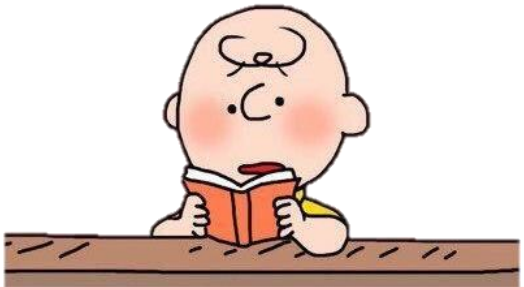
Data Mining Final Round

유(나)종(연)의 美

INDEX



1. 1 Round

1라운드 계획

1. 하나의 데이터셋에 수많은 피쳐를
   만들어 적용해보자!

2. 직접 피쳐를 만들기보다는 수업에서
   배운 feature engineering 기법을 활용하자

(※피쳐에 대한 이해보다는 피쳐가 많아야 잘 구분할 수 있다고 생각함)

1. 1 Round

데이터셋 생성

분프 파생변수에 mba 커널 변수를 추가적으로 추가

Pred 1

분프2 파생변수 + MBA feature (ex.브랜드 편중도 등..)

+ MBA feature (ex.치우친 데이터 log치환)

+ MBA feature (ex.dummy PCA)

1. 1 Round

데이터셋 생성

수업시간 배운 Feature engineering기법 적용

1) " Polynomial "

Try 1. Pred 1 + 피벗테이블, 수치형 피쳐에 대하여

-> 1만여개 이상의 피쳐 생성

Try 2. Pred 1 + 수치형 데이터만

-> 1만여개 이상의 피쳐 생성

1. 1 Round

데이터셋 생성

수업시간 배운 Feature engineering기법 적용

1) " Polynomial "

Try 1. Pred 1 + 피벗테이블, 수치형 피쳐에 대하여

튜닝없이 XGB 단일 모델 적용 : 0.70927

Try 2. Pred 1 + 수치형 데이터만

튜닝없이 XGB 단일 모델 적용 : 0.70883

1. 1 Round

데이터셋 생성

수업시간 배운 Feature engineering기법 적용

2) " AutoFE "

튜닝없이 XGB 단일 모델 적용 : 0.62554

3) " TPOT "

튜닝없이 XGB 단일 모델 적용 : 0.61278

1. 1 Round

데이터셋 생성

Feature Engineering은 큰 효과가 없음….
오히려, MBA 커널 변수를 추가적으로 추가했을 때 성능이 증가

+ 수업시간 Private 점수 공개, 등수 급락…..
  (-> 오버피팅 발생!)

즉, 피쳐 기법이 아니라 의미있는 피쳐 생성이 중요!

1. 1 Round

오버피팅을 막자!

오버피팅이 발생한 이유 분석

1. 변수에 대한 이해를 하지 않은 채 poly로 수를 뻥튀기 해버림.
2. 변수 하나하나에 대한 성능 테스트를 거치지 않음
3. 튜닝도 거치지 않은 단일 모델로 성능을 높이려고함.

1. 1 Round

오버피팅을 막자!

Step 1.

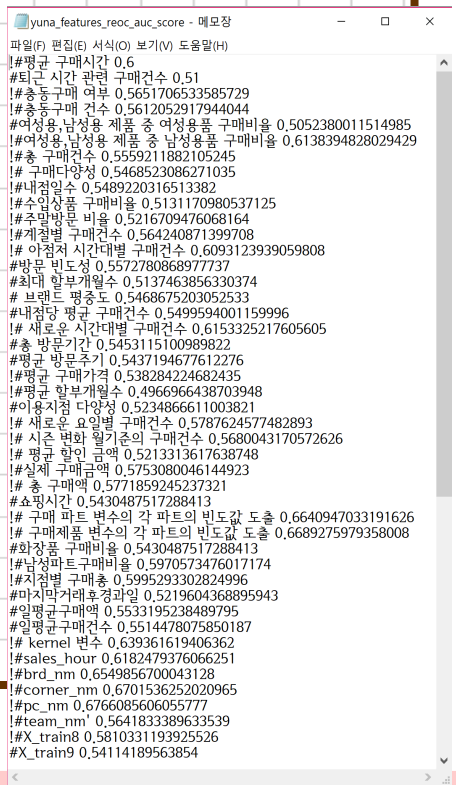
사용한 모든 변수에 대한 a/b Test 진행

Step 2.

성능이 좋은(0.56) 변수만 사용

Step 3.

Boosting 계열 모델들
사용 후 앙상블


yuna_features_reoc_auc_score - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

!#평균 구매시간 0.6
#퇴근 시간 관련 구매건수 0.51
!#충동구매 여부 0.5651706533585729
!#충동구매 건수 0.5612052917944044
#여성용,남성용 제품 중 여성용품 구매비율 0.5052380011514985
!#여성용,남성용 제품 중 남성용품 구매비율 0.6138394828029429
!#총 구매건수 0.5559211882105245
!# 구매다양성 0.5468523086271035
!#내점일수 0.5489220316513382
!#수입상품 구매비율 0.5131170980537125
!#주말방문 비율 0.5216709476068164
!#계절별 구매건수 0.564240871399708
!# 아점저 시간대별 구매건수 0.6093123939059808
#방문 빈도성 0.5572780868977737
#최대 할부개월수 0.5137463856330374
브랜드 평충도 0.5468675203052533
#내점당 평균 구매건수 0.5499594001159996
!# 새로운 시간대별 구매건수 0.6153325217605605
#총 방문기간 0.5453115100989822
#평균 방문주기 0.5437194677612276
!#평균 구매가격 0.538284224682435
!#평균 할부개월수 0.4966966438703948
#이용지점 다양성 0.5234866611003821
!# 새로운 요일별 구매건수 0.5787624577482893
!# 시즌 변화 월기준의 구매건수 0.5680043170572626
!# 평균 할인 금액 0.5213131617638748
!#실제 구매금액 0.5753080046144923
!# 총 구매액 0.5771859245237321
#쇼핑시간 0.5430487517288413
!# 구매 파트 변수의 각 파트의 빈도값 도출 0.6640947033191626
!# 구매제품 변수의 각 파트의 빈도값 도출 0.6689275979358008
#화장품 구매비율 0.5430487517288413
!#남성파트구매비율 0.5970573476017174
!#지점별 구매총 0.5995293302824996
#마지막거래후경과일 0.5219604368895943
#일평균구매액 0.5533195238489795
#일평균구매건수 0.5514478075850187
!# kernel 변수 0.639361619406362
!#sales_hour 0.6182479376066251
!#brd_nm 0.6549856700043128
!#corner_nm 0.6701536252020965
!#pc_nm 0.6766085606055777
!#team_nm' 0.5641833389633539
!#X_train8 0.5810331193925526
#X_train9 0.54114189563854

1. 1 Round

데이터셋 생성

New Data Set :

0.56 이상 변수

+ 연속구간 분할 변수

+ 변수 그룹화

+ Log 치환 + PCA

1. 1 Round

모델 선정

Gradient Boosting

Gmean Ensemble 0.72435

XGB

Light GBM

2. 1라운드 후 문제점 분석

1라운드 후 문제점 분석 및 해결 초점

1라운드 이후 느낀 문제점

1. 데이터셋이 1개.
2. 피쳐 수를 늘리는데 급급했음.
3. 모델 튜닝이 이뤄지지 않음.
4. 커널의 변수를 모두 활용하지 않음.
5. 데이터에 대한 이해 완전치 못함.

3. Round 2

모델 선정

XGB

Light GBM

Gradient Boosting

3. Round 2

데이터셋 생성

"1차 1,2,3등 변수 + 우리가 만든 변수"

에서 랜덤 중복 추출하여

6개의 데이터셋을 만듦.

(※ booststrap 방식 응용)

3. Round 2

데이터셋 생성

Gmean	Private	Public
Pred1	0.70501	0.70300
Pred2	0.70684	0.69990
Pred3	0.69042	0.69308
Pred4	(NN만 진행한거라 제출 안함)	
Pred5	0.68594	0.68122
Pred6	0.68401	0.68825

3. Round 2

데이터셋 생성

Pred1 변수가 무엇이길래?

1차 때도 느꼈던,

Dummy PCA 와
치우친 데이터 log처리 !!

3. Round 2

데이터셋 생성

변수 하나하나 마다 a/b Test 진행

yuna_features_reoc_auc_score - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
!#평균 구매시간 0.6
#퇴근 시간 관련 구매건수 0.51
!#충동구매 여부 0.5651706533585729
!#충동구매 건수 0.5612052917944044
#여성용,남성용 제품 중 여성용품 구매비율 0.5052380011514985
!#여성용,남성용 제품 중 남성용품 구매비율 0.6138394828029429
!#총 구매건수 0.5559211882105245
!# 구매다양성 0.5468523086271035
!#내점일수 0.5489220316513382
!#수입상품 구매비율 0.5131170980537125
!#주말방문 비율 0.5216709476068164
!#계절별 구매건수 0.564240871399708
!# 아점저 시간대별 구매건수 0.6093123939059808
#방문 빈도성 0.5572780868977737
#최대 할부개월수 0.5137463856330374
# 브랜드 평중도 0.5468675203052533
#내점당 평균 구매건수 0.5499594001159996
!# 새로운 시간대별 구매건수 0.6153325217605605
#총 방문기간 0.5453115100989822
#평균 방문주기 0.5437194677612276
!#평균 구매가격 0.538284224682435
!#평균 할부개월수 0.4966966438703948
#이용지점 다양성 0.5234866611003821
!# 새로운 요일별 구매건수 0.5787624577482893
!# 시즌 변화 월기준의 구매건수 0.5680043170572626
!# 평균 할인 금액 0.5213313617638748
!#실제 구매금액 0.5753080046144923
!# 총 구매액 0.5771859245237321
#쇼핑시간 0.5430487517288413
!# 구매 파트 변수의 각 파트의 빈도값 도출 0.6640947033191626
!# 구매제품 변수의 각 파트의 빈도값 도출 0.6689275979358008
#화장품 구매비율 0.5430487517288413
!#남성파트구매비율 0.5970573476017174
!#지점별 구매총 0.5995293302824996
#마지막거래후경과일 0.5219604368895943
#일평균구매액 0.5533195238489795
#일평균구매건수 0.5514478075850187
!# kernel 변수 0.639361619406362
!#sales_hour 0.6182479376066251
!#brd_nm 0.6549856700043128
!#corner_nm 0.6701536252020965
!#pc_nm 0.6766085606055777
!#team_nm' 0.5641833389633539
!#X_train8 0.5810331193925526
#X_train9 0.54114189563854
```

Pred7 :

1차

만 선출

+

+ 연속

= 선택적 추가

Pred8 :

1차

수

+

(※

있어서)

3. Round 2

데이터셋 생성

Pred7 :

1차 우리 변수 중 0.56 이상만 선출
+ log + dummy PCA

+ 연속변수 분할, 변수 묶기 위해 필요한 피쳐는 선택적 추가

Pred8 :

1차 2등 조 변수 + 우리 변수
+ log + dummy PCA

(※ 1차 1,3등 변수는 모두 2등 변수에 들어가 있어서)

3. Round 2

데이터셋 생성

Pred7 & Pred8 에 1차 1등 조 기법
" Imbalanced data" 처리를 함!

-> 1. SMOTE 2. TomekLink 사용

3. Round 2

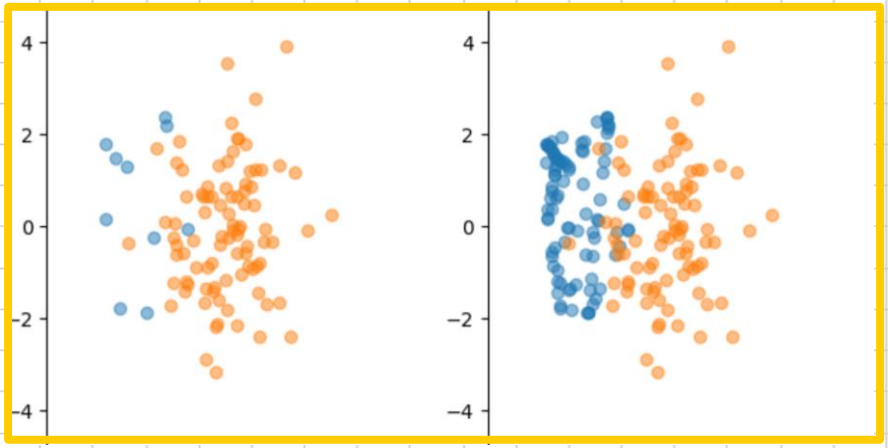
데이터셋 생성

	Public	Private
Pred7 -		
SMOTE	0.68379	0.70598
TomekLink	0.71616	0.71711
Pred8 -		
SMOTE	(※오래걸리고, 메모리 오류로 진행이 어려웠음)	
TomekLink		

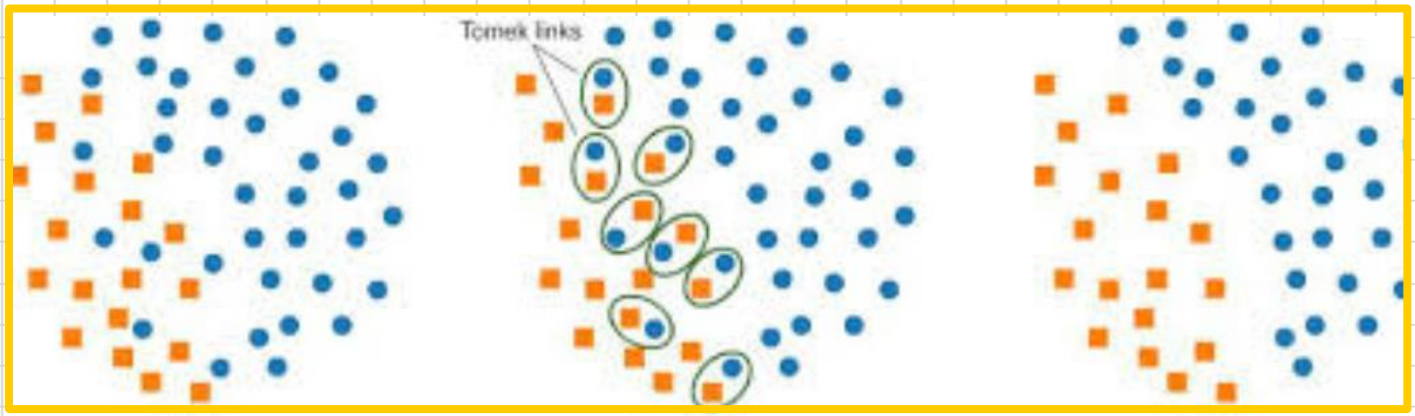
3. Round 2

데이터셋 생성

SMOTE



TomekLink



3. Round 2

데이터셋 생성

		Public	Private
Pred7 -			
	SMOTE :	0.68379	0.70598
	TomekLink :	0.71616	0.71711
Pred8 -			
	SMOTE :	진행 안함	
		(※오래걸리고, 메모리 오류로 진행이 어려웠음)	
	TomekLink :	0.71672	0.72504

3. Round 2

데이터셋 생성

Pred9 : 수치형 데이터 (ex,평균구매액, 내점일수) 만 가지고 poly -> 0. 63513

Pred10 : 원핫인코딩한 변수들 가지고 word2vec -> 0.64420

3. Round 2

데이터셋 생성

Pred12 : 1차 2등조의 pred1
+ 우리 변수 중 0.56이상 선출

	Private	Public
-> SMOTE :	0.70598	0.69816
TomekLink :	0.71711	0.71081

3. Round 2

데이터셋 생성

Pred12 : 1차 2등조의 pred1
+ 우리 변수 중 0.56이상 선출

	Private	Public
-> SMOTE	0.70598	0.69816
TomekLink	0.71711	0.71081
Imbalanced X	0.72706	0.71706

3. Round 2

앙상블

지금까지 만든 pred 중 0.7이상 파일만 가지고 앙상블

1. pred12 임벨런스 처리 없이 앙상블
2. pred7 tomeklink 앙상블
3. Pred1 gmean 앙상블
4. Pred1 Voting

(※ 이때까지 pred8 tomek값이 나오지 않아서 함께 앙상블 못함)

Private : 0.72016 Public : 0.71685

3. Round 2

앙상블

"지금 앙상블한 값들간의 **유사도가 높아서**
앙상블 했을 때 오히려 값이 않좋구나!"

지금까지 만든 pred 중 0.7이상 파일만 가지고 앙상블

1. pred12 임벨런스 처리 없이 앙상블
2. pred7 tomeklink 앙상블
3. Pred1 gmean 앙상블
4. Pred1 Voting
5. 1차 1등 & 2등 → 0.72838

(※ 이때까지 pred8 tomek값이 나오지 않아서 함께 앙상블 못함)

Private : 0.72643 Public : 0.72123

3. Round 2

앙상블

"Power mean (멱평균)"

1. pred12 임벨런스 처리 없이 앙상블
2. Pred1 gmean 앙상블
3. Pred1 Voting
4. 1차 1등 & 2등
5. Pred8 tomeklink 앙상블
6. 앞장의 앙상블한 데이터

Private : 0.73282 **Public : 0.72461**

(※ 최적의 P를 찾기위해 상하로 조정하였으나, 교수님이 주신 2.56이 최고)

3. Round 2 – 딥러닝 기법

1. Neural Net

```
# input node
max_features = X_train_scaled.shape[1]

# modeling
model = Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(max_features,), kernel_constraint=max_norm(2.), kernel_initializer="he_normal"))
model.add(Dropout(0.2))
model.add(layers.Dense(16, activation='relu'))

model.add(layers.Dense(1, activation='sigmoid'))

# compile
model.compile(optimizer=RMSprop(lr=1e-4), loss='binary_crossentropy', metrics=['acc'])

# train
history = model.fit(X_train_scaled, y_train.gender,
                    epochs=100,
                    batch_size = 64,
                    validation_split = 0.2,
                    callbacks=[EarlyStopping(patience = 7)])
```

```
Epoch 12/100
24000/24000 [==============================] - 4s 168us/step - loss: 0.5494 - acc: 0.7270 - val_loss: 0.5710 - val_acc: 0.7167
Epoch 13/100
24000/24000 [==============================] - 4s 163us/step - loss: 0.5481 - acc: 0.7287 - val_loss: 0.5711 - val_acc: 0.7163
Epoch 14/100
24000/24000 [==============================] - 4s 168us/step - loss: 0.5464 - acc: 0.7283 - val_loss: 0.5712 - val_acc: 0.7187
Epoch 15/100
24000/24000 [==============================] - 4s 167us/step - loss: 0.5449 - acc: 0.7301 - val_loss: 0.5714 - val_acc: 0.7180
Epoch 16/100
24000/24000 [==============================] - 4s 166us/step - loss: 0.5452 - acc: 0.7301 - val_loss: 0.5715 - val_acc: 0.7178
Epoch 17/100
24000/24000 [==============================] - 4s 177us/step - loss: 0.5442 - acc: 0.7316 - val_loss: 0.5720 - val_acc: 0.7173
Epoch 18/100
24000/24000 [==============================] - 4s 166us/step - loss: 0.5434 - acc: 0.7316 - val_loss: 0.5720 - val_acc: 0.7190
Epoch 19/100
24000/24000 [==============================] - 4s 177us/step - loss: 0.5416 - acc: 0.7332 - val_loss: 0.5727 - val_acc: 0.7187
Wall time: 1min 18s
```

3. Round 2 – 딥러닝 기법

1. Neural Net

Neural Net :

NeuralNet을 사용하여
다양한 Dense, 규제 등 방법을 시도

Private: 0.70944 Public : 0.69572

3. Round 2

앙상블

지금까지 만든 pred 중 0.7이상 파일만 가지고 앙상블

1. pred12 임벨런스 처리 없이 앙상블
2. pred7 tomeklink 앙상블
3. Pred1 gmean 앙상블
4. Pred1 Voting
5. 1차 1등 & 2등
6. NeuralNet

(※ 이때까지 pred8 tomek값이 나오지 않아서 함께 앙상블 못함)

Private : 0.73227 Public : 0.72437

3. Round 2 – 딥러닝 기법

2. CNN

```
# Define the Model & its Architecture
in_low = Input(shape=(max_len,), dtype='int32', name='low')
x = layers.Embedding(max_features, emb_dim)(in_low)
x = layers.Conv1D(32, 5, activation='elu')(x)
x = layers.MaxPooling1D(5)(x)
x = layers.Conv1D(32, 5, activation='elu')(x)
x = layers.GlobalMaxPooling1D()(x)
out_low = layers.Dropout(0.5)(x)

in_brd_nm = Input(shape=(max_len,), dtype='int32', name='brd_nm')
x = layers.Embedding(max_features//10, emb_dim)(in_brd_nm)
x = layers.Conv1D(32, 3, activation='elu')(x)
x = layers.MaxPooling1D(3)(x)
x = layers.Conv1D(32, 3, activation='elu')(x)
x = layers.GlobalMaxPooling1D()(x)
out_brd_nm = layers.Dropout(0.5)(x)

in_corner_nm = Input(shape=(max_len,), dtype='int32', name='corner_nm')
x = layers.Embedding(max_features//100, emb_dim)(in_corner_nm)
x = layers.Conv1D(32, 1, activation='elu')(x)
x = layers.MaxPooling1D(1)(x)
x = layers.Conv1D(32, 1, activation='elu')(x)
x = layers.GlobalMaxPooling1D()(x)
out_corner_nm = layers.Dropout(0.5)(x)

in_pc_nm = Input(shape=(max_len,), dtype='int32', name='pc_nm')
x = layers.Embedding(max_features//100, emb_dim)(in_pc_nm)
x = layers.Conv1D(32, 1, activation='elu')(x)
x = layers.MaxPooling1D(1)(x)
x = layers.Conv1D(32, 1, activation='elu')(x)
x = layers.GlobalMaxPooling1D()(x)
out_pc_nm = layers.Dropout(0.5)(x)

x = layers.add([out_low, out_brd_nm, out_corner_nm, out_pc_nm, out_part_nm ,out_team_nm , out_buyer_nm, out_season,
                        out_str, out_day , out_amount, out_weekend , out_returnsum])
out = layers.Dense(1, activation='sigmoid')(x)

model = Model([in_low, in_brd_nm, in_corner_nm, in_pc_nm,in_part_nm ,in_team_nm , in_buyer_nm,
 in_season, in_str , in_day , in_amount,
              in_weekend ,in_returnsum], out)
model.summary()

# Choose the Optimizer and the Cost function
model.compile(optimizer=RMSprop(lr=1e-4), loss='binary_crossentropy', metrics=['acc'])

# Train the Model
history = model.fit([X_train_low , X_train_brd_nm , X_train_corner_nm , X_train_pc_nm , X_train_part_nm,
                     X_train_team_nm , X_train_buyer_nm , X_train_season , X_train_str , X_train_day,
                     X_train_amount , X_train_weekend , X_train_returnsum], y_train, epochs=50, batch_size=16,
                    validation_split=0.2, callbacks=[EarlyStopping(patience=5)])

plt.plot(history.history["loss"], label="train loss")
plt.plot(history.history["val_loss"], label="validation loss")
plt.legend()
plt.title("Loss")
plt.show()
```

Private : 0.66419 **Public : 0.66079**

3. Round 2

앙상블

지금까지 만든 pred 중 0.7이상 파일만 가지고 앙상블

1. pred12 임벨런스 처리 없이 앙상블
2. pred7 tomeklink 앙상블
3. Pred1 gmean 앙상블
4. Pred1 Voting
5. 1차 1등 & 2등
6. CNN

(※ 이때까지 pred8 tomek값이 나오지 않아서 함께 앙상블 못함)

Private : 0.70746 Public : 0.70171

3. Round 2 – 딥러닝 기법

3. LSTM

In [25]:
```
# Converts a "brd_nm" to a sequence of indexes in a fixed-size hashing space
X_train = df_train.groupby('custid')['brd_nm'].apply(lambda x: [one_hot(i, max_features//100)[0] for i in x]).values
X_test = df_test.groupby('custid')['brd_nm'].apply(lambda x: [one_hot(i, max_features//100)[0] for i in x]).values

# Pads sequences to the same length
X_train_brd_nm = sequence.pad_sequences(X_train, maxlen=max_len)
X_test_brd_nm = sequence.pad_sequences(X_test, maxlen=max_len)

X_train_brd_nm.shape, X_test_brd_nm.shape
```

Out [25]:
```
((30000, 100), (19995, 100))
```

In [37]:
```
# Converts a "corner_nm" to a sequence of indexes in a fixed-size hashing space
X_train = df_train.groupby('custid')['corner_nm'].apply(lambda x: [one_hot(i, max_features//100)[0] for i in x]).values
X_test = df_test.groupby('custid')['corner_nm'].apply(lambda x: [one_hot(i, max_features//100)[0] for i in x]).values

# Pads sequences to the same length
X_train_corner_nm = sequence.pad_sequences(X_train, maxlen=max_len)
X_test_corner_nm = sequence.pad_sequences(X_test, maxlen=max_len)

X_train_corner_nm.shape, X_test_corner_nm.shape
```

Out [37]:
```
((30000, 100), (19995, 100))
```

In [38]:
```
# Converts a "pc_nm" to a sequence of indexes in a fixed-size hashing space
X_train = df_train.groupby('custid')['pc_nm'].apply(lambda x: [one_hot(i, max_features//100)[0] for i in x]).values
X_test = df_test.groupby('custid')['pc_nm'].apply(lambda x: [one_hot(i, max_features//100)[0] for i in x]).values

# Pads sequences to the same length
X_train_pc_nm = sequence.pad_sequences(X_train, maxlen=max_len)
X_test_pc_nm = sequence.pad_sequences(X_test, maxlen=max_len)

X_train_pc_nm.shape, X_test_pc_nm.shape
```

Out [38]:
```
((30000, 100), (19995, 100))
```

Private : 0.67907 **Public : 0.67615**

3. Round 2

앙상블

지금까지 만든 pred 중 0.7이상 파일만 가지고 앙상블

1. pred12 임벨런스 처리 없이 앙상블
2. pred7 tomeklink 앙상블
3. Pred1 gmean 앙상블
4. Pred1 Voting
5. 1차 1등 & 2등
6. LSTM

(※ 이때까지 pred8 tomek값이 나오지 않아서 함께 앙상블 못함)

Private : 0.73125 Public : 0.72338

3. Round 2

과정 정리

Data set	Imbalanced	Model	Ensemble
Pred1 . . . pred12	X TomekLink SMOTE	XGB LightGBM GBM NN CNN	Gmean Voting Powermean

3. Round 2

느낀점

1. Feature에 대한 이해가 중요함

2. ImbalInced, feature선택 등
   다양한 데이터셋 생성이 중요

3. 데이터간 유사도를 살펴 앙상블하는 것이 중요

4. 모든 과정에서 이루어진 A/B Test

THANK YOU

피드백은 살살 ^^