Instagram

@정유경 @김연지 @강창균

#이태원 맛집

#이태원

#맛스타그램

#ITEWON

인스타그램 태그를 이용한 실제 상권과

인스타그램 맛집의 위치 비교 분석

목차

Instagram

프로젝트 개요

프로젝트 배경

INSTAGRAM

Insta_matjip

최근 한달 내 이용 및 주 이용 소셜미디어

	전체					Gap	성별		연령별				
■ 주 이용 ■ 최근 한달 내 이용	2018년		2019년		2020년	주 이용 ('20-'19)	남	여	10대	20대	30대	40대	50대
Base	(500)		(500)		(500)		(310)	(310)	(120)	(124)	(126)	(124)	(126)
유튜브	27.6	78.8	32.6	83.8	37.6	89.2 +5.0	49.7	34.2	60.0	42.7	38.9	33.9	34.9
네이버 블로그	17.0	65.8	21.8	71.8	22.2	78.2 +0.4	15.2	21.3	1.7	12.9	19.0	29.8	27.0
인스타그램	14.8	48.4	19.2	53.0	19.0	58.6 -0.2	11.6	26.1	18.3	28.2	27.8	12.9	7.1
페이스북	15.6	60.2	9.0	55.2	7.0	53.0 -2.0	12.3	4.8	15.0	8.1	7.1	4.0	8.7
밴드	11.2	50.4	6.2	45.4	6.0	47.2 -0.2	6.1	3.5	0.0	1.6	4.8	4.8	12.7

평소 자주 보거나 공유하는 게시물

	전체	성별		연령별				
		남	여	10대	20대	30대	40대	50대
Base	(383)	(174)	(209)	(90)	(93)	(89)	(58)	(53)
맛집/음식	41.5	36.8	45.5	25.6	52.7	40.4	43.1	49.1
여행	28.5	31.0	26.3	20.0	33.3	21.3	29.3	45.3
일상생활	27.9	25.3	30.1	34.4	28.0	29.2	20.7	22.6
패션/의류/잡화	25.6	16.7	33.0	31.1	24.7	27.0	27.6	13.2
리빙/라이프스타일	19.3	14.9	23.0	4.4	19.4	27.0	24.1	26.4
TV/연예	18.0	19.0	17.2	27.8	20.4	10.1	20.7	7.5
뷰티	13.8	2.9	23.0	22.2	15.1	10.1	12.1	5.7
동물	13.3	10.3	15.8	17.8	8.6	16.9	12.1	9.4
스포츠	11.7	21.8	3.3	12.2	8.6	13.5	12.1	13.2
유머/개그	11.0	13.2	9.1	14.4	15.1	9.0	6.9	5.7
음악/뮤직비디오	10.7	13.2	8.6	18.9	10.8	10.1	1.7	7.5
운동/건강	10.4	12.6	8.6	6.7	8.6	13.5	12.1	13.2
도서/공연/문화	7.8	6.3	9.1	7.8	7.5	5.6	12.1	7.5
육아/교육	7.8	5.2	10.0	2.2	4.3	21.3	6.9	1.9

▲ [Base : 최근 1개월 내 인스타그램 이용자, N=383 , 단위 : %, 순위형 응답(1+2+3순위)] * 5% 이하인 경우 제시하지 않음 * 하늘색 음영 : 평균 대비 +5%P 이상인 데이터

출처 :https://www.cosinkorea.com/mobile/article.html?no=34894

문제점 :
길 건너 가면 가게가 있는데 지리적인 문제로 상권으로 포함되지 않은 경우가 다수이다.
즉, 국가 지정 발달 상권과 소비자 이용 맛집 장소가 차이가 존재

사실 :
1. 20-30대가 많이 이용하는 SNS : 인스타그램
2. 인스타그램에서 자주 공유하는 게시물 1위 : 맛집, 음식 관련 게시물
사진을 게시하기 위해 맛집을 일부러 찾아가는 경우가 있을 정도이다.

가정 :
진짜 맛집 데이터는 인스타그램에 더 많이 있지 않을까?

프로젝트 목표

이태원상권

그림 1 이태원 해시태그/위치정보를 활용한 상권지도

A구역 : 현재 #이태원맛집, #이태원핫플 등의 가장 많은 해시태그가 밀집된 지역

B구역 : #한남동맛집 이라는 신규 상권으로 #이태원디저트 #이태원브런치 등의 해시태그 밀집

C구역 : #우사단길이라는 떠오르는 상권으로 이태원상권과 이어지는 트렌드 상권으로 부상

1.
인스타그램 내 실제 이용 맛집 지역과
국가 지정 상권 지역 차이 확인

2.
실질적인 인스타그램 맛집 지도 예상

3.
맛집 지도의 비즈니스적 활용 방안 도출

Instagram

분석 계획 수립

분석 환경 및 사용 기술

처리툴

CO

pandas

jupyter

NumPy

RegEx

분석툴

Keras

python

TensorFlow 2.0

scikit learn

시각화툴

Word Clouds

Folium

QGIS® trademark

분석 절차

데이터 수집

인스타그램
공공데이터포털

전처리

해시태그/이모티콘 /특수기호 처리

EDA

라벨링 해주기
타입별로 비교

모델 학습

머신러닝 (SVM,NB,LR)
딥러닝 (RNN,LSTM,CNN)

예측

학습된 모델 바탕으로 type 예측

실제 상권 비교

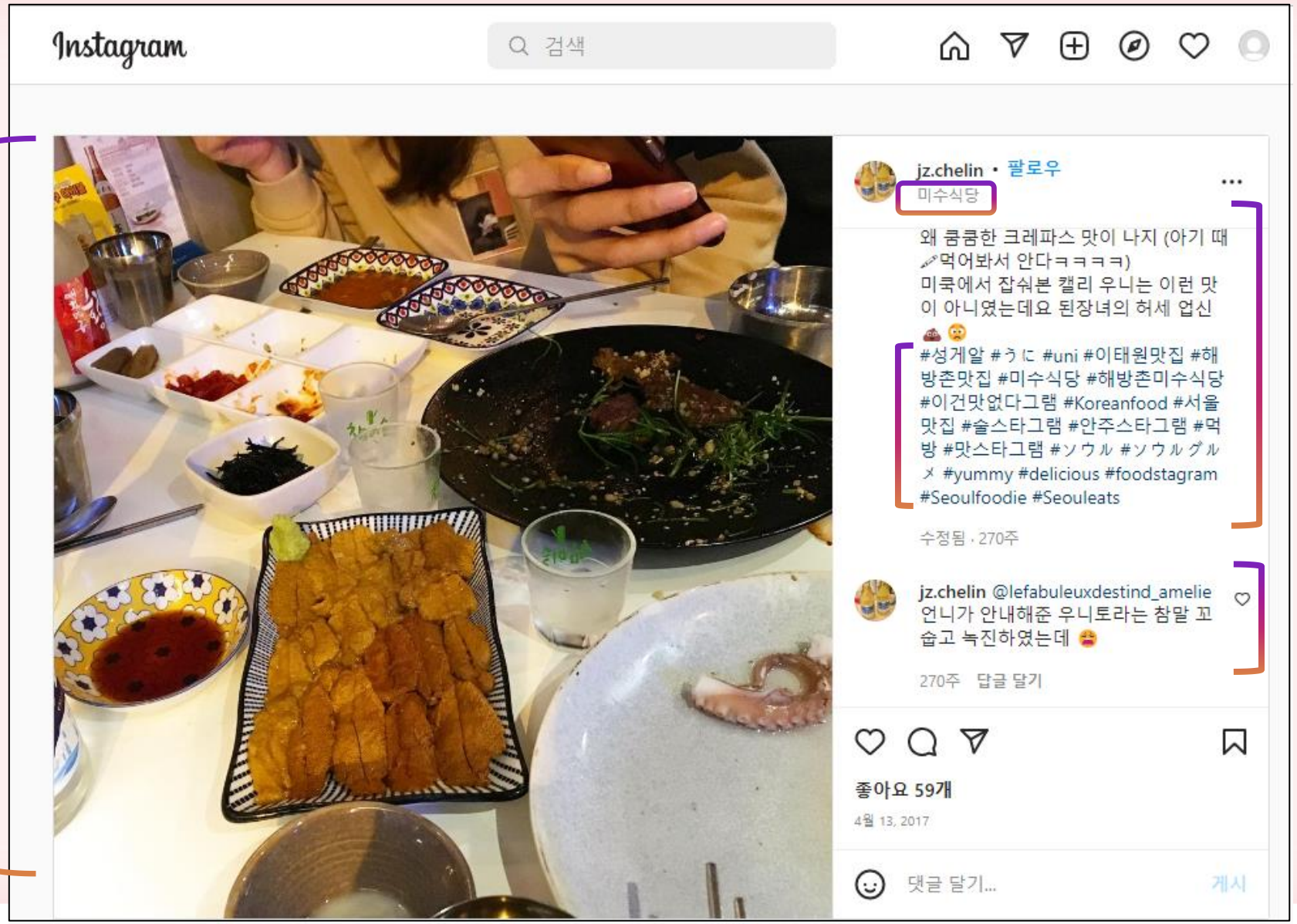
위치데이터화 상권비교
QGIS로 시각화

Instagram

데이터 분석 및 결과

데이터 수집

♦필수 게시
-사진
♦선택 게시
-위치
-본문
-태그
-댓글



데이터 수집

멘토님

Instagram

이태원 근처 데이터만

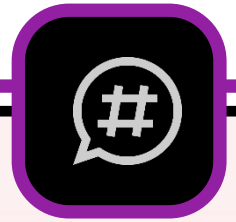
DATA 공공데이터포털 .GO.KR

실제 상권 데이터

전처리

RegEx

"#" 해시태그
빼주기



이모티콘
빼주기

예시:
<U+653C>

특수기호
들어간 것들
빼주기

라벨링 - Train set 형성

라벨링 0

#

"#" 해시태그 포함되지 않은 데이터

관련X

이태원맛집, 경리단길맛집, 해방촌맛집 등 이태원 주변 맛집을 태그 하지 않은 관련 없는 데이터

선정적

쩍벌녀,호빠,출장마사지, ...등 선정적인 글

광고성

1. 상담, 문의, 전화, 프로필링크 , 택배, 플러스친구,공구,대관, ..
2. 다이어트, 디톡스, 젤네일, 속눈썹, 강아지옷, 맛집 외 다른 광고글
3. 다른 지역 태그 EX) 홍대,강남, 부산 건대, 마포, 광화문, …

라벨링 1

나머지

라벨링 0에 해당되지 않은 데이터들은 라벨링 1

라벨링 - Train set 라벨 구성 확인

Insta_matjip Follow me

total

전체 데이터
이태원
이태원맛집
먹스타그램
우사단길
이태원카페
일상

Insta_matjip Follow me

type1

정확한 데이터
이태원
이태원맛집
이태원카페
먹스타그램
한남동맛집
경리단길카페

Insta_matjip Follow me

type0

광고 데이터
이태원
우사단길
이태원맛집
일상
맛팔
셀카

라벨링 - Validation 성능 확인

Train : Validation = 8 : 2

a
p

XGBoost가 가장 높은 성능을 가지는 이유가 뭘까?
여러 개의 약한 의사결정나무(Decision Tree)를 조합해서 사용하는 앙상블(Ensemble) 기법 중 하나이기 때문이다.

accuracy : 0.9608
precision : 0.9793
recall : 0.9653
f1 score : 0.9722

accuracy : 0.9511
precision : 0.9723
recall : 0.9585
f1 score : 0.9653

라벨링 - Validation 성능 확인 코드

1 RandomForest분류 모델

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier()
RF_clf = clf.fit(X_train_padded, y_train)
RF_pred = RF_clf.predict(X_test_padded)
RF_pred
print(accuracy_score(y_test, RF_pred))
print(precision_score(y_test, RF_pred))
print(recall_score(y_test, RF_pred))
print(f1_score(y_test, RF_pred))
```

```
executed in 3.02s, finished 15:30:36 2022-06-22
```

```
0.9169844529187445
0.9389312977099237
0.9573758339510748
0.9480638649293448
```

XGBOOST 분류 모델

```
xgb = XGBClassifier()
xgb.fit(train_X_vect,train_Y)
pred_Y = xgb.predict(test_X_vect)
print(accuracy_score(test_Y, pred_Y))
print(precision_score(test_Y, pred_Y))
print(recall_score(test_Y, pred_Y))
print(f1_score(test_Y, pred_Y))
```

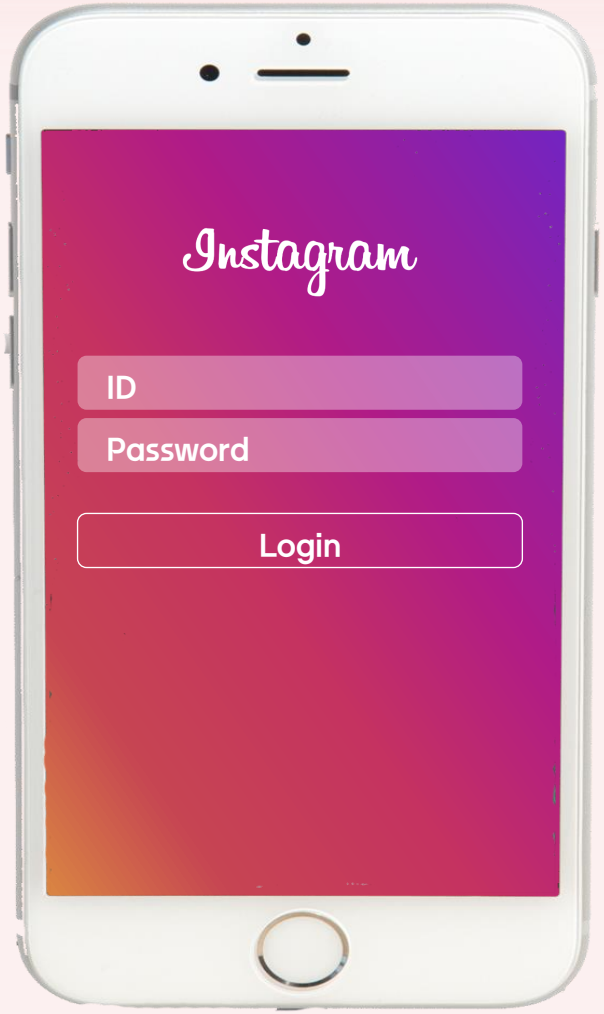
```
C:₩Users₩medici₩anaconda3₩lib₩site-packages₩
and will be removed in a future release. To
g XGBClassifier object; and 2) Encode your l
  warnings.warn(label_encoder_deprecation_ms

[15:19:16] WARNING: C:/Users/Administrator/w
lt evaluation metric used with the objective
like to restore the old behavior.
0.960844698636163 6
0.9792713567839196
0.9653250773993808
0.9722482070470845
```

로지스틱회귀모델

```
lr = LogisticRegression()
lr.fit(train_X_vect,train_Y)
pred_Y = lr.predict(test_X_vect)
print(accuracy_score(test_Y, pred_Y))
print(precision_score(test_Y, pred_Y))
print(recall_score(test_Y, pred_Y))
print(f1_score(test_Y, pred_Y))
```

```
0.9511658600967884
0.9723618090452262
0.958513931888545
0.9653882132834425
```

최종 데이터 구성도



‘ id ’
인스타 링크

‘ word’
피드 내용과
첫번째 댓글 까지 내용

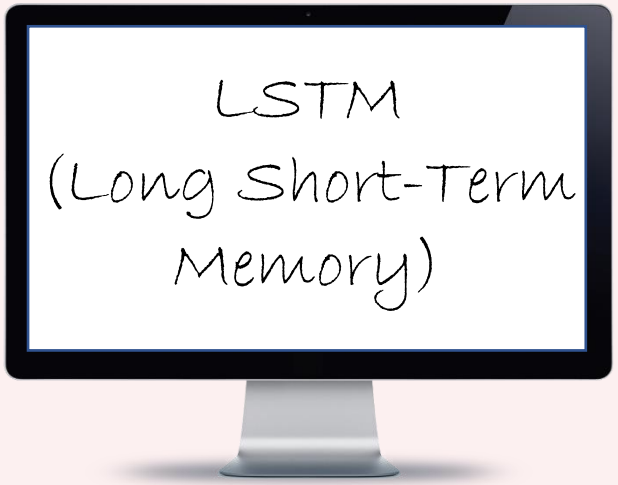
‘ type ’
정확한 데이터 : 1
광고성 데이터 : 0

‘ 위치 ’
가게 위치 : 위도 경도

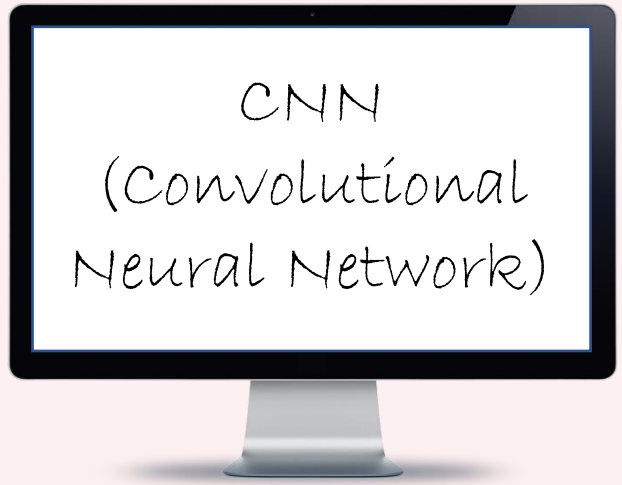
‘ 상권 ’
나라에서 지정한
상권 구역 지도 데이터

모델링

accuracy: 0.8678
precision: 0.8768
recall: 0.9226
f1 score: 0.8970

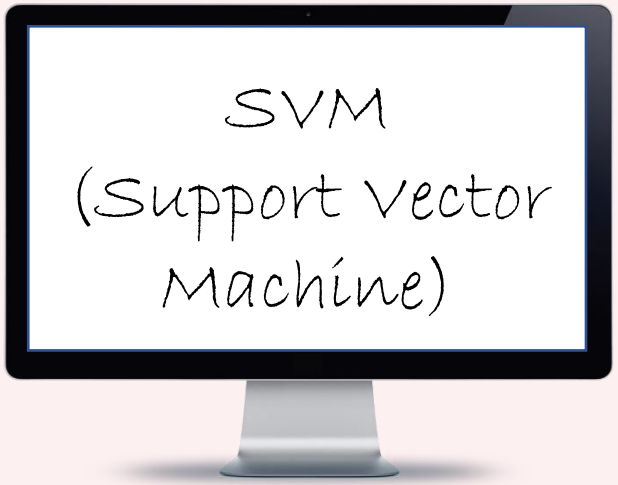


accuracy: 0.8875
precision: 0.8872
recall: 0.9414
f1 score: 0.9117



accuracy: 0.8894
precision: 0.8897
recall: 0.9424
f1 score: 0.9134

모델링



accuracy : 0.8905
precision : 0.8942
recall : 0.9395
f1 score : 0.9163

accuracy : 0.8568
precision : 0.8452
recall : 0.9494
f1 score : 0.8943

accuracy : 0.8848
precision : 0.8915
recall : 0.9320
f1 score : 0.9113

모델링 코드

2 RNN

```
from tensorflow.keras.layers import SimpleRNN, Embedding, Dense
from tensorflow.keras.models import Sequential
import tensorflow as tf
embedding_dim = 30
hidden_units = 50

tf.random.set_seed(1234)
model = Sequential()
model.add(Embedding(vocab_size, embedding_dim))
model.add(SimpleRNN(hidden_units))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy', precision, recall, f1score])
history = model.fit(X_train_padded, y_train, epochs=4 ,batch_size=64, validation_split=0.3)
```

executed in 5.79s, finished 09:24:14 2022-06-13

```
print("\n 테스트 f1score: %.4f" % (model.evaluate(X_test_padded, y_test)[4]))
```

executed in 366ms, finished 09:24:15 2022-06-13

```
98/98 [==============================] - 0s 3ms/step - loss: 0.3685 - accuracy: 0.8678 - precision: 0.8768 - recall: 0.9226 - f1
score: 0.8970

 테스트 f1score: 0.8970
```

3 LSTM

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, SimpleRNN, LSTM, Embedding
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

tf.random.set_seed(1234)
model = Sequential()
model.add(Embedding(vocab_size, 60))
model.add(LSTM(60))
model.add(Dense(1, activation='sigmoid'))

# 테스트 데이터 손실함수값(val_loss)이 patience회 이상 연속 증가하면 학습을 조기 종료하는 콜백
early_stop = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=5)
# 훈련 도중 테스트 데이터 정확도(val_acc)가 높았던 순간을 체크포인트로 저장해 활용하는 콜백
model_check = ModelCheckpoint('the_best.h5', monitor='val_acc', mode='max', verbose=1, save_best_only=True)

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy', precision, recall, f1score])
model.fit(X_train_padded, y_train, validation_split=0.3 ,epochs=5, batch_size=64, callbacks=[early_stop, model_check])
```

executed in 18.5s, finished 09:47:44 2022-06-13

```
print("\n 테스트 f1score: %.4f" % (model.evaluate(X_test_padded, y_test)[4]))
```

executed in 5.66s, finished 15:49:01 2022-06-02

```
98/98 [==============================] - 6s 57ms/step - loss: 0.3912 - accuracy: 0.8875 - precision: 0.8872 - recall: 0.9414 - f
1score: 0.9117

 테스트 f1score: 0.9117
```

5 CNN

```
from tensorflow.keras.layers import Dense, Conv1D, GlobalMaxPooling1D, Embedding, Dropout, MaxPooling1D
from tensorflow.keras.models import Sequential
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
import tensorflow as tf

embedding_dim = 70
dropout_ratio = 0.2
num_filters = 60
kernel_size = 20

tf.random.set_seed(1234)
model = Sequential()
model.add(Embedding(vocab_size, embedding_dim))
model.add(Dropout(dropout_ratio))
model.add(Conv1D(num_filters, kernel_size, padding='valid', activation='relu'))
model.add(GlobalMaxPooling1D())
model.add(Dropout(dropout_ratio))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy', precision, recall, f1score])

es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=3)
mc = ModelCheckpoint('best_model.h5', monitor = 'val_acc', mode='max', verbose=1, save_best_only=True)

history = model.fit(X_train_padded, y_train, epochs=4, batch_size=64, validation_split=0.2, callbacks=[es, mc])
```

executed in 11.7s, finished 10:13:00 2022-06-13

```
print("\n 테스트 f1score: %.4f" % (model.evaluate(X_test_padded, y_test)[4]))
```

executed in 371ms, finished 10:29:19 2022-06-13

```
98/98 [==============================] - 0s 3ms/step - loss: 0.3176 - accuracy: 0.8894 - precision: 0.8897 - recall: 0.9424 - f1
score: 0.9134

 테스트 f1score: 0.9134
```

모델링 코드

SVC 모델링

```
from sklearn.svm import SVC
svc = SVC(probability=True)
svc.fit(train_X_vect, train_Y)
pred_Y_svc = svc.predict(test_X_vect)
print("정확도 : ",accuracy_score(test_Y, pred_Y_svc))
print("정밀도 : ",precision_score(test_Y, pred_Y_svc))
print("재현율 : ",recall_score(test_Y, pred_Y_svc))
print("f1스코어 : ",f1_score(test_Y, pred_Y_svc))
```

```
정확도 :  0.890549662487946
정밀도 :  0.8942446043165467
재현율 :  0.9395313681027967
f1스코어 :  0.9163287873203096
```

NB 모델링

```
from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB()
nb.fit(train_X_vect, train_Y)
pred_Y_nb=nb.predict(test_X_vect)
print("정확도 : ",accuracy_score(test_Y, pred_Y_nb))
print("정밀도 : ",precision_score(test_Y, pred_Y_nb))
print("재현율 : ",recall_score(test_Y, pred_Y_nb))
print("f1스코어 : ",f1_score(test_Y, pred_Y_nb))
```

```
정확도 :  0.8567984570877532
정밀도 :  0.845222072678331
재현율 :  0.9493575207860923
f1스코어 :  0.8942684229263084
```

LR 모델링

```
from sklearn.linear_model import LogisticRegression

# 분류용 평가 함수
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import roc_auc_score

lr = LogisticRegression()
lr.fit(train_X_vect,train_Y)
pred_Y = lr.predict(test_X_vect)
print("정확도 : ",accuracy_score(test_Y, pred_Y))
print("정밀도 : ",precision_score(test_Y, pred_Y))
print("재현율 : ",recall_score(test_Y, pred_Y))
print("f1스코어 : ",f1_score(test_Y, pred_Y))
```

```
정확도 :  0.8842815814850531
정밀도 :  0.8915401301518439
재현율 :  0.9319727891156463
f1스코어 :  0.9113082039911309
```

모델링 - 모델 선택

SVM
(Support Vector Machine) + CountVectorizer

f1 score : 0.9163

CNN
(Convolutional Neural Network) + Tokenizer

f1 score : 0.9134

Naïve Bayes Classifier
각 토큰의 확률값을 곱해서 라벨을 결정하는 알고리즘으로 토큰이 많아지면 0으로 판단할 확률이 높아진다.

Logistic Regression
평면에 선형으로 데이터를 시상한다.

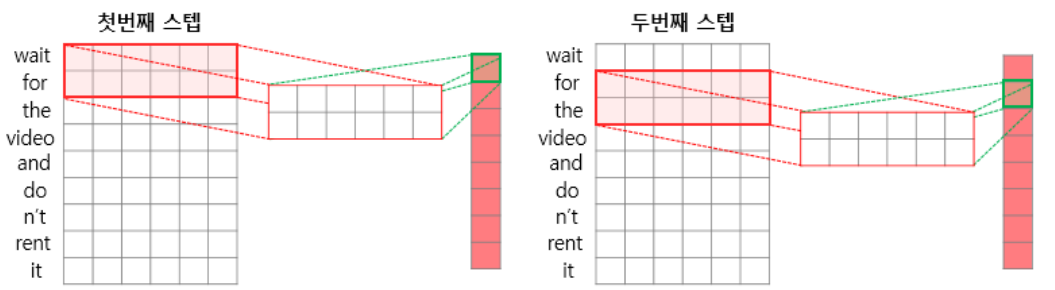
SVM은 고차원의 공간에 데이터를 시상할 수 있다.

RNN & LSTM
시계열 데이터에 적합한 알고리즘을 가지고 있어, 시계열 데이터가 아닌 태그 데이터에 적합하지 않다.

CNN은 이미지 처리에 탁월한 성능을 보이는 신경망이지만, 텍스트 처리도 가능하다. CNN의 첫 단계는 '특징'에 대해 학습하고, 두 번째 단계는 학습한 특징을 기반으로 '분류' 한다.

불균형 데이터라서
정확도는 사용하기 힘들고,
정밀도와 재현율은
trade off 관계라서
둘 다 좋은 점수를 맞아야 좋은 모델이므로
f1 score를 선택했다.

CNN 추가 설명



모델링 - 단어 임베딩 선택

프로젝트 데이터
1. 시계열 데이터가 아님
2. 문장이 아님

→ Word2Vec, GloVe, FastText,ELMo,GPT,BERT 등을 사용하지 않은 것은 문맥을 고려하는 임베딩 방법에 적합한 데이터가 아니기 때문이다.

→ CountVectorizer 방법 사용 or Tokenizer 방법 사용

TDIDFVectorizer vs CountVectorizer

TDIDF 방식은 한 문서 당(인스타 글 하나 당) 자주 나오는 단어가 다른 문서 (인스타 글)들에서도 자주 나오면 해당 단어의 가중치를 줄여버린다.
하지만, 우리의 데이터는 맛집, 카페 와 같이 다른 문서들에도 자주 나오는 단어들이 오히려 중요한 단어이므로 가중치가 줄어들면 안된다.

→ 오히려 자주 나올수록 가중치를 더 주는 CountVectorizer 임베딩 방법이 더 적합하다.

Out Of Vocabulary (OOV)가 있는데에도 머신이 제대로 돌아간 이유
CountVectorizer 같은 경우:
train으로 학습하고 test에 적용할 때, OOV가 나오면, 그 단어는 무시한 채 처리해주는 형태의 알고리즘

Tokenizer 같은 경우:
oov_Token라는 파라미터를 이용해서 전부 해당 인덱스로 임베딩 시킨다.

EDA - 틀린 10% 분석

예측 틀린 10%

5%는
데이터 문제

Type 설정
오류

내용 오류

5%는
머신 성능 문제

Instagram

예측을 잘못한
것 중 모든
머신이 같은
예측을 했다.

Instagram

예측을 잘못한
것 중 머신이
각각 다른
예측을 했다.

EDA - 틀린 10% 분석

Instagram

Type 설정 오류 예시

└>type은 “0”이라 되어있고 예측은 “1” 혹은 type은 “1” 예측은 “0”이라 해서 예측값이 오히려 맞는 상황

EDA - 틀린 10% 분석

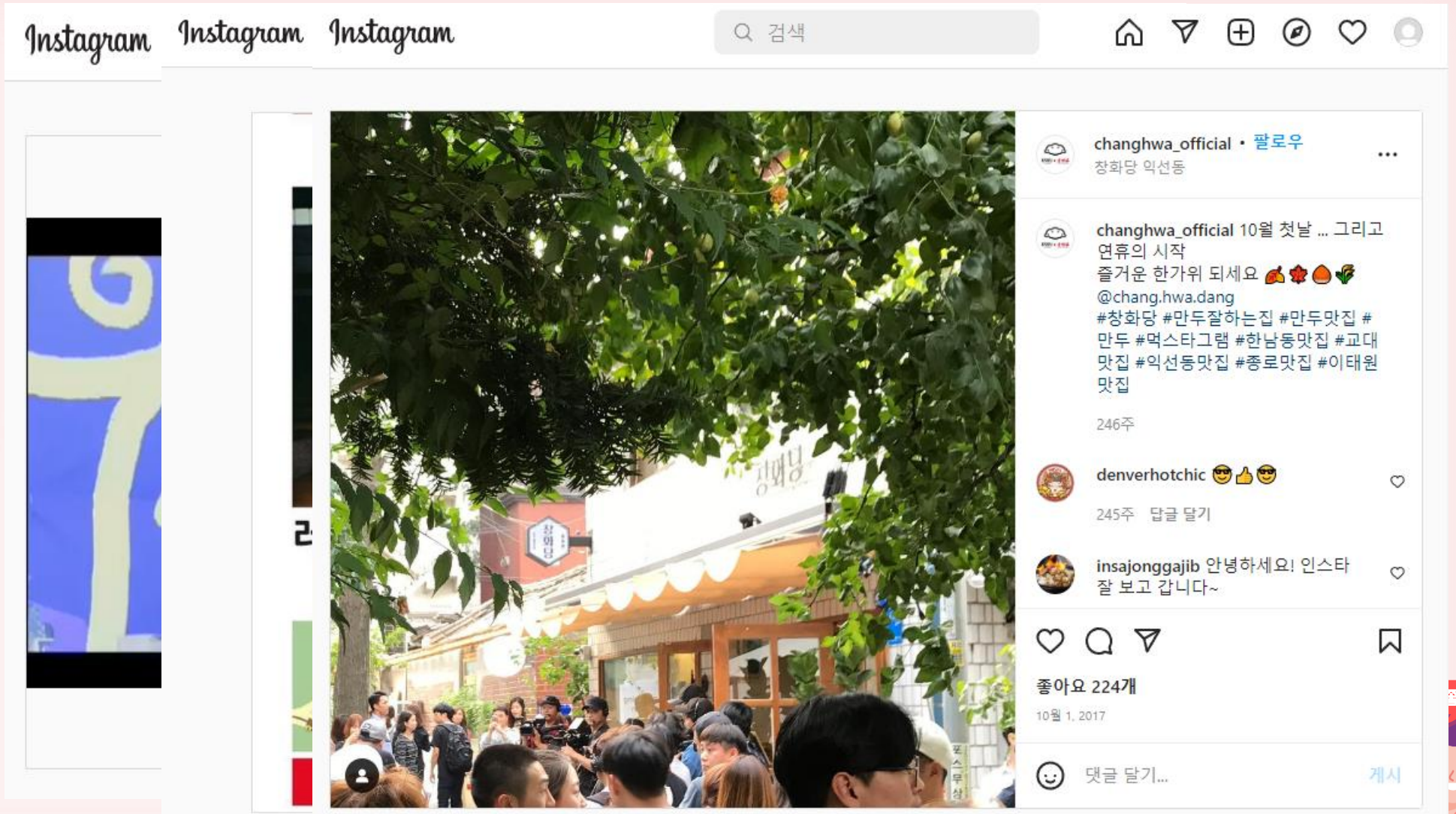
Instagram

내용 오류
Test type 0
예측 1
해시태그
키워드

└> 페스티벌 홍보인데 해시 태그에는 카페 내용을 적어서 “1”이라 잘못 예측

└> 본문 내용은 고양이를 찾는 것인데 사람들 많이 보게 하기 위해 이태원 맛집들을 태그해서 “1” 이라 잘못 예측

└> 익선동 맛집인데 이태원맛집, 한남동 맛집 태그해서 “1” 이라 잘못 예측



EDA - 틀린 10% 분석

Instagram

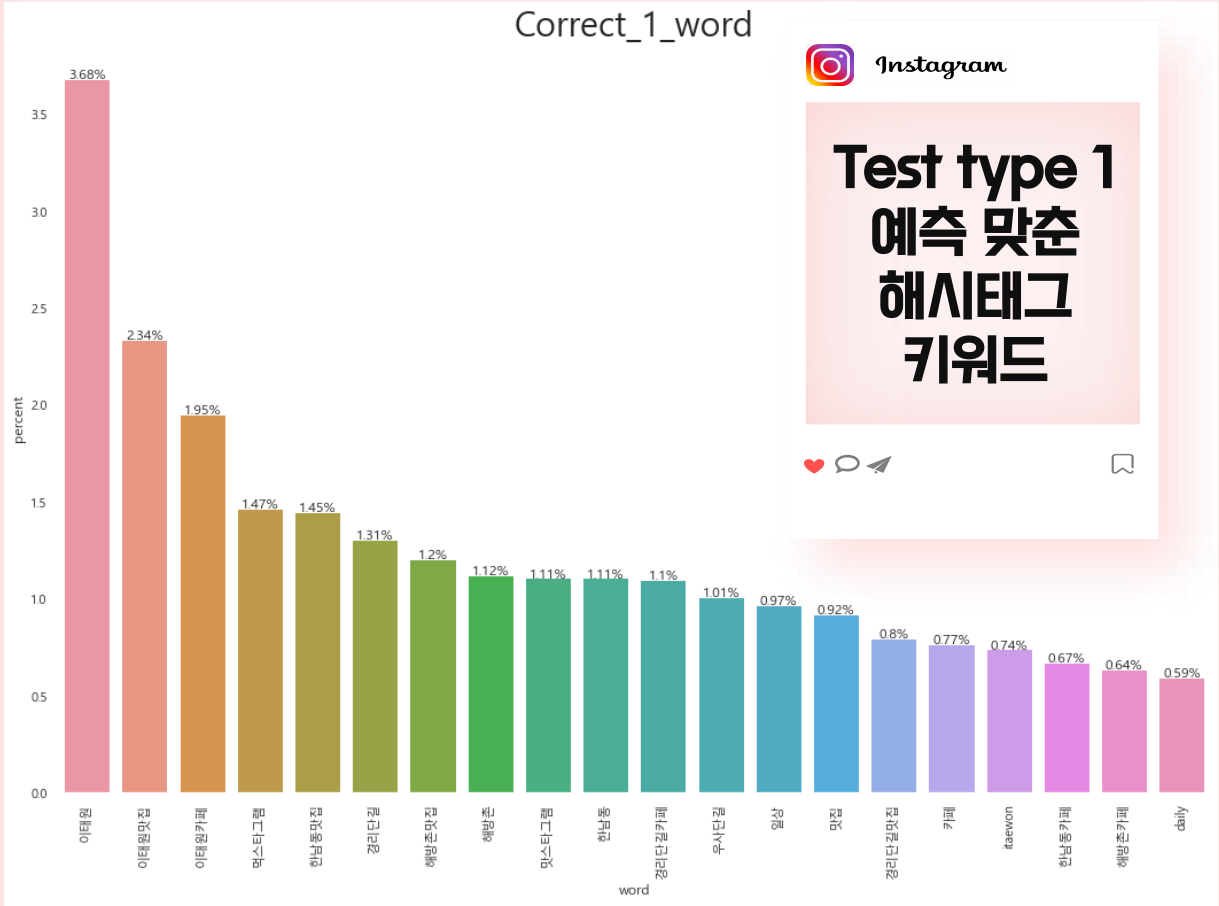
내용 오류 Test type 1 예측 O 해시태그 키워드

↳ 내용은 카페 맛집이지만 태그는 셀카에 관련 이라서 "O"이라 잘못 예측

↳ 댓글에는 태그내용이 추가되어 있지만 본문 내용에는 이태원 관련 없어 "O"이라 잘못 예측

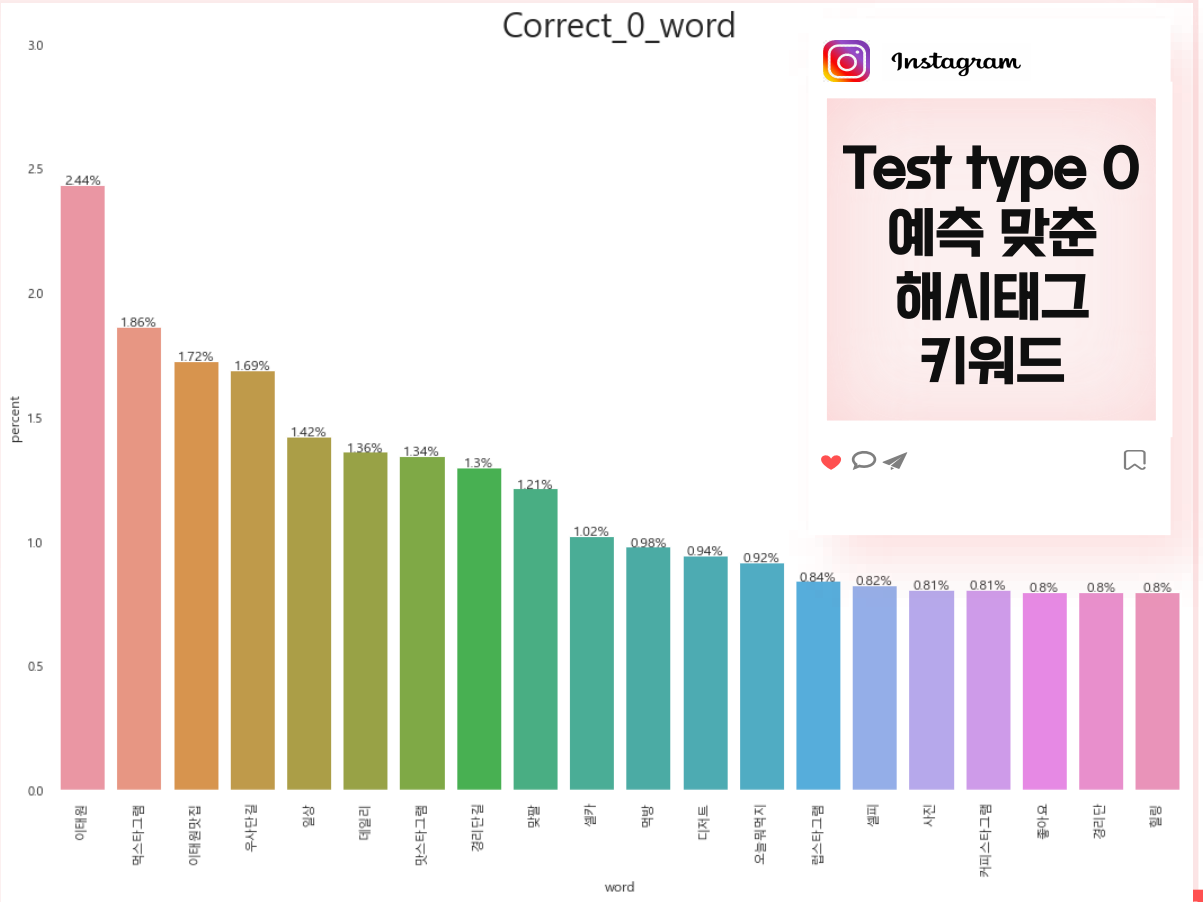
↳ 실제는 "1"이지만 댓글에는 광고성 태그를 달아서 "O" 으로 잘못 예측

EDA - 맞은 90% 분석



Instagram

Test type 1
예측 맞춘
해시태그
키워드

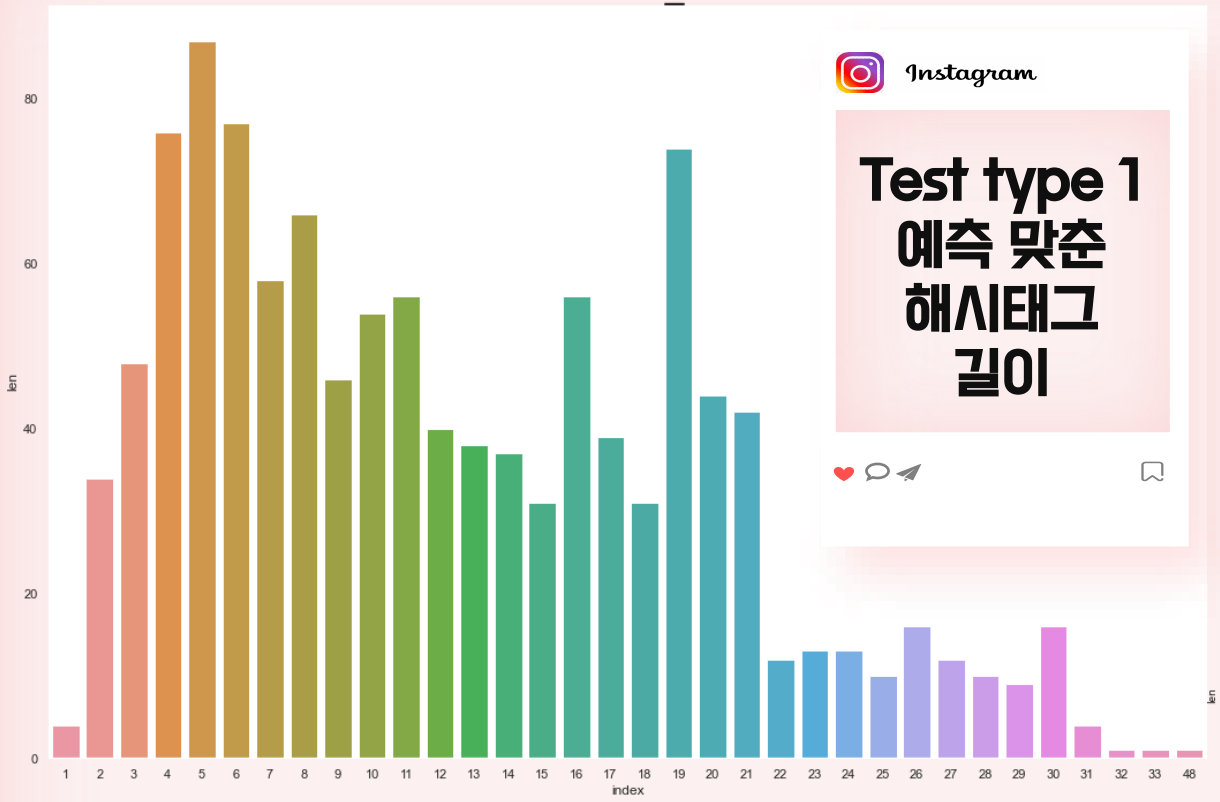


Instagram

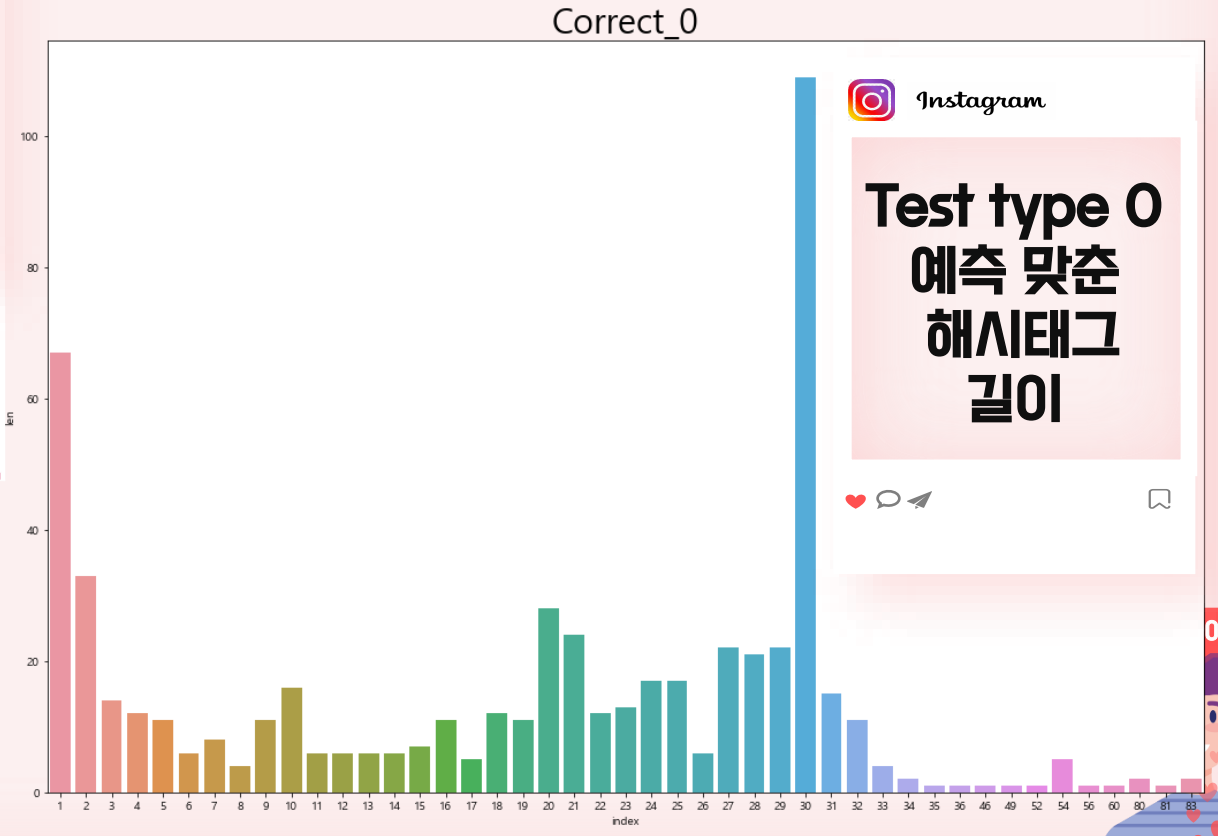
Test type 0
예측 맞춘
해시태그
키워드

EDA - 맞은 90% 분석

Correct_1



Correct_0

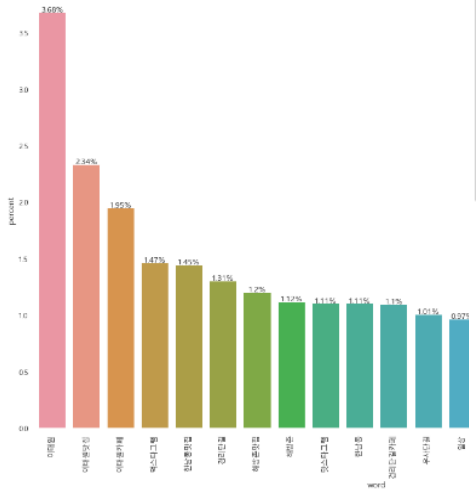


EDA - 맞은 90% 분석 시각화 코드

Test 해시태그 키워드

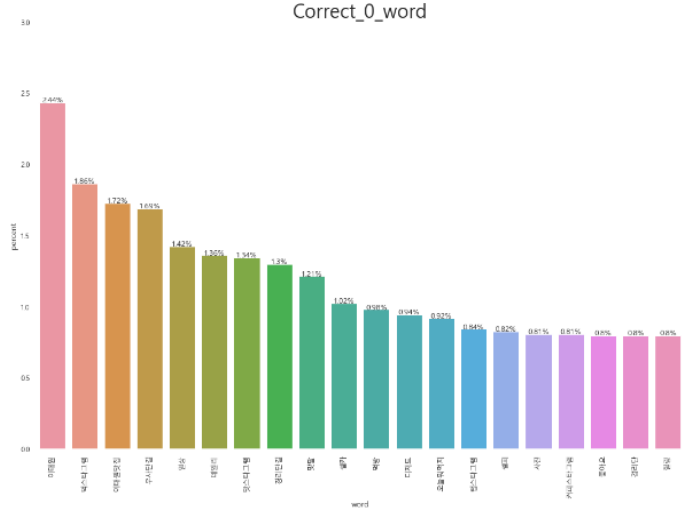
```
plt.figure( figsize = (18,12) )
sns.barplot(x = AA.word, y = AA.percent).set_title('Correct_1_word', fontsize = 30)
plt.xticks( rotation=90)
plt.gca().set_facecolor('white')
ncnt=0
for i, sr in AA.iterrows():
    plt.text(ncnt, sr['percent'], f"{round(sr['percent'],2)}%", ha='center')
    ncnt +=1
plt.show()
```

Correct_1_word



```
plt.figure( figsize = (18,12) )
sns.barplot(x = BB.word, y = BB.percent).set_title('Correct_0_word', fontsize = 30)
plt.xticks( rotation=90)
plt.ylim([0,3])
plt.gca().set_facecolor('white')
ncnt=0
for i, sr in BB.iterrows():
    plt.text(ncnt, sr['percent'], f"{round(sr['percent'],2)}%", ha='center')
    ncnt +=1
plt.show()
```

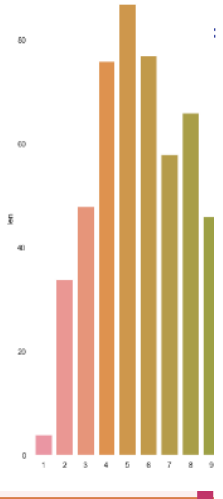
Correct_0_word



Test 해시태그 길이

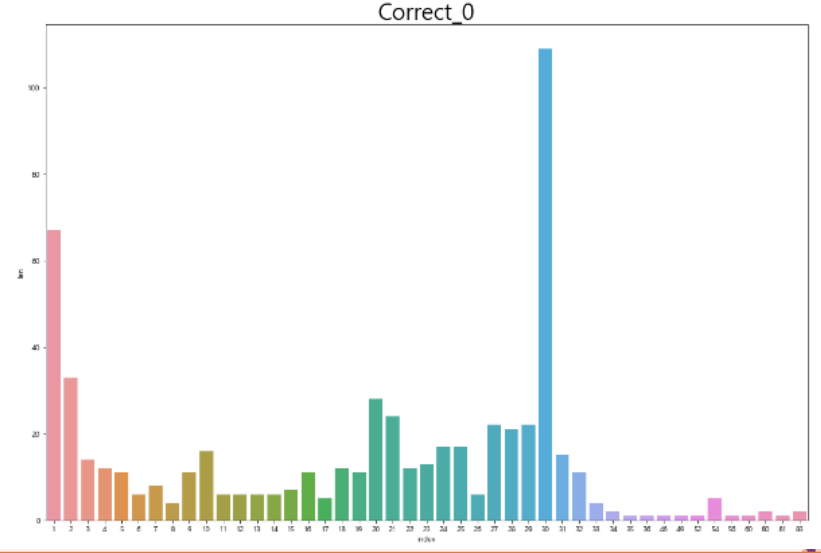
```
plt.figure( figsize = (18,12) )
sns.set()
sns.barplot(x = correct_1_cnt.index, y = correct_1_cnt.len).set_title('Correct_1', for
plt.xlabel( 'index')
plt.gca().set_facecolor('white')
plt.show()
```

Correct_1



```
plt.figure( figsize = (18,12) )
# plt.xticks( rotation=90)
sns.barplot(x = correct_0_cnt.index, y = correct_0_cnt.len).set_title('Correct_0', for
plt.xlabel( 'index')
plt.gca().set_facecolor('white')
plt.show()
```

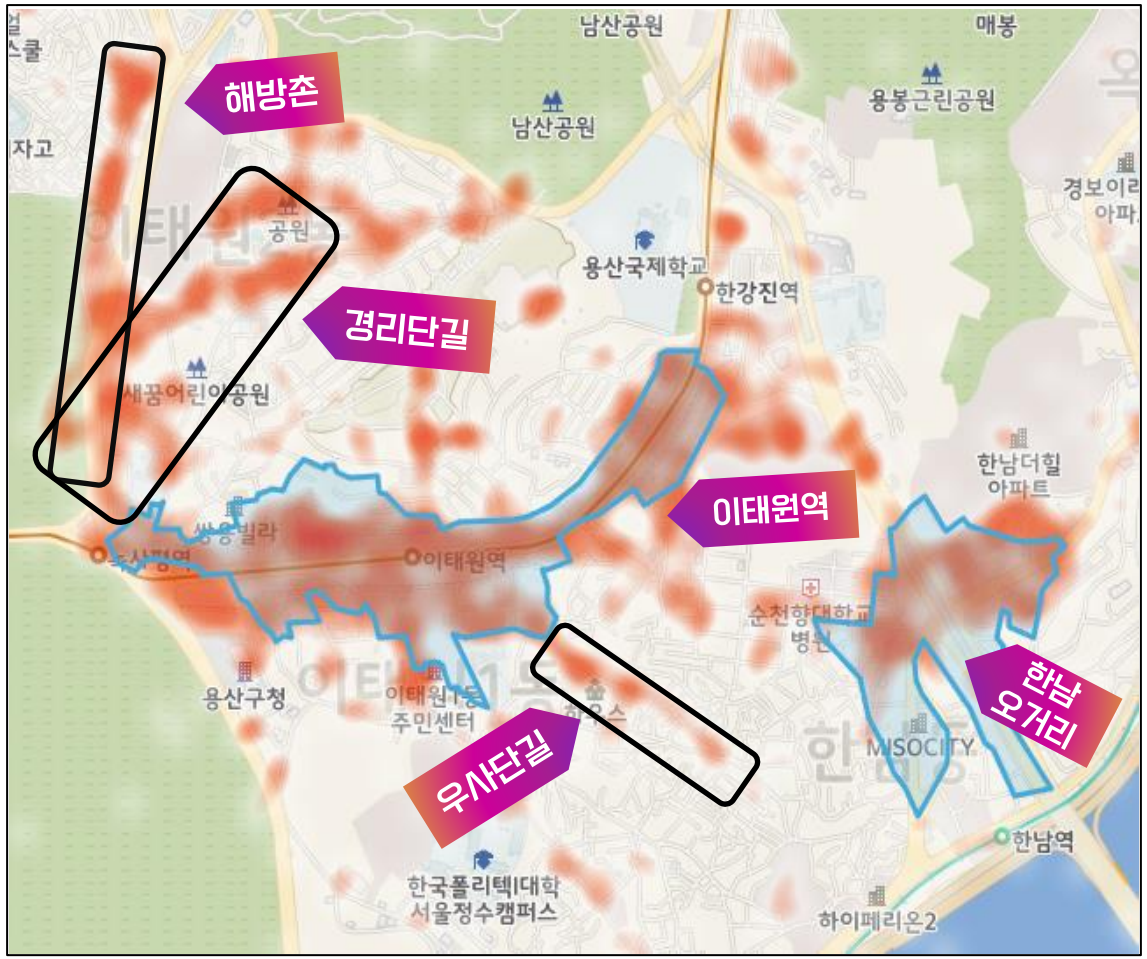
Correct_0



결론(실제 상권과 인스타그램 맛집 비교)

INSTAGRAM

Insta_matjip



*공데 상권 지도와 인스타 지도 데이터의 차이점

1.경리단길, 해방촌을 발달 상권으로 추가

2.우사단길을 발달 상권으로 추가

3.이태원역 근처 발달상권 구체적으로 넓게 재구획화

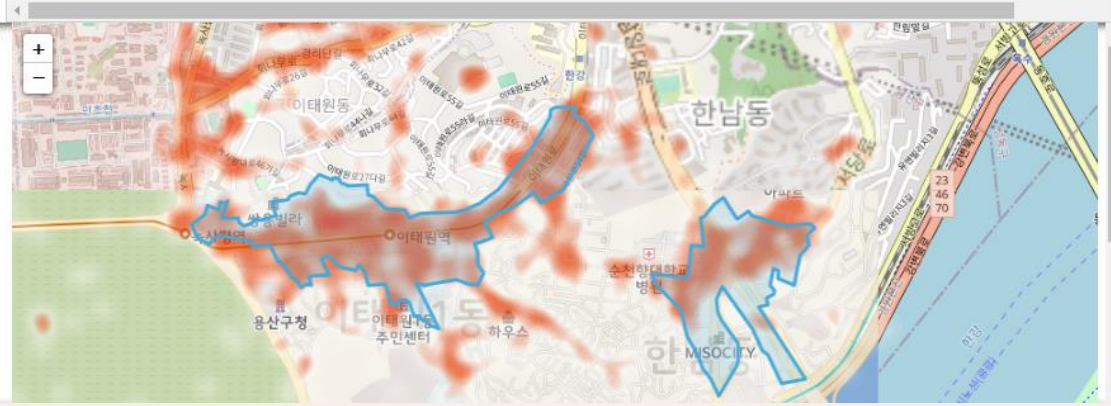
4.한남동 발달상권 아래부분은 잘라내고 위에 붙어진 부분을 더 넓게 구획화

5.다른 옅은 부분들은 아파트나 상가지역으로 상권보다는 배달로 인한 위치 태그로 판단되어 제외

결론(지도 시각화 코드 - 포리움)

In [38]:
```
from folium.plugins import HeatMap
map = folium.Map(location=[ sum(location['latitude'])/len(location['latitude']), sum(location['longitude'])/len(location['lon
                           zoom_start=15)
# 인스타 지도
# for j in range(len(location)) :
#     folium.Circle(
#     location = [location.latitude[j], location.longitude[j]],
#     popup = location.name[j], color='black', radius=5).add_to(map)

folium.GeoJson(roads,
               style_function=lambda feature:{'fillColor' : '#44a9db',
                                              'color': '#44a9db'}).add_to(map)
# 지도 타일 바꾸기
vworld_key='3542FD13-9702-395C-87E3-FEBB361F057B'
layer='Base'
tileType = 'png'
tiles = f"http://api.vworld.kr/req/wmts/1.0.0/{vworld_key}/{layer}/{{z}}/{{y}}/{{x}}.{tileType}"
attr="Vworld"
folium.TileLayer(
tiles=tiles,
attr=attr,
overlay=True,
control=True).add_to(map)

# 히트맵으로 시각화 하기
heatmap = HeatMap(location.drop(['link','name'],axis=1), radius=8, blur=13, gradient={0.4:'white',0.65:'#e8430c',1:'red'})
map.add_child(heatmap)

# 누르면 위,경도 볼 수 있게 하기
map.add_child(folium.LatLngPopup())

# 지도 저장하기
# map.save('공대지도vs인스타지도.html')

map
```

Out[38]:



결론 및 시사점

위치기반서비스(LBS) , GIS 이용하는 스타트업 회사들은 내부 데이터가 쌓이지 않았기 때문에 공공기반 상권데이터를 사용할 수 밖에 없다. 공공기반 상권데이터는 실제와 맞지 않는 부분들이 있기 때문에 우리가 만든 상권 지도를 사용하면 실질적인 상권 구역을 이용하게 되는 것이므로 비즈니스적으로 더 높은 활용 가치가 있을 것으로 예상된다.

혹은 경쟁력이 약화되고 있는 소상공인 분들에게 신뢰도 높고 정확한 트렌드 정보를 제공해 창업, 마케팅 등에 실질적으로 도움이 될 수 있다. 지역적 잠재력을 발굴하여 경쟁력을 확보하고, 지역 발전에 기여할 수 있을 것으로 예상된다.

EX) 맛집 추천 어플 사이트

배달 사이트

데이트 코스 추천 어플

결혼정보회사 등

Instagram

프로젝트 마무리

한계점 및 개선방향

한계점

- 인스타그램 데이터를 직접 가지고 오지 못 한 것이 아쉬웠다.
- 2017년 인스타그램 데이터였기 때문에 코로나로 인해 폐업한 곳들도 있고 현재와 달라진 곳들이 존재한다.
- 라벨링을 더 많이 못 해서 아쉬웠다.

개선방향

- 단어별 혹은 좋아요 수, 댓글 수로 가중치를 두어 SVD 계산을 할 수 있었다면 좋았을 것 같다.
- 인스타 맛집 지도가 히트맵이 아니라 폴리곤(Polygon) 형식으로 변환하여 표현하였으면 좋았을 것 같다.

Instagram

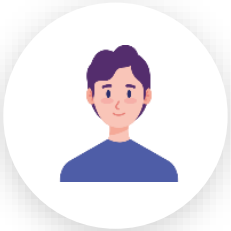
THANK YOU

#메디치

#수고하셨습니다:D

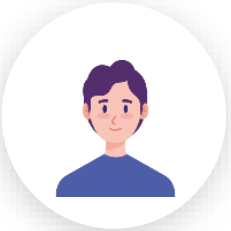
INSTAGRAM

팔로잉 | 내 소식



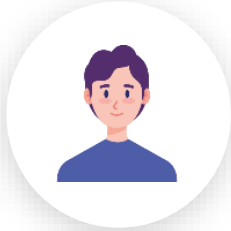
마지막_조님이 회원님을 팔로우 하기 시작했습니다.

팔로잉



마지막_조님이 댓글에서 회원님을 언급했습니다 :

@insta_matjip Q&A 시간입니다



마지막_조님이 댓글을 남겼습니다 :

질문이 있으시다면 질문해주세요

MEDICI님이 회원님을 팔로우 하기 시작했습니다.

팔로우