



Grupo Bimbo Inventory

Demand

Grupo Bimbo 재고 수요량 예측 프로젝트 [Version 1]

작성자/작성일 : 류현철 - 2024년 3월 22일

<https://github.com/HyunCheol-Ryu/Grupo-Bimbo-Inventory-Demand>

사전 정보

프로젝트 개요

Grupo Bimbo는 베이커리 제품 제조를 전문으로 하는 멕시코의 다국적 기업입니다. 북미에서 가장 큰 규모의 베이커리를 운영하고 있으며 미국과 멕시코에 광범위한 유통망을 보유하고 있습니다. 멕시코 내 45,000개 경로를 따라 1백만 개 이상의 매장에 신선한 베이커리 제품을 공급하고 있습니다.

Grupo Bimbo는 머신 러닝 실무자들에게 자사 제품에 대한 정확한 재고 수요 예측을 위한 예측 모델 구축 과제를 부여하는 Kaggle 경진대회를 개최했습니다.

문제 설명

대회 설명 :

"이번 대회에서 Grupo Bimbo는 과거 판매 데이터를 기반으로 재고 수요를 정확하게 예측하는 모델을 개발하도록 Kagglers를 초대합니다. 이를 통해 100개가 넘는 베이커리 제품의 소비자들이 빈 진열대를 바라보지 않도록 하는 한편, 판매에 적합하지 않은 잉여 제품을 보유한 점주에게 환불하는 데 지출되는 금액을 줄일 수 있을 것입니다."

즉, 향후 판매 데이터에서 반품으로 조정된 순매출(조정 수요)에 대한 예측 오차를 최소화하는 것이 목표입니다. 저는 머신 러닝 모델을 구축하기 위해 LightGBM 알고리즘(그라데이션 부스팅의 변형)을 사용하여 향후 몇 주 동안의 조정 수요를 예측할 계획입니다.

평가 구조

대회의 평가 지표는 평균 제곱근 로그 오차(RMSLE)입니다. 경연 대회의 목표는 보류된 경연 데이터에 대한 모델 예측의 RMSLE를 최소화하는 것입니다. 이는 다음과 같이 계산됩니다 :

$$\epsilon = \sqrt{\frac{1}{n} \sum_{i=1}^n (\log(p_i + 1) - \log(a_i + 1))^2}$$

수식 :

- ϵ 는 RMSLE 값(점수)
- n 은 (공개/비공개) 데이터 세트의 총 관측 수
- p_i 는 예측치

- ai는 i에 대한 실제 응답
- $\log(x)$ 는 x의 자연 로그

RMSLE는 과소 예측된 예상치에 대해 과대 예측된 예상치보다 더 큰 불이익을 준다는 점에 유의 할 것. 즉, 대상을 과대 예측한 예측이 과소 예측한 예측보다 RMSLE가 더 커진다는 것을 의미합니다.

데이터 요약

대회에서 제공하는 데이터는 6개의 테이블로 구성됩니다 : train, test, town_state, cliente_tabla, producto_tabla 그리고 sample_submission.

Grupo Bimbo는 멕시코 회사이므로 설명에 영어가 아닌 스페인어로 표시되어 있습니다. 각 기능의 영어 설명은 아래 그림과 같습니다.

Data fields	
Semana	Week number (From Thursday to Wednesday)
Agencia_ID	Sales Depot ID
Canal_ID	Sales Channel ID
Ruta_SAK	Route ID (Several routes = Sales Depot)
Cliente_ID	Client ID
NombreCliente	Client name
Producto_ID	Product ID
NombreProducto	Product Name
Venta_uni_hoy	Sales unit this week (integer)
Venta_hoy	Sales this week (unit: pesos)
Dev_uni_proxima	Returns unit next week (integer)
Dev_proxima	Returns next week (unit: pesos)
Demanda_uni_equil	Adjusted Demand (integer) (This is the target you will predict)

분석 과정

첫번째 단계 : 데이터 전처리

기본적으로 제공된 데이터들을 살펴보고 크게 세가지 과정을 거쳤습니다.

1. 중복 데이터 제거

사전 설명에 명시했던대로 중복된 데이터가 존재하였습니다. Cliente_ID 테이블에서 중복 데이터를 확인하고 제거하였습니다.

2. 데이터 트리밍 및 추가 데이터 생성

Producto_ID	NombreProducto	Agencia_ID	Town	State
0	0 NO IDENTIFICADO 0	0	1110 2008 AG. LAGO FILT	MÉXICO, D.F.
1	9 Capuccino Moka 750g NES 9	1	1111 2002 AG. AZCAPOTZALCO	MÉXICO, D.F.
2	41 Bimbollos Ext sAjonjoli 6p 480g BIM 41	2	1112 2004 AG. CUAUTITLAN	ESTADO DE MÉXICO
3	53 Burritos Sincro 170g CU LON 53	3	1113 2008 AG. LAGO FILT	MÉXICO, D.F.
4	72 Div Tira Mini Doradita 4p 45g TR 72	4	1114 2029 AG.IZTAPALAPA 2	MÉXICO, D.F.

그림 1 : 데이터 트리밍 전 Producto_ID와 Town_State 테이블

Product_ID 테이블과 Town_State 테이블에서 다음과 같이 여러 텍스트와 ID가 함께 포함된 지저분한 데이터가 존재했습니다. 향후 분석에 사용할 피처를 넓히기 위해 트리밍을 실시하여 다음과 같이 분리하였습니다.

	Producto_ID	NombreProducto	Name	Weight	Pieces	Code
0	0	NO IDENTIFICADO 0	NaN	NaN	NaN	NaN
1	9	Capuccino Moka 750g NES 9	Capuccino Moka	750g	NaN	NES
2	41	Bimbollos Ext sAjonjoli 6p 480g BIM 41	Bimbollos Ext sAjonjoli	480g	6p	BIM
3	53	Burritos Sincro 170g CU LON 53	Burritos Sincro	170g	NaN	CU LON
4	72	Div Tira Mini Doradita 4p 45g TR 72	Div Tira Mini Doradita	45g	4p	TR

	Agencia_ID	Town	State	Town_ID
0	1110	2008 AG. LAGO FILT	MÉXICO, D.F.	2008
1	1111	2002 AG. AZCAPOTZALCO	MÉXICO, D.F.	2002
2	1112	2004 AG. CUAUTITLAN	ESTADO DE MÉXICO	2004
3	1113	2008 AG. LAGO FILT	MÉXICO, D.F.	2008
4	1114	2029 AG. IZTAPALAPA 2	MÉXICO, D.F.	2029

그림 2: 데이터 트리밍 후 *Producto_ID*와 *Town_State* 테이블

또한, 판매 가격과 반품 가격을 계산하여 상품당 평균 가격 테이블을 만들고 반영하였습니다.

3. 데이터 타입 변경

train 데이터셋이 csv파일로 3gb가 넘고, 이를 로딩 및 분석시 메모리 이슈가 빈번하게 발생하였습니다. 사용 메모리 감소와 속도 향상을 위해 데이터 타입을 가능한 작게 조정하였습니다.

<pre><class 'pandas.core.frame.DataFrame'> RangeIndex: 74180464 entries, 0 to 74180463 Data columns (total 11 columns): # Column Dtype --- --- 0 Semana int64 1 Agencia_ID int64 2 Canal_ID int64 3 Ruta_SAK int64 4 Cliente_ID int64 5 Producto_ID int64 6 Venta_uni_hoy int64 7 Venta_hoy float64 8 Dev_uni_proxima int64 9 Dev_proxima float64 10 Demanda_uni_equil int64 dtypes: float64(2), int64(9) memory usage: 6.1 GB</pre>	<pre><class 'pandas.core.frame.DataFrame'> RangeIndex: 74180464 entries, 0 to 74180463 Data columns (total 11 columns): # Column Dtype --- --- 0 Semana int8 1 Agencia_ID int16 2 Canal_ID int8 3 Ruta_SAK int16 4 Cliente_ID int32 5 Producto_ID int32 6 Venta_uni_hoy int16 7 Venta_hoy float32 8 Dev_uni_proxima int32 9 Dev_proxima float32 10 Demanda_uni_equil int16 dtypes: float32(2), int16(4), int32(3), int8(2) memory usage: 2.1 GB</pre>
--	---

그림 3: train 테이블의 데이터 타입 변경 전/후

이 과정은 이 후, 피처 엔지니어링 이후 추가 데이터 프로세싱 과정에서도 반복해서 진행하였습니다.

두번째 단계 : 탐색적 분석(EDA)

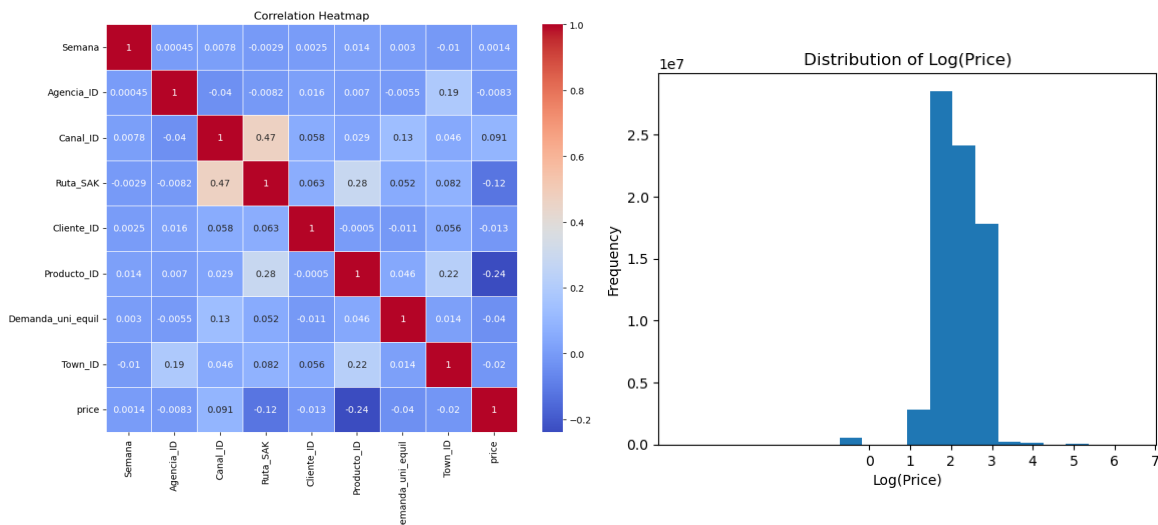


그림 4 : 피쳐 간 상관관계 히트맵

그림 5 : 계산된 가격 특성의 로그 변환을 통한 왜도 보정 시각화

1. 피쳐 간 상관관계를 살펴보았으나, 대부분의 특성에서 낮은 상관관계를 보였습니다. 예측을 위해 보다 필요한 특성을 만들어낼 필요를 확인하였습니다.
2. price 피쳐를 앞서 생성하였습니다. 하지만 결측치가 다수 존재하였습니다. 기존 분포는 치우침이 커, 로그 변환을 통해 정규분포에 가깝게 변환하였습니다. 이로써, 평균값으로 결측치를 채울 수 있었습니다.
3. 이 외에 수치형 데이터를 가진 두 특성[수요량, 가격]에 대한 두드러진 이상치를 박스플롯 차트로 확인하였고, 해당 데이터를 검토한 후 제거하였습니다.

세번째 단계 : 모델링

우선 첫번째 버전부터 사용할 알고리즘은 **LightGBM**을 통해 회귀 분석을 수행하였습니다.

LightGBM은 무거운 데이터를 사용시에 다른 트리계열 알고리즘에 비해 속도가 빠르면서도 성능을 잃지 않아 가장 우선순위로 고려하였습니다. 또한, 결측치를 자체적으로 처리할 수 있고, 이상치에 민감하지 않아 우선적으로 사용하기에 적합하다고 생각하였습니다. 대부분이 ID와 같은 범주형 데이터로 선형적 특성이 크지 않다고 판단하였습니다. 따라서 선형회귀 계열을 배제하고 트리계열 알고리즘 중에 선택하고자 하였습니다.

수행 과정에서 기존 RMSE 평가 함수를 사용할 수 없어 직접 평가 함수를 작성하여 사용하였습니다.

```
score = rmsle(y_val, y_pred)
print("Score:", score)
```

Score: 0.7206183202331504

무작위 추출로 분리한 훈련/검증 데이터 셋을 통한 최초 스코어는 0.72로 저조한 성과를 나타냈습니다. 이는 경진 대회 약 1700팀 중 1500위 내외의 순위에 해당합니다.

모델링 개선

모델링을 개선하기 위해 추가적인 과정은 다음과 같습니다.

1. 모델링 과정 수정

주차별 데이터가 큰 의미를 가지지 못하고 있습니다. 훈련 데이터와 테스트 데이터를 분리할 때, 주차별 특성을 보다 잘 반영할 수 있도록 분리합니다.

1~9주차 데이터를 통해 10,11주차 데이터를 예측하는 목적에 맞게, 훈련데이터를 1~7주차 데이터로, 검증데이터를 8,9주차 데이터로 나누어 학습합니다.

추가 피쳐 엔지니어링에서도 주차별 특성을 보다 잘 반영할 수 있도록 수정합니다.

2. 추가 피쳐 엔지니어링

2.1. 다른 변수에 대한 타겟 변수의 평균값을 변수로 추가

- 제품별 평균 수요량, 고객별 평균 수요량, 도시-제품별 평균 수요량, 고객-제품별 평균 수요량을 변수로 추가합니다.

2.2. 시간적 순서를 고려한 시차 특성 추가

- 주차별 특성을 반영하기 위해, 주차별 특성을 추가합니다.
- 2주 전 제품별 평균 수요량, 2주전 고객-제품별 평균 수요량을 변수로 추가합니다.

3. 하이퍼 파라미터 튜닝

최적 하이퍼 파라미터를 탐색합니다.

이를 위해 직접 훈련/검증 데이터 셋을 분리하였으며, 학습 성능 강화 및 시계열 특성을 보다 반영시키기 위해 6가지 특성을 생성하였습니다.

	Semana	Agencia_ID	Canal_ID	Ruta_SAK	Ciente_ID	Producto_ID	Demanda_uni_equil	Town_ID	price	Product_Mean_Demand	Product_Mean_Demand_2w	Client_Mean_Demand	Town_Product_Mean_Demand	Client_Product_Mean_Demand	Client_Product_Mean_Demand_2w
0	3	1110	7	3301	15766	1212	3	2008	2.125848	2.871166	2.804188	5.525	6.521588	3.400000	NaN
1	3	1110	7	3301	15766	1216	4	2008	2.125848	3.056116	3.023282	5.525	4.782525	2.833333	1.5
2	3	1110	7	3301	15766	1238	4	2008	2.285439	3.175030	3.105704	5.525	6.558471	2.428571	2.0
3	3	1110	7	3301	15766	1240	4	2008	2.125848	5.668002	5.621343	5.525	11.204265	4.600000	8.0
4	3	1110	7	3301	15766	1242	3	2008	2.033398	5.083143	5.142427	5.525	7.031367	2.142857	2.5

그림 6 : 피쳐 엔지니어링 이후 train 테이블

```
C:\Users\333cu\AppData\Local\Temp\ipykernel_8264\1056675118.py:10: RuntimeWarning: invalid value encountered in log1p
    return np.sqrt(np.mean(np.square(np.log1p(y_pred) - np.log1p(y_true))))
Score: 0.6188782864453775
```

그림 7 : 피쳐 엔지니어링을 통한 개선된 스코어

하이퍼 파라미터 튜닝의 경우 optuna 라이브러리를 이용하여 최적화를 시도하였습니다.

하지만, 메모리 부족 이슈 발생하여 csv데이터로 내보내고 새로 불러오는 방식으로 해결을 시도하였습니다.

메모리 이슈는 해결하였으나, 피쳐 엔지니어링을 통한 추가로 인해 데이터 테이블의 크기가 커졌고, 최적화 시간이 지나치게 길어져서 진행이 불가능하였습니다.

시도 횟수를 10까지 줄여 보았으나, 도출된 하이퍼 파라미터는 오히려 성능을 더 감소시켰습니다.

따라서, 기존 모델링에 사용한 기본 하이퍼 파라미터를 일단 사용하였습니다.

최종 모델링 결과

전체 train 데이터로 다시 학습하고, test 데이터에 예측한 결과를 반영해 제출하였습니다.

```
test['id'] = test.index
test = test[['id', 'Demanda_uni_equil']]
test.head()
```

	id	Demanda_uni_equil
0	0	3.987271
1	1	1.180406
2	2	2.077539
3	3	1.180406
4	4	1.180406

```
test.to_csv('submission.csv', index=False)
```


Submission and Description		Private Score	Public Score	Selected
 submission.csv	Complete (after deadline) · 3m ago · version 1	0.61814	0.56933	<input type="checkbox"/>

그림 8 : 모델링 결과 제출 및 성과

평가 스코어는 개선이 있었습니다. (0.72 → 0.62 / 0.57)

하지만, 여전히 리더보드 기준 1900명 중 1300등에 불과합니다.

결론

스코어 개선을 위해 각 피처에 대해 좀 더 검토해보고 추가 피처 엔지니어링을 통해 성능을 개선할 필요가 있습니다.

최신 알고리즘과 복잡한 알고리즘이 항상 더 나은 성과를 보장하지 않는 것을 확인하였습니다.

향후 업데이트에서 반영할 몇 가지 개선사항은 다음과 같습니다 :

피처 엔지니어링

1. Cliente_ID 테이블의 텍스트 데이터를 보다 면밀히 조사하여 고객에 대한 카테고리를 생성
 - 월마트와 같은 대형마트 부터, 프랜차이즈 식당, 학교, 중소 브랜드 슈퍼마켓, Bimbo 직매장 등으로 카테고리화를 시도
2. 보다 시계열 특성을 반영할 수 있도록 주차별 수요량을 계산하여 추가
3. 텍스트 트리밍 과정에서 결측치로 인해 포기한 데이터들을 피처로 활용할 방법 모색(무게, 한 개당 조각 수 등)

알고리즘

1. 회귀 분석 부터 여러 알고리즘을 비교하여 선택
2. 앙상블 기법을 통한 성능 개선
3. 샘플링을 통해 더 빈번하고 빠른 속도로 모델링 반복 수행