Technical Portfolio

for Data Science

Dong Gyu, Lee

written in LaTeX

Growth Timeline

2017년 - 첫 프로그래밍, Hangman

2019년 - 첫 데이터 분석, Kaggle

2019년 - 프로젝트, Impact Echo

2020년 - 도시가스 수요량 예측 대회

2020년 - 카카오 아레나

2020년 - 추천시스템 구현 프로젝트

Portfolio Timeline

2017년 - 첫 프로그래밍, Hangman

2019년 - 첫 데이터 분석, Kaggle

2019년 - 프로젝트, Impact Echo

2020년 - 도시가스 수요량 예측 대회

2020년 - 카카오 아레나

2020년 - 추천시스템 구현 프로젝트

Hangman

Hangman. 무작위의 영어 단어를 맞추는 게임으로, 26개의 알파벳 중 하나를 대면 틀릴 때마다 Life가 하나씩 줄어듭니다. 총 Life는 8개이며 글자를 맞추는 순서는 상관이 없습니다.

예시) 'GAME' 이라는 단어를 맞추어야 하고 플레이어는 아직 이 단어를 아직 알지 못합니다. 플레이어에게는 오직 철자의 개수만 주어져 있습니다.

	A		E
—	—	—	—

input	boolean	Life
⋮	⋮	4

Hangman

Hangman. 무작위의 영어 단어를 맞추는 게임으로, 26개의 알파벳 중 하나를 대면 틀릴 때마다 Life가 하나씩 줄어듭니다. 총 Life는 8개이며 글자를 맞추는 순서는 상관이 없습니다.

예시) 'GAME' 이라는 단어를 맞추어야 하고 플레이어는 아직 이 단어를 아직 알지 못합니다. 플레이어에게는 오직 철자의 개수만 주어져 있습니다.

G A E

input	boolean	Life
⋮	⋮	4
'G'	true	4

Hangman

Hangman. 무작위의 영어 단어를 맞추는 게임으로, 26개의 알파벳 중 하나를 대면 틀릴 때마다 Life가 하나씩 줄어듭니다. 총 Life는 8개이며 글자를 맞추는 순서는 상관이 없습니다.

예시) 'GAME' 이라는 단어를 맞추어야 하고 플레이어는 아직 이 단어를 아직 알지 못합니다. 플레이어에게는 오직 철자의 개수만 주어져 있습니다.

G A E

input	boolean	Life
⋮	⋮	4
'G'	true	4
'T'	false	3

Hangman

Hangman. 무작위의 영어 단어를 맞추는 게임으로, 26개의 알파벳 중 하나를 대면 틀릴 때마다 Life가 하나씩 줄어듭니다. 총 Life는 8개이며 글자를 맞추는 순서는 상관이 없습니다.

예시) 'GAME' 이라는 단어를 맞추어야 하고 플레이어는 아직 이 단어를 아직 알지 못합니다. 플레이어에게는 오직 철자의 개수만 주어져 있습니다.

You win!

G	A	M	E
—	—	—	—

input	boolean	Life
⋮	⋮	4
'G'	true	4
'T'	false	3
'M'	true	3

Hangman

개발 환경

- PC : LG 노트북(LG15U56)
- OS : Ubuntu 16.04.2.(*Window 10에서 VMware 가상머신을 이용해 설치)
- S/W : Python3 (Spyder or sublime text3 이용)

주요 코드 설명

- loadWords() : 단어들을 wordslist.txt 파일로 부터 불러와 list 형태로 reshape
- chooseWord(list) : list에서 무작위로 단어 하나를 뽑는 함수
- isWordGuessed() : 플레이어가 추측한 철자들과 정답을 비교하는 boolean 함수
- getGuessedWord() : 플레이어가 맞춘 철자들을 분류하기 위해 만든 함수
- getAvailableLetters() : 플레이어가 아직 입력하지 않은, 남은 철자들을 보여주는 함수
- hangman(word) : Hangman 게임을 실행. 이 때의 정답은 word

hangman 코드 예시

```
def hangman(secretWord):
    length = len(secretWord); entered_letters = []; your_letters = [];
    life = 8  # set the number of lifes
    print("There are {0} letters in here.".format(length))  # notice how long letter the word has.
    while life :        # loop due to run out the lifes
        x, y = 0, 0     # to case check
        print("\n")
        letter = input("Guess a letter: ")
        if (letter in secretWord) and (len(letter) == 1):
            x = 1
        if letter in entered_letters:
            y = 1
        if y == 0:
            entered_letters += [letter]
        if (x == 1) and (y == 0):
            your_letters += [letter]
        if (x == 0) and (y == 0):
            life += -1
        print("You have {0} guesses left".format(life))   # notice how many lifes you have.
        print("Available letters: " + getAvailableLetters(entered_letters))
        if (x == 1) and (y == 0):   # if your guess is good, then...
            print("Good guess: " + getGuessedWord(secretWord,entered_letters))
        elif y == 1:                # if you guess same letter, then...
            print("You have already guessed this letter.\nPlease guess another letter: "
                + getGuessedWord(secretWord,entered_letters))
        else:                       # if your guess is bad, then...
            if len(letter) > 1:     # the case of that your guess is not valid.
                print("Please enter only one letter")
            print("You missed: " + getGuessedWord(secretWord,entered_letters))
        if isWordGuessed(secretWord,your_letters) == True:
            break
    if isWordGuessed(secretWord,your_letters) == True:   # if your letters are correct, then...
        print("\nYou won!")
    else:                                                # otherwise,
        print("\nYou lose...\nThe answer is " + secretWord)

secretWord = chooseWord(wordList).lower() # choose random word and make it lower.
hangman(secretWord) # start hangman game.
```

hangman 함수 요약 설명

Step1. 플레이어에게 'secretWord'의 철자 개수를 출력

Step2. Life가 0이 될 때까지 Loop를 돌려 플레이어에게 철자를 입력받음

Step3. 입력 받은 철자가 유효하고, 플레이어의 추측이 맞는가를 검사 (변수 x의 역할)

Step4. 입력 받은 철자가 이미 전에 입력된 적이 있다면 플레이어에게 경고메세지 출력 (변수 y의 역할)

Step5. 입력 받은 철자가 틀리거나 유효하지 않으면 Life를 깎고 Loop 재시작

Step6. 매 Loop마다 플레이어가 모든 철자를 찾았는지 검사하여 만약 Life가 0이 되기전에 모든 철자를 찾았다면, Loop를 탈출하고 "You won" 출력. 그렇지 못하면 "You lose" 출력

Output in console 예시

```
Loading word list from file...
   55909 words loaded.

There are 10 letters in here.

Guess a letter:
```

첫 화면

```
Loading word list from file...
   55909 words loaded.

There are 10 letters in here.

Guess a letter: a
You have 8 guesses left
Available letters: _ b c d e f g h i j k l m n o p q r s t u v w x y z
Good guess: _ _ a _ _ _ _ _ _

Guess a letter:
```

맞췄을 때

```
Loading word list from file...
   55909 words loaded.

There are 10 letters in here.

Guess a letter: a
You have 8 guesses left
Available letters: _ b c d e f g h i j k l m n o p q r s t u v w x y z
Good guess: _ _ a _ _ _ _ _ _

Guess a letter: z
You have 7 guesses left
Available letters: _ b c d e f g h i j k l m n o p q r s t u v w x y _
You missed: _ _ a _ _ _ _ _ _

Guess a letter:
```

틀렸을 때

Portfolio Timeline

2017년 - 첫 프로그래밍, Hangman

2019년 - 첫 데이터 분석, Kaggle

2019년 - 프로젝트, Impact Echo

2020년 - 도시가스 수요량 예측 대회

2020년 - 카카오 아레나

2020년 - 추천시스템 구현 프로젝트

데이터 분석

Summary. 페이스북에서 보게된 대회 공고에 흥미를 느껴 참여하게 되으며, 본격적으로 데이터 사이언스를 공부하게된 계기가 되었습니다.

문제

- ▷ Seattle 지역에서 집과 관련된 데이터가 주어졌을 때, 집값을 예측하는 문제입니다.
- ▷ 약 15,000개의 데이터와 43개 가량의 feature들이 주어졌습니다.
- ▷ 적당한 feature을 골라내 Decision Tree 기반의 모델을 이용하여 학습시킨 후, 5-fold CV으로 overfitting을 감지하고자 했습니다.

방향

- ▷ 당시에는 데이터를 접한게 처음이라 참가에만 의의를 두었습니다.
- ▷ 다른 참가자분들의 커널을 참고하며 단일 모델 하나만 목표로 하여 공부했습니다.
- ▷ 전처리부터 hyperparameter tuning까지의 과정에만 집중했습니다.

데이터 분석

분석 환경

- ▷ Lang : Python 3
- ▷ Env : Kaggle Notebook

사용 라이브러리

- ▷ 전처리: pandas, sklearn
- ▷ 시각화: matplotlib, seaborn, geojson, folium
- ▷ 학습: LightGBM, XGBoost

데이터 불러오기

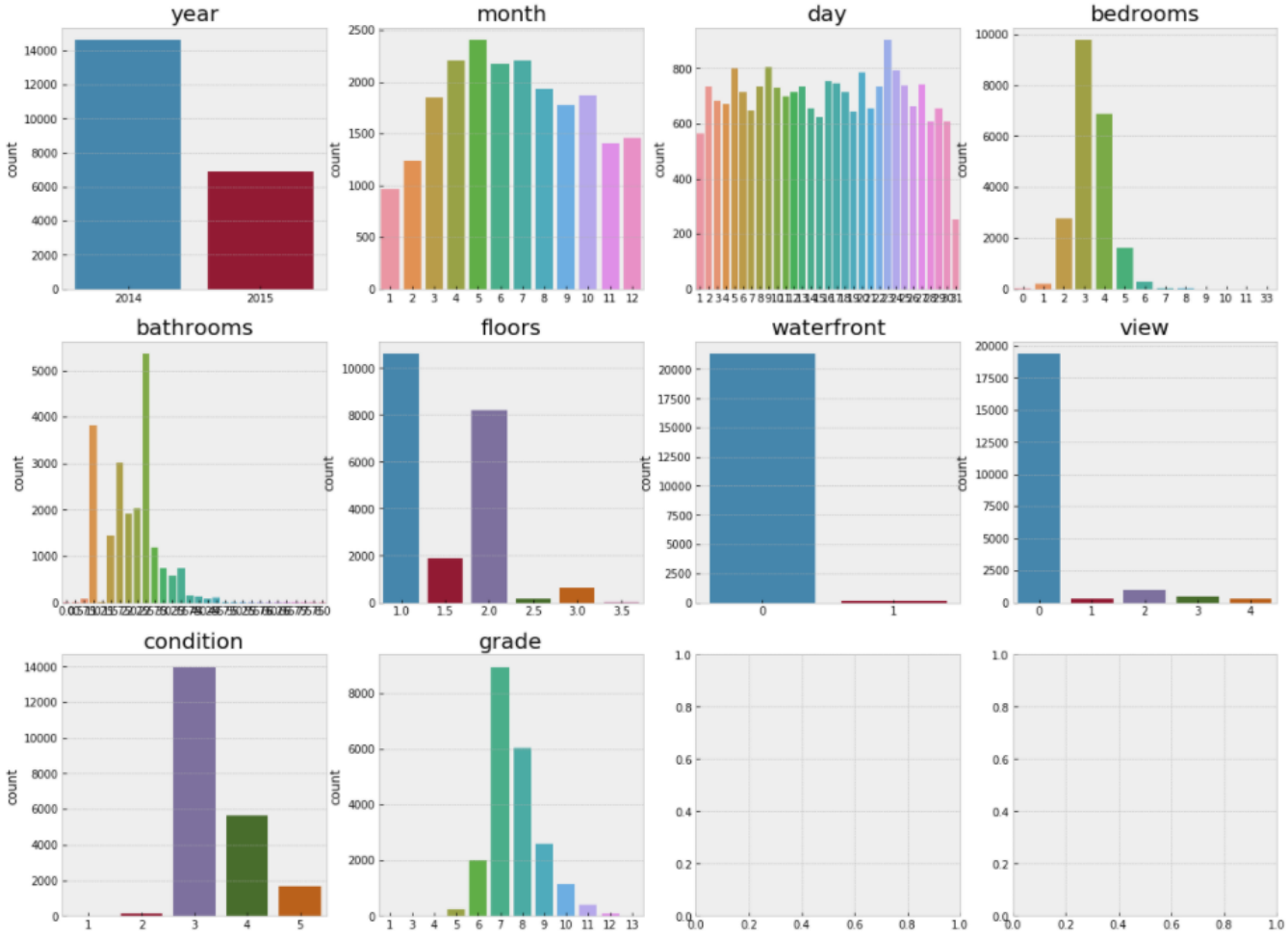
```
train = pd.read_csv('input/train.csv')

print(train.shape)
train.head()
```

```
(15035, 21)
```

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_built	yr_re
0	0	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	0	1955	
1	1	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	0	1933	
2	2	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680	0	1987	
3	3	20140627T000000	257500.0	3	2.25	1715	6819	2.0	0	0	...	7	1715	0	1995	
4	4	20150115T000000	291850.0	3	1.50	1060	9711	1.0	0	0	...	7	1060	0	1963	

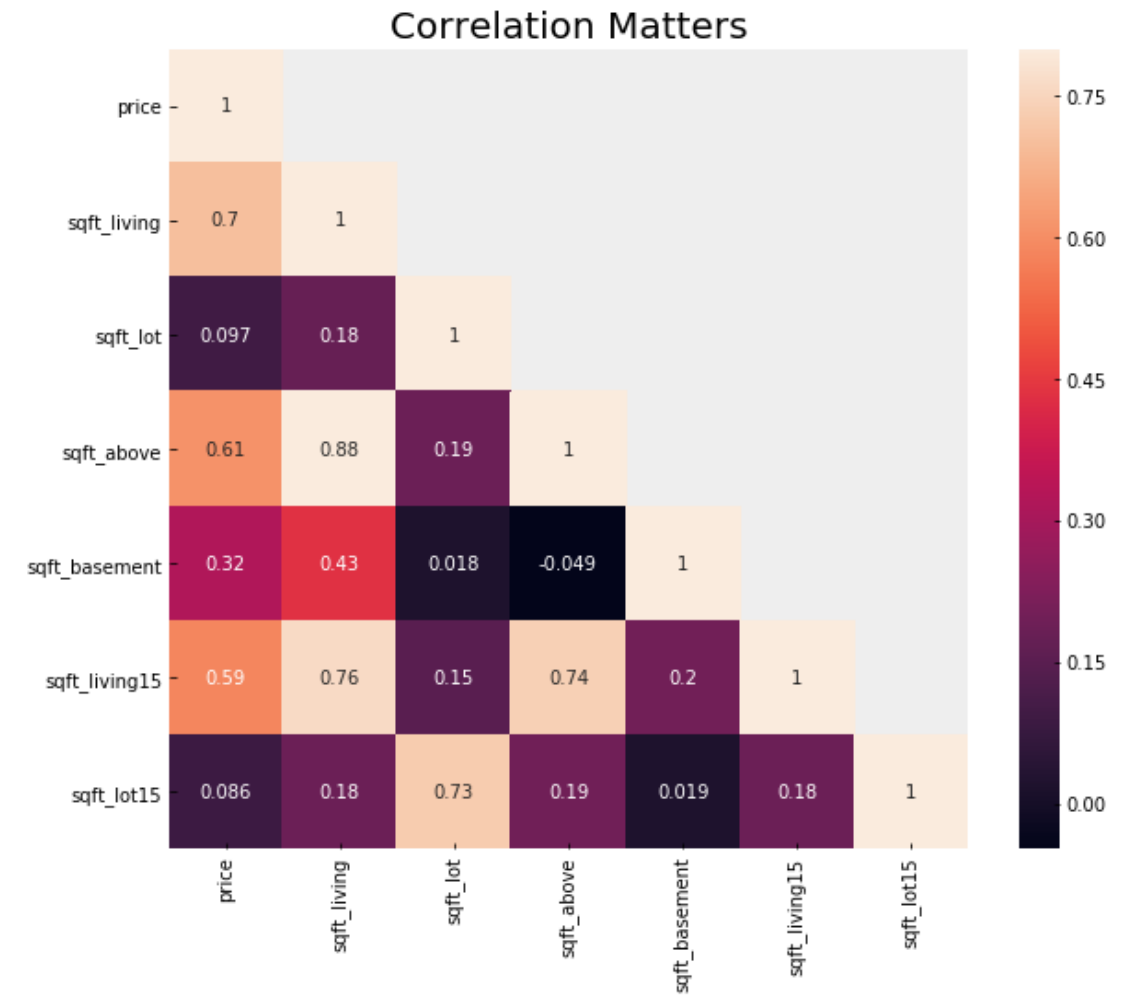
시각화


year
month
day
bedrooms
bathrooms
floors
waterfront
view
condition
grade
count

시각화

```
fig, ax = plt.subplots(figsize=(10, 8))

sns.heatmap(corrMatt, mask=mask, vmax=0.8, square=True, annot=True)
ax.set_title('Correlation Matters', fontsize=20)
plt.show()
```



탐험적 데이터 분석

Step1. 결측치를 확인하고, feature 간의 상관관계를 알기 위해 시각화

Step2. 경도, 위도 좌표에 대한 이상치 제거

Step3. Log scale 등을 통해 원활한 regression 준비

Step4. 분석 및 시각화를 통해 적절한 feature 선택 및 생성

학습 모델 소개- XGBoost

```
%%time
# transform
print('Start Transforming...')
dtrain = xgb.DMatrix(X_train, y_train)
dtest = xgb.DMatrix(X_test)

# cross validation
print('Valid the Model...')
cv_output = xgb.cv(xgb_params,
                   dtrain,
                   num_boost_round=5000,
                   early_stopping_rounds=100,
                   nfold=5,
                   verbose_eval=100,
                   feval=rmse_exp,
                   maximize=False,
                   show_stdv=False,
                   )
```

학습 후...

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_{true} - y_{pred})^2}$$

Step1. 캐글에서 요구하는 지표인 RMSE로 성능 측정

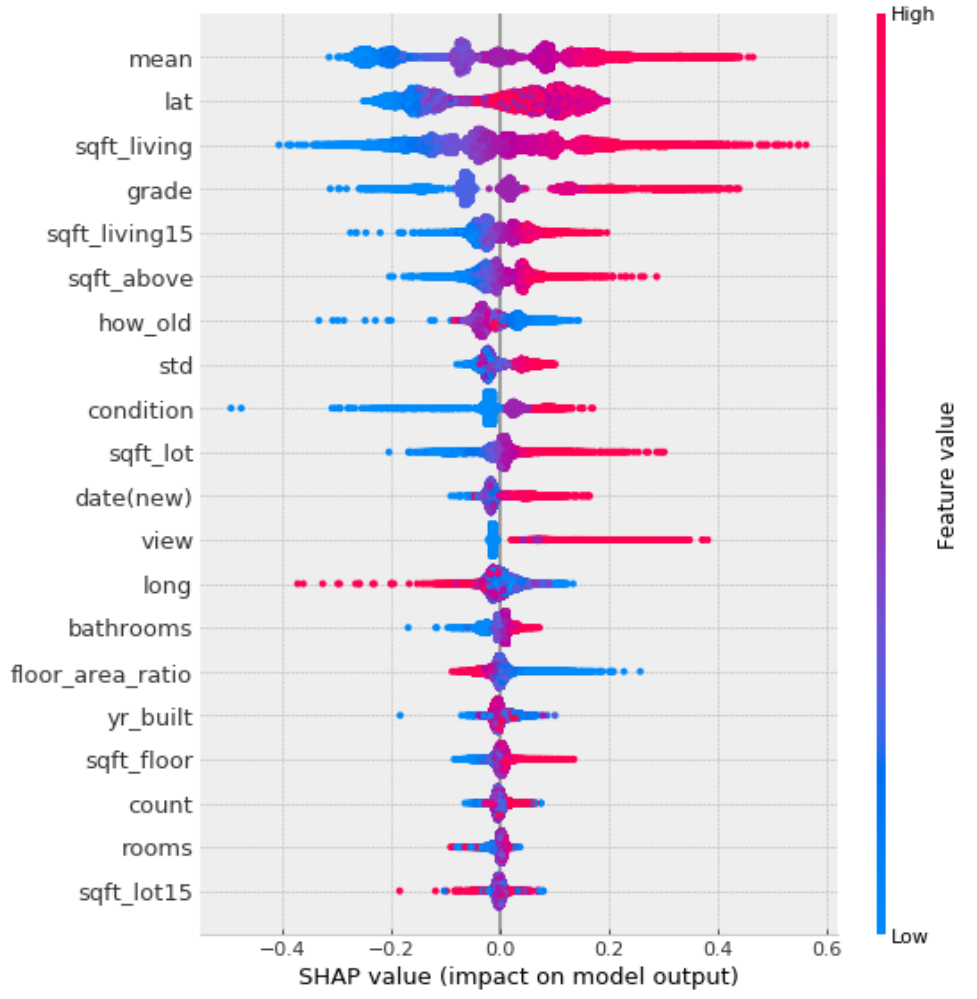
Step2. CV를 이용해 hyperparameter tuning

Step3. Shapley value 등을 이용해 feature 재선택

Step4. Step1 $\sim$ Step3 반복 시행

게임이론- Shapley Value

```
explainer = TreeExplainer(model)
shap_values = explainer.shap_values(X_train)
summary_plot(shap_values, X_train, title='Train')
```



결과

- Private Score: 108067.72 (6등)
- Public Score: 104647.13 (47등)
- Overfitting을 해결하지 못하여 굉장히 편향된 결과를 제출

- [자세한 분석 노트 링크]

Portfolio Timeline

2017년 - 첫 프로그래밍, Hangman

2019년 - 첫 데이터 분석, Kaggle

2019년 - 프로젝트, Impact Echo

2020년 - 도시가스 수요량 예측 대회

2020년 - 카카오 아레나

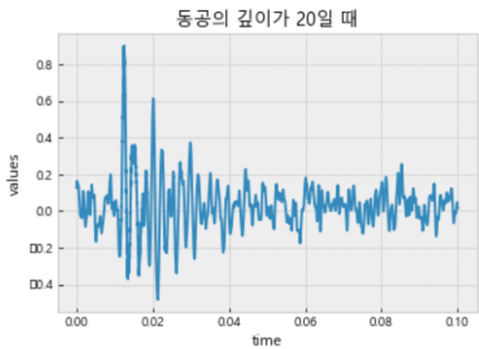
2020년 - 추천시스템 구현 프로젝트

Impact Echo

Summary. 기초과학연구소에서 다음과 같은 프로젝트를 경험하였습니다.

문제

- ▷ 땅 속에 동공(균열)을 감지하기 위해 타격기로 충격을 발생시키고, 그 충격파(impact echo)를 센서로 수집하여 지하에 어떤 종류의 동공이 있는가를 '분류'하고자 합니다.
- ▷ 한번의 실험에서 약 12개의 센서로 충격파를 감지하고, 각 실험마다 동공의 종류를 다르게 하여 측정합니다.
- ▷ 이렇게 얻은 데이터는 시계열 데이터로, 총 길이가 10,000에 달하는 수열입니다.
- ▷ 타격기(좌)와 데이터(우)의 모습입니다. 데이터는 CSV파일로 주어졌습니다.



	time	values
0	0.00000	0.129394
1	0.00001	0.133667
2	0.00002	0.137634
3	0.00003	0.141602
4	0.00004	0.144958
5	0.00005	0.146790
6	0.00006	0.149536
7	0.00007	0.152893
8	0.00008	0.155945
9	0.00009	0.158997

개발환경

- PC : LG 노트북(LG15U56) → 이후 GPU 탑재된 서버 제공받음
- OS : Window 10 → 이후 Ubuntu
- S/W : Python3

사용 라이브러리

- 전처리: pandas, scikit-learn
- 신호 처리: scipy.signal
- 시각화: matplotlib, seaborn
- 학습: keras

데이터 가공

Step1. 하나의 관측치가 하나의 CSV 파일로 처리되있었기 때문에, tidy data로 만들어주는 작업이 필요

Step2. 카테고리 별로 데이터를 나눈 후, NaN값을 가지는 데이터 제거하고 저장

```
file = d['20']

df = pd.DataFrame()
n = 0

bar_total = tqdm_notebook(range(len(file)))
for i in bar_total:
    temp = pd.read_csv(file[i], names=['*', str(n)])
    temp = temp[str(n)]
    df = pd.concat([df, temp], axis=1)
    n += 1

df = df.transpose()
print(df.shape)
df.head()
```

100% 1000/1000 [02:35<00:00, 3.98it/s]

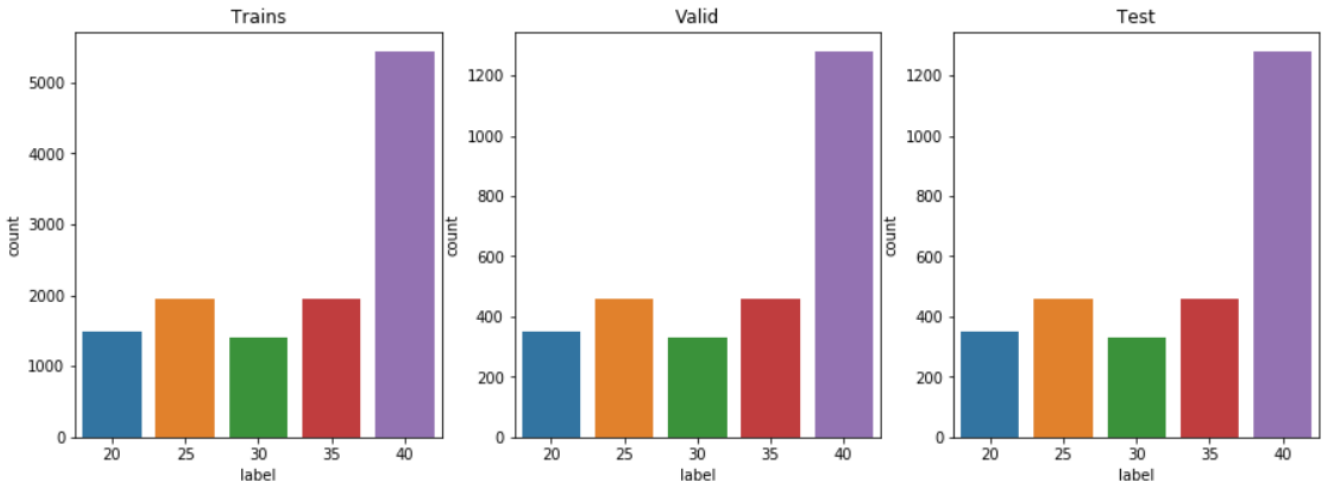
(1000, 10000)

	0	1	2	3	4	5	6	7	8	9	...	9990	9991	9992	9993	9994	9995	9996
0	-0.00641	-0.00458	-0.00397	-0.00366	-0.00366	-0.00366	-0.00214	0.000000	0.00000	0.000916	...	0.0494	0.0500	0.0494	0.0491	0.0497	0.0500	0.0494
1	-0.00183	-0.00305	-0.00366	-0.00549	-0.00519	-0.00397	-0.00214	-0.000305	0.00153	0.003050	...	0.0650	0.0635	0.0620	0.0601	0.0577	0.0552	0.0543
2	0.01070	0.01250	0.01400	0.01400	0.01460	0.01500	0.01680	0.018300	0.01980	0.022000	...	0.0464	0.0458	0.0449	0.0433	0.0418	0.0418	0.0421
3	0.02900	0.02930	0.03080	0.03080	0.03050	0.03020	0.03140	0.033000	0.03360	0.033300	...	0.0525	0.0528	0.0516	0.0507	0.0497	0.0482	0.0467
4	0.02560	0.02470	0.02440	0.02320	0.02230	0.02110	0.02080	0.020800	0.01920	0.017100	...	0.0839	0.0812	0.0797	0.0754	0.0717	0.0720	0.0674

데이터 분리

Step1. 모델의 성능을 평가하기 위해 데이터를 8:2 비율로 train data와 test data로 분리

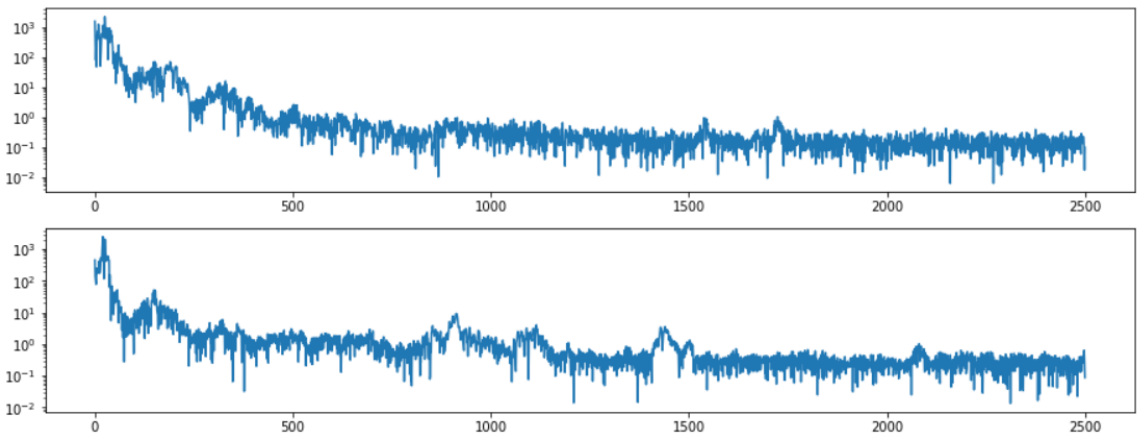
Step2. 다만, train data와 test data의 상관성을 줄이기 위해 서로 다른 날짜에 실험되도록 분리



데이터 전처리

신호 데이터에 관한 각종 전처리 작업 수행

- ▷ DC offset: 신호의 영점 조절을 위해 데이터의 평균값을 빼서 제거
- ▷ Normalization: 실험마다 타격기의 에너지가 달라, 각 데이터의 신호 에너지를 계산 후 1이 되도록 정규화
- ▷ 푸리에 변환 후, 상위 75%의 스펙트럼만 이용 → 노이즈 제거 효과
- ▷ Power Spectral Density(PSD)



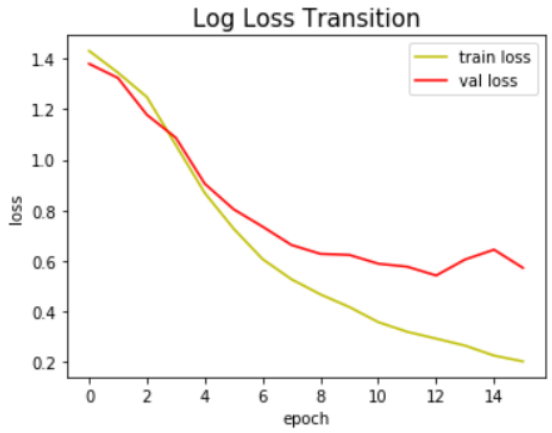
기본 모델

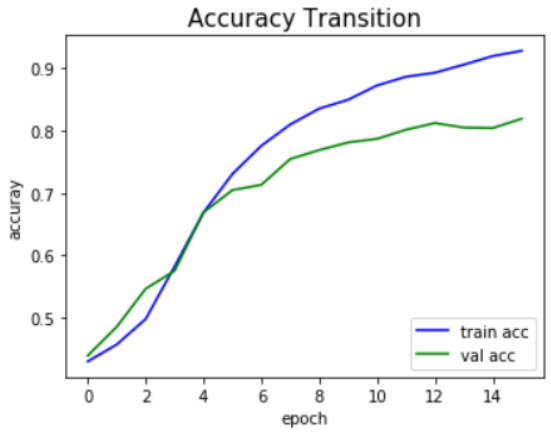
- ▷ 시계열 데이터의 PSD를 학습데이터로 사용
- ▷ 수열이 일정한지, 증가하는지, 감소하는지 공간정보를 학습시키고자 1D-CNN 사용
- ▷ 시퀀스를 500단위로 끊어 각각에 대한 추세를 학습(reshape)
- ▷ Smoothing 효과를 위해 Global Average Pooling 사용
- ▷ 5 Classification 문제이므로 마지막 Activation Layer은 Softmax로 설정

Layer (type)	Output Shape	Param #
reshape_15 (Reshape)	(None, 500, 10)	0
conv1d_108 (Conv1D)	(None, 492, 64)	5824
average_pooling1d_59 (Averag	(None, 246, 64)	0
conv1d_109 (Conv1D)	(None, 246, 64)	4160
conv1d_110 (Conv1D)	(None, 238, 64)	36928
average_pooling1d_60 (Averag	(None, 119, 64)	0
conv1d_111 (Conv1D)	(None, 119, 64)	4160
conv1d_112 (Conv1D)	(None, 113, 64)	28736
average_pooling1d_61 (Averag	(None, 56, 64)	0
conv1d_113 (Conv1D)	(None, 56, 64)	4160
conv1d_114 (Conv1D)	(None, 50, 64)	28736
average_pooling1d_62 (Averag	(None, 25, 64)	0
conv1d_115 (Conv1D)	(None, 25, 64)	4160
conv1d_116 (Conv1D)	(None, 21, 64)	20544
flatten_15 (Flatten)	(None, 1344)	0
dropout_15 (Dropout)	(None, 1344)	0
dense_15 (Dense)	(None, 5)	6725

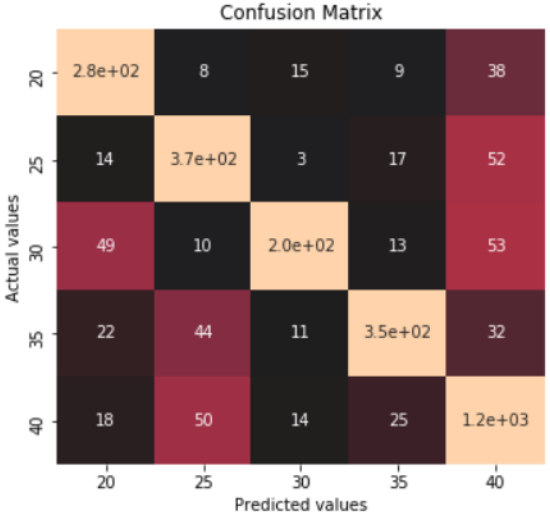
Total params: 144,133
Trainable params: 144,133
Non-trainable params: 0

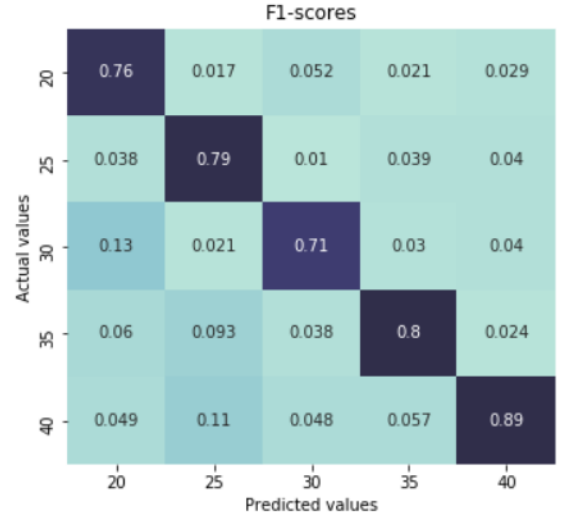
결과





- ▷ train accuracy: 약 99.1%
- ▷ test accuracy: 약 82.7 %
- ▷ 깊이가 40인 동공이 훨씬 많기 때문에 f1-score도 측정해보자





고찰

- 신호 처리의 도메인 지식을 이용하여 데이터 어그멘테이션을 시도하였으나 효과가 없었습니다.
- 테스트 데이터 또한 정형화된 실험 환경에서 얻은 것이기 때문에 현장에서 사용하기에는 high variance로 인한 문제가 발생할 수 있다고 생각했습니다. 그러나 아이러니하게도 실제 현장에서는 가장 overfitting이 심했던 모델이 가장 좋은 성능을 보였습니다.
- 데이터의 개수가 가장 부족한 class의 정확도는 약 70%로, 데이터 불균형 문제를 충분히 해결하지 못한 것이 아쉬웠습니다.

Portfolio Timeline

2017년 - 첫 프로그래밍, Hangman

2019년 - 첫 데이터 분석, Kaggle

2019년 - 프로젝트, Impact Echo

2020년 - 도시가스 수요량 예측 대회

2020년 - 카카오 아레나

2020년 - 추천시스템 구현 프로젝트

도시가스 수요량 예측

Summary. 국가수리과학연구소에서 주관한 도시가스 수요량 예측 대회에 참여하였습니다. 대회는 약 1달간 진행되었으며, 군인이라는 신분임에도 2등을 기록해 입상할 수 있었습니다.

문제

- 부산지역의 각 세대별 도시가스 소비량이 주어졌을 때 다음 12개월 동안의 수요량을 예측하는 문제입니다.
- 약 8만 가구의 10년치 도시가스 소비량이 학습데이터로 주어졌으며, 시계열 데이터입니다.
- 거주자가 소비량을 허위계량하는 등의 인위적인 이상치가 섞여있는 데이터입니다.

방향

- 이상치를 제거하기 위해 EDA를 진행하였고, 전처리 과정에서 다양한 통계량 및 클러스터링 기법을 사용하였습니다.
- 시계열 분석이 가능한 모델에 대하여 최대한 다양한 baseline을 만들었습니다.
- 효과가 높은 모델들은 hyperparameter tuning 후, 앙상블 하여 성능을 높였습니다.

자세한 분석내용 링크

- [깃허브 링크]

Portfolio Timeline

2017년 - 첫 프로그래밍, Hangman

2019년 - 첫 데이터 분석, Kaggle

2019년 - 프로젝트, Impact Echo

2020년 - 도시가스 수요량 예측 대회

2020년 - 카카오 아레나

2020년 - 추천시스템 구현 프로젝트

카카오 아레나

Summary. 카카오-멜론에서 주최한 3번째 추천시스템 경진대회로, 비록 좋은 성적을 내지는 못했지만 스스로 성장할 수 있었던 계기가 되었습니다.

문제

- 멜론의 유명 DJ들이 생성한 플레이리스트들이 주어졌을 때, 나머지 비어있는 플레이리스트를 완성하는 문제입니다.
- 플레이리스트의 성격에 알맞는 음악과 태그를 추천해야 합니다.
- 곡 및 장르에 대한 메타데이터가 주어졌으며, 추가로 곡에 대한 Mel spectrogram을 사용할 수 있었습니다.

방향

- item2vec 모델을 이용하여 곡에 대한 임베딩을 추출하였고, 이를 바탕으로 협업필터링 (Collaborative Filtering) 모델을 구축하였습니다.
- 플레이리스트-곡에 대한 interaction 정보뿐만 아니라, 곡에 대한 메타데이터도 활용하고자 노력했습니다.

자세한 분석내용 링크

- [깃허브 링크]

고찰

- 추천시스템에 관심만 있었지 체계적으로 공부하지는 못한 저에겐 굉장히 좋은 경험이었습니다. 특히 실제 데이터를 가지고 전처리부터 모델링까지 경험해볼 수 있어서 좋았습니다.

- 3달이라는 넉넉한 일정에도 불구하고 군인이라는 신분과 과도한 의욕 때문에 컨디션 관리에 실패했었습니다.

- 추천시스템을 학습하기 위한 데이터 형식을 공부할 수 있었고 메타데이터를 학습에 반영시키는 방법에 대해 고민해 볼 수 있었습니다.

- 대회가 끝난 후, 추천시스템 논문을 읽고 구현하는 프로젝트를 시작하여 부족했던 부분을 보완하려 했습니다.

Portfolio Timeline

2017년 - 첫 프로그래밍, Hangman

2019년 - 첫 데이터 분석, Kaggle

2019년 - 프로젝트, Impact Echo

2020년 - 도시가스 수요량 예측 대회

2020년 - 카카오 아레나

2020년 - 추천시스템 구현 프로젝트

추천시스템 논문 구현

- 8월에 카카오 아레나가 끝난 후, 추천시스템 분야의 논문 구현 능력을 키워 폭넓은 코드 스피넷을 확보해야겠다는 생각이 들었습니다.
- AutoRec과 NCF라는 두 모델을 시작으로 8월부터 12월까지 약 20편이 넘는 논문을 읽었습니다.
- Colab에서 tensorflow를 활용하여 baseline을 만든 뒤, 이를 적절히 변형한 top-N 추천시스템을 구현 및 검증하고 있습니다.
- [자세한 프로젝트 내용 링크]

Thank You!