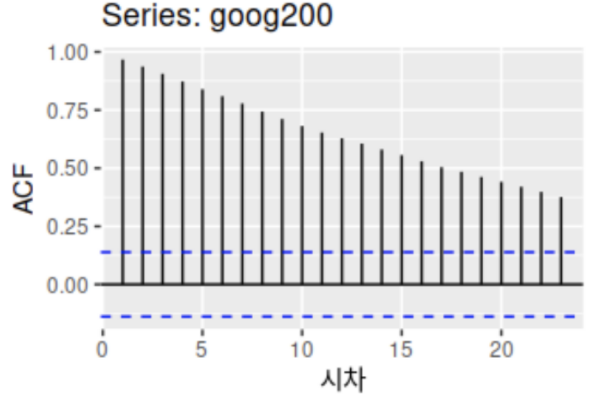
자동 생성된 설명

시차간의 유의미한 상관성을 판단하는 방법이 몇 개 있다. 예를 들어 ACF(자기 상관 함수), PACF(부분 자기 상관 함수) 그림을 사용하는 것은 좋은 방법이 될 수 있다.

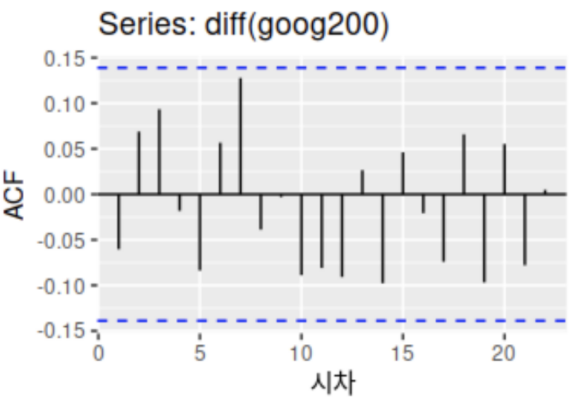
ACF: 시계열과 시차 간의 상관 관계를 측정한 것. 시차에 따른 자기상관을 의미하며 시차가 커질수록 ACF는 0에 가까워진다. 정상성을 가진 시계열은 상대적으로 빠르게 0에 수렵하며, 비정상 시계열은 천천히 감소하고, 종종 큰 양의 값을 가진다.



위 그래프는 증가하는 추세를 가진 비정상성 시계열이다. 이 데이터에 대해 ACF를 그려보면 다음과 같다.



ACF가 느리게 감소한다. 이제 정상성을 갖도록 차분을 한 뒤 ACF를 다시 그리면 다음과 같다.



전보다 0에 가까우며 빠르게 감소한다.

PACF: 간섭 비교에 의해 설명되는 변동성을 제거한 뒤 상관 관계를 측정한 것. 시차에 따른 편자기상관이며, 순수한 상호 연관성이다. 예를 들어 t시점과 t-7시점 간의 상관성을 볼 때, 그 사이에 존재하는 시점들의 영향을 제거한 뒤 순수한 상관성을 보는 것이다.

테이블이(가) 표시된 사진

자동 생성된 설명

텍스트, 장치이(가) 표시된 사진

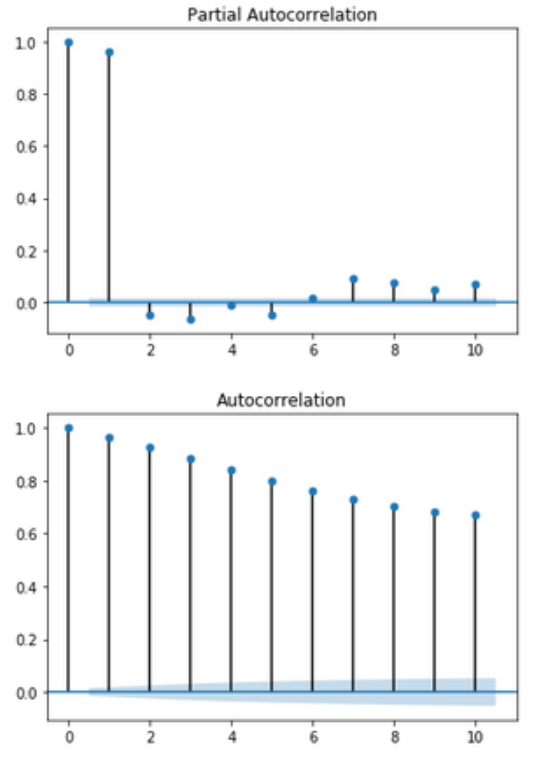
자동 생성된 설명

ACF 상에서 1시점 이후 0에 수렴한다. PACF에서는 2시점 이후 0에 수렴한다. 따라서 MA(1), AR(2), ARMA(2, 1)을 활용할 수 있다.

from statsmodels.graphics.tsaplots import plot\_acf

plot\_acf(data['Count'], lags=10)

plot\_pacf(data['Count'], lags=10)

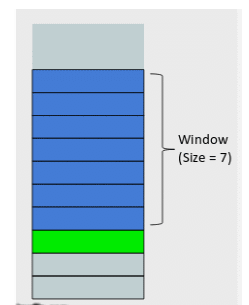


다시 제트레일의 데이터를 확인해보자.

ACF에서는 1시점 이후에 감소한다. 즉 t시점의 값은 t-1시점까지 상관성이 높다는 것이다. 이 경우에는 AR(1)을 활용해볼 수 있다. PACF에서는 점진적으로 감소한다.

**윈도우 롤링**

이제 우리는 과거의 값을 어떻게 특징으로 사용할 수 있는지 살펴볼 것이다. 과거의 값을 기반으로 하는 통계 수치를 어떻게 계산할 수 있을까? 이 방법은 롤링 윈도우라고 불린다. 왜냐하면 윈도우가 밀려나가면서 서로 다른 일정량의 데이터 포인트를 가지기 때문이다.



윈도우가 미끄러지면서 새로운 특징들이 생성되며, 이들은 ‘롤링 윈도우 특징’이라고 불린다.

우리는 윈도우 사이즈를 설정하고, 윈도우 내부의 값들의 평균을 계산할 수 있다.

data['rolling\_mean'] = data['Count'].rolling(window=7).mean()

테이블이(가) 표시된 사진

자동 생성된 설명

평균 대신 합계, 최소, 최대값 등을 사용할 수 있다. 또는 가중 평균을 사용할 수 있다. t시점과 가까운 시점의 포인트들에 더 큰 가중치를 줘서 평균을 계산할 수 있다.



**윈도우 특징 확장**

이제 좀 더 고급진 롤링 기법을 살펴보자. 앞서 살펴본 롤링 윈도우 기법은 윈도우의 크기는 일정하지만 시간이 지날수록 윈도우가 미끄러져나가기 때문에 최근의 값만 고려하고 과거의 값을 무시한다.

윈도우를 확장하는 것은 과거의 값까지 고려하겠다는 것이다.

data['expanding\_mean'] = data['Count'].expanding(2).mean()

테이블이(가) 표시된 사진

자동 생성된 설명

**도메인 지식**

도메인 지식을 사용하는 것은 특성공학에서 필수이다.

문제에 대한 뛰어난 이해, 명확한 목적, 가용 데이터에 대한 지식은 모델을 위한 도메인 기반 특성공학의 필수 요소들이다.

아래의 테이블은 소매업자가 여러 점포와 제품에 대해 제공한 자료이다. 우리의 업무는 제품의 향후 수요를 예측하는 것이다. 이 때 lag1에서 lag7로 지연 특성을 추가하는 것이 옳은 방법일까?

테이블이(가) 표시된 사진

자동 생성된 설명

정답은 ‘아니다’이다.

위 데이터에는 서로 다른 가게들과 서로 다른 제품들이 존재한다. 그리고 각 가게와 제품의 수요는 다를 것이다. 이 경우, 우리는 가게와 상품의 조합으로 데이터를 구분한 뒤 lag 특징을 생성해야 한다. 게다가 만약 우리가 시장에서의 특정 제품의 트렌드를 알고 있다면, 더 정확한 특징을 생성할 수 있다.

도메인과 데이터에 대해 잘 이해하는 것은 우리가 적절한 lag값을 찾고 윈도우 사이즈를 정할 수 있게 도와준다. 게다가 도메인 지식에 기반하여 우리는 모델에 필요한 외부 데이터를 가져올 수도 있다.

지금까지 논의만 모든 특성공학 기술을 사용하면 시계열 문제를 지도 학습 문제로 변환할 수 있다.

우리는 선형 회귀나 랜덤 포레스트와 같은 기계 학습 알고리즘을 진행할 수 있다. 그러나 모델을 생성하기 전에 꼭 알아야 할 중요한 단계가 있다. 바로 시계열 검증 세트를 만드는 것이다.

기존의 기계 학습 문제의 경우 검증 데이터를 훈련 데이터에서 임의로 선택한다. 그러나 시계열의 경우 시간에 따라 값이 달라지기 때문에 무작위로 섞을 경우 제대로 된 검증을 할 수 없게 된다.

시계열 문제를 처리할 때 데이터 내의 순차적인 순서를 파괴하지 않고 검증 세트를 신중하게 구축하는 것이 중요하다.

data.index = data.Datetime

Train=data.loc['2012-08-25':'2014-06-24']

valid=data.loc['2014-06-25':'2014-09-25']

**마무리**

시계열은 어려운 주제이다. 날짜와 시간을 가지고 작업할 때 변동되거나 연관된 의미가 많기 때문이다. 그러나 기본 개념을 숙지하고 특성공학을 수행할 수 있게 되면 수월하게 분석을 진행할 수 있을 것이다.

특성공학 기술을 사용하여 시계열 문제를 지도 학습 문제로 변환하고 회귀 모델을 구축할 수 있다.

**레퍼런스**

1. <https://www.analyticsvidhya.com/blog/2019/12/6-powerful-feature-engineering-techniques-time-series/>
2. <https://leedakyeong.tistory.com/entry/ARIMA%EB%9E%80-ARIMA-%EB%B6%84%EC%84%9D%EA%B8%B0%EB%B2%95-AR-MA-ACF-PACF-%EC%A0%95%EC%83%81%EC%84%B1%EC%9D%B4%EB%9E%80>