Online Retail

코호트, 잔존률, RFM, 군집 분석

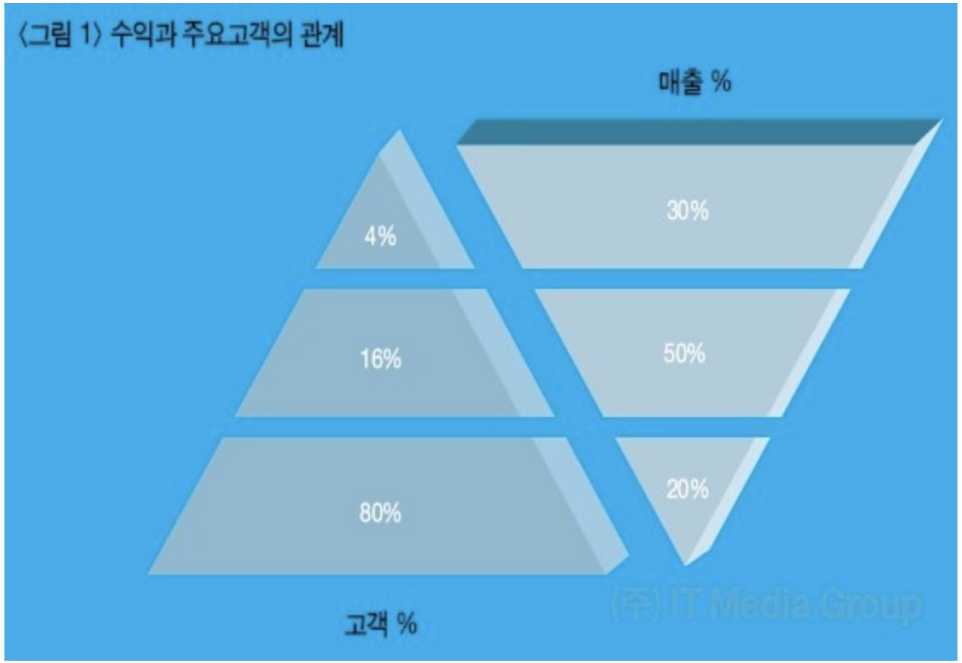
@ 멋쟁이사자처럼

Machine Learning Examples

Retail	Marketing	Healthcare	Telco	Finance
• Demand forecasting • Supply chain optimization • Pricing optimization • Market segmentation and targeting • Recommendations	• Recommendation engines & targeting • Customer 360 • Click-stream analysis • Social media analysis • Ad optimization	• Predicting Patient Disease Risk • Diagnostics and Alerts • Fraud	• Customer churn • System log analysis • Anomaly detection • Preventative maintenance • Smart meter analysis	• Risk Analytics • Customer 360 • Fraud • Credit scoring

https://www.bigdatavietnam.org/2017/11/demystifying-ai-machine-learning-and.html

왜 고객의 가치를 평가해야 할까요?



▲ <그림 1> 수익과 주요고객의 관계

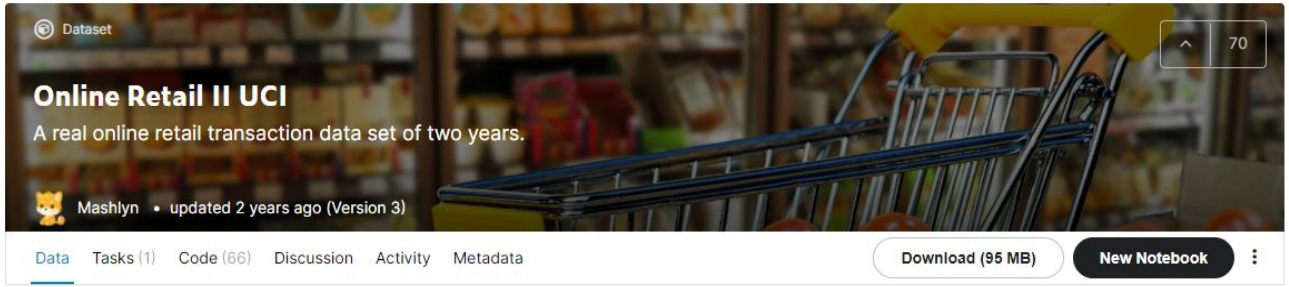
"기업의 첫 번째 과업은 고객창출이다."
피터 드러커(Peter Drucker)

✔따라서 고객확보와 유지가 매우 중요!
✔모든 고객이 동일한 가치를 갖는가?

💡 한정된 자원을 가지고 효율을 극대화하기위해 고객들의 가치를 평가, 차별화된 마케팅 전략 수립

"상위 20%의 고객이
전체 매출의 80% 를 차지한다"
파레토 법칙(Pareto's law)

http://www.itdaily.kr/news/articleView.html?idxno=8371



Usability 10.0 License CC0: Public Domain Tags business, education, economics

Description

Context

This Online Retail II data set contains all the transactions occurring for a UK-based and registered, non-store online retail between 01/12/2009 and 09/12/2011.The company mainly sells unique all-occasion gift-ware. Many customers of the company are wholesalers.

Content

Attribute Information:

InvoiceNo: Invoice number. Nominal. A 6-digit integral number uniquely assigned to each transaction. If this code starts with the letter 'c', it indicates a cancellation.
StockCode: Product (item) code. Nominal. A 5-digit integral number uniquely assigned to each distinct product.
Description: Product (item) name. Nominal.
Quantity: The quantities of each product (item) per transaction. Numeric.
InvoiceDate: Invice date and time. Numeric. The day and time when a transaction was generated.
UnitPrice: Unit price. Numeric. Product price per unit in sterling (Â£).
CustomerID: Customer number. Nominal. A 5-digit integral number uniquely assigned to each customer.
Country: Country name. Nominal. The name of the country where a customer resides.

https://www.kaggle.com/mashlyn/online-retail-ii-uci

📗문제유형: 회귀, 분류, 군집화
📘캐글유형: 데이터셋
📙데이터유형: 정형

📝영국에 기반을 두고 등록된 비매장 온라인 소매에 대해 발생한 모든 거래 데이터

💾 Data Explorer (94.85 MB)
online_retail_II.csv

Online Retail Data Set

Download: Data Folder, Data Set Description

Abstract: This is a transnational data set which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail.

Data Set Characteristics:	Multivariate, Sequential, Time-Series	**Number of Instances:**	541909	**Area:**	Business
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	8	**Date Donated**	2015-11-06
Associated Tasks:	Classification, Clustering	**Missing Values?**	N/A	**Number of Web Hits:**	648634

Online Retail II Data Set

Download: Data Folder, Data Set Description

Abstract: A real online retail transaction data set of two years.

Data Set Characteristics:	Multivariate, Sequential, Time-Series, Text	**Number of Instances:**	1067371	**Area:**	Business
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	8	**Date Donated**	2019-09-21
Associated Tasks:	Classification, Regression, Clustering	**Missing Values?**	Yes	**Number of Web Hits:**	128505

https://archive.ics.uci.edu/ml/datasets/online+retail

📝캐글에 올라온 해당 데이터의 출처는 대부분 UCI Repository로 관련 논문을 참고해 볼 수 있음

📝Online Retail, Online Retail2가 있음

📝Online Retail2의 데이터는 2년간의 데이터를 제공

코호트, 잔존률 분석

코호트 분석(Cohort analysis)

- 코호트 분석은 분석 전에 데이터 세트 의 데이터 를 관련 그룹으로 나누는 일종의 행동 분석
- 이러한 그룹 또는 집단은 일반적으로 정의된 시간 범위 내에서 공통된 특성이나 경험을 공유
- 코호트 분석을 통해 회사는
  - "고객이 겪는 자연적 주기를 고려하지 않고 맹목적으로 모든 고객을 분할하는 대신 고객(또는 사용자)의 수명 주기 전반에 걸쳐 패턴을 명확하게 볼 수 있습니다."
- 이러한 시간 패턴을 보고 회사는 특정 집단에 맞게 서비스를 조정 가능

	0일	1일	2일	3일	4일	5일	6일	7일
모든 사용자	100.00%	2.22%	2.50%	0.96%	0.00%	0.00%	0.00%	0.00%
2020. 9. 1.	100.00%	4.35%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
2020. 9. 2.	100.00%	3.70%	3.70%	0.00%	0.00%	0.00%	0.00%	
2020. 9. 3.	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%		
2020. 9. 4.	100.00%	0.00%	5.00%	5.00%	0.00%			
2020. 9. 5.	100.00%	5.26%	5.26%	0.00%				
2020. 9. 6.	100.00%	0.00%	0.00%					
2020. 9. 7.	100.00%	0.00%						

✔동질 집단 분류 기준: 획득날짜
✔잔존율(Retention Rate): 9월4일 사용자가 가장 높다.

https://en.wikipedia.org/wiki/Cohort_analysis

코호트 분석의 유형

시간집단

시간 집단은 특정 기간 동안 제품이나 서비스에 가입한 고객입니다. 이러한 집단을 분석하면 고객이 회사의 제품이나 서비스를 사용하기 시작한 시점을 기준으로 고객의 행동이 나타납니다. 시간은 월별 또는 분기별 또는 매일일 수 있습니다.

✅ Online Retail에 적용

행동집단

행동 집단은 과거에 제품을 구매했거나 서비스에 가입한 고객입니다. 가입한 제품 또는 서비스 유형에 따라 고객을 그룹화합니다. 기본 서비스에 가입한 고객은 고급 서비스에 가입한 고객과 요구 사항이 다를 수 있습니다. 다양한 코호트의 요구 사항을 이해하면 비즈니스에서 특정 세그먼트에 대한 맞춤형 서비스 또는 제품을 설계하는 데 도움이 될 수 있습니다.

규모집단

규모 집단은 회사의 제품이나 서비스를 구매하는 다양한 규모의 고객을 나타냅니다. 이 분류는 획득 후 특정 기간의 지출 금액 또는 고객이 주어진 기간 동안 주문 금액의 대부분을 지출한 제품 유형을 기반으로 할 수 있습니다.

https://thecleverprogrammer.com/2021/01/08/cohort-analysis-with-python/

잔존율 분석(Retention rate analysis)

- 리텐션 분석은 고객이 이탈하는 방법과 이유를 이해하기 위해 사용자 메트릭을 분석하는 과정
- 유지 분석은 유지 및 신규 사용자 확보율을 개선하여 수익성 있는 고객 기반을 유지방법 확보
- 일관된 유지 분석을 실행하여 알 수 있는 항목
  - 고객이 이탈하는 이유
  - 고객이 떠날 가능성이 더 높을 때
  - 이탈이 수익에 미치는 영향
  - 유지 전략을 개선하는 방법

https://www.profitwell.com/customer-retention/retention-analysis

데이터셋 소개

Attribute Information

- InvoiceNo: 송장번호. 해당 거래에 할당된 6자리 정수
  - 이 코드가 문자 'c'로 시작하면 취소를 나타냅니다.
- StockCode: 제품 코드. 각 고유 제품에 고유하게 할당된 5자리 정수
- Description: 제품 이름
- Quantity: 거래당 각 제품의 수량
  - 이 코드가 '-'(마이너스)로 시작하면 취소를 나타냅니다.
- InvoiceDate: 송장 날짜 및 시간. 숫자, 각 거래가 생성된 날짜 및 시간
- UnitPrice: 단가. 숫자, 스털링(영국 화폐) 단위의 제품 가격
- CustomerID: 고객 번호. 해당 고객에게 고유하게 할당된 5자리 정수
- Country: 국가 이름. 해당 고객이 거주하는 국가의 이름

https://archive.ics.uci.edu/ml/datasets/Online+Retail#

라이브러리 로드 및 폰트 설정

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt

%matplotlib inline
```

📌 pandas : 데이터분석 라이브러리로 대용량의 데이터를 처리하는데 매우 편리

📌 numpy : Numerical Python의 약자
파이썬의 고성능 과학 계산용 패키지로 Matrix와 Vector와 같은 Array 연산을 할 때 사용하며 pandas와 같이 표준 라이브러리처럼 사용

📌 matplotlib : numpy나 pandas에서 사용되는 자료 구조를 쉽게 시각화할 수 있음

📌 seaborn : Matplotlib을 기반으로 다양한 색상 테마와 통계용 차트 등의 기능을 추가한 시각화 패키지로 기본적인 시각화 기능은 Matplotlib 패키지에 의존하며 통계 기능은 Statsmodels 패키지에 의존

📌 datetime : 파이썬의 내장 함수로 날짜와 시간을 조작

```
def get_font_family():
    """
    시스템 환경에 따른 기본 폰트명을 반환하는 함수
    """
    import platform
    system_name = platform.system()
    # colab 사용자는 system_name이 'Linux'로 확인

    if system_name == "Darwin" :
        font_family = "AppleGothic"
    elif system_name == "Windows":
        font_family = "Malgun Gothic"
    else:
        # Linux
        !apt-get install fonts-nanum -qq  > /dev/null
        !fc-cache -fv

        import matplotlib as mpl
        mpl.font_manager._rebuild()
        findfont = mpl.font_manager.fontManager.findfont
        mpl.font_manager.findfont = findfont
        mpl.backends.backend_agg.findfont = findfont

        font_family = "NanumBarunGothic"
    return font_family

plt.rc("font", family=get_font_family())
plt.rc("axes", unicode_minus=False)
```

데이터 불러오기 및 요약

```
# pd.read_excel 로 데이터를 불러옵니다.
# 데이터의 용량이 커서 로드하는데 1분 이상 걸릴 수도 있습니다. 또 read_excel은 시간이 오래 걸립니다.
# csv로 로드하는 것이 훨씬 빠릅니다.
# df = pd.read_excel("http://archive.ics.uci.edu/ml/machine-learning-databases/00352/Online%20Retail.xlsx")
# df = pd.read_excel("data/Online Retail.xlsx")
df = pd.read_csv("data/online_retail.csv")
df.shape
```

```
(541909, 8)
```

📌 541909의 행과 8개의 열로 구성되어 있음

📌 8개의 변수가 존재

📌 Description, CustomerID에 결측치 존재

📌 데이터 타입은 float64, int64, object로 구성

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   InvoiceNo    541909 non-null  object 
 1   StockCode    541909 non-null  object 
 2   Description  540455 non-null  object 
 3   Quantity     541909 non-null  int64  
 4   InvoiceDate  541909 non-null  object 
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

데이터 미리보기

```
# 상단에 위치한 데이터
df.head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

```
# 하단에 위치한 데이터
df.tail()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	12680.0	France
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0	France
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09 12:50:00	4.15	12680.0	France
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0	France
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0	France

기술통계

```
# describe 를 통해 수치형 타입의 변수들의 기술통계를 구합니다.
df.describe()
```

	Quantity	UnitPrice	CustomerID
count	541909.000000	541909.000000	406829.000000
mean	9.552250	4.611114	15287.690570
std	218.081158	96.759853	1713.600303
min	-80995.000000	-11062.060000	12346.000000
25%	1.000000	1.250000	13953.000000
50%	3.000000	2.080000	15152.000000
75%	10.000000	4.130000	16791.000000
max	80995.000000	38970.000000	18287.000000

```
# describe 를 통해 범주형(np.object)타입의 변수들의 기술통계를 구합니다.
df.describe(include=np.object)
```

	InvoiceNo	StockCode	Description	InvoiceDate	Country
count	541909	541909	540455	541909	541909
unique	25900	4070	4223	23260	38
top	573585	85123A	WHITE HANGING HEART T-LIGHT HOLDER	2011-10-31 14:41:00	United Kingdom
freq	1114	2313	2369	1114	495478

결측치 계산 및 시각화

```
# 결측치 합계를 구합니다.
df.isnull().sum()
```

```
InvoiceNo          0
StockCode          0
Description     1454
Quantity           0
InvoiceDate        0
UnitPrice          0
CustomerID    135080
Country            0
dtype: int64
```

```
# 결측치 비율을 구합니다.
df.isnull().mean() * 100
```

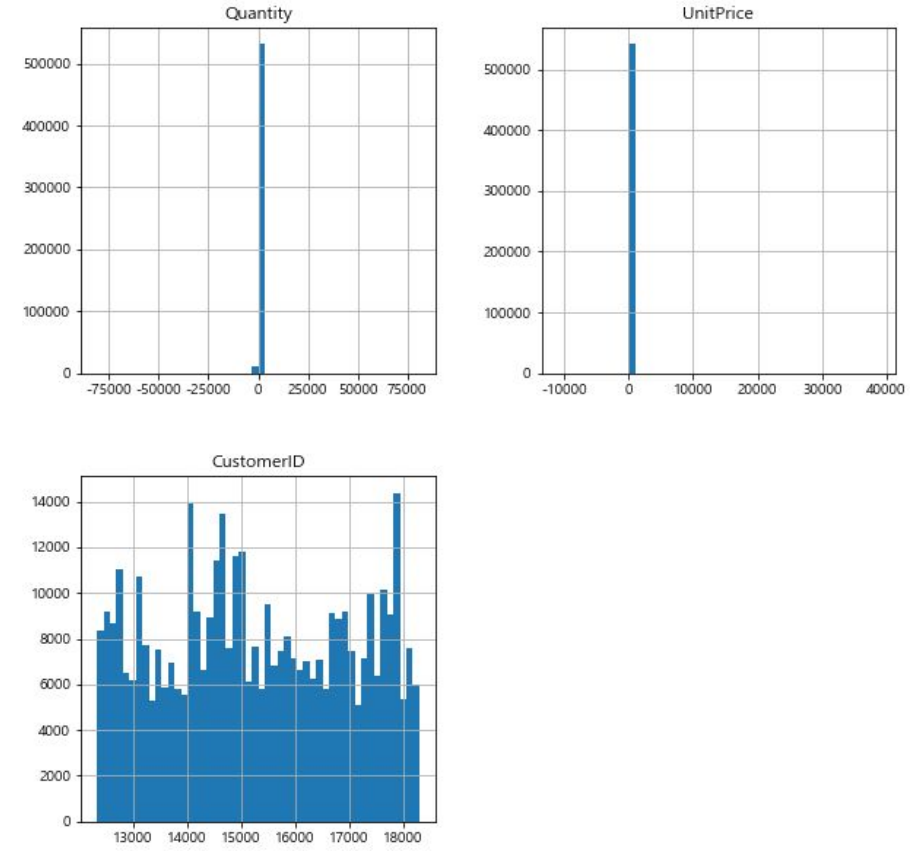
```
InvoiceNo      0.000000
StockCode      0.000000
Description    0.268311
Quantity       0.000000
InvoiceDate    0.000000
UnitPrice      0.000000
CustomerID    24.926694
Country        0.000000
dtype: float64
```

```
# 결측치를 시각화 합니다.
plt.figure(figsize=(12, 4))
sns.heatmap(df.isnull(), cmap="Greys_r")
```

```
<AxesSubplot:>
```

히스토그램으로 전체 수치형 변수 시각화

```
_ = df.hist(figsize=(10, 10), bins=50)
```

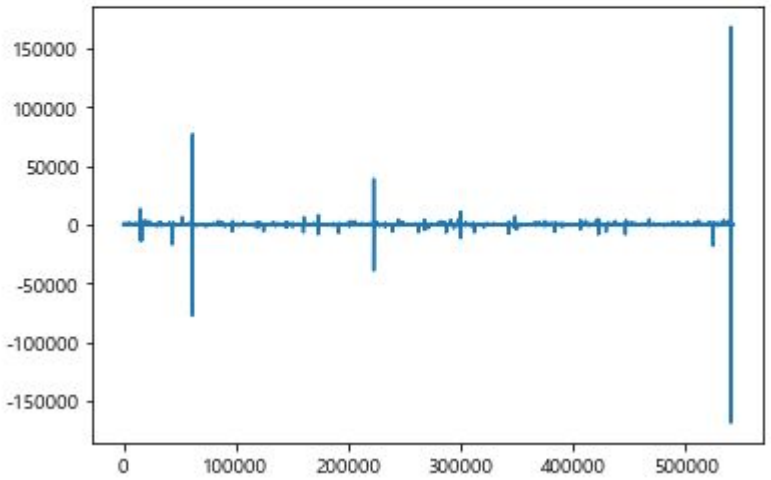


파생변수① 전체 주문 금액(TotalPrice)

```
df["TotalPrice"] = df["Quantity"] * df["UnitPrice"]
```

```
df["TotalPrice"].plot()
```

```
<AxesSubplot:>
```



✅ 전체 주문 금액(TotalPrice)

= 수량(Quantity) X 단가(UnitPrice)

✅ 전체 주문 금액의 분포를 확인하기 위해 그래프로 나타냄

❓ 전체 주문 금액이 마이너스 값이 나오는 이유는?

❓ 전체 주문 금액이 마이너스 값을 가지는 이유는?

```
df[df["TotalPrice"] < 0]
```
➔TotalPrice가 0보다 작은 데이터 추출

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	TotalPrice	id_null	Cancel	Invoi
141	C536379	D	Discount	-1	2010-12-01 09:41:00	27.50	14527.0	United Kingdom	-27.50	False	True	
154	C536383	35004C	SET OF 3 COLOURED FLYING DUCKS	-1	2010-12-01 09:49:00	4.65	15311.0	United Kingdom	-4.65	False	True	
235	C536391	22556	PLASTERS IN TIN CIRCUS PARADE	-12	2010-12-01 10:24:00							
236	C536391	21984	PACK OF 12 PINK PAISLEY TISSUES	-24	2010-12-01 10:24:00							
237	C536391	21983	PACK OF 12 BLUE PAISLEY TISSUES	-24	2010-12-01 10:24:00	0.29	17548.0	United Kingdom	-6.96	False	True	
...	...	...	...	...	...	...	...	...	...	...	...	...
540449	C581490	23144	ZINC T-LIGHT HOLDER STARS SMALL	-11	2011-12-09 09:57:00	0.83	14397.0	United Kingdom	-9.13	False	True	

수량(Quantity)이 마이너스라면,
이것을 통해 취소된 주문이란 것을
유추해 볼 수 있음

비회원 vs 회원 구매

```
# CustomerID값이 결측치인 값에 대한 Country값을 가져와 빈도수를 구합니다.
df.loc[df["CustomerID"].isnull(), "Country"].value_counts()
```

```
# CustomerID값이 결측치가 **아닌** 값에 대한 Country값을 가져와 빈도수를 구합니다.
df.loc[df["CustomerID"].notnull(), "Country"].value_counts()
```

파생변수 ② id_null

```
# 결측치 여부 컬럼 생성
df["id_null"] = df["CustomerID"].isnull()
```

```
# 비회원 구매 빈도수
df["id_null"].value_counts()
```

```
False    406829
True     135080
Name: id_null, dtype: int64
```

```
# 비회원 구매 비율
df["id_null"].value_counts()[1] / df["id_null"].value_counts()[0] *100
```

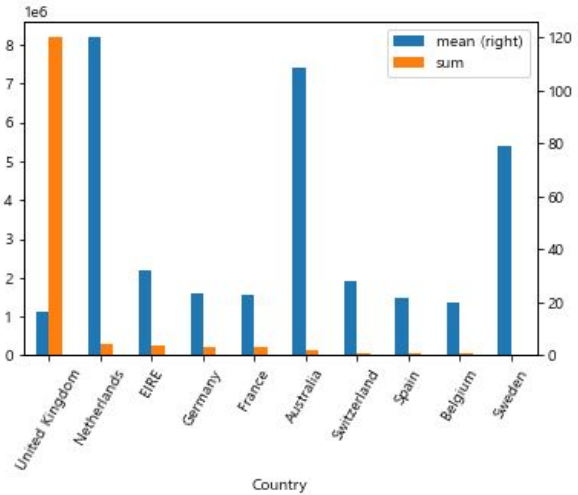
```
33.203139402549965
```

매출액 상위 국가

```
# TotalPrice를 통해 매출액 상위 10개 국가를 구합니다.
# 매출액의 평균과 합계를 구합니다.
top_sale_country = df.groupby(
    "Country")["TotalPrice"].agg(["mean", "sum"]).sort_values("sum", ascending=False).head(10)
top_sale_country.style.format("{:,}")
plt.show()
```

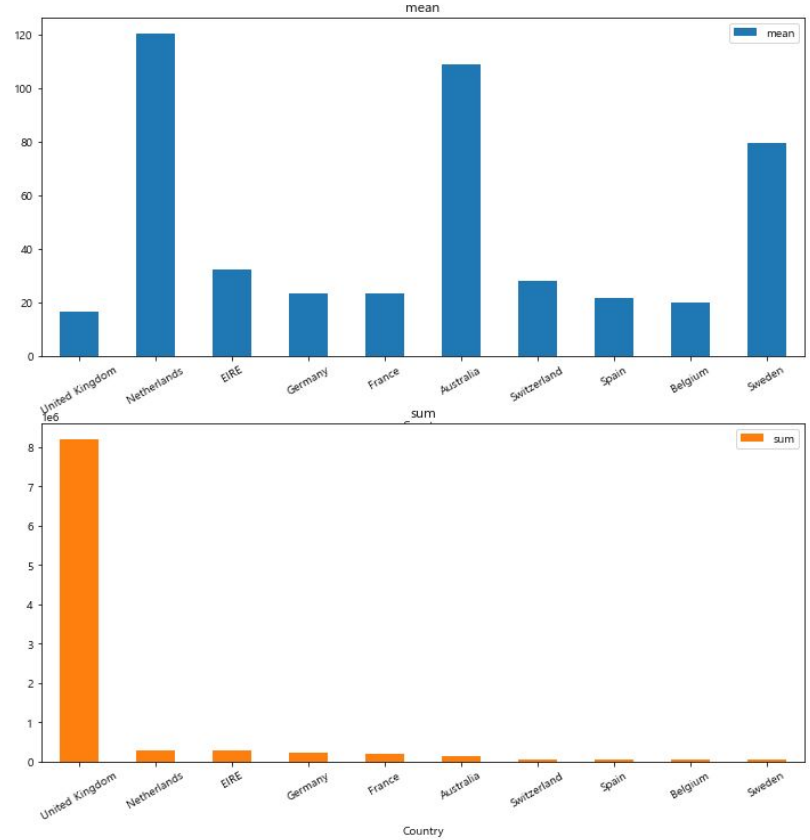
```
# 위에서 구한 결과를 barplot으로 시각화 합니다.
# 스케일이 다르기 때문에 2축 그래프를 사용하지만 두 가지 값을 비교하기에 적절해 보이진 않습니다.
top_sale_country.plot.bar(secondary_y="mean", rot=60)
```

```
<AxesSubplot:xlabel='Country'>
```



매출액 상위 국가

```
# 서브플롯으로 다시 시각화 해봅니다.
_ = top_sale_country.plot.bar(subplots=True, figsize=(12, 12), sharex=False, rot=30)
```



고객별 구매 취소 비율

```
# Quantity 가 0보다 작다면 Cancel로 True, False 값을 Cancel 컬럼으로 생성
df["Cancel"] = df["Quantity"] < 0
```

```
# CustomerID, Cancel로 그룹화 하여 InvoiceNo 의 빈도수를 구함
# 고객별 Cancel 빈도수를 구함
cancel_customer = df.groupby(["CustomerID", "Cancel"])["InvoiceNo"].count().unstack()
cancel_customer["cancel_ratio"] = (cancel_customer[True] / cancel_customer[False]) * 100
cancel_customer.sort_values(True, ascending=False).head(15)
cancel_customer.sample(10)
```

Cancel	False	True	cancel_ratio
CustomerID			
12453.0	43.0	NaN	NaN
14039.0	6.0	NaN	NaN
15664.0	17.0	2.0	11.764706
15506.0	164.0	NaN	NaN
17291.0	1.0	NaN	NaN
12736.0	4.0	NaN	NaN
14264.0	17.0	2.0	11.764706
14125.0	167.0	NaN	NaN
16401.0	186.0	12.0	6.451613
16995.0	NaN	1.0	NaN

특정 고객의 구매 건 조회

```
# 취소 건은 InvoiceNo 에 C를 붙이고 Quantity 를 마이너스로 표기하는 것을 확인
df[df["CustomerID"] == 14311.0]
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	TotalPrice	id_null	Cancel
152427	549592	79000	MOROCCAN TEA GLASS	12	2011-04-11 10:14:00	0.85	14311.0	United Kingdom	10.20	False	False
152428	549592	47566	PARTY BUNTING	4	2011-04-11 10:14:00	4.95	14311.0	United Kingdom	19.80	False	False
152429	549592	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2011-04-11 10:14:00	2.95	14311.0	United Kingdom	17.70	False	False
152430	549592	22993	SET OF 4 PANTRY JELLY MOULDS	12	2011-04-11 10:14:00	1.25	14311.0	United Kingdom	15.00	False	False
152431	549592	22829	SWEETHEART WIRE WALL TIDY	2	2011-04-11 10:14:00	9.95	14311.0	United Kingdom	19.90	False	False
152432	549592	21318	GLASS CHALICE BLUE SMALL	36	2011-04-11 10:14:00	0.39	14311.0	United Kingdom	14.04	False	False
152433	549592	22221	CAKE STAND LOVEBIRD 2 TIER PINK	2	2011-04-11 10:14:00	9.95	14311.0	United Kingdom	19.90	False	False
152434	549592	84968A	SET OF 16 VINTAGE ROSE CUTLERY	1	2011-04-11 10:14:00	12.75	14311.0	United Kingdom	12.75	False	False
152435	549592	22766	PHOTO FRAME CORNICE	8	2011-04-11 10:14:00	2.95	14311.0	United Kingdom	23.60	False	False
152436	549592	21756	BATH BUILDING BLOCK WORD	3	2011-04-11 10:14:00	5.95	14311.0	United Kingdom	17.85	False	False
152437	549592	21205	MULTICOLOUR 3D BALLS GARLAND	6	2011-04-11 10:14:00	2.55	14311.0	United Kingdom	15.30	False	False
152438	549592	22219	LOVEBIRD HANGING DECORATION WHITE	12	2011-04-11 10:14:00	0.85	14311.0	United Kingdom	10.20	False	False
152439	549592	21441	BLUE BIRDHOUSE DECORATION	12	2011-04-11 10:14:00	0.85	14311.0	United Kingdom	10.20	False	False
152440	549592	84879	ASSORTED COLOUR BIRD ORNAMENT	8	2011-04-11 10:14:00	1.69	14311.0	United Kingdom	13.52	False	False

제품별 구매 취소 비율

```
# "StockCode", "Cancel" 로 그룹화 하고 "InvoiceNo" 로 빈도수를 구함
cancel_stock = df.groupby(["StockCode", "Cancel"])["InvoiceNo"].count().unstack()
cancel_stock["cancel_ratio"] = (cancel_stock[True] / cancel_stock[False]) * 100
cancel_stock.sort_values(True, ascending=False).head(10)
```

Cancel StockCode	False	True	cancel_ratio
M	327.0	244.0	74.617737
22423	2019.0	184.0	9.113422
POST	1130.0	126.0	11.150442
22960	1142.0	87.0	7.618214
D	NaN	77.0	NaN
22720	1401.0	76.0	5.424697
21232	843.0	61.0	7.236062
S	2.0	61.0	3050.000000
22699	1084.0	54.0	4.981550
22197	1426.0	50.0	3.506311

국가별 구매 취소 비율

```
# "Country", "Cancel" 로 그룹화 하고 "InvoiceNo" 로 빈도수를 구함
cancel_country = df.groupby(["Country", "Cancel"])["InvoiceNo"].count().unstack()
cancel_country["cancel_ratio"] = (cancel_country[True] / cancel_country[False]) * 100
cancel_country.sort_values(True, ascending=False).head(10)
```

Cancel	False	True	cancel_ratio
Country			
United Kingdom	486286.0	9192.0	1.890246
Germany	9042.0	453.0	5.009954
EIRE	7894.0	302.0	3.825690
France	8408.0	149.0	1.772122
USA	179.0	112.0	62.569832
Australia	1185.0	74.0	6.244726
Spain	2485.0	48.0	1.931590
Italy	758.0	45.0	5.936675
Belgium	2031.0	38.0	1.871000
Japan	321.0	37.0	11.526480

파생변수 ③ 날짜와 시간

```
df["InvoiceDate"] = pd.to_datetime(df["InvoiceDate"])
```

```
# year, month, day, dayofweek 를 구해서 파생변수로 생성합니다.
df["InvoiceYear"] = df["InvoiceDate"].dt.year
df["InvoiceMonth"] = df["InvoiceDate"].dt.month
df["InvoiceDay"] = df["InvoiceDate"].dt.day
df["InvoiceDow"] = df["InvoiceDate"].dt.dayofweek
```

```
# day_name() 을 통해 요일명을 생성할 수도 있습니다.
df["InvoiceDayname"] = df["InvoiceDate"].dt.day_name()
```

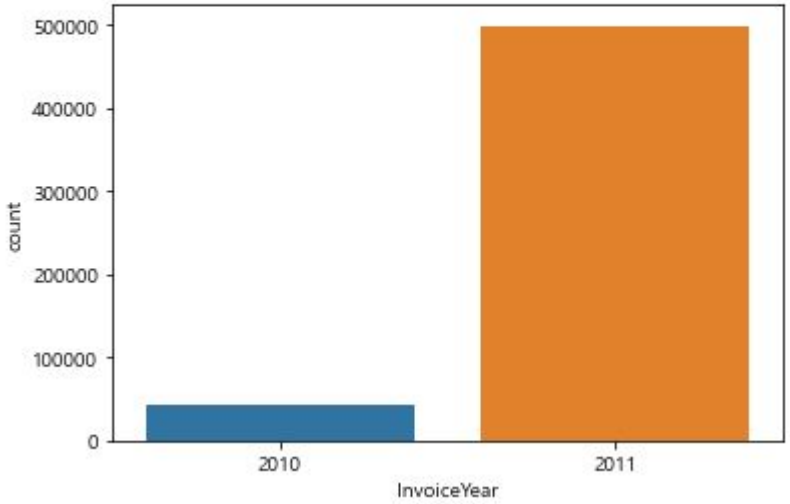
```
# 연도에서 앞에서 7개문자만 가져오면 연, 월만 따로 구할 수 있습니다.
df["InvoiceYM"] = df["InvoiceDate"].astype(str).str[:7]
```

```
# time, hour 에 대한 파생변수도 구합니다.
df["InvoiceTime"] = df["InvoiceDate"].dt.time
df["InvoiceHour"] = df["InvoiceDate"].dt.hour
```

파생변수 ③ 날짜와 시간 시각화 - 연도별 & 월별

```
# countplot으로 연도(InvoiceYear)별 빈도수 시각화
sns.countplot(data=df, x="InvoiceYear")
```

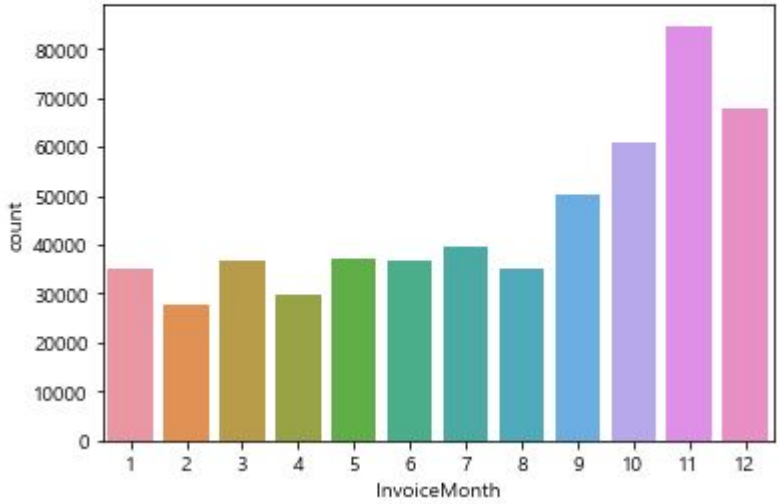
```
<AxesSubplot:xlabel='InvoiceYear', ylabel='count'>
```



✅ 2011년도의 주문량이 2010년도에 비해 월등히 많다.

```
# countplot으로 월(InvoiceMonth)별 빈도수 시각화
sns.countplot(data=df, x="InvoiceMonth")
```

```
<AxesSubplot:xlabel='InvoiceMonth', ylabel='count'>
```

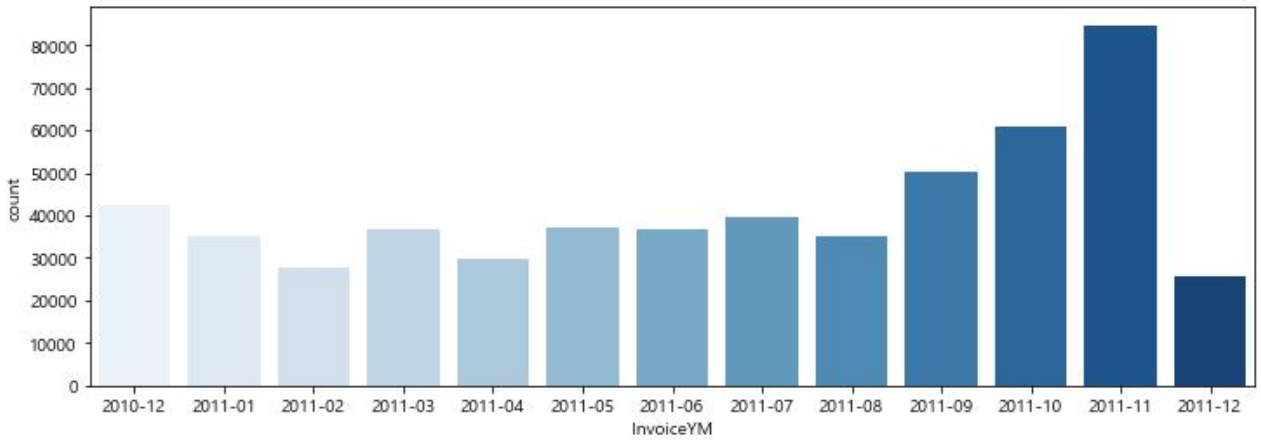


✅ 연중 주문량이 가장 많은 달은 11월이다.

파생변수 ③ 날짜와 시간 시각화 - 연도별 & 월별

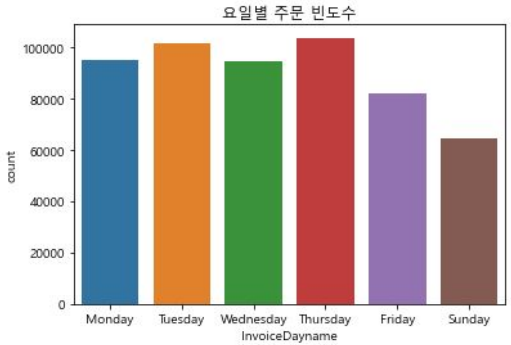
```
# countplot으로 연도-월별(InvoiceYM) 빈도수 시각화
plt.figure(figsize=(12, 4))
sns.countplot(data=df, x="InvoiceYM", palette="Blues")
```

```
<AxesSubplot:xlabel='InvoiceYM', ylabel='count'>
```

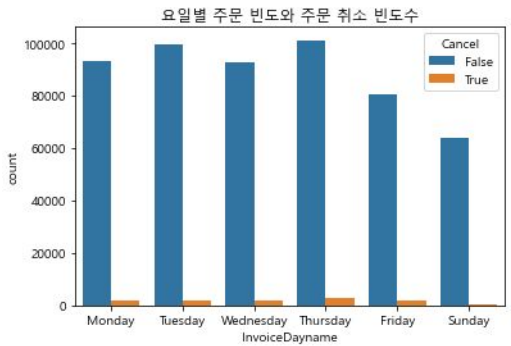


✅ 연중 가장 많은 주문이 발생한 달은 2011년 11월 이다.

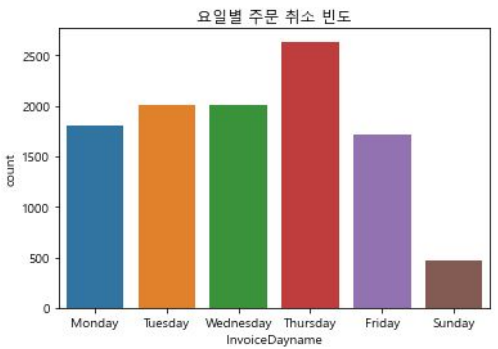
파생변수 ③ 날짜와 시간 시각화 - 요일별



✅ 주문이 가장 많은 요일은 목요일



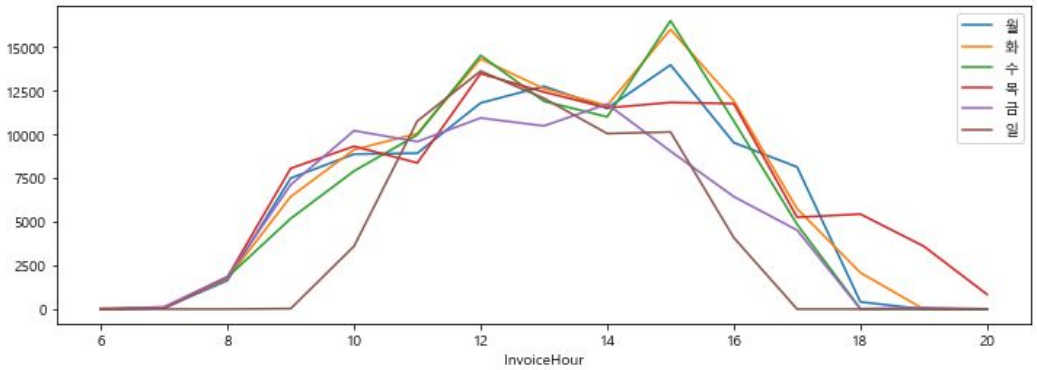
✅ 주문 빈도와 주문 취소 빈도가 가장 많은 요일은 목요일

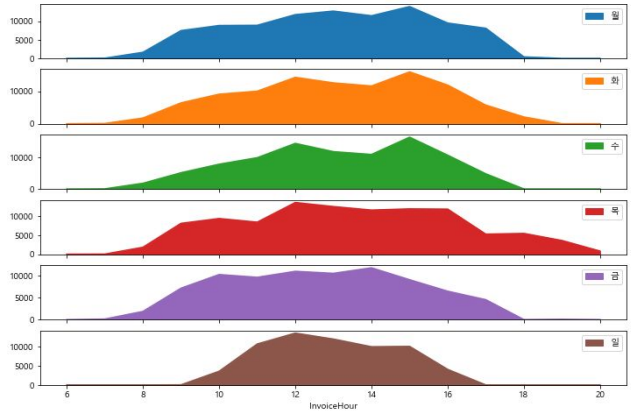


✅ 요일별 주문 취소 빈도는 목요일이 가장 많음

파생변수 ③ 날짜와 시간 시각화 - 시간별

InvoiceHour	월	화	수	목	금	일
6	0	0	0	41	0	0
7	61	65	59	64	134	0
8	1641	1751	1826	1834	1857	0
9	7497	6440	5186	8060	7119	30
10	8872	9125	7904	9321	10215	3600
11	8926	10052	9973	8369	9589	10765
12	11801	14327	14527	13485	10942	13627
13	12756	12619	11901	12428	10488	12067
14	11502	11656	11006	11518	11737	10052
15	13977	16017	16510	11830	9043	10142
16	9529	11924	10768	11765	6438	4092
17	8136	5746	4849	5257	4521	0
18	413	2086	23	5439	13	0
19	0	0	15	3601	89	0
20	0	0	18	845	8	0





✅ 요일들의 시간대 주문 빈도수를 구해 어떤 요일, 어떤 시간대에 구매가 많이 이뤄졌는지 각각의 그래프를 통해 살펴 볼 수 있음

코호트 월별 빈도수

```
# 함수를 통해 연도-월-1 로 일자를 모두 1로 만듭니다.
# 1일자로 만드는 이유는 월별 잔존률을 구하기 위함입니다.
def get_month_day1(x):
    return dt.datetime(x.year, x.month, 1)
```

```
# map을 통해 위에서 만든 함수를 적용합니다.
# InvoiceDate1 이라는 파생변수를 생성합니다.
df_valid["InvoiceDate1"] = df_valid["InvoiceDate"].map(get_month_day1)
```

```
# 고객별로 가장 첫 구매일을 구합니다.
# 1일자로 만든 구매일 - 첫구매일을 구하면 첫 구매 후 몇달 후 구매인지를 구할 수 있습니다.
df_valid.groupby('CustomerID')['InvoiceDate1'].transform('min')
```

```
0         2010-12-01
1         2010-12-01
2         2010-12-01
3         2010-12-01
4         2010-12-01
             ...
541904    2011-08-01
541905    2011-08-01
541906    2011-08-01
541907    2011-08-01
541908    2011-08-01
Name: InvoiceDate1, Length: 387488, dtype:
```

```
# 위에서 구한 첫 구매일인 InvoiceDate1의 최솟값을 구해 InvoiceDateMin 변수에 할당합니다.
df_valid["InvoiceDateMin"] = df_valid.groupby('CustomerID')['InvoiceDate1'].transform('min')
```

```
# 1일자로 만든 구매일 - 첫구매일을 구하면 첫 구매 후 몇달 후 구매인지를 구할 수 있습니다.
# 연도별 차이 - 월별 차이를 구합니다.
# 날짜는 모두 1일자 기준이기 때문에 이 계산으로 첫 구매로 부터 해당 구매 데이터가 몇 달 후 데이터인지를 알 수 있습니다.
year_diff = df_valid["InvoiceDate1"].dt.year - df_valid["InvoiceDateMin"].dt.year
month_diff = df_valid["InvoiceDate1"].dt.month - df_valid["InvoiceDateMin"].dt.month
```

```
# 구매 후 몇 달이 지났는지를 계산합니다.
# CohortIndex 변수를 생성합니다. 연도차이 * 12개월 + 월차이 + 1
df_valid["CohortIndex"] = year_diff * 12 + month_diff + 1
```

월별, 주문건, 주문제품 종류 수, 중복을 제외한 고객 수, 총 주문금액

```
df_valid.groupby('InvoiceMonth').agg({"InvoiceNo":"count",
                                      "StockCode":"nunique",
                                      "CustomerID":"nunique",
                                      "UnitPrice":"sum",
                                      "Quantity":"sum",
                                      "TotalPrice":"sum"
                                     }).style.format("{:,}")
```

InvoiceMonth	InvoiceNo	StockCode	CustomerID	UnitPrice	Quantity	TotalPrice
1	21,027	2,147	783	70,891.61	266,291	469,202.27
2	19,518	2,134	798	64,538.26	260,358	433,166.39
3	26,732	2,231	1,020	92,455.76	340,630	574,851.69
4	22,473	2,220	899	81,559.001	275,526	422,658.871
5	28,059	2,225	1,079	123,511.72	364,940	643,165.53
6	26,973	2,339	1,051	127,268.75	352,460	599,568.36
7	26,562	2,360	992	91,000.771	357,698	569,722.031
8	26,774	2,359	980	85,502.65	382,728	610,788.59
9	39,444	2,547	1,301	122,745.571	531,839	922,708.871
10	47,709	2,656	1,425	167,563.41	563,236	966,535.28
11	60,533	2,697	1,711	182,541.27	654,119	1,108,030.06
12	41,684	3,086	1,374	129,688.24	491,023	882,064.11

✅ InvoiceNo : 빈도수

✅ StockCode, CustomerID : 고유값(유일값)

✅ UnitPrice, Quantity, TotalPrice : 합계

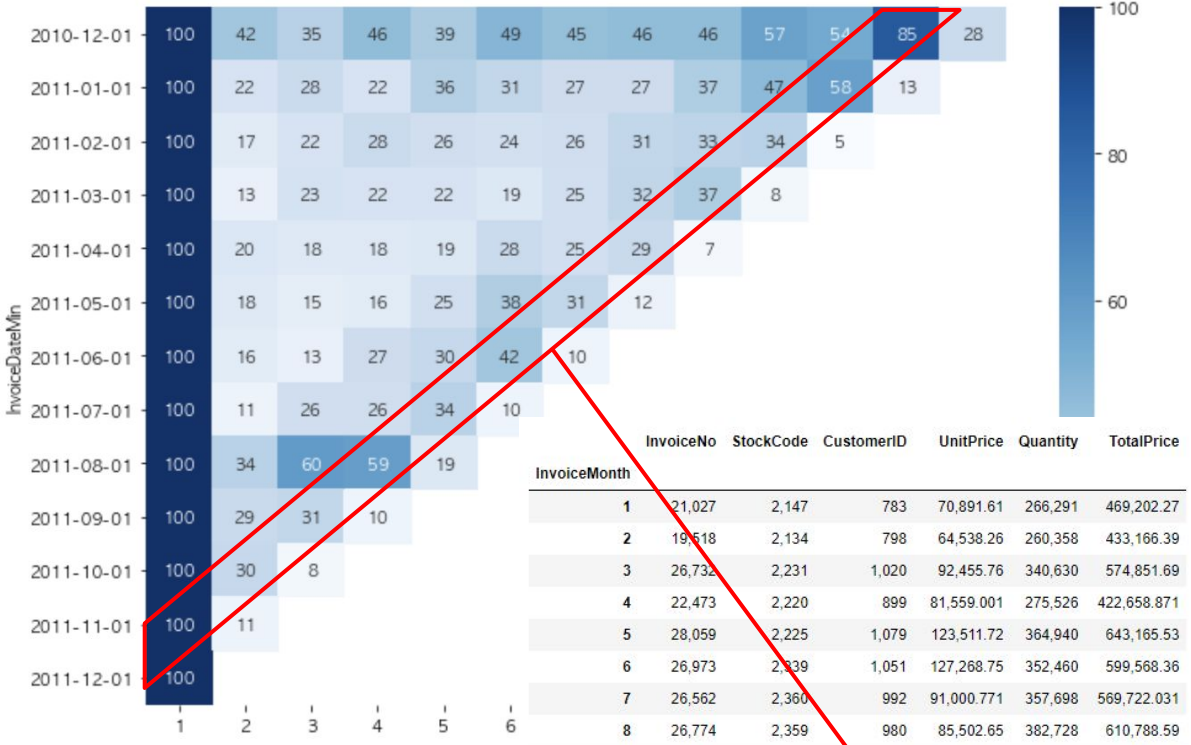
👉 각 변수들의 값을 agg(다중집계작업 multiple aggregation operation)메서드를 이용해 집계함

Online Retail 코호트 분석

📌 InvoiceDateMin : 월별 일자 중 첫 구매일의 최솟값

📌 CohortIndex : 첫 구매와 다음 구매의 경과 기간(월단위)

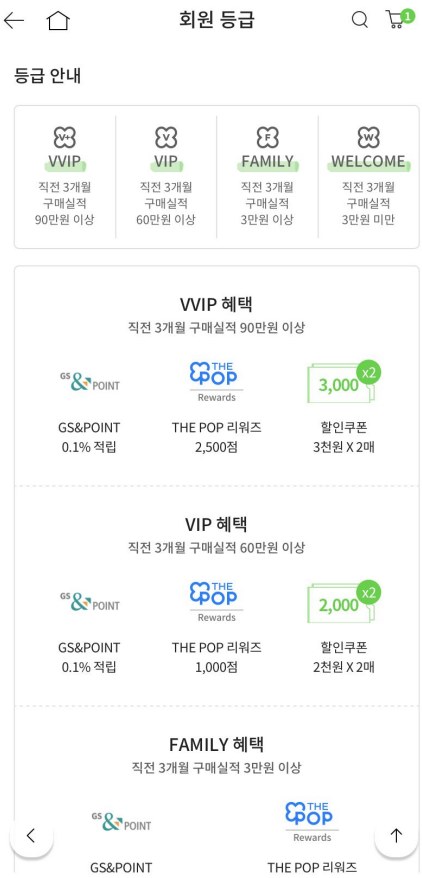
📝 Online Retail 코호트 분석 그래프를 해석해보면, 2010년 12월을 기준으로 CohortIndex가 1이면 당월의 구매빈도는 100%이고, 이것을 제외한 나머지들 중 가장 높은 85%가 해당하는 월을 유추해보면 11월이라는 것을 알 수 있습니다. 따라서, 다른 구매월의 행동 패턴들도 고려해보면, 고객마다 첫 구매일은 다르지만, 가장 많이 물건을 구매하는 월은 11월이라는 것을 알 수 있습니다.

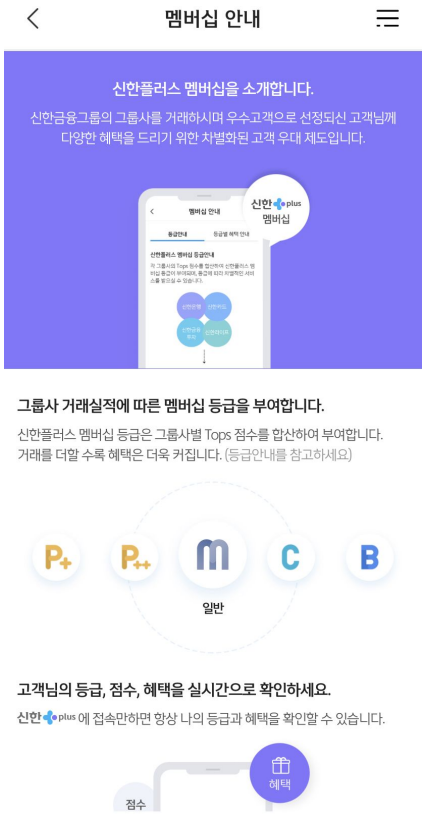


InvoiceMonth	InvoiceNo	StockCode	CustomerID	UnitPrice	Quantity	TotalPrice
1	21,027	2,147	783	70,891.61	266,291	469,202.27
2	19,518	2,134	798	64,538.26	260,358	433,166.39
3	26,732	2,231	1,020	92,455.76	340,630	574,851.69
4	22,473	2,220	899	81,559.001	275,526	422,658.871
5	28,059	2,225	1,079	123,511.72	364,940	643,165.53
6	26,973	2,239	1,051	127,268.75	352,460	599,568.36
7	26,562	2,360	992	91,000.771	357,698	569,722.031
8	26,774	2,359	980	85,502.65	382,728	610,788.59
9	39,444	2,547	1,301	122,745.571	531,839	922,708.871
10	47,709	2,656	1,425	167,563.41	563,236	966,535.28
11	60,533	2,697	1,711	182,541.27	654,119	1,108,030.06
12	41,684	3,086	1,374	129,688.24	491,023	882,064.11

RFM 분석

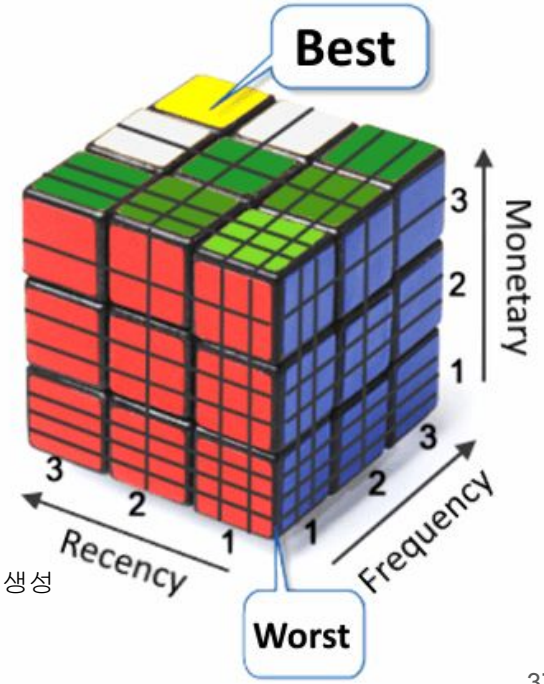
Customer Segmentation





RFM 분석(Recency, Frequency, Monetary analysis)

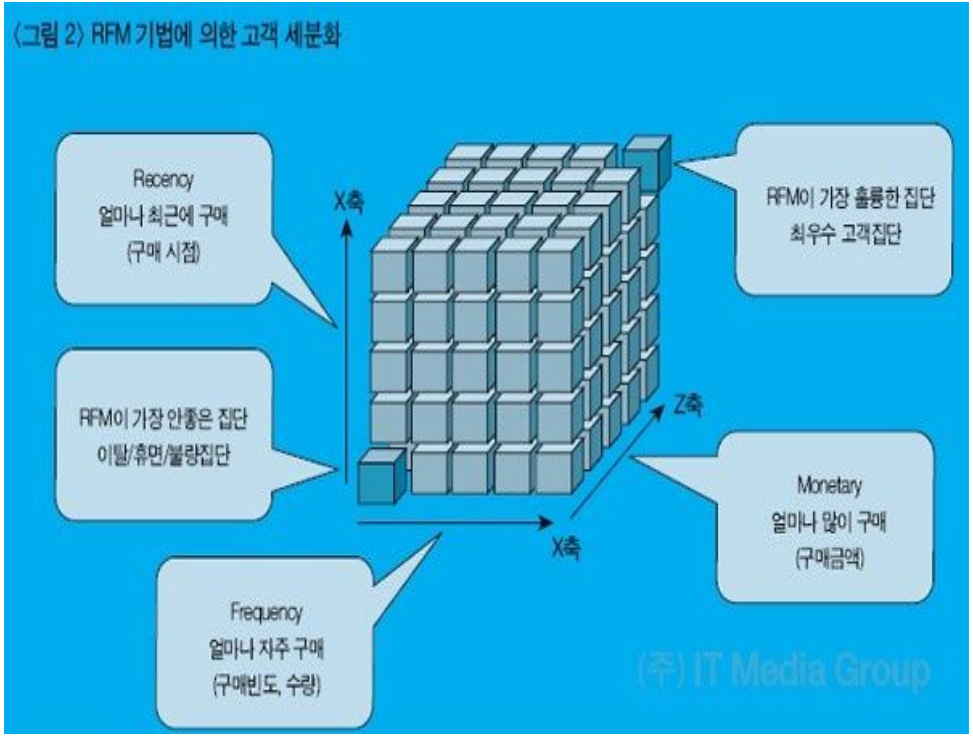
- RFM은 가치있는 고객을 추출해내어 이를 기준으로 고객을 분류할 수 있는 분석 방법
- 구매 가능성이 높은 고객을 선정하기 위한 데이터 분석방법
- RFM 분석에서 우리는 세 가지 지표 또는 차원에 따라 각 고객을 분석
  - Recency - 거래의 최근성: 고객이 얼마나 최근에 구입했는가?
  - Frequency - 거래빈도: 고객이 얼마나 빈번하게 우리 상품을 구입했나?
  - Monetary - 거래규모: 고객이 구입했던 총 금액은 어느 정도인가?
- 가장 큰 과제는 그룹의 경계를 정의하는 것



✔RFM 분석에 의한 고객 세분화의 3D 표현
✔그룹의 교차점이 우리의 고객세분화 기준
✔3가지 차원을 각각 4개의 그룹으로 나누면 64(4x4x4) 고객 세그먼트가 생성

https://ko.wikipedia.org/wiki/RFM
https://mindbox.cloud/blog/product/ml-driven-rfm-analysis/

RFM 기법에 의한 고객 세분화



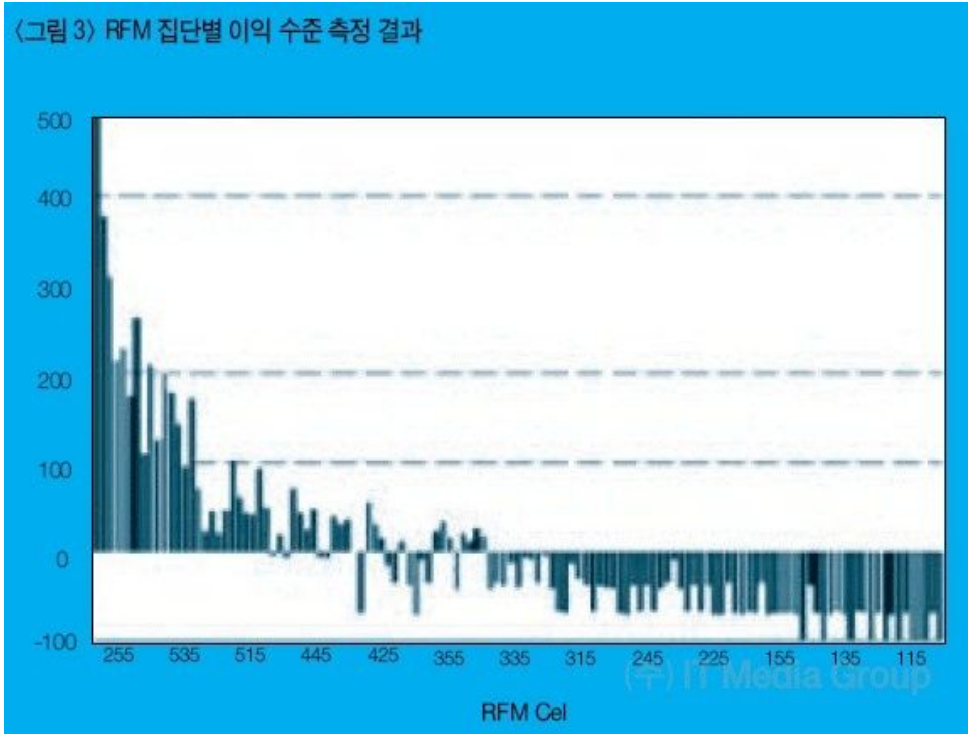
고객 세분화의 항목

✔ 판매정보: 상품판매 자료 거래 금액, 횟수
✔ 인구통계학적 정보: 나이, 성별, 직업, 학력,거주지역, 소득수준
✔ 라이프스타일 정보: 순차적, 구매 정보, RFM 정보
✔ 심리 정보: 구매욕구
✔ 행동 정보: 구매패턴 Life Time Value

💡 구매 행동별로 고객을 묶어 각 고객집단별로 차별화된 마케팅 전략을 수립

http://www.itdaily.kr/news/articleView.html?idxno=8371

RFM 집단별 이익 수준 측정 결과



고객 세분화의 예시

한 기업에서 RFM에 의하여 고객을 분류한 후에 각 고객들에게 DM을 발송하고 측정한 각 집단별 이익 수준을 구한 결과가 다음 <그림 3>과 같다.

✔ 가장 많은 이익을 가져다주는 집단은?
✔ 손해를 주는 집단의 특징은?
✔ 각 집단별 서열화는 어떻게 할것인가?(등급)

💡 RFM 모형 분석에서는 이와 같이 제 2의 분석을 통하여 각 집단의 중요도를 측정하고 그 결과를 미래의 마케팅에 활용하는 것이 RFM 모형의 목적

http://www.itdaily.kr/news/articleView.html?idxno=8371

Online Retail의 RFM

RFM_score	Recency mean	Frequency mean	MonetaryValue mean	MonetaryValue sum
3	278	7	138	37,309
4	204	11	200	52,138
5	183	16	295	108,729
6	126	20	371	142,014
7	103	26	898	345,746
8	87	36	628	227,483
9	70	46	858	309,676
10	59	62	1,123	392,016
11	45	80	1,445	487,016
12	35	108	1,794	592,058
13	23	140	3,080	973,379
14	16	230	4,797	1,467,897
15	5	439	11,596	3,583,278

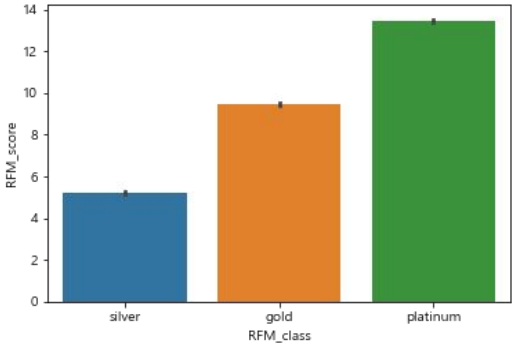
RFM_class	Recency mean	Frequency mean	MonetaryValue mean	MonetaryValue sum	MonetaryValue count
silver	170	17	411	685,935	1,668
gold	65	55	1,005	1,416,191	1,409
platinum	19	226	5,247	6,616,612	1,261

고객 세분화의 항목

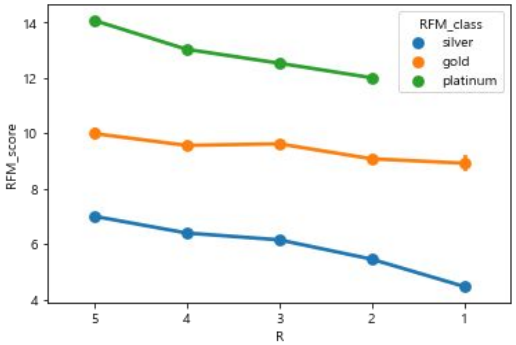
✔ Recency: 최근 구매일(구매일 - x.max()).days)
✨최근일수록 높은 스코어
✔ Frequency: 구매 빈도수(count)
✔ Monetary: 총 구매금액(sum)

💡 RFM_score을 기준으로 고객등급 설정
Silver, gold, platinum

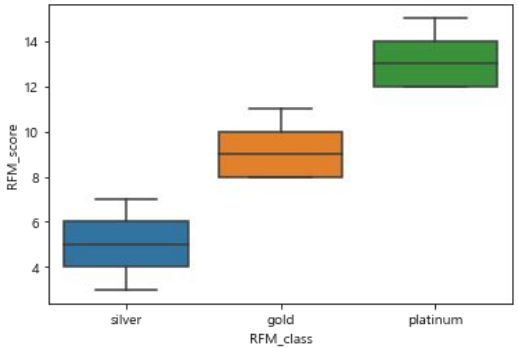
RFM을 이용한 그래프 시각화



✅ 막대그래프(bar)

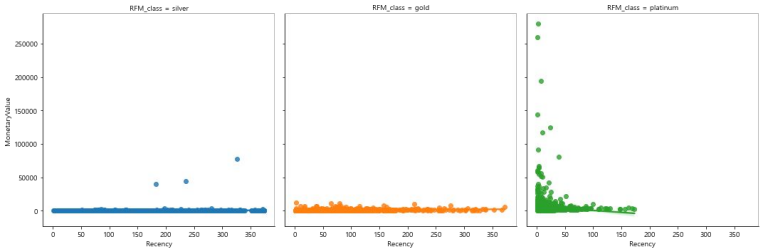


✅ 포인트 그래프(point)

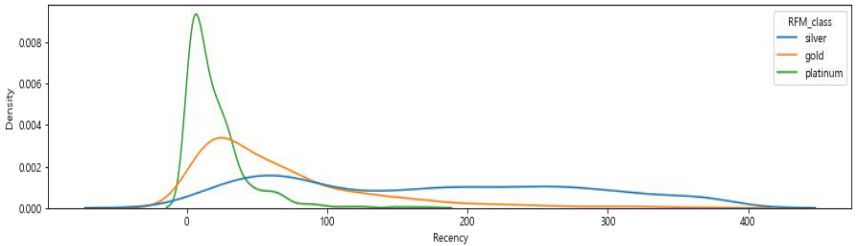


✅ 박스 그래프(box)

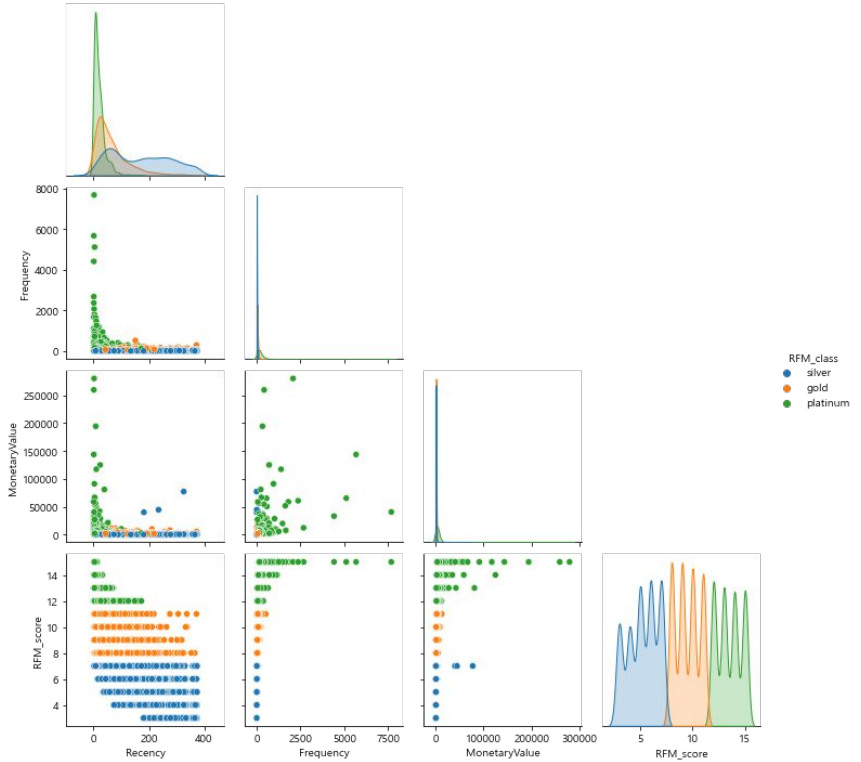
RFM을 이용한 그래프 시각화



✅ 회귀선·플롯 그래프(lm)



✅ 커널밀도 그래프(kde)



✅ 여러 변수간 산점도 그래프(pair)

머신러닝 비지도 학습 고객 군집화

Cluster analysis(군집화 분석)

👉 대표적인 비지도 학습 방법

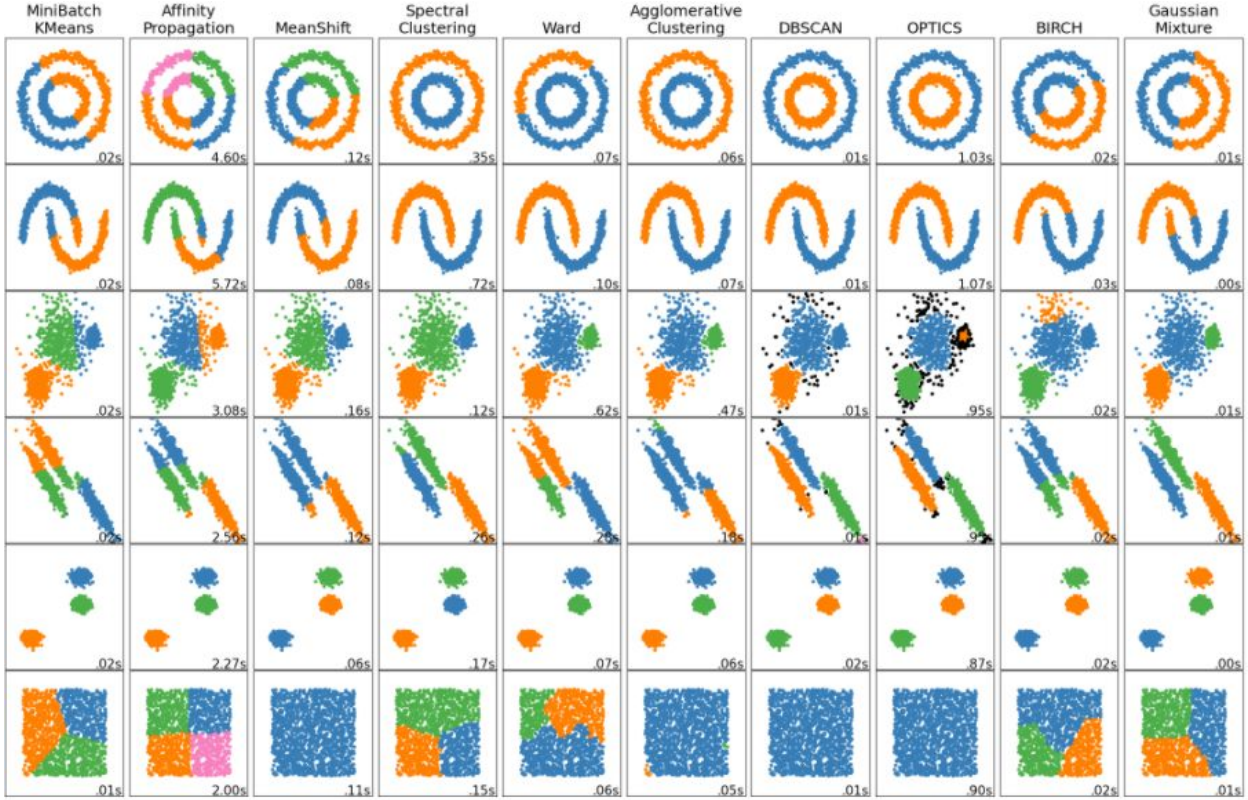
👉 Cluster analysis(군집화 분석)이란 주어진 데이터들의 특성을 고려해 데이터 집단(Cluster)을 정의하고 데이터 집단의 대표할 수 있는 대표점을 찾는 것으로 데이터 마이닝의 한 방법

👉 Cluster란 비슷한 특성을 가진 데이터들의 집단. 반대로 데이터의 특성이 다르면 다른 클러스터에 속해야 함

👉 효율성 : Cluster 분석을 통해 수백만의 데이터를 직접 확인하지 않고 각각 Cluster의 대표값만 확인해 전체 데이터의 특성을 파악할 수 있음

https://ko.wikipedia.org/wiki/%ED%81%B4%EB%9F%AC%EC%8A%A4%ED%84%B0_%EB%B6%84%EC%84%9D

scikit-learn의 클러스터링 알고리즘 비교



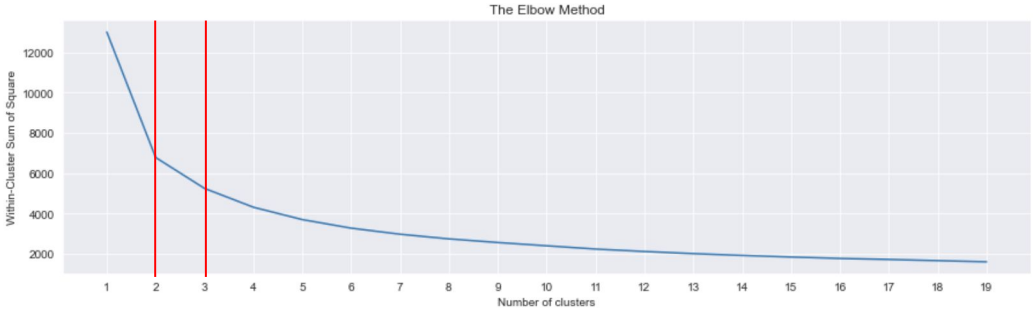
https://scikit-learn.org/stable/modules/clustering.html#clustering

scikit-learn의 클러스터링 알고리즘 비교

종류	매개변수	확장성	사례	알고리즘
K-Means	number of clusters	가장 큰 n_samples, 중간 n_clusters에 MiniBatch code	범용, 균일한 클러스터 크기, 평평한 기하학, 클러스터가 너무 많지 않음, 유도성	점 사이의 거리
Affinity propagation	damping, sample preference	n_samples로 확장 불가능	많은 클러스터, 고르지 않은 클러스터 크기, 평평하지 않은 기하학, 귀납적	그래프 거리(예: 가장 가까운 이웃 그래프)
Mean-shift	bandwidth	n_samples로 확장 불가능	많은 클러스터, 고르지 않은 클러스터 크기, 평평하지 않은 기하학, 귀납적	점 사이의 거리
Spectral clustering	number of clusters	중형 n_samples, 소형n_clusters	적은 수의 클러스터, 심지어 클러스터 크기, 평평하지 않은 기하학, 변형	그래프 거리(예: 가장 가까운 이웃 그래프)
DBSCAN	neighborhood size	매우 큰 n_samples, 중간n_clusters	평평하지 않은 기하학, 고르지 않은 클러스터 크기, 변형	가장 가까운 지점 간의 거리

https://scikit-learn.org/stable/modules/clustering.html#clustering

Elbow Method



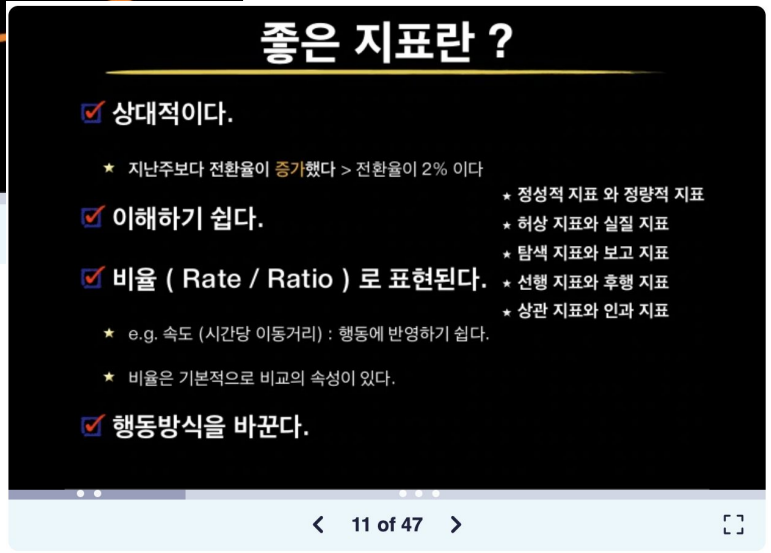
✅ 데이터 세트의 클러스터의 개수를 결정하는 단계

✅ Cluster 간의 거리의 합을 나타내는 inertia가 급격히 떨어지는 지점에서 K값을 군집의 개수로 사용

✅ 왼쪽의 그래프에서는 2 또는 3이 K값으로 적절해 보임

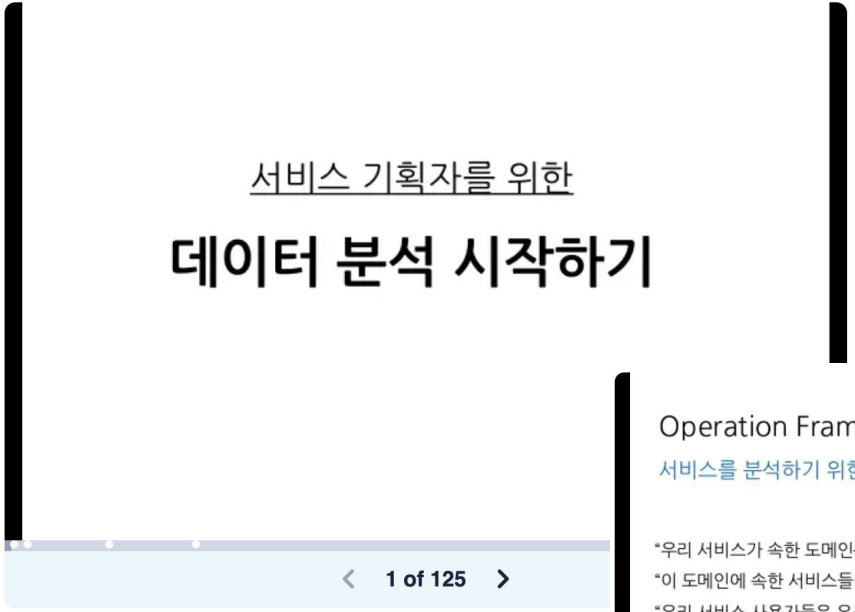
✅ Inertia_메서드를 이용하면, 더욱 정확한 K값을 구해 볼 수 있음

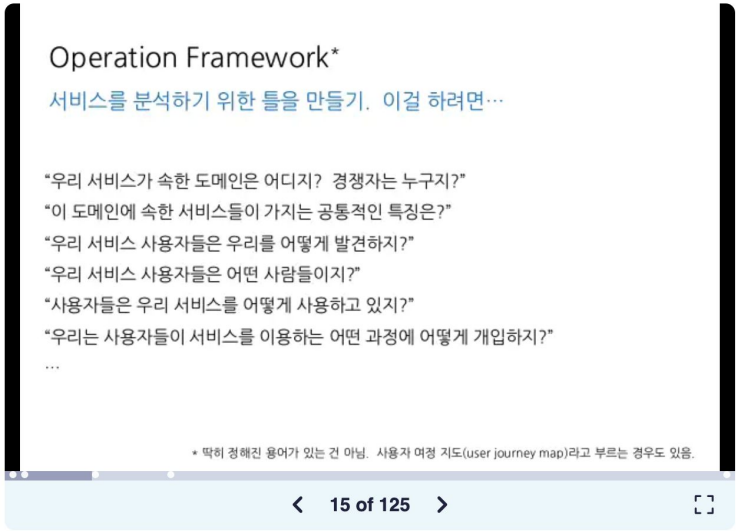
추천 자료



https://www.slideshare.net/xguru/with-lean-analytics-with-lezhin-comics

린분석 책 요약
린분석 책을 감수하신 분이
만든 슬라이드

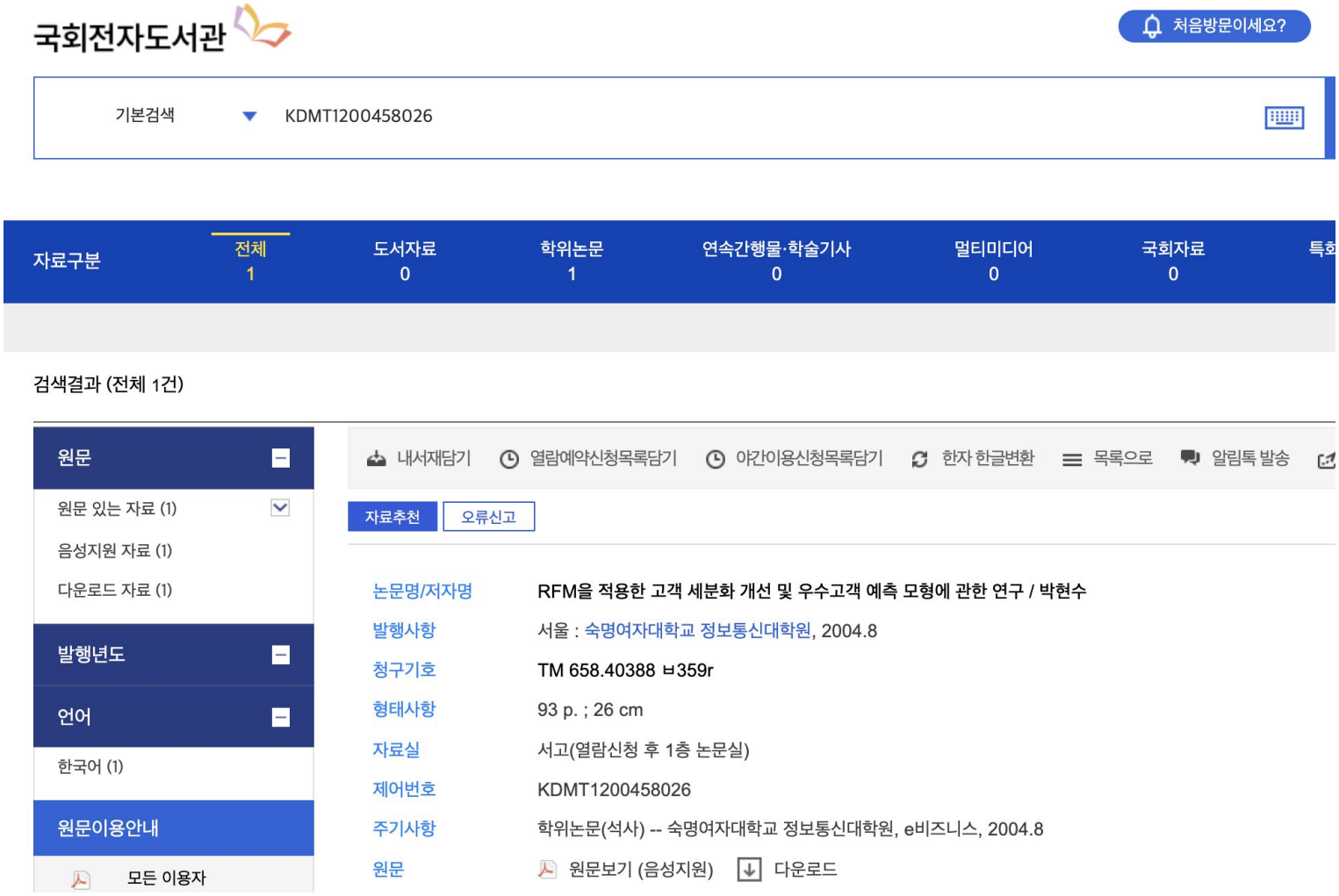




실제 스타트업이 어떻게 데이터분석을 하는지 가이드를 쉽게 알려주는 슬라이드

https://www.slideshare.net/leoyang991/ss-90038927

관련 논문



http://dl.nanet.go.kr/law/SearchDetailView.do?cn=KDMT1200458026#none

http://www.riss.kr/link?id=T11360772

RFM에서 등급부여 방법에 관한 연구

A study on proposing a method for grouping R, F, and M in RFM model

류귀열 (서경대학교 컴퓨터과학과) ; 문영수 isni (한국과학기술정보연구원)
Ryu, Gui-Yeol (Department of Computer Science, SeoKyeong University) ; Moon, Young-Soo isni (Department of AST

투고 : 2013.01.21 심사 : 2013.02.13 발행 : 2013.03.31

https://doi.org/10.7465/jkdi.2013.24.2.245 복사 인용 KSCI

초록

본 논문은 RFM (recency frequency monetary) 모델에서 등급을 매기는 방법을 정규분포를 이용하여 6등급모델을 제안하고 NDSL (national discovery for science leaders) 자료를 이용하여 현재 많이 사용되고 있는 5등급모델과 10등급모델을 비교하였다. 제안 모델이 5등급모델과 10등급모델에 비해 고객그룹들을 쉽게 세분화할 수 있다는 사실을 알 수 있었다. 제안된 모델은 대칭적으로 등급을 부여하고, 고객분포를 이용하기 때문에 고객특성을 잘 반영함으로써 경계값들이 명확하게 구분되는 특징을 가지고 있다. 또한 등급 값을 보면 쉽게 어느 위치에 속하고 있는 지 알 수 있으며, 고객세분화 후 고객의 RFM값들의 확인으로 고객의 특성을 쉽게 알 수 있는 장점이 있다. 향후 군집분석 등의 통계적 등급부여 방법들과 비교연구와 가중치 부여 방법에 관한 연구가 필요하다.

https://www.koreascience.or.kr/article/JAKO201311637858167.pdf