

QoE Data 를 이용한 이상감지

2019.01.15
인턴 김지희, 장예훈

Project Instruction

Project Instruction

프로젝트 목적

QoE Data 패턴 분석을 통해 주기적으로 발생하는 에러를 예측하고 분류하여 최종적으로 에러의 원인을 신속하게 파악할 수 있도록 함

프로젝트 기간

2018.11.28 - 2019.1.14

프로젝트 과정

1. QoE 및 데이터 분석 Tool 사전조사
2. Data 수집
3. Feature Engineering
4. 실험
 - a. Resampling
 - b. SMOTE
 - c. Decision Tree
 - d. Random Forest
5. 문서화 및 발표

QoE Data Description

QoE Data Description

Data Details

2018.11.06 00:00 - 2018.11.07 23:59 (2일)

s3 , qoe-dtv

Event 별 분류

QoE Event	설명	
PlaybackStarted	Playback session 이 시작될 때의 event	본 프로젝트에서 사용하지 않음
PlaybackUpated	Playback session이 시작되고 끝나기 전까지 1분 간격으로 발생하는 event	X , 독립변수
PlaybackEnded	Playback session 종료 시 발생하는 event . 정상 종료시 errorcode == 0, error 발생 시 12가지 종류의 errorCode 수반	X, errorCode : Y

QoE Data Description

Field Name	Playback Updated	Playback Ended
sessionId	O	O
bufferedDuration	O	
bufferings	O	
configurations		
contentUrl	O	O
currentPosition	O	
device	O	O
device.*	O	O
duration		
status		
timestamp	O	O

	Playback Updated	Playback Ended
errorCode		O
errorString		O
estimatedBandwidth	O	
event	O	O
fragmentSum.*	O	
fragments	O	
frameDropped	O	
licenseInfo.*	O	O
maxDecodingTime	O	
tracks		
userEvents	O	

	Playback Updated	Playback Ended
networkChanged	O	
networkErrors	O	
networkInfo.*	O	O
position		O
qoeContentId	O	O
qualityChanged	O	
redirectUrl	O	O
avgDecodingTime	O	
startupTime		
userInfo		

QoE Data Description

Data Merging Rule

sessionId	event	time stamp	C	D	E	F	G
123	playbackEnded	15632	c	d	e	f	g

sessionId	event	time stamp	C	D	K	L	M
123	playbackUpdated	15000	c	d	k	f	g
123	playbackUpdated	15123	a	d	b	e	g
123	playbackUpdated	15345	d	a	m	d	h
198	playbackUpdated	15589	t	j	f	n	m



sessionId	timestamp	C	D	E	F	G	K	L	M
123	15632	c	d	e	f	g	m	d	h

Feature Selection

Features Selection

1) Invalid Feature 삭제

A) Too Sparse Column

```
In [2]: rdf = pd.read_pickle('data/concated_all_1.pkl')
rdf.shape, rdf.columns

Out[2]: ((25050, 39),

In [14]: len(rdf['deviceInfo.deviceId'].value_counts())

Out[14]: 5566
```

총 25050개의 data 중 5566개의 카테고리 생성 시 **data의 분포가 너무 희박해져 학습에 악영향**

+ qoeContentId 동일한 문제로 삭제

B) Duplicated Column

```
In [22]: rdf.iloc[1]['deviceInfo.os']

Out[22]: 'Android'

In [23]: rdf.iloc[1]['deviceInfo.osVersion']

Out[23]: '7.0'

In [25]: rdf.iloc[1]['device']

Out[25]: 'Android 7.0'
```

하나의 column 에서 파생된 column 에 대하여 가장 많은 정보를 포함한 column을 남기고 삭제

C) Meaningless Column

```
In [16]: rdf['licenseInfo.laUrl'].value_counts()

Out[16]: https://app.video.dmkt-sp.jp/license/widevine-challenge    21309
NaN                                                                1
Name: licenseInfo.laUrl, dtype: int64
```

모든 data에 같은 값이 매핑되어있는 column은 의미가 없어서 삭제

+ deviceInfo.player,
networkChanged
networkInfo.carrier.name
fragments .etc 삭제

Features Selection

2) pearson 상관관계 분석 결과

avgDecodingTime	0.070473
bufferedDuration	0.041627
currentPosition	0.126110
device	-0.018956
errorCode	1.000000
estimatedBandwidth	0.072219
fragmentSum.bitrate	0.051129
fragmentSum.downloadTime	0.011791
fragmentSum.duration	0.011543
fragmentSum.fragmentIndex	-0.005489
fragmentSum.size	0.048179
frameDropped	0.021447
licenseInfo.drmSystem	NaN
licenseInfo.elapsedTime	0.009947
maxDecodingTime	0.129162
networkErrors	0.022833
networkInfo.carrier.mnc	-0.031805
networkInfo.type	-0.012458
position	0.126110
timestamp	-0.031551
qualityChangedCount	-0.171506
content_type	-0.027799
bufferingTime	0.044721

r = X와 Y가 함께 변하는 정도 / X와 Y가 각각 변하는 정도

$0.7 < |r| \leq 1$: 강한 선형관계
 $0.3 < |r| \leq 0.7$: 뚜렷한 선형관계
 $0.1 < |r| \leq 0.3$: 약한 선형관계
 $|r| \leq 0.1$: 거의 무시할 수 있는 선형관계



errorCode와 다른 feature들은 거의 무시할 수 있는 선형관계

데이터에 대한 배경지식을 기반으로 **manual하게 feature selection** 진행

Data Preprocessing

Data Preprocessing

Data preprocessing ?

- 1) Raw Data를 모델이 이해할 수 있는 (학습에 사용할 수 있는) 형식으로 바꿔주는 데이터마이닝 기법
- 1) 현실의 데이터는 불완전하고 에러를 많이 포함하며 불연속적인 부분이 존재하기 때문에 이런 이슈를 해결하는 과정이 필요
- 1) 결측치 처리(삭제, 다른값으로 대체, 예측값 삽입) ,이상치 처리(삭제, 다른값으로 대체, 리샘플링, 케이스 분리 분석), Transforming(기존의 데이터로 다른 변수 제작) 등



Data Preprocessing

Categorical value → label encoding (범주형 자료 라벨링)

```
In [8]: df['device'].value_counts()
```

```
Out[8]: Android 8.0.0      193
        Android 5.0.2      31
        Android 8.1.0      30
        Android 6.0.1      24
        Android 7.0        22
        Android 7.1.2       8
        Android 5.0         6
        Android 6.0         5
        Android 5.0.1        2
        Android 7.1.1        2
        Android 5.1.1        1
        Name: device, dtype: int64
```



```
In [18]: le_device = preprocessing.LabelEncoder()
df['device'] = le_device.fit_transform(df['device'].astype(str))
print('encoding 된 device : \n', le_device.classes_)
print('encoding 후 device 값 : \n', df['device'].value_counts())
```

```
encoding 된 device :
['Android 5.0' 'Android 5.0.1' 'Android 5.0.2' 'Android 5.1.1'
 'Android 6.0' 'Android 6.0.1' 'Android 7.0' 'Android 7.1.1'
 'Android 7.1.2' 'Android 8.0.0' 'Android 8.1.0' 'None']
encoding 후 device 값 :
9      193
2       31
11      30
10      30
5       24
6       22
8        8
0        6
4        5
7        2
1        2
3        1
        Name: device, dtype: int64
```

device, networkInfo.type (LTE, None, WIFI), licenseInfo.drmSystem(데이터 유무) 를 위와 같은 방법으로 처리함

Data Preprocessing

Data counting (json row 갯수 count)

```
In [21]: df['qualityChanged'].iloc[17]
```

```
Out[21]: [{'bandwidth': 431581, 'position': 0, 'timestamp': 1541721419},  
          {'bandwidth': 2725906, 'position': 10076, 'timestamp': 1541721420}]
```

```
In [18]: df['qualityChangedCount'] = None
```

```
In [19]: for row, val in enumerate(df['qualityChanged']):  
         if isinstance(val, float):  
             df['qualityChangedCount'][row] = 0  
         else:  
             df['qualityChangedCount'][row] = len(val)
```

```
/home/centos/anaconda3/envs/dev/lib/python3.5/site-packages  
A value is trying to be set on a copy of a slice from a Data  
See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/10min/05min.html#copy  
"""
```

```
In [20]: df['qualityChangedCount'].value_counts()
```

```
Out[20]: 0    202  
         1     96  
         2     56  
         Name: qualityChangedCount, dtype: int64
```

Event 가 발생한 횟수를 반영하기 위해 counting 방법을 이용
frameDropped 와 qualityChanged 를 위와 같은 방법으로 처리
함

Data Preprocessing

Extract valid part (유효한 부분만 추출)

```
In [111]: qoe_ended['contentUrl'].head(3).tolist()
```

```
Out[111]: ['http://abc-streaming.akamaized.net/g/mpd/d/main/video/1065/10656340/10656340.mpd?p=80&cf=1541728796&e=1541804396&h=c3d0063630dc0d664be09c31a08e0b97',  
          'http://abc-streaming.akamaized.net/g/mpd/d/main/video/1065/10656983/10656983.mpd?p=80&cf=1541728803&e=1541804403&h=32fc8d807a3afac2f701ac1f0f671d8f',  
          'http://abc-streaming.akamaized.net/g/mpd/d/main/video/1065/10656983/10656983.mpd?p=80&cf=1541728808&e=1541804408&h=d7d7ade43a2d010733545c602b44e7c1']
```



```
In [25]: le_content = preprocessing.LabelEncoder()  
df['content_type'] = le_content.fit_transform(df['content_type'].astype(str))  
le_content.classes_
```

```
Out[25]: array(['contents', 'video'], dtype=object)
```

```
In [26]: df['content_type'].value_counts()
```

```
Out[26]: 1    237  
         0     87  
         Name: content_type, dtype: int64
```


contentUrl 에서 content_type 추출 후 label encoding

Data Preprocessing

Sum of data

```
In [118]: qoe_updated[qoe_updated.index==42159755583668]['bufferings'].tolist()
```

```
Out[118]: [{ 'currentTime': 1541719621, 'duration': 120},  
            { 'currentTime': 1541719621, 'duration': 975}]
```



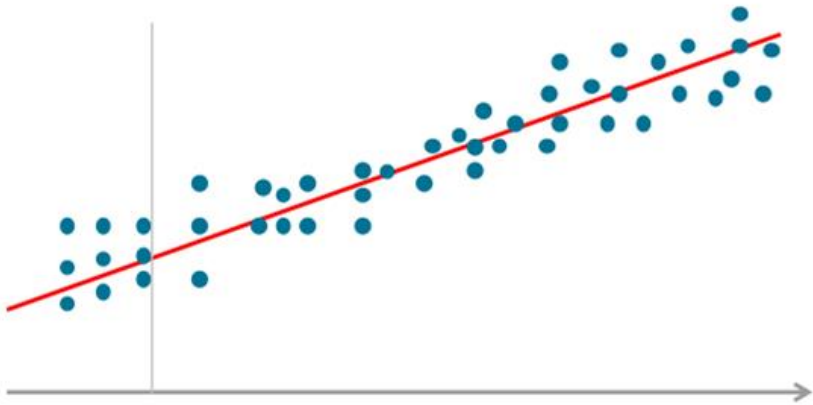
```
In [36]: df['bufferingTime'].head(10)
```

```
Out[36]: timestamp  
2018-11-08 23:08:53      31  
2018-11-08 23:22:38       0  
2018-11-08 23:20:46       0  
2018-11-08 23:44:54       0  
2018-11-08 23:09:01    1528  
2018-11-08 23:56:49       0  
2018-11-08 23:21:16       0  
2018-11-08 23:22:43       0  
2018-11-08 23:48:10       0  
2018-11-08 23:31:47       0  
Name: bufferingTime, dtype: int64
```

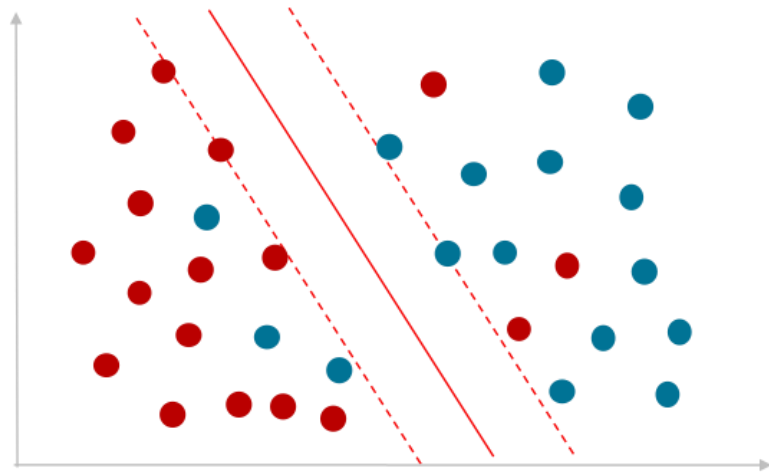
bufferings 의 duration 값만 추출한 후 더하여 total duration값을 반영함

Decision Tree & Random Forest

Machine Learning?



Linear Regression



Classification

과거/기존의 데이터를 **학습**하여 미래/새로운의 데이터를 **예측**

Machine Learning Step

1. 문제 정의
2. Train Data 수집
3. Exploratory Data Analysis – Cleaning & Preprocessing
4. Data 분류 – Labeling
5. 모델 설계
6. 모델 학습
7. 성능 Test

Decision Tree

Q. 겨울에 찍은 가족사진은?



Decision Tree

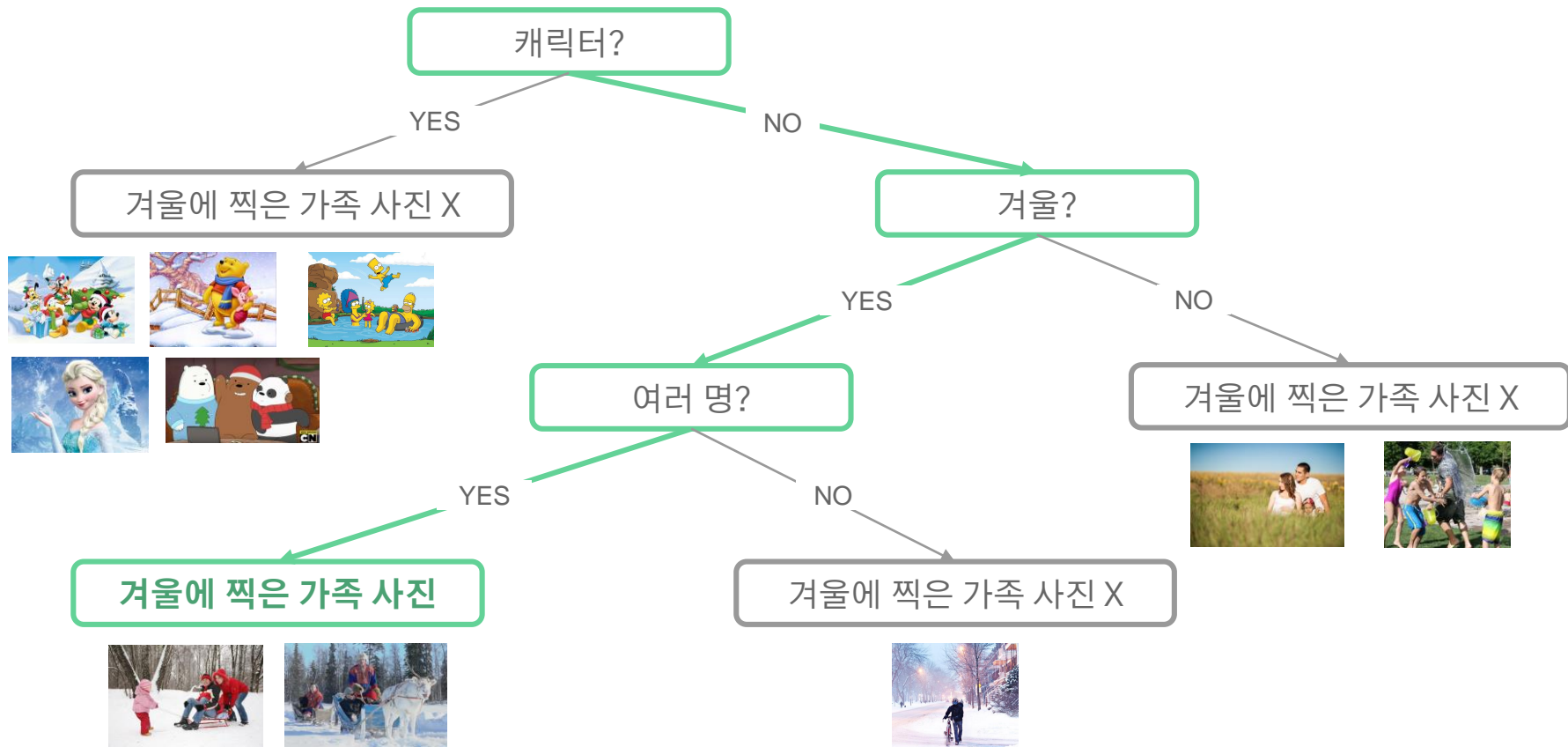


캐릭터?	겨울?	여러 명?	겨울 가족 사진
1	1	1	0
0	0	1	0
1	0	1	0
0	1	1	1
0	0	1	0



캐릭터?	겨울?	여러 명?	겨울 가족 사진
0	1	1	1
1	1	1	0
1	1	0	0
0	1	0	0
1	1	1	0

Decision Tree



Decision Tree

Information Gain

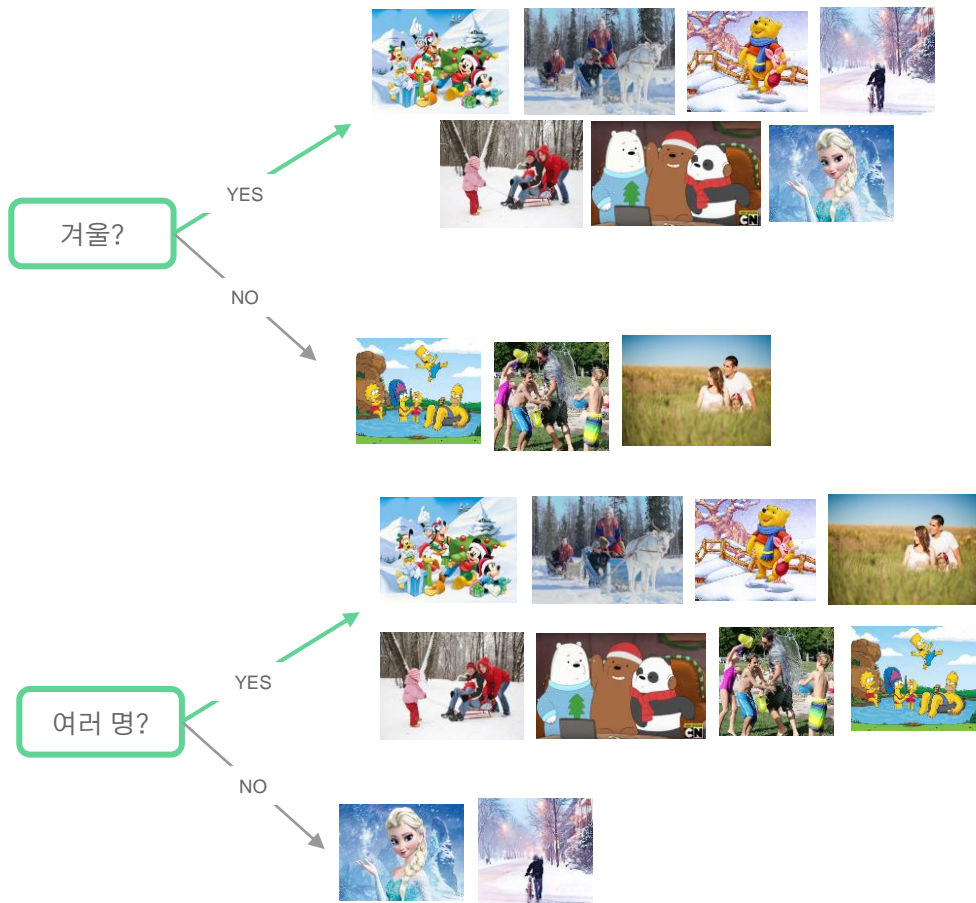
$$= \text{Base Entropy} - \text{New Entropy}$$

Information Gain이 높을 수록 효율적인 Split Option

$$* \text{캐릭터 I.G} = 10 - 5 = 5$$

$$* \text{겨울 I.G} = 10 - 7 = 3$$

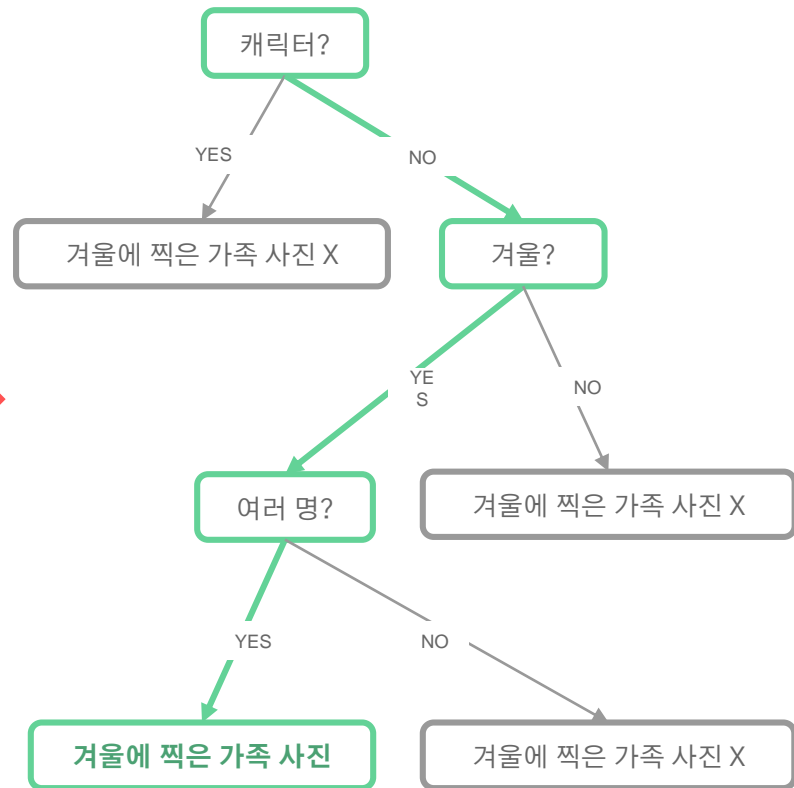
$$* \text{여러 명 I.G} = 10 - 8 = 2$$



Decision Tree



캐릭터?	겨울?	여러 명?
0	1	1
1	1	1
0	0	1



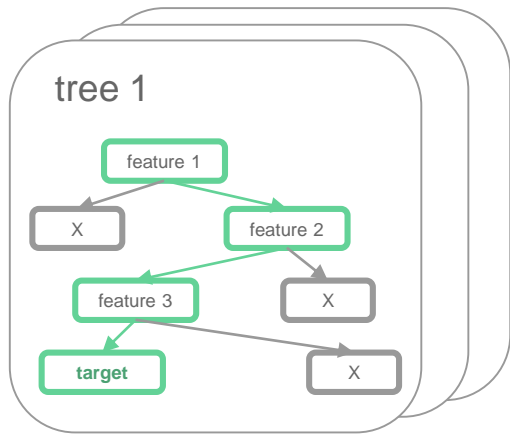
Random Forest



Tree

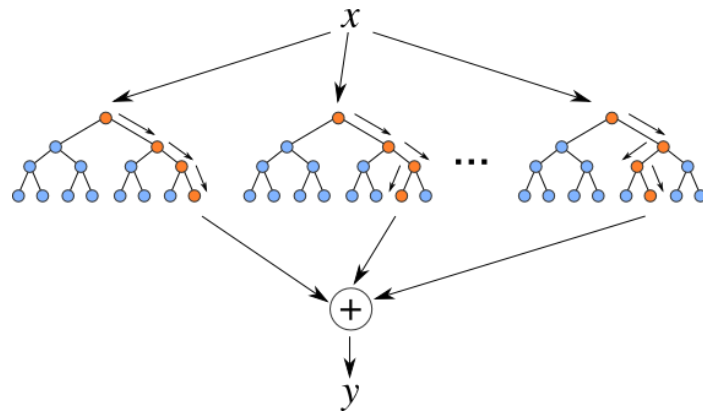


Forest



Decision Tree

....



Random Forest

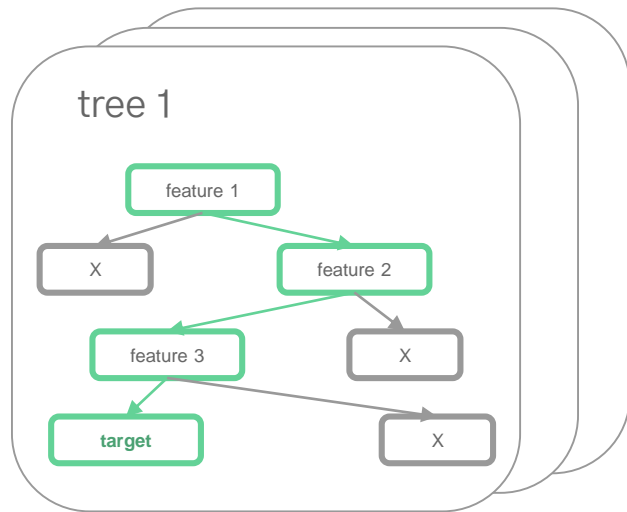
Random Forest

겨울에 찍은 가족사진인지 구분할 수 있는 요소

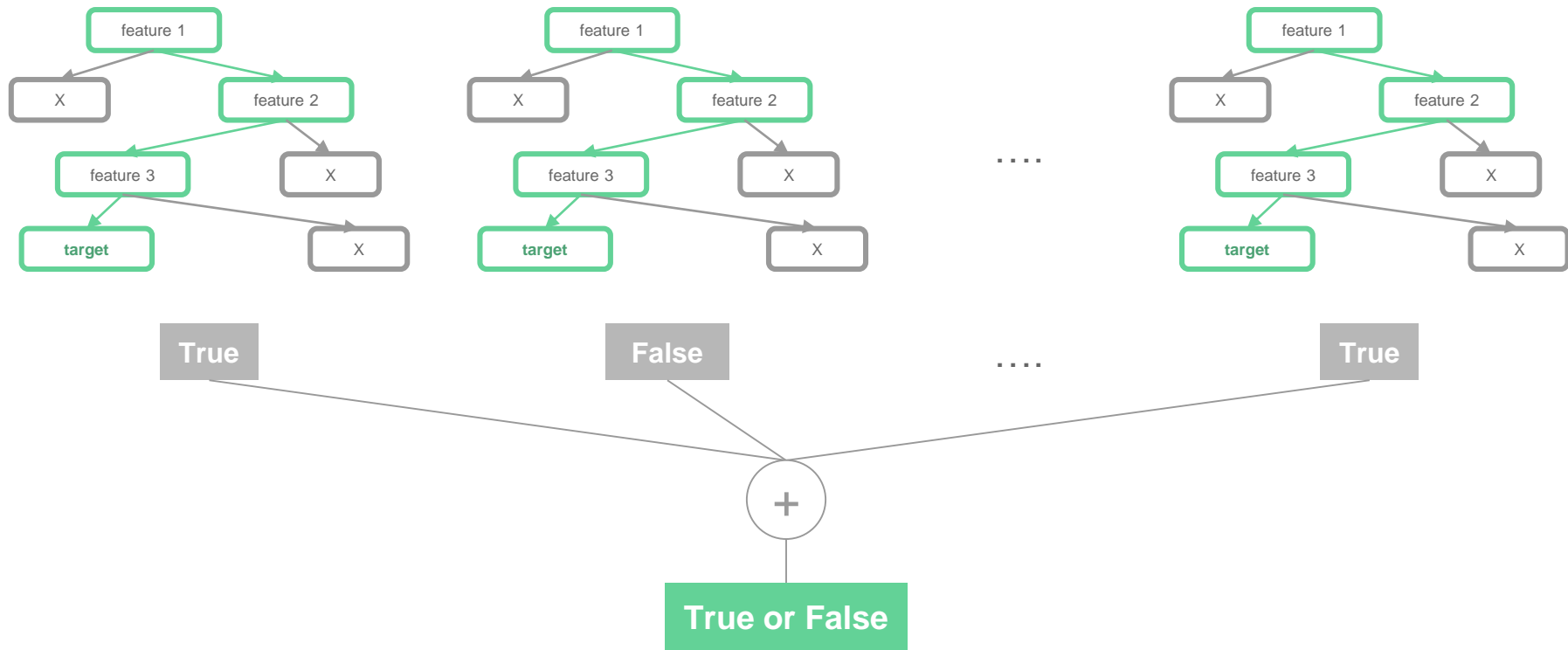
눈의 여부
옷의 두께
모자, 장갑 등의 겨울 소품
아이의 여부
야외, 실내
인원 수
사진 속 인물들의 평균 연령
성별 비율
.
.
.

Radom

Tree 생성



Random Forest



Experiments & Reports

Performance Evaluation Metrics

		Actual	
		Positives(1)	Negatives(0)
Predicted	Positives(1)	TP	FP
	Negatives(0)	FN	TN

- Accuracy(정확도)

: 전체 중 **맞춘 것**의 비율

$$\frac{TP + TN}{TP + FN + FP + TN}$$

- Precision(정밀도)

: True라고 분류한 샘플 중의 **실제 True**의 비율

$$\frac{TP}{TP + FP}$$

- Recall(재현율)

: 실제 True 중에 **True라고 분류한 샘플**의 비율

$$\frac{TP}{TP + FN}$$

Decision Tree - Naive

- Accuracy: 0.96
- 대부분의 ErrorCode를 0으로 예측하였기 때문에 전체 ErrorCode에 대한 Accuracy 점수는 높지만 다른 ErrorCode 각각의 Metric 점수(Precision, Recall)는 낮음
- Error가 아닌 데이터와 Error인 데이터의 비율이 불균형 (Data Imbalance)하기 때문에 전체 데이터 개수로 계산하는 Accuracy보다 Precision과 Recall 점수가 높을수록 성능이 좋은 모델

ErrorCode	Precision	Recall	Support
-101416.0	0.00	0.00	1
-100874.0	0.00	0.00	4
-100870.0	0.00	0.00	3
-100859.0	0.00	0.00	1
-100702.0	0.00	0.00	90
-2005.0	0.00	0.00	11
-1004.0	1.00	0.90	10
-1002.0	0.67	0.40	5
-38.0	0.00	0.00	53
0.0	0.96	1.00	3554

Resampling

- 모델의 성능을 높이기 위한 방법 중 하나
- 데이터의 timestamp 간격을 무작위 → **일정 간격**으로 재정렬
- 데이터 resampling 후 NaN 값이 있는 시간에는 각각 앞 시간의 데이터, 최빈값, 평균값 등으로 채움

	errorCode	estimatedBandwidth	fragmentSum.bitrate
timestamp			
2018-11-06 02:14:03	0.0	66791384	0
2018-11-06 02:14:14	0.0	30158938	0
2018-11-06 02:14:37	0.0	80715832	106455214
2018-11-06 02:14:37	0.0	40284572	6255458
2018-11-06 02:14:40	0.0	49529664	0


Resample

	errorCode	estimatedBandwidth	fragmentSum.bitrate
timestamp			
2018-11-06 02:14:00	0.0	66791384.0	0.0
2018-11-06 02:14:15	0.0	66791384.0	0.0
2018-11-06 02:14:30	0.0	80715832.0	106455214.0
2018-11-06 02:14:45	0.0	104205000.0	0.0
2018-11-06 02:15:00	0.0	89998664.0	0.0

Decision Tree - 15s

- Accuracy: 0.96
- 15초 간격으로 resampling
- 15초 사이에 데이터가 여러 개 있을 경우,
15초 간격의 해당 시간 데이터가 NaN 값일 경우
: 시간 순으로 정렬했을 때 **가장 처음 값**

timestamp	errorCode	estimatedBandwidth	fragmentSum.bitrate
2018-11-06 02:14:03	0.0	66791384	0
2018-11-06 02:14:14	0.0	30158938	0



timestamp	errorCode	estimatedBandwidth	fragmentSum.bitrate
2018-11-06 02:14:00	0.0	66791384.0	0.0

ErrorCode	Precision	Recall	Support
-100874	0.00	0.00	1
-100859	0.00	0.00	1
-100702	0.00	0.00	51
-2005	0.00	0.00	1
-1004	0.33	1.00	1
-1002	1.00	0.50	6
-38	0.00	0.00	22
0	0.96	1.00	2114

Decision Tree & Random Forest - 10min

- Accuracy: DT - 0.75 / RF - 0.82
- 10분 간격으로 resampling
- 10분 사이에 데이터가 여러 개 있을 경우: **평균값** (numeric) / **최빈값** (categorical)

ErrorCode	Precision	Recall	Support
-100874.0	0.00	0.00	1
-100702.0	0.79	0.70	27
-1004.0	0.75	1.00	3
-1002.0	0.00	0.00	2
-38.0	0.70	0.93	15
0.0	0.71	0.71	7

Decision Tree

ErrorCode	Precision	Recall	Support
-100874.0	0.00	0.00	1
-100702.0	0.80	0.89	27
-1004.0	1.00	1.00	3
-1002.0	0.00	0.00	2
-38.0	0.81	0.87	15
0.0	0.83	0.71	7

Random Forest

Decision Tree & Random Forest - 15min

- Accuracy: DT - 0.676 / RF - 0.68
- 15분 간격으로 resampling
- 15분 사이에 데이터가 여러 개 있을 경우: **평균값** (numeric) / **최빈값** (categorical)

ErrorCode	Precision	Recall	Support
-100874.0	0.00	0.00	1
-100870.0	0.00	0.00	1
-100702.0	0.82	0.88	16
-1004.0	1.00	0.60	5
-38.0	0.40	0.86	7
0.0	1.00	0.29	7

Decision Tree

ErrorCode	Precision	Recall	Support
-100874.0	0.00	0.00	1
-100870.0	0.00	0.00	1
-100702.0	0.76	0.81	16
-1004.0	1.00	0.60	5
-38.0	0.33	0.57	7
0.0	1.00	0.71	7

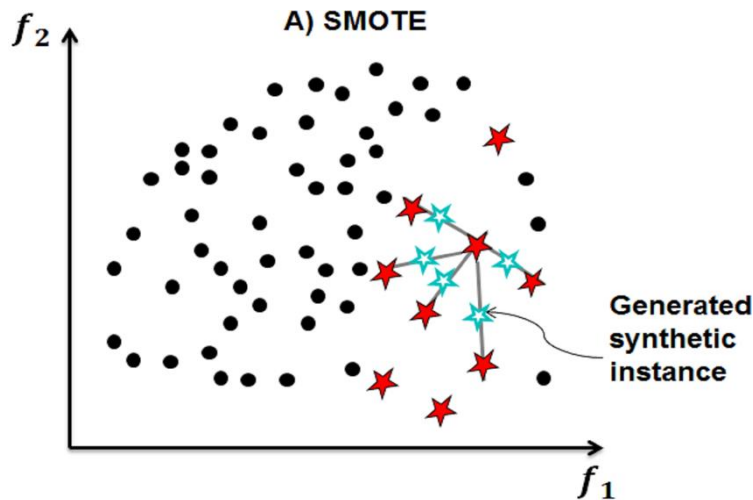
Random Forest

Oversampling - SMOTE

- SMOTE(Synthetic Minority Over-sampling Technique)
- 데이터 불균형 (Data Imbalance) 문제를 해결하기 위한 방법 중 하나

“ The minority class is over-sampled by taking each minority class sample and **introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors.** ”

<SMOTE: Synthetic Minority Over-sampling Technique> 중 일부



Decision Tree & Random Forest - 15s

- Accuracy: DT – 0.631 (10000 iter) / RF – 0.98
- train dataset 의 Y data 중 가장 많은 수의 데이터 (ErrorCode: 0 / 7375개) 기준으로 SMOTE

ErrorCode	Precision	Recall	Support
-100702	0.23	0.41	63
-2005.0	0.01	1.00	3
-1004.0	1.00	0.50	2
-1002.0	0.11	0.33	3
-38.0	0.03	0.61	36
0.0	1.00	0.63	3184

Decision Tree

ErrorCode	Precision	Recall	Support
-100702	0.76	0.90	63
-2005.0	0.50	0.33	3
-1004.0	1.00	0.50	2
-1002.0	0.75	1.00	3
-38.0	0.38	0.36	36
0.0	0.99	0.99	3184

Random Forest

Conclusion

- 2일 치의 데이터로만 학습하여 ErrorCode의 패턴 트렌드 및 error 발생 시간 예측 불가
- QoE Data 와 ErrorCode간의 상관관계가 뚜렷한 양상을 띄지 않음
- feature selection을 manual하게 진행하여 신뢰성 하락

- 충분한 컴퓨팅 자원과 실험 조건(데이터의 개수, feature 등) 하에 딥러닝 모델(RNN, LSTM, GRU 등)을 활용하여 학습한다면 ErrorCode의 패턴 트렌드 파악 및 발생 확률을 실시간으로 예측할 수 있는 모델 구축 가능

Q&A

Thank You :)
