수DA쟁이 1기 개인 프로젝트

2020 DACON CUP

수학과 2019010718 강수연

2020 DACON CUP

지난 2년 동안의 DACON에 관한 여러가지 데이터를 통해

DACON 사용자들의 행동 패턴을 예측하기

Contents

주제 선정 이유 및 목표

주제 선정 이유 : 지난 팀 프로젝트에서 코로나 데이터를 시각화 하면서
시계열 분석에 흥미를 느껴 2020 DACON CUP의 데이터를 주제로 선정

목표 : 2018-09-09 ~ 2020-11-08까지의 DACON의 데이터를 이용하여
2020-11-09~2020-01-08의 데이터를 예측해 보기

데이터 소개

Train.csv

	DateTime	사용자	세션	신규방문자	페이지뷰
0	2018-09-09 00:00:00	19	19	8	206
1	2018-09-09 01:00:00	20	19	9	259
2	2018-09-09 02:00:00	12	9	1	48
3	2018-09-09 03:00:00	10	10	2	102
4	2018-09-09 04:00:00	6	5	3	18
...	...	...	...	...	...
19003	2020-11-08 19:00:00	124	123	19	3128
19004	2020-11-08 20:00:00	166	159	29	4864
19005	2020-11-08 21:00:00	184	173	32	3426
19006	2020-11-08 22:00:00	163	155	34	2845
19007	2020-11-08 23:00:00	160	152	33	3293

19008 rows × 5 columns

훈련시킬 데이터

2018년 9월 9일 ~ 2020년 11월 8일 기간 동안 기록된
한 시간 간격의 사용자 행동 데이터

columns : 사용자 수, 세션 수, 신규 방문자 수, 페이지 뷰 수

데이터 소개

Info_user.csv

	유저 id	아이디 생성 시점	대회 참여 횟수	코드 공유 횟수	토론 횟수	국가	파이썬 수준	데이터 사이언스 수준
1	19195.0	2018-08-08 18:21	1.0	0.0	0.0	82.0	1.0	1.0
2	16339.0	2018-08-08 19:57	1.0	0.0	0.0	82.0	1.0	1.0
3	7290.0	2018-08-08 19:59	0.0	0.0	0.0	82.0	1.0	1.0
4	21287.0	2018-08-08 20:26	0.0	0.0	0.0	82.0	1.0	1.0
5	3828.0	2018-08-08 20:28	0.0	0.0	0.0	82.0	1.0	1.0
...	...	...	...	...	...	...	...	...
21687	11584.0	2020-12-08 22:53	1.0	0.0	0.0	82.0	1.0	1.0
21688	6337.0	2020-12-08 22:54	1.0	0.0	0.0	82.0	1.0	1.0
21689	2505.0	2020-12-08 23:07	1.0	0.0	0.0	82.0	1.0	1.0
21690	14182.0	2020-12-08 23:09	1.0	0.0	0.0	82.0	1.0	1.0
21691	8843.0	2020-12-08 23:13	2.0	0.0	0.0	82.0	1.0	1.0

21691 rows × 8 columns

데이터 소개

Info_competition.csv

	대회 id	대회 시작 시점	대회 종료 시점	대회 이름	키워드	참여자 수	최대 팀 멤버	하루 최대 제출 횟수	상금	상금 정보	팀 병합 데드라인	보여지는 위너 수
0	136	2018-08-14 0:00	2018-09-13 23:59	대출 상점 총 매출 예측 경진대회	금융 | 소상공인 신용카드 가맹점 빅데이터와 AI로 매출 예측 | 시계열, 회귀 ...	303	5	5	850	$8,500 + 100,000ZPR	2018-09-13 23:59	3
1	9565	2018-09-15 0:00	2018-10-13 23:59	병원 개/폐업 분류 예측 경진대회	금융 | 병원 재무 데이터와 AI로 개업|폐업 예측 분석 | 분류 | Accuracy	448	10	3	350	$3,500 + 40,000ZPR	2018-10-13 23:59	3
2	17801	2018-10-18 0:00	2018-12-31 23:59	아파트 경매가격 예측 경진대회	금융 | 부동산 아파트 경매 빅데이터와 AI로 경매가 예측 분석 | 회귀 | RMSE	316	10	3	1000	$10,000 + 120,000ZPR	2018-12-31 23:59	3
3	21265	2018-11-13 0:00	2019-01-31 23:59	아파트 실거래가 예측	금융 | 부동산 빅데이터와 AI를 이용하여 실거래가를 예측 분석 | 회귀 | RMSE	568	10	3	800	$8,000 + 80,000ZPR	2019-01-31 23:59	3
4	42473	2018-12-25 0:00	2019-01-10 23:59	신용카드 거래 데이터 시각화	금융 | 신용카드, 시계열, 시각화 | Python, R, Tableau, Spot...	252	5	3	100	$1,000 + 100,000ZPR	2019-01-10 23:59	3
5	62540	2019-02-08 0:00	2019-07-18 23:59	KBO 타자 OPS 예측 경진대회	스포츠 | KBO 타자 빅데이터와 AI로 OPS 예측 | 시계열, 회귀 | RMSE...	341	5	3	800	800만원	2019-07-18 23:59	3
6	68346	2019-03-26 0:00	2019-05-20 23:59	KBO 외국인 투수 스카우팅 최적화 경진대회	스포츠 | MLB 데이터와 AI를 이용 투수 스카우트 분석 | KBO, 회귀 | ...	98	1	3	200	200만원	2019-05-20 23:59	3
7	82407	2019-05-06 0:00	2019-07-08 23:59	KCB 금융스타일 시각화 경진대회	금융 | 구인 | 개인 신용카드 빅데이터 AI 분석 시각화 | Python, R, ...	269	5	3	1000	1,000만원	2019-07-08 23:59	3
8	140472	2019-07-11 0:00	2019-10-21 23:59	상점 신용카드 매출 예측 경진대회	금융 | 구인 | 소상공인 가맹점 신용카드 빅데이터와 AI로 매출 예측 분석 | 시...	624	5	100	200	200만원	2020-10-28 23:59	3
9	196878	2019-10-01 0:00	2019-10-27 23:59	전력 수요량 예측 경진대회	공공 | 전력 기상 빅데이터와 AI로 수요량 분석 | 시계열, 회귀 | SMAPE	478	4	3	600	600만원	2019-10-27 23:59	3

데이터 소개

Info_submission.csv

	제출 아이디	제출 대회 아이디	팀 아이디	유저 아이디	제출 시점
0	-250918.0	229255.0	5019.0	15880.0	2019-11-24 20:40
1	-250904.0	229611.0	5162.0	19772.0	2019-11-24 20:36
2	-250881.0	235401.0	5042.0	17801.0	2019-11-24 20:24
3	-250873.0	235401.0	5024.0	439.0	2019-11-24 20:23
4	-250860.0	229255.0	4890.0	13722.0	2019-11-24 20:19
...	...	...	...	...	...
101582	483653.0	235658.0	44883.0	11628.0	2020-12-08 23:53
101583	483654.0	235671.0	45009.0	10592.0	2020-12-08 23:56
101584	483655.0	235671.0	45013.0	5987.0	2020-12-08 23:56
101585	483656.0	235658.0	42355.0	6586.0	2020-12-08 23:57
101586	483657.0	235658.0	42355.0	6586.0	2020-12-08 23:57

99203 rows × 5 columns

데이터 소개

Info_login.csv

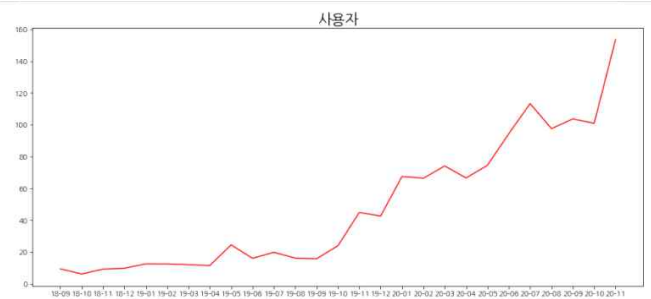
	로그인 id	유저 id	로그인 시점	로그인한 플랫폼	로그인한 브라우저
0	14196.0	19195.0	2018-09-23 1:30	Windows 8	Internet Explorer 10.0
1	14234.0	22045.0	2018-09-23 2:46	Windows	Chrome 67.0.3396.99
2	14256.0	8790.0	2018-09-23 3:35	Windows	Chrome 69.0.3497.100
5	14280.0	10206.0	2018-09-23 13:24	Windows	Chrome 68.0.3440.106
6	14288.0	488.0	2018-09-23 14:05	Windows 7	Chrome 68.0.3440.106
...	...	...	...	...	...
64649	328208.0	16088.0	2019-12-30 23:19	Windows	Firefox 71.0
64650	328209.0	14107.0	2019-12-30 23:21	Windows	Chrome 79.0.3945.88
64651	328210.0	1283.0	2019-12-30 23:21	Windows	Chrome 79.0.3945.88
64652	328218.0	4077.0	2019-12-30 23:24	Windows	Chrome 79.0.3945.88
64653	328219.0	21531.0	2019-12-30 23:32	Apple	Chrome 78.0.3904.108

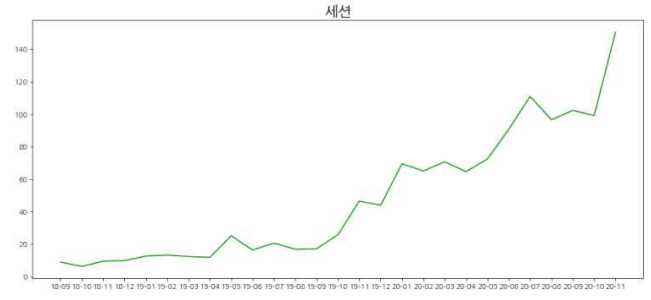
60397 rows × 5 columns

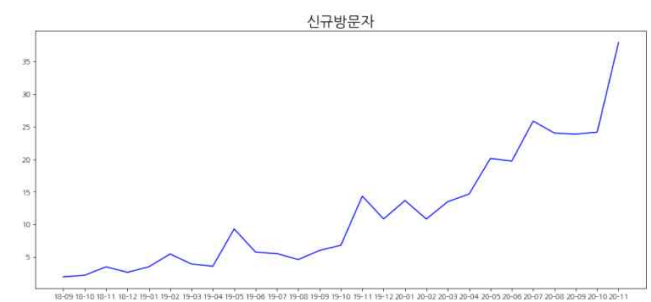
EDA (Exploratory Data Analysis)

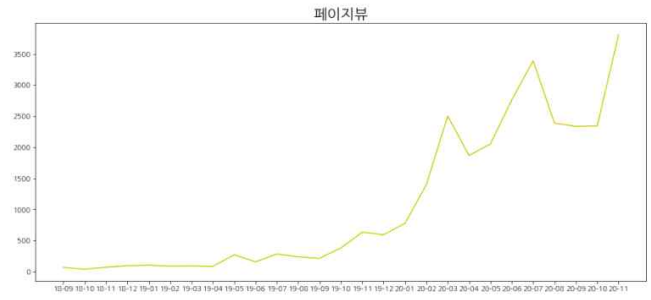
Train.csv

월별에 따른 사용자 데이터





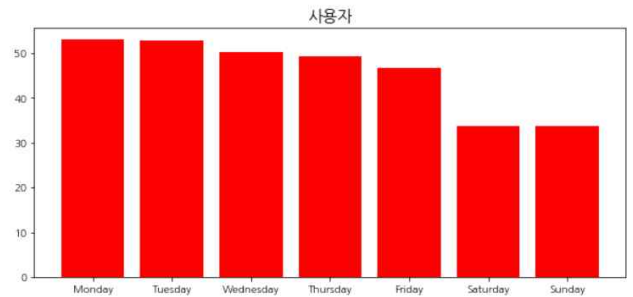


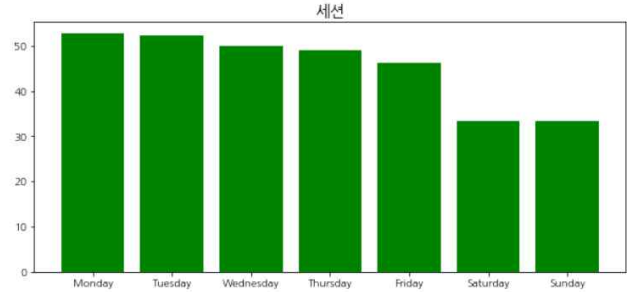


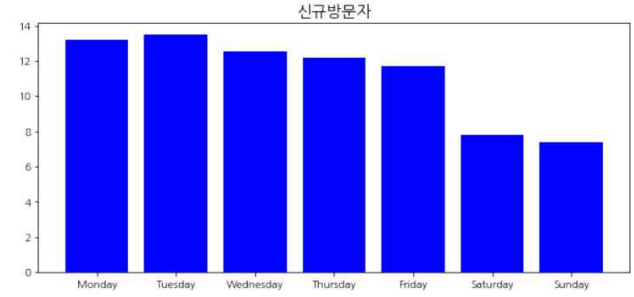
EDA (Exploratory Data Analysis)

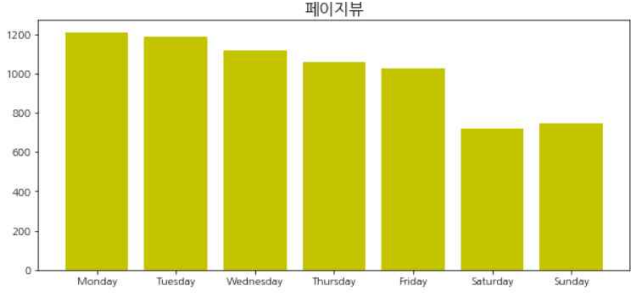
Train.csv

요일별에 따른 사용자 데이터





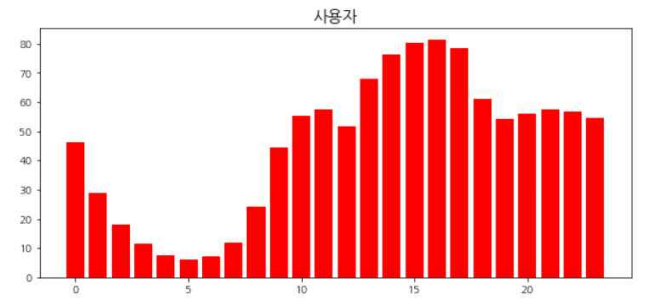


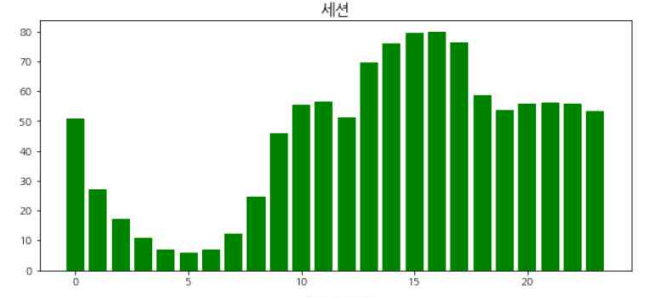


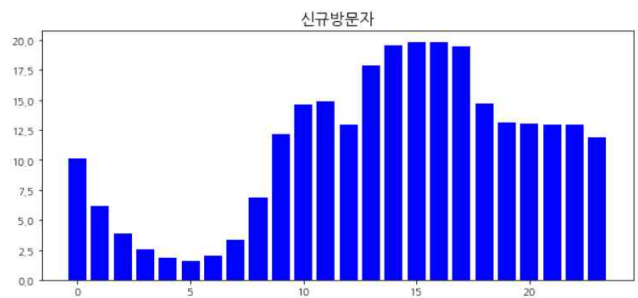
EDA (Exploratory Data Analysis)

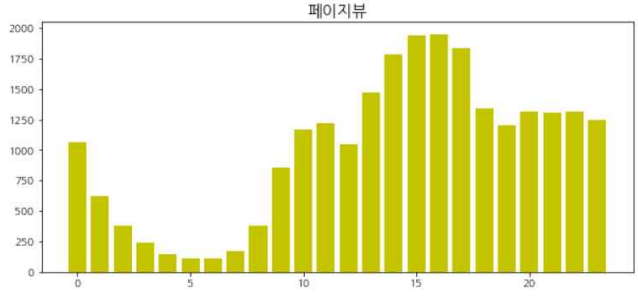
Train.csv

시간대별에 따른 사용자 데이터



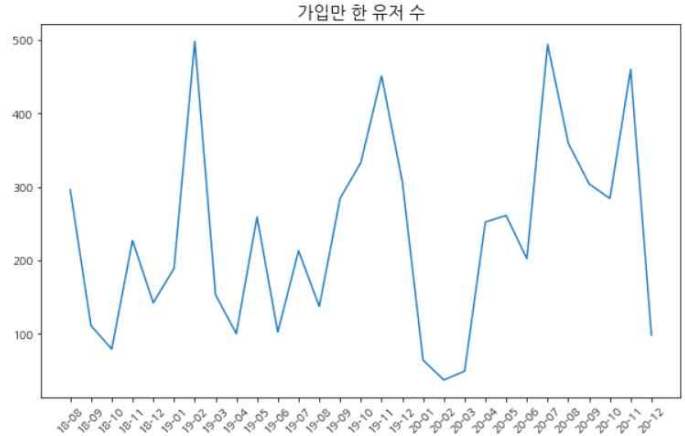






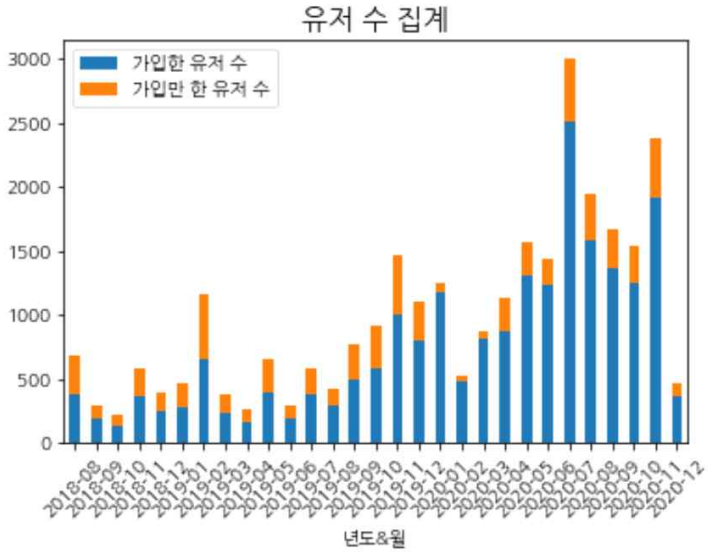
EDA (Exploratory Data Analysis)

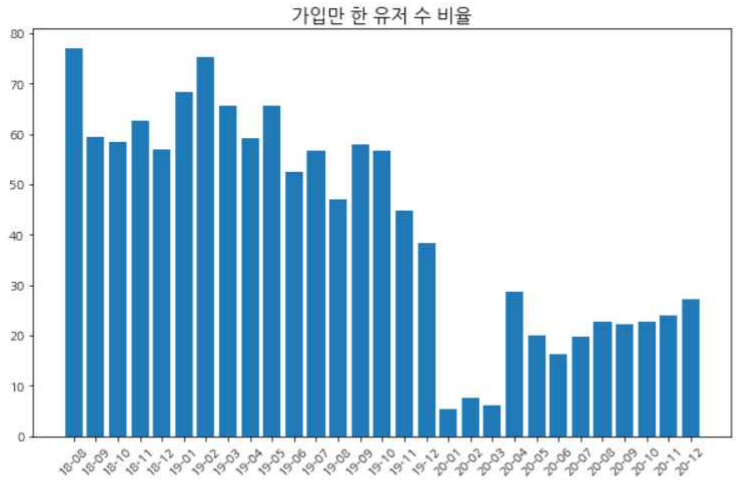
Info_user.csv



EDA (Exploratory Data Analysis)

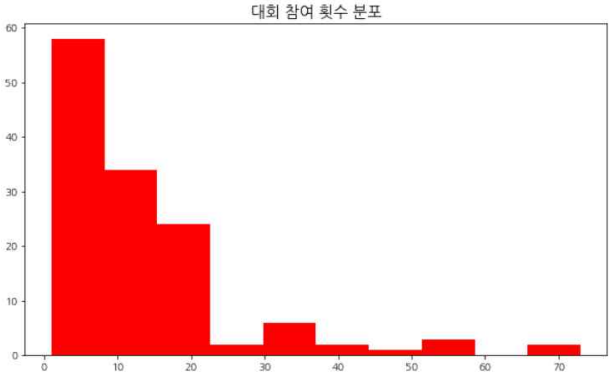
Info_user.csv

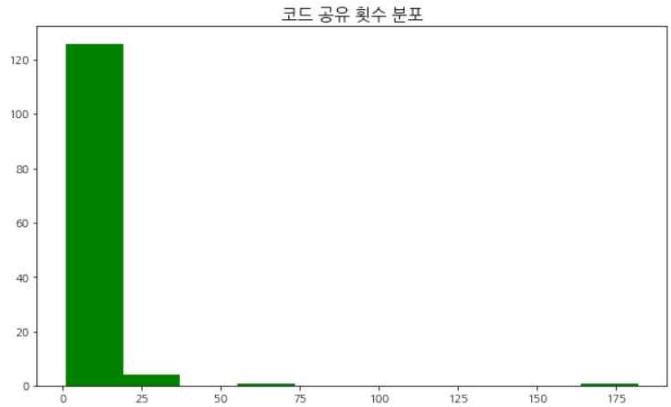


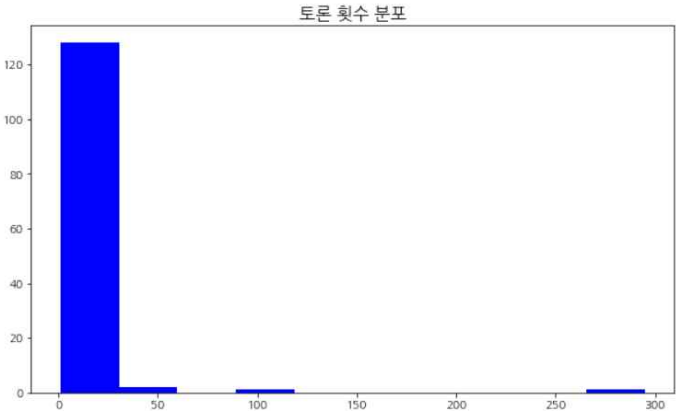


EDA

Info_competition.csv

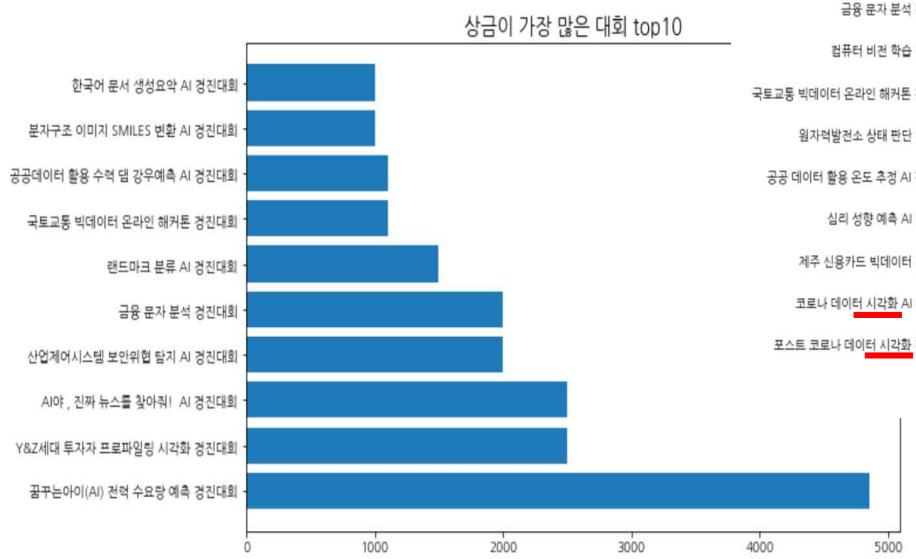


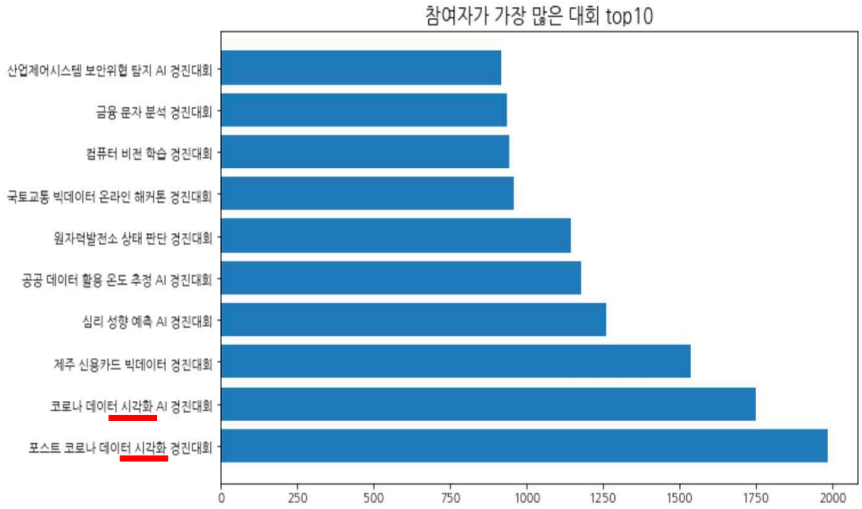




EDA (Exploratory Data Analysis)

Info_competition.csv





020_DACON_CUP Last Checkpoint: 16시간 전 (autosaved)

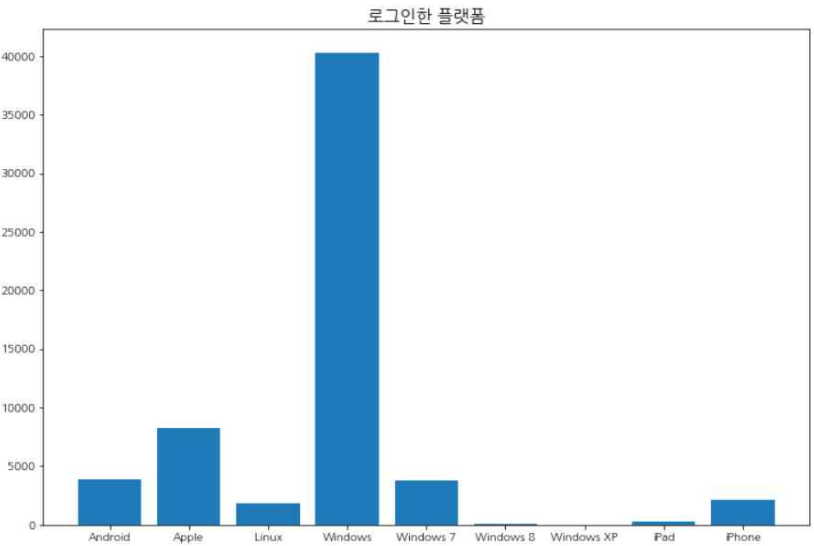
Logout

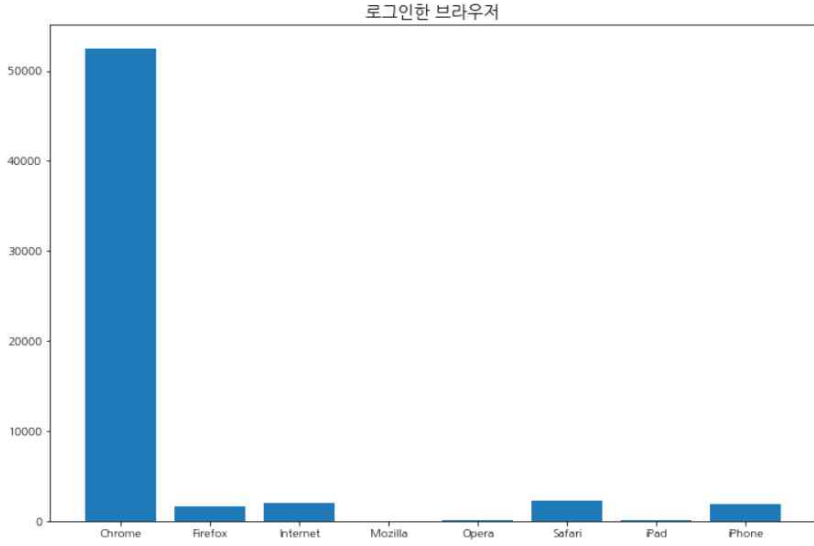
ew Insert Cell Kernel Widgets Help

Trusted Python 3

Run Code

```
lt.figure(figsize=(12,8))
lt.bar(login_platform['로그인한 플랫폼'],login_platform['유저 id'])
lt.title('로그인한 플랫폼',size=15)
lt.show()
```



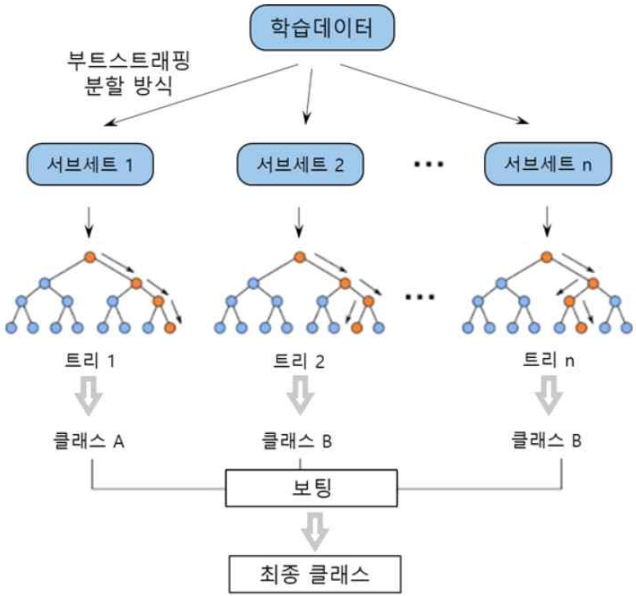



```
ogin['로그인한 브라우저']=login['로그인한 브라우저'].str.split(' ').str[0]
```

모두 표시

오후 6:25
2021-01-12

Random Forest



이미지 출처 : 파이썬 머신러닝 완벽 가이드, https://en.wikipedia.org/wiki/Random_forest

랜덤 포레스트

여러 개의 결정 트리를 랜덤으로 학습

하나의 데이터 집합에서 임의복원 추출을 통해 여러 개의 훈련용 데이터를 생성한다.

여러 번의 학습을 통해 여러 개의 트리를 생성하고, 이를 결합하여 최종적으로 목표변수를 예측

Random Forest

사용자

In [530]:
```
train1=pd.DataFrame(train_["사용자"])
```

In [531]:
```
split_date = pd.Timestamp('08-30-2020')
train = train1.loc[:split_date, ]
test = train1.loc[split_date:, ]
```

In [532]:
```
sc = MinMaxScaler()
train_sc = sc.fit_transform(train)
test_sc = sc.transform(test)
```

MinMaxScaler : 0~1 사이의 값으로 조정

In [533]:
```
train_sc_df = pd.DataFrame(train_sc, columns=["사용자"], index=train.index)
test_sc_df = pd.DataFrame(test_sc, columns=["사용자"], index=test.index)
```

In [534]:
```
WINDOW_SIZE = 8
train_shift = train_sc_df.copy()
test_shift = test_sc_df.copy()
```

In [535]:
```
for s in range(1, WINDOW_SIZE + 1):
    train_shift["shift_사용자_{}".format(s)] = train_shift["사용자"].shift(s)
    test_shift["shift_사용자_{}".format(s)] = test_shift["사용자"].shift(s)
```

In [536]:
```
X_train = train_shift.dropna().drop("사용자", axis=1)
y_train = train_shift.dropna()[["사용자"]]

X_test = test_shift.dropna().drop("사용자", axis=1)
y_test = test_shift.dropna()[["사용자"]]

X_train_val = X_train.values
y_train_val = y_train.values

X_test_val = X_test.values
y_test_val = y_test.values
```

dropna() : 결측치 제거

Random Forest

In [536]:
```
X_train = train_shift.dropna().drop("사용자", axis=1)
y_train = train_shift.dropna()[["사용자"]]

X_test = test_shift.dropna().drop("사용자", axis=1)
y_test = test_shift.dropna()[["사용자"]]

X_train_val = X_train.values
y_train_val = y_train.values

X_test_val = X_test.values
y_test_val = y_test.values
```

In [537]:
```
rf_model= RandomForestRegressor(bootstrap=True, n_estimators=30, n_jobs=1, random_state=1104, verbose=0)
rf_model.fit(X_train, y_train)
```

Out[537]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=30, n_jobs=1,
                      oob_score=False, random_state=1104, verbose=0,
                      warm_start=False)
```

In [538]:
```
rf_pred = rf_model.predict(X_test)
```

In [539]:
```
print(mean_squared_error(y_test, rf_pred))
```

mean_squared_error : 평균 제곱근 오차(RMSE)

```
0.013144491712390776
```

In [541]:
```
rf_result=np.expm1(rf_model.predict(X_test))
test_og["사용자"] = sc.inverse_transform((rf_result[:-2]).reshape(-1, 1))
```

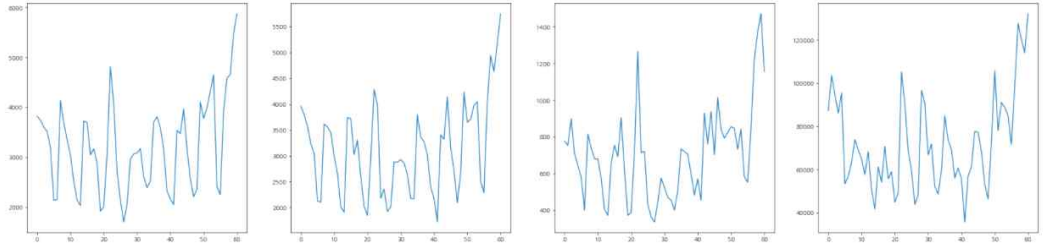
Random Forest

In [576]:
```
test_og.head()
```

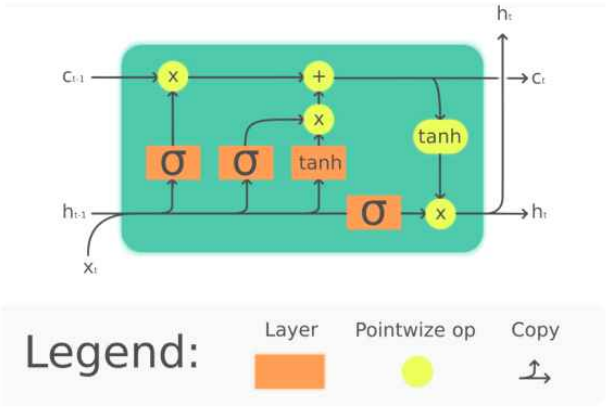
Out[576]:

	DateTime	사용자	세션	신규방문자	페이지뷰
0	2020-11-09	3820.350466	3961.643505	776.467362	87333.536196
1	2020-11-10	3739.961893	3790.078818	752.681909	103618.290320
2	2020-11-11	3603.498245	3569.890892	899.655905	94620.311819
3	2020-11-12	3513.894406	3233.407925	707.313907	86050.551724
4	2020-11-13	3190.634622	3040.363786	644.040415	95461.189955

In [606]:
```
plt.figure(figsize=(30,7))
plt.subplot(141)
plt.plot(test_og["사용자"])
plt.subplot(142)
plt.plot(test_og["세션"])
plt.subplot(143)
plt.plot(test_og["신규방문자"])
plt.subplot(144)
plt.plot(test_og["페이지뷰"])
plt.show()
```



LSTM



LSTM

RNN의 한 종류로 순차적인 방식으로 입력값을 처리하는 딥러닝 기법

RNN의 기울기 소실 문제를 극복해줌

LSTM

LSTM

In [298]:
```
from keras.models import Sequential, Model, load_model
from keras.layers import Input, Dense, Activation, Flatten, Dropout
from keras.layers import LSTM
```

필요한 라이브러리 import

```
Using TensorFlow backend.
```

In [299]:
```
train_
```

Out[299]:

date	사용자	세션	신규방문자	페이지뷰
2018-09-09	281	266	73	1826
2018-09-10	264	247	51	2092
2018-09-11	329	310	58	1998
2018-09-12	300	287	45	2595
2018-09-13	378	344	50	3845
...	...	...	...	...
2020-11-04	4516	4472	1196	112683
2020-11-05	4155	4037	1044	102901
2020-11-06	3663	3576	825	88015
2020-11-07	2472	2417	531	57386
2020-11-08	2492	2420	522	50486

792 rows × 4 columns

LSTM

In [301]:
```
from sklearn import preprocessing
scaler = preprocessing.MinMaxScaler()
scaler_fit = scaler.fit(train_)
scaled_train_=scaler_fit.transform(train_)

scaled_train_
```

스케일링

Out[301]:
```
array([[0.05168913, 0.04908316, 0.04260355, 0.01173496],
       [0.04855086, 0.04556399, 0.0295858 , 0.01344631],
       [0.06055012, 0.05723282, 0.03372781, 0.01284155],
       ...,
       [0.67601994, 0.66215966, 0.48757396, 0.56624398],
       [0.45615654, 0.44749028, 0.31360947, 0.36918801],
       [0.45984862, 0.44804593, 0.30828402, 0.32479589]])
```

In [303]:
```
size=len(train_)
sequence=7
dropout=0.3
epoch=180
batch_size=14
verbose=1


dt_index = pd.date_range(start='20201109', end='20210108')
mylter=len(dt_index)
```

파라미터 설정

예측 구간 갯수

LSTM

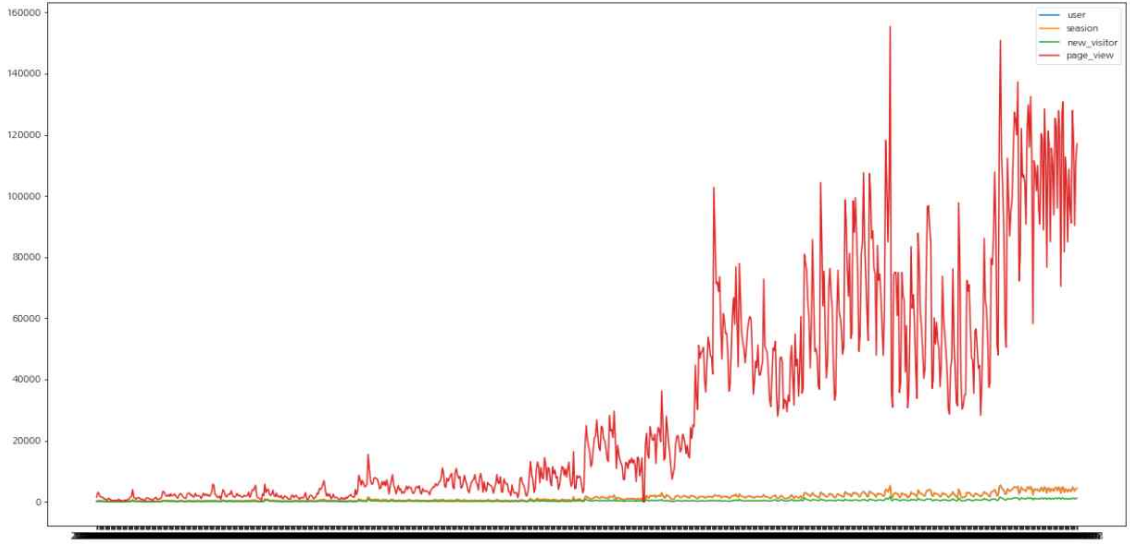
In [305]:
```
X_train,Y_train=[],[]

for i in range(size-sequence):
    X_train.append(np.array(scaled_train_[i:i+sequence]))
    Y_train.append(np.array(scaled_train_[i+sequence]))

X_train=np.array(X_train)
Y_train=np.array(Y_train)
print(len(X_train),len(Y_train))

for i in range(mylter):
    print(i+1,"반복")
    model = Sequential()
    model.add(LSTM(128, input_shape=(X_train.shape[1], X_train.shape[2]), activat
    model.add(Dropout(dropout))
    model.add(LSTM(64, activation="relu", return_sequences=False))
    model.add(Dropout(dropout))
    model.add(Dense(4))
    model.compile(optimizer='adam', loss='mean_squared_error')

    model_fit = model.fit(X_train, Y_train,
                          batch_size=batch_size, epochs=epoch,
                          verbose=verbose)

    last_X=np.concatenate((X_train[-1][1:],np.array([Y_train[-1]])),axis=0)
    last_Y = model.predict(np.array([last_X]))

    new_X_train=np.concatenate((X_train[:],np.array([last_X])),axis=0)
    new_Y_train=np.concatenate((Y_train[:],last_Y),axis=0)

    X_train=np.array(new_X_train)
    Y_train=np.array(new_Y_train)

model.summary()
```

1스텝 교차검사

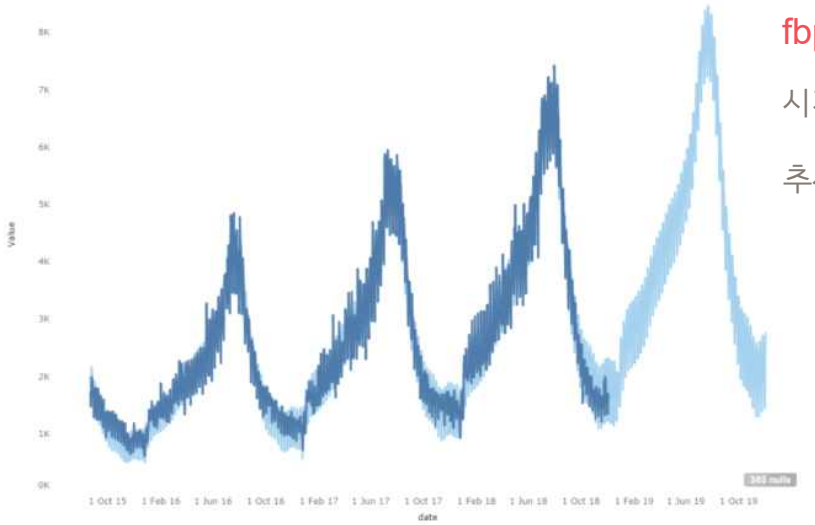
예측 개수만큼 반복

예측값 train set에 추가

LSTM


160000
140000
120000
100000
80000
60000
40000
20000
0
user
session
new_visitor
page_view

Facebook prophet



fbprophet

시계열 자료를 예측하는데 쓰이는 모델

추세의 변화를 파악하는 데 용이

Facebook prophet

facebook Prophet

In [588]:
```
train_og=pd.read_csv('train.csv',encoding='cp949')
```

In [589]:
```
from fbprophet import Prophet
train_fb = train_og.copy()
train_fb['DateTime'] = pd.to_datetime(train_fb.DateTime)
train_fb['ds'] = train_fb.DateTime.dt.date
train_fb = train_fb.groupby('ds').sum().reset_index()
train_fb = train_fb.set_index('ds')
```

In [590]:
```
train_fb
```

Out[590]:

ds	사용자	세션	신규방문자	페이지뷰
2018-09-09	281	266	73	1826
2018-09-10	264	247	51	2092
2018-09-11	329	310	58	1998
2018-09-12	300	287	45	2595
2018-09-13	378	344	50	3845
...	...	...	...	...
2020-11-04	4516	4472	1196	112683
2020-11-05	4155	4037	1044	102901
2020-11-06	3663	3576	825	88015
2020-11-07	2472	2417	531	57386
2020-11-08	2492	2420	522	50486

792 rows × 4 columns

Facebook prophet

In [591]:
```
train1 = pd.DataFrame(train_fb["사용자"])
train1_sc = sc.fit_transform(train1)
```

In [592]:
```
train1_sc_df = pd.DataFrame(train1_sc, columns=["사용자"], index=train1.index)
```

In [593]:
```
train1_sc_df.rename(columns={'사용자':'y'},inplace=True)
```

In [594]:
```
train1_sc_df=train1_sc_df.reset_index('ds')
train1_sc_df['ds']=pd.to_datetime(train1_sc_df['ds'])
```

In [595]:
```
train1_sc_df
```

Out[595]:

	ds	y
0	2018-09-09	0.051689
1	2018-09-10	0.048551
2	2018-09-11	0.060550
3	2018-09-12	0.055197
4	2018-09-13	0.069596
...	...	...
787	2020-11-04	0.833487
788	2020-11-05	0.766845
789	2020-11-06	0.676020
790	2020-11-07	0.456157
791	2020-11-08	0.459849

792 rows × 2 columns

Facebook prophet

In [596]:
```
model = Prophet()
model.fit(train1_sc_df)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-596-fe430ea0e77a> in <module>
      1 model = Prophet()
----> 2 model.fit(train1_sc_df)

C:₩Anaconda3₩lib₩site-packages₩fbprophet₩forecaster.py in fit(self, df, **kwargs)
   1088         self.history_dates = pd.to_datetime(df['ds']).sort_values()
   1089
-> 1090         history = self.setup_dataframe(history, initialize_scales=True)
   1091         self.history = history
   1092         self.set_auto_seasonalities()

C:₩Anaconda3₩lib₩site-packages₩fbprophet₩forecaster.py in setup_dataframe(self, df, initialize_scales)
    320         df['t'] = (df['ds'] - self.start) / self.t_scale
    321         if 'y' in df:
--> 322             df['y_scaled'] = (df['y'] - df['floor']) / self.y_scale
    323
    324         for name, props in self.extra_regressors.items():

C:₩Anaconda3₩lib₩site-packages₩pandas₩core₩ops₩__init__.py in wrapper(left, right)

C:₩Anaconda3₩lib₩site-packages₩pandas₩core₩ops₩__init__.py in na_op(x, y)

C:₩Anaconda3₩lib₩site-packages₩pandas₩core₩computation₩expressions.py in evaluate(op, a, b, use_numexpr)
    230     op_str = _op_str_mapping[op]
    231     if op_str is not None:
--> 232         use_numexpr = use_numexpr and _bool_arith_check(op_str, a, b)
    233         if use_numexpr:
    234             # error: "None" not callable

ValueError: The truth value of an array with more than one element is ambiguous. Use a.any() or a.all()
```

Facebook prophet

In [287]:
```
train_pred = model.predict(train1_sc_df)
model.plot(train_pred)
```

```
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
<ipython-input-287-37e655ded476> in <module>
----> 1 train_pred = model.predict(train1_sc_df)
      2 model.plot(train_pred)

C:\Anaconda3\lib\site-packages\fbprophet\forecaster.py in predict(self, df)
   1165         """
   1166         if self.history is None:
-> 1167             raise Exception('Model has not been fit.')
   1168
   1169         if df is None:

Exception: Model has not been fit.
```

느낀 점

여러 적절한 컬럼을 조합한 뒤 학습시켜보고 싶었지만 전처리 과정에서 어려움을 겪어 결국 실패

Facebook prophet 모델이 피팅되지 않는 원인을 찾지 못함

다른 모델도 적용시켜보려 했지만 제대로 구현해내지 못함

뭔가 할 수 있을 것 같은데 해결하지 못해 아쉬움이 크다. 아직까지 많이 부족하다는 생각이 들었고 좀 더 분발해야겠다고 느꼈다.

감사합니다