

2020 AI

자치구별 상권 변화 예측

2020 AI Defense Game

경영학과

16010064 이주희



CONTENTS

001 프로젝트 소개

002 데이터 소개

003 데이터 가공 및 전처리

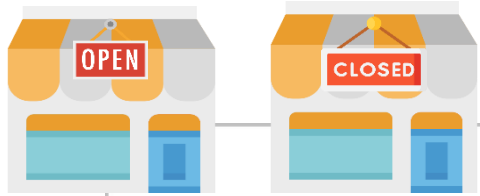
004 베이스라인 구축

005 캐글 리더보드 제작



1. 프로젝트 소개

PART.1



Purpose

서울 25개의 자치구에 대해서 분기별 개·폐점률에 따른 상권변화 정도를 예측 분류한다.

프로젝트 선정 이유

코로나로 인한 상권마비가 사회의 큰 문제로 대두되고있는 가운데, 도움이 되는 분석 및 예측문제를 선정하고자 하였다. 근본적인 원인 해결은 불가능하지만, 각 상권에 대한 이해와 분석이 분명 긍정적으로 작용할 것이다.

마침 서울시의 우리마을가게 상권분석서비스에서 지역/상권별 현황과 동향, 점포 분석 등의 여러가지 정보를 제공하고 있음을 알게 되었고 이를 활용하게 되었다.



1. 프로젝트 소개

Research

문제해결을 위한 조사



상권 변화에 대한 이해

다이나믹: 점포 개·폐점률이 높다.

정체: 점포 개·폐점률이 낮다.

- 다양한 업종의 점포가 집적한 상권에서는 점포 교체가 **다이나믹**하게 일어난다.
- 반면, 동종 점포가 집적한 상권의 경우 점포 변화가 **정체**된 특징을 나타낸다.



2. 데이터 소개

PART.2



Guest님 로그인

제공정보안내

이용방법

댓글과 대화해 보세요



창업신희동

서울 상권 검색

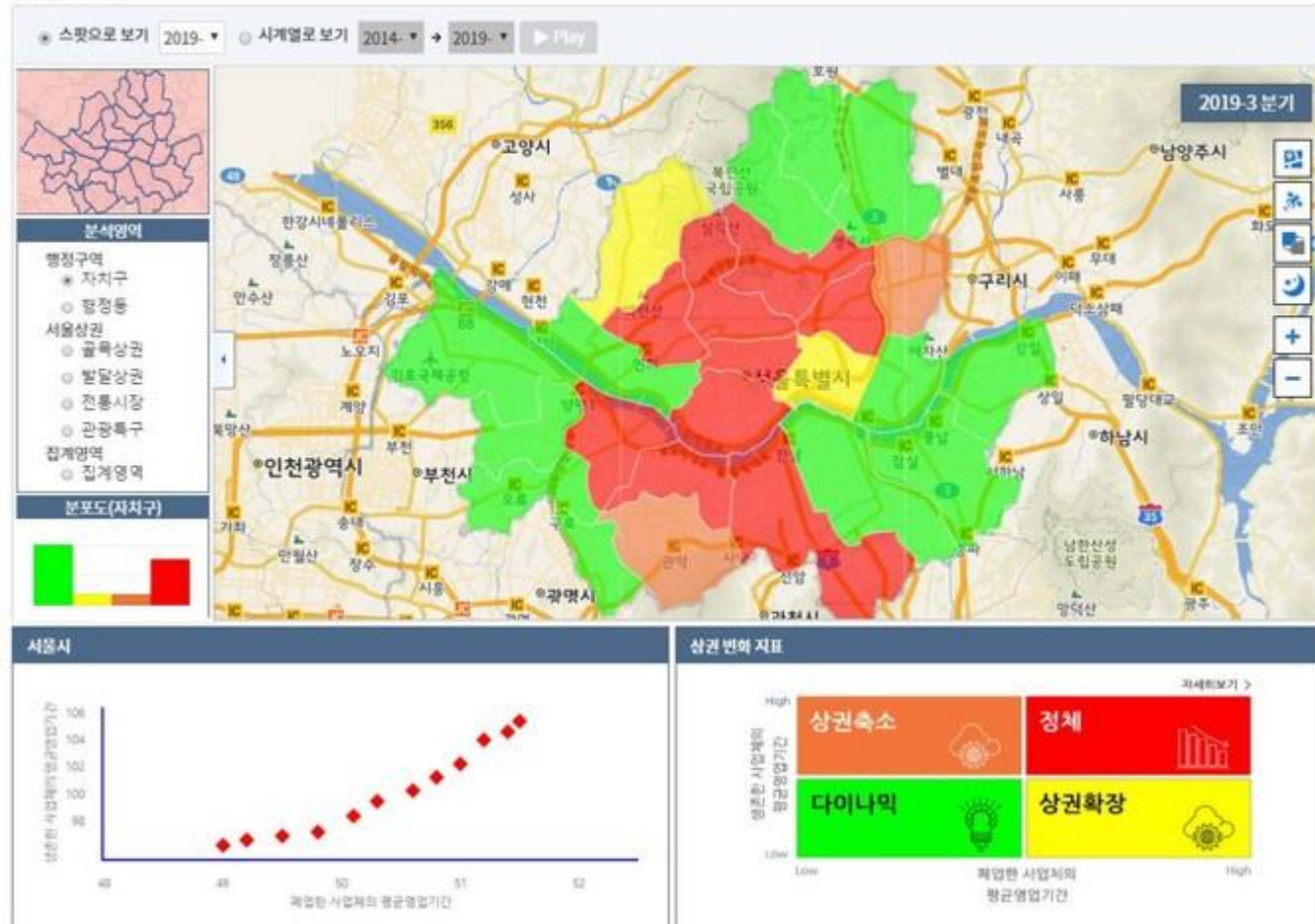
내 점포 분석

지역별 현황

상권 동향

도움말

상권변화 시각화



출처 서울 열린데이터광장

서울시 우리마을가게 상권분석서비스(자치구별 상권변화지표)

- 서울시의 우리마을가게 상권분석서비스에서 제공하는 자치구별 상권변화지표이다.
- 서울 25개의 자치구에 대해서 운영 영업개월 평균, 폐업 영업개월 평균에 대한 데이터를 제공한다.
- 이를 통해 분기별 각 자치구의 개·폐점률과 상권 변화 정도를 예측할 수 있다.



2. 데이터 소개

	기준_년_코드	기준_분기_코드	시군구_코드	시군구_코드_명	상권_변화_지표	상권_변화_지표_명	운영_영업_개월_평균	폐업_영업_개월_평균	서울_운영_영업_개월_평균	서울_폐업_영업_개월_평균
0	2019	4	11740	강동구	LL	다이나믹	105	50	110	54
1	2019	4	11710	송파구	LL	다이나믹	110	53	110	54
2	2019	4	11680	강남구	LL	다이나믹	106	50	110	54
3	2019	4	11650	서초구	HH	정체	118	54	110	54
4	2019	4	11620	관악구	LL	다이나믹	108	52	110	54

Target value

613 rows * 10개의 Columns

- 기준 년 코드: 2014년 1분기 ~ 2019년 4분기
- 시군구 코드: 자치구 25개
- 상권 변화 지표 다이나믹(LL), 상권축소(HL), 상권확장(LH), 정체(HH)
- 운영 영업 개월 평균, 폐업 영업 개월 평균: 해당 자치구의 평균적인 운영, 폐업 개월 수
- 서울 운영 영업 개월 평균, 서울 폐업 영업 개월 평균: 서울의 평균적인 운영, 폐업 개월 수



3. 데이터 가공 및 전처리

PART.3

1) 불필요한 컬럼, 예측에 방해되는 컬럼 제거

```
[ ] df.columns
```

```
↳ Index(['기준_년_코드', '기준_분기_코드', '시군구_코드', '시군구_코드_명', '상권_변화_지표', '상권_변화_지표_명',  
        '운영_영업_개월_평균', '폐업_영업_개월_평균', '서울_운영_영업_개월_평균', '서울_폐업_영업_개월_평균'],  
        dtype='object')
```

```
[ ] # 시군구 코드와 시군구 코드명은 같은 정보를 가지고 있으므로 둘 중 하나를 제거  
del df['시군구_코드_명']  
  
# 상권변화지표와 상권변화지표명은 같은 정보를 가지고 있으므로 둘 중 하나를 제거  
del df['상권_변화_지표']
```

```
[ ] df.head()
```

```
↳
```

	기준_년_코드	기준_분기_코드	시군구_코드	상권_변화_지표_명	운영_영업_개월_평균	폐업_영업_개월_평균	서울_운영_영업_개월_평균	서울_폐업_영업_개월_평균
0	2019	4	11740	다이나믹	105	50	110	54
1	2019	4	11710	다이나믹	110	53	110	54
2	2019	4	11680	다이나믹	106	50	110	54
3	2019	4	11650	정체	118	54	110	54
4	2019	4	11620	다이나믹	108	52	110	54

Step 1

시군구 코드 = 시군구 코드명

상권변화지표 = 상권변화지표명

두 컬럼이 같은 정보를 가지고 있으므로 하나를 제거한다.



3. 데이터 가공 및 전처리

서울_운영_영업_개월_평균 서울_폐업_영업_개월_평균

110	54
110	54
110	54
110	54
110	54

```
df['서울 대비 운영 영업개월 평균']=df['운영_영업_개월_평균']/df['서울_운영_영업_개월_평균']  
df['서울 대비 운영 영업개월 평균'].head()
```

```
0    0.954545  
1    1.000000  
2    0.963636  
3    1.072727  
4    0.981818  
Name: 서울 대비 운영 영업개월 평균, dtype: float64
```

```
df['서울 대비 폐업 영업개월 평균']=df['폐업_영업_개월_평균']/df['서울_폐업_영업_개월_평균']  
df['서울 대비 폐업 영업개월 평균'].head()
```

```
0    0.925926  
1    0.981481  
2    0.925926  
3    1.000000  
4    0.962963  
Name: 서울 대비 폐업 영업개월 평균, dtype: float64
```

2) 의미있는 데이터 만들기

['운영_영업_개월_평균'] = 110

['폐업_영업_개월_평균'] = 54

로 모든 row에 동일한 데이터가 들어있다.

따라서 이 데이터는 분류에 도움이 되지 않으므로, 아래와 같이 의미 있는 feature로 변경한다.

- 서울 대비 해당 도시의 운영개월 평균
- 서울 대비 해당 도시의 폐업개월 평균



3. 데이터 가공 및 전처리

3) ['상권 변화지표명'] 컬럼 데이터를 숫자로 변경

상권 변화: 다이나믹 (3) 상권확장 (2) 상권축소 (1) 정체 (0)

```
df['상권_변화_지표명']=df['상권_변화_지표명'].replace(['다이나믹','상권확장','상권축소','정체'],[3,2,1,0])  
df.head()
```

	기준_년_코드	기준_분기_코드	시군구_코드	상권_변화_지표명	운영_영업_개월_평균	폐업_영업_개월_평균	서울_운영_영업_개월_평균	서울_폐업_영업_개월_평균
0	2019	4	11740	3	105	50	110	54
1	2019	4	11710	3	110	53	110	54
2	2019	4	11680	3	106	50	110	54
3	2019	4	11650	0	118	54	110	54
4	2019	4	11620	3	108	52	110	54



3. 데이터 가공 및 전처리

컬럼명 변경

```
df.columns=['year','quarter','city','Commercial change','Operating months','Closing months', 'Operating months_Average', 'Closing months_Average']  
df.head()
```

	year	quarter	city	Commercial change	Operating months	Closing months	Operating months_Average	Closing months_Average
0	2019	4	11740	3	105	50	0.954545	0.925926
1	2019	4	11710	3	110	53	1.000000	0.981481
2	2019	4	11680	3	106	50	0.963636	0.925926
3	2019	4	11650	0	118	54	1.072727	1.000000
4	2019	4	11620	3	108	52	0.981818	0.962963

결측치는 없다.

```
df.isna().sum()
```

```
year          0  
quarter       0  
city          0  
Commercial change  0  
Operating months  0  
Closing months  0  
Operating months_Average  0  
Closing months_Average  0  
dtype: int64
```

- 편의를 위해서 컬럼명을 변경해준다.
- 결측치는 존재하지 않는다.



3. 데이터 가공 및 전처리

Train과 Test set 나누기

```
] from sklearn import model_selection
   from sklearn.model_selection import train_test_split

   x_data=np.array(x_data)
   y_data=np.array(y_data)

   x_train, x_test, y_train, y_test = model_selection.train_test_split(x_data,y_data,test_size=0.1, random_state=0)
```

```
] x_train
```

```
> array([[2.01600000e+03, 1.00000000e+00, 1.12900000e+04, ...,
         5.10000000e+01, 1.03125000e+00, 1.06250000e+00],
        [2.01900000e+03, 4.00000000e+00, 1.12900000e+04, ...,
         5.60000000e+01, 1.00909091e+00, 1.03703704e+00],
        [2.01500000e+03, 4.00000000e+00, 1.13800000e+04, ...,
         4.80000000e+01, 9.68421053e-01, 1.00000000e+00],
        ...,
        [2.01600000e+03, 3.00000000e+00, 1.12000000e+04, ...,
         5.10000000e+01, 9.89583333e-01, 1.04081633e+00],
        [2.01800000e+03, 1.00000000e+00, 1.16200000e+04, ...,
         5.00000000e+01, 9.90000000e-01, 1.00000000e+00],
        [2.01400000e+03, 3.00000000e+00, 1.12000000e+04, ...,
         4.70000000e+01, 1.01075269e+00, 1.04444444e+00]])
```





Linear regression



NN Layer#3



DNN Layer#5



DNN Layer#5
+
Dropout 0.3



4. 베이스라인 구축

Linear regression

- 예상했던 대로 전혀 성과가 없었다.
- Epoch과 learning rate를 바꿔보아도 Cost는 낮아지지 않았다.

```
Epoch    0/10000 cost: 1.386295
Epoch 1000/10000 cost: 1.199204
Epoch 2000/10000 cost: 1.199204
Epoch 3000/10000 cost: 1.199204
Epoch 4000/10000 cost: 1.199204
Epoch 5000/10000 cost: 1.199204
Epoch 6000/10000 cost: 1.199204
Epoch 7000/10000 cost: 1.199204
Epoch 8000/10000 cost: 1.199204
Epoch 9000/10000 cost: 1.199204
Epoch 10000/10000 cost: 1.199204
```

```
hypothesis=F.softmax(x_test.matmul(W)+b, dim=1)
predict=torch.argmax(hypothesis, dim=1)
```

predict

[illegible]

```
solution=pd.read_csv('solution.csv')
v_test=solution.iloc[:,1]
```

```
y_test=y_test.to_numpy()
y_test=torch.LongTensor(y_test)
y_test
```

```
tensor([3, 0, 0, 0, 3, 0, 0, 1, 3, 0, 0, 3, 3, 0, 3, 0, 0, 3, 0, 0, 0, 3,
        0, 0, 3, 3, 0, 0, 0, 3, 0, 3, 3, 3, 1, 0, 3, 3, 1, 1, 3, 3, 0, 3, 3, 3,
        3, 3, 0, 0, 3, 3, 2, 3, 3, 3, 3, 1, 3, 0])
```

```
correct = prediction == y_test
accuracy=correct.float().mean()
```

```
print('Accuracy:', accuracy.item())
```

Accuracy: 0.5

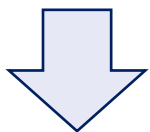


4. 베이스라인 구축

NN Layer#3

- xavier init
- Adam
- 7 => 512 => 4

→ Accuracy: 0.5483871102333069



hidden layer의 노드 개수 & epoch 변경

- 7 => 256 => 4
- training_epochs = 55

→ Accuracy: 0.725806474685669

prediction

```
• tensor([3, 0, 0, 3, 3, 0, 0, 3, 3, 0, 0, 3, 0, 0, 3, 0, 3, 3, 0, 0, 3, 3,
          0, 3, 3, 3, 0, 0, 3, 3, 0, 3, 0, 3, 3, 3, 3, 3, 0, 3, 0, 3, 3, 3, 3, 3,
          3, 3, 3, 0, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0])
```

```
] y_test=y_test.to_numpy()
   y_test=torch.LongTensor(y_test)
   y_test
```

```
• tensor([3, 0, 0, 0, 3, 0, 0, 1, 3, 0, 0, 3, 3, 0, 3, 0, 3, 0, 0, 3, 0, 0, 0, 3,
          0, 0, 3, 3, 0, 0, 0, 3, 0, 3, 3, 3, 1, 0, 3, 3, 1, 1, 3, 3, 0, 3, 3, 3,
          3, 3, 0, 0, 3, 3, 2, 3, 3, 3, 3, 1, 3, 0])
```

```
] correct = prediction == y_test
   accuracy=correct.float().mean()

   print('Accuracy: ', accuracy.item())
```

Accuracy: 0.725806474685669



4. 베이스라인 구축

DNN Layer#5

- xavier init
- Adam
- $7 \Rightarrow 215 \Rightarrow 4$

→ Accuracy: 0.5

DNN Layer#5 + Dropout 0.3

```
print(prediction)
```

```
tensor([3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
        3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
        3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3])
```

```
with torch.no_grad():
    model.eval()
    y_test=solution.iloc[:,1]

    y_test=y_test.to_numpy()
    y_test=torch.LongTensor(y_test)
    print(y_test)
```

```
tensor([3, 0, 0, 0, 3, 0, 0, 1, 3, 0, 0, 3, 3, 0, 3, 0, 0, 3, 0, 0, 0, 3,
        0, 0, 3, 3, 0, 0, 0, 3, 0, 3, 3, 3, 1, 0, 3, 3, 1, 1, 3, 3, 0, 3, 3, 3,
        3, 3, 0, 0, 3, 3, 2, 3, 3, 3, 3, 1, 3, 0])
```

→ Accuracy: 0.4516128897666931

0, 3,
3, 3,
Init, optim, 변경
Dropout 0.5 적용
학습 파라미터 변경
Hidden layer 노드 수 변경
→ 정확도가 나아지지 않았다.

→ 정확도가 나아지지 않았다.



4. 베이스라인 구축

Linear regression

50%

3-layer NN

72.58%

5layer DNN

50%

5layer DNN +
Dropout(0.3)

45.16%


60%~80%의 정확도를 갖는
3layer NN을 베이스라인으로 구축

#	Team Name	Notebook	Team Members	Score ?	Entries	Last
📍	baseline.csv			0.72580		



5. Kaggle 리더보드 제작

PART.5

 InClass Prediction Competition

자치구별 상권변화 분류

서울 25개의 자치구에 대해서 분기별 개·폐점률에 따른 상권변화 정도를 예측한다.

14 days to go

[Overview](#) [Data](#) [Notebooks](#) [Discussion](#) [Leaderboard](#) [Rules](#) [Team](#) [Host](#) [My Submissions](#) [Submit Predictions](#)

Overview Edit

Description

Evaluation

[+ Add Page](#)

자치구별 상권변화 정도를 분류한다.

서울 25개의 자치구에 대해서 분기별 개·폐점률에 따른 상권변화 정도를 예측한다.
상권 변화정도는 다이내믹, 상권확장, 상권축소, 정체 4가지로 나눌 수 있다.
다이내믹: 점포 개·폐점률이 높다. 정체: 점포 개·폐점률이 낮다.

활용 데이터

- 서울시 우리마을가게 상권분석서비스(자치구별 상권변화지표)
- <http://data.seoul.go.kr/dataList/OA-15567/S/1/datasetView.do>



5. Kaggle 리더보드 제작

[Overview](#) [Data](#) [Notebooks](#) [Discussion](#) [Leaderboard](#) [Rules](#) [Team](#) [Host](#) [My Submissions](#) [Submit Predictions](#)

제공되는 데이터의 설명

- 구성 데이터: 기준 년, 기준 분기, 시군구코드, 운영영업개월 평균, 폐업 영업개월 평균, 서울 대비 운영영업개월 평균, 서울 대비 폐업 영업개월평균
- target value: 상권 변화를 나타내는 분류 카테고리

상권 변화 정도: 다이나믹 > 상권확장 > 상권축소 > 정체 = 3>2>1>0

각 데이터 셋의 설명

- train.csv : 자치구별 상권변화 지표를 7개의 특징으로 표현한 train데이터
- test.csv : 자치구별 상권변화 지표를 7개의 특징으로 표현한 test데이터.
- submit.csv : submission 파일의 예시



5. Kaggle 리더보드 제작

[Overview](#) [Data](#) [Notebooks](#) [Discussion](#) [Leaderboard](#) [Rules](#) [Team](#) [Host](#) [New Notebook](#)

GPU quota: 28h 6m remaining

Public

Your Work

Shared With You

Favorites

Sort by

Hotness


Outputs


Languages

Tags

Search notebooks

1



baseline code
6h ago  gpu

Py

0

[Overview](#) [Data](#) [Notebooks](#) [Discussion](#) [Leaderboard](#) [Rules](#) [Team](#) [Host](#) [My Submissions](#) [Submit Predictions](#)

[Public Leaderboard](#) [Private Leaderboard](#)

This leaderboard is calculated with all of the test data. [Raw Data](#) [Refresh](#)

#	Team Name	Notebook	Team Members	Score ?	Entries	Last
📍	baseline.csv			0.72580		







5. Kaggle 리더보드 제작

[Overview](#) [Data](#) [Notebooks](#) [Discussion](#) [Leaderboard](#) [Rules](#) [Team](#) [Host](#) [My Submissions](#) [New Topic](#)

4 topics [Following](#) Sort by [Hotness](#) ▼

[All](#) [Mine](#) | [Upvoted](#) Q

▲ 0		1. 데이터 가공 및 전처리 이주희 10 minutes ago	last comment by 이주희 10m ago	💬 0
▲ 0		2. 베이스라인 구축 과정 이주희 6 minutes ago	last comment by 이주희 6m ago	💬 0
▲ 0		3. 발표자료 이주희 a minute ago	last comment by 이주희 1m ago	💬 0
▲ 0		4. 발표 영상 이주희 just now	last comment by 이주희 now	💬 0



**Thank you
for listening**

