머신러닝 프로젝트

강동엽, 안은선, 이상훈, 주한솔

목차

1. 프로젝트 주제소개
2. 탐색적 데이터 분석 및 전처리
3. 머신러닝 성능 올리기
4. 최종 예측과 성능평가

프로젝트 주제

제주도 버스의 퇴근시간 승차인원 예측

알고리즘 | 회귀 | RMSE

성능평가지표 (RMSE)

기본 데이터

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

```
X = train.drop(['18~20_ride', 'date', 'in_out', 'station_name'], axis = 1)
y = train['18~20_ride']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

```
rf = RandomForestRegressor(random_state=42, n_estimators = 100)
neg_mse_scores = cross_val_score(rf, X, y, scoring = "neg_mean_squared_error", cv=3, n_jobs=-1)
rmse_scores = np.sqrt(-1 * neg_mse_scores)
avg_rmse = np.mean(rmse_scores)

print('Negative MSE scores: ', np.round(neg_mse_scores, 2))
print('개별 RMSE scores : ', np.round(rmse_scores, 2))
print('평균 RMSE : {0:.3f}'.format(avg_rmse))
```

```
Negative MSE scores:  [-12.06 -10.43 -11.  ]
개별 RMSE scores :  [3.47 3.23 3.32]
평균 RMSE : 3.340
```

- 3.340

탐색적 데이터 분석

데이터 훑어보기

train & test 컬럼	설명
id	해당 데이터에서의 고유한 ID(train, test와의 중복은 없음)
date	날짜
bus_route_id	노선ID
in_out	시내버스, 시외버스 구분
station_code	해당 승하차 정류소의 ID
station_name	해당 승하차 정류소의 이름
latitude	해당 버스 정류장의 위도 (같은 정류장 이름이어도 버스의 진행 방향에 따라 다를 수 있음)
longitude	해당 버스 정류장의 경도 (같은 정류장 이름이어도 버스의 진행 방향에 따라 다를 수 있음)
X~Y_ride	X:00:00부터 X:59:59까지 승차한 인원 수
X~Y_takeoff	X:00:00부터 X:59:59까지 하차한 인원 수
18~20_ride	18:00:00부터 19:59:59까지 승차한 인원 수 (train data에만 존재)

```
train.info()
```

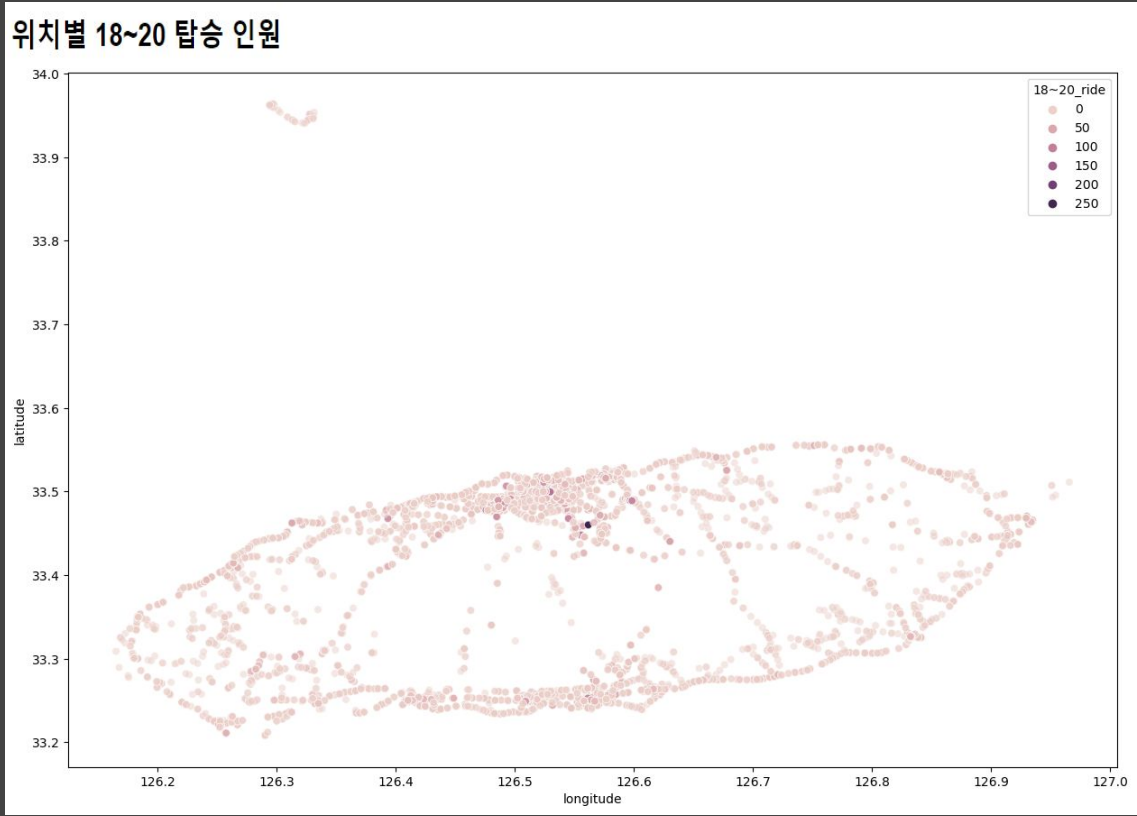
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 415423 entries, 0 to 415422
Data columns (total 21 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   id              415423 non-null  int64
 1   date            415423 non-null  object
 2   bus_route_id    415423 non-null  int64
 3   in_out          415423 non-null  object
 4   station_code    415423 non-null  int64
 5   station_name    415423 non-null  object
 6   latitude        415423 non-null  float64
 7   longitude       415423 non-null  float64
 8   6~7_ride        415423 non-null  float64
 9   7~8_ride        415423 non-null  float64
 10  8~9_ride        415423 non-null  float64
 11  9~10_ride       415423 non-null  float64
 12  10~11_ride      415423 non-null  float64
 13  11~12_ride      415423 non-null  float64
 14  6~7_takeoff     415423 non-null  float64
 15  7~8_takeoff     415423 non-null  float64
 16  8~9_takeoff     415423 non-null  float64
 17  9~10_takeoff    415423 non-null  float64
 18  10~11_takeoff   415423 non-null  float64
 19  11~12_takeoff   415423 non-null  float64
 20  18~20_ride      415423 non-null  float64
dtypes: float64(15), int64(3), object(3)
memory usage: 66.6+ MB
```

```
test.info()
```

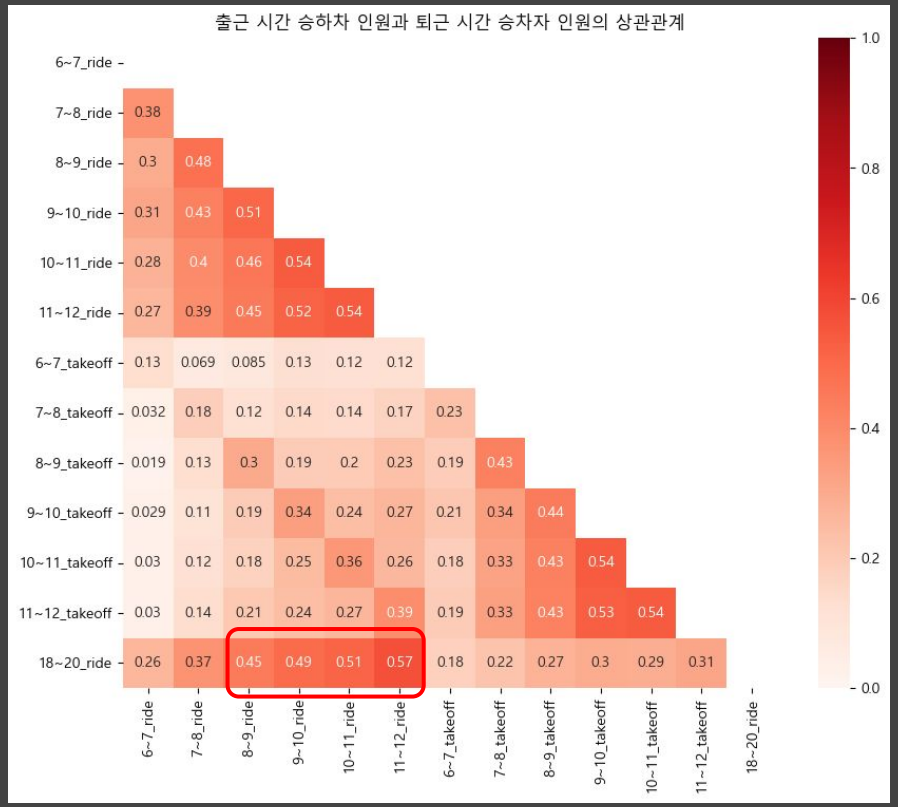
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 228170 entries, 0 to 228169
Data columns (total 20 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   id              228170 non-null  int64
 1   date            228170 non-null  object
 2   bus_route_id    228170 non-null  int64
 3   in_out          228170 non-null  object
 4   station_code    228170 non-null  int64
 5   station_name    228170 non-null  object
 6   latitude        228170 non-null  float64
 7   longitude       228170 non-null  float64
 8   6~7_ride        228170 non-null  float64
 9   7~8_ride        228170 non-null  float64
 10  8~9_ride        228170 non-null  float64
 11  9~10_ride       228170 non-null  float64
 12  10~11_ride      228170 non-null  float64
 13  11~12_ride      228170 non-null  float64
 14  6~7_takeoff     228170 non-null  float64
 15  7~8_takeoff     228170 non-null  float64
 16  8~9_takeoff     228170 non-null  float64
 17  9~10_takeoff    228170 non-null  float64
 18  10~11_takeoff   228170 non-null  float64
 19  11~12_takeoff   228170 non-null  float64
dtypes: float64(14), int64(3), object(3)
memory usage: 34.8+ MB
```

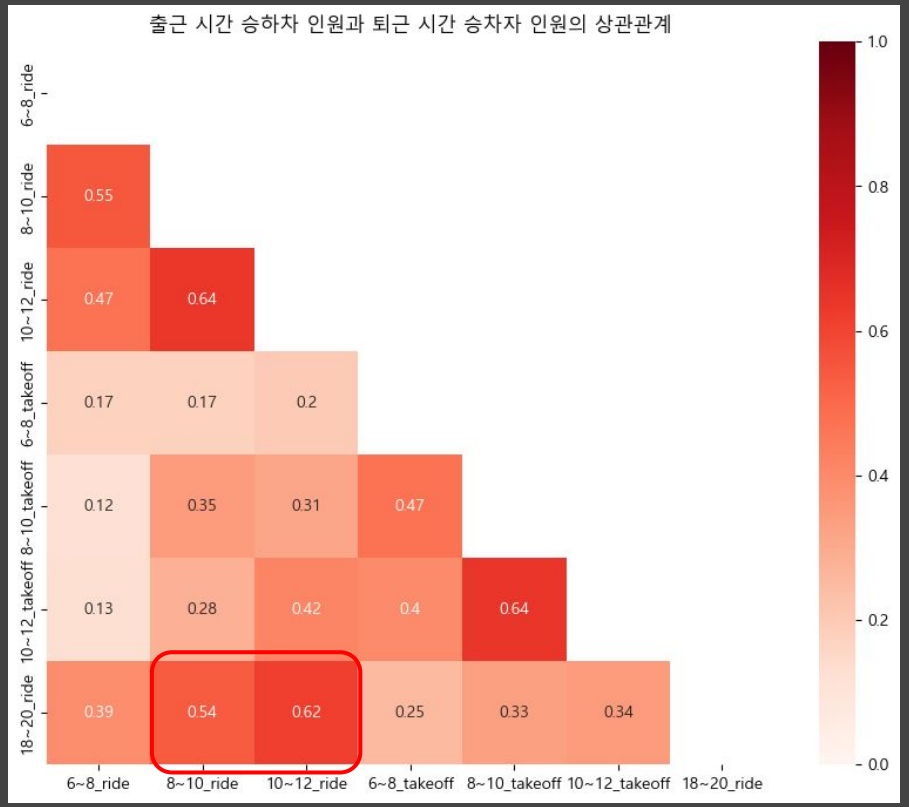
데이터 시각화

데이터 시각화

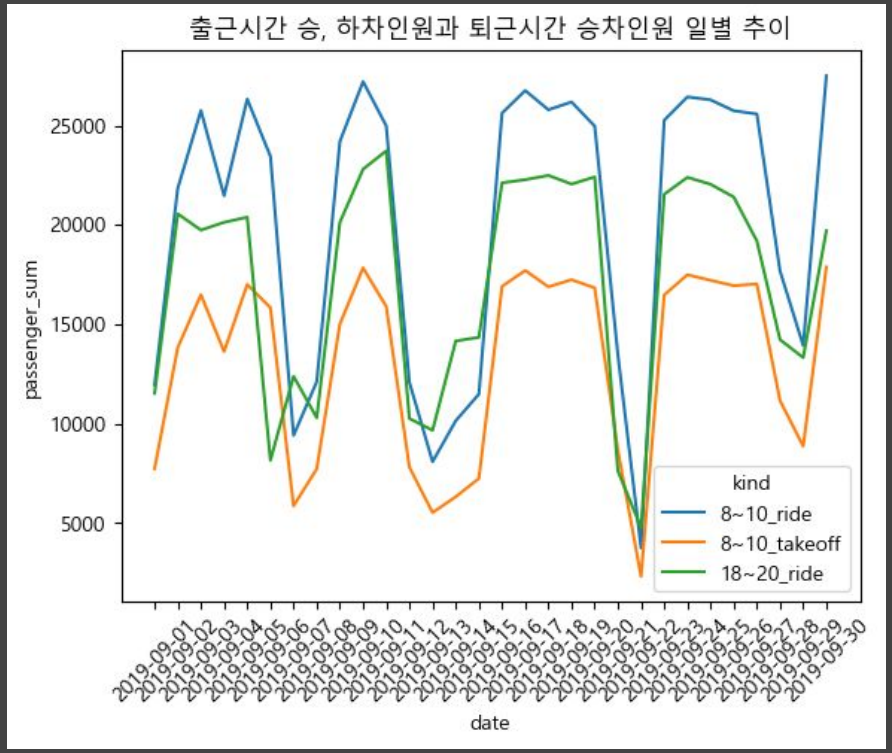

위치별 18~20 탑승 인원
18~20_ride
0
50
100
150
200
250
34.0
33.9
33.8
33.7
33.6
33.5
33.4
33.3
33.2
latitude
126.2
126.3
126.4
126.5
126.6
126.7
126.8
126.9
127.0
longitude

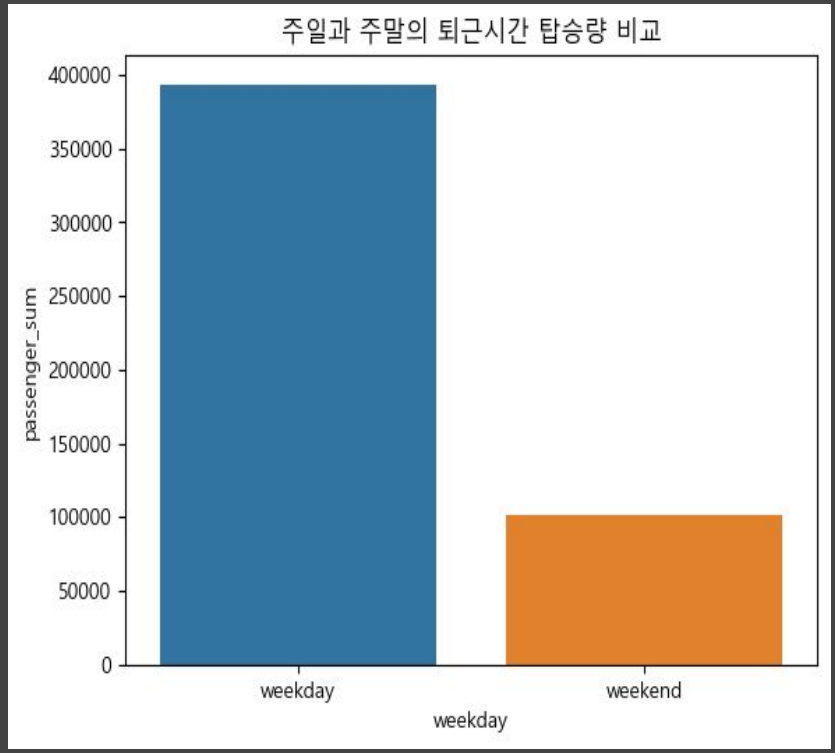
데이터 시각화


출근 시간 승하차 인원과 퇴근 시간 승차자 인원의 상관관계
6~7_ride
7~8_ride 0.38
8~9_ride 0.3 0.48
9~10_ride 0.31 0.43 0.51
10~11_ride 0.28 0.4 0.46 0.54
11~12_ride 0.27 0.39 0.45 0.52 0.54
6~7_takeoff 0.13 0.069 0.085 0.13 0.12 0.12
7~8_takeoff 0.032 0.18 0.12 0.14 0.14 0.17 0.23
8~9_takeoff 0.019 0.13 0.3 0.19 0.2 0.23 0.19 0.43
9~10_takeoff 0.029 0.11 0.19 0.34 0.24 0.27 0.21 0.34 0.44
10~11_takeoff 0.03 0.12 0.18 0.25 0.36 0.26 0.18 0.33 0.43 0.54
11~12_takeoff 0.03 0.14 0.21 0.24 0.27 0.39 0.19 0.33 0.43 0.53 0.54
18~20_ride 0.26 0.37 0.45 0.49 0.51 0.57 0.18 0.22 0.27 0.3 0.29 0.31
6~7_ride 7~8_ride 8~9_ride 9~10_ride 10~11_ride 11~12_ride 6~7_takeoff 7~8_takeoff 8~9_takeoff 9~10_takeoff 10~11_takeoff 11~12_takeoff 18~20_ride
1.0 0.8 0.6 0.4 0.2 0.0

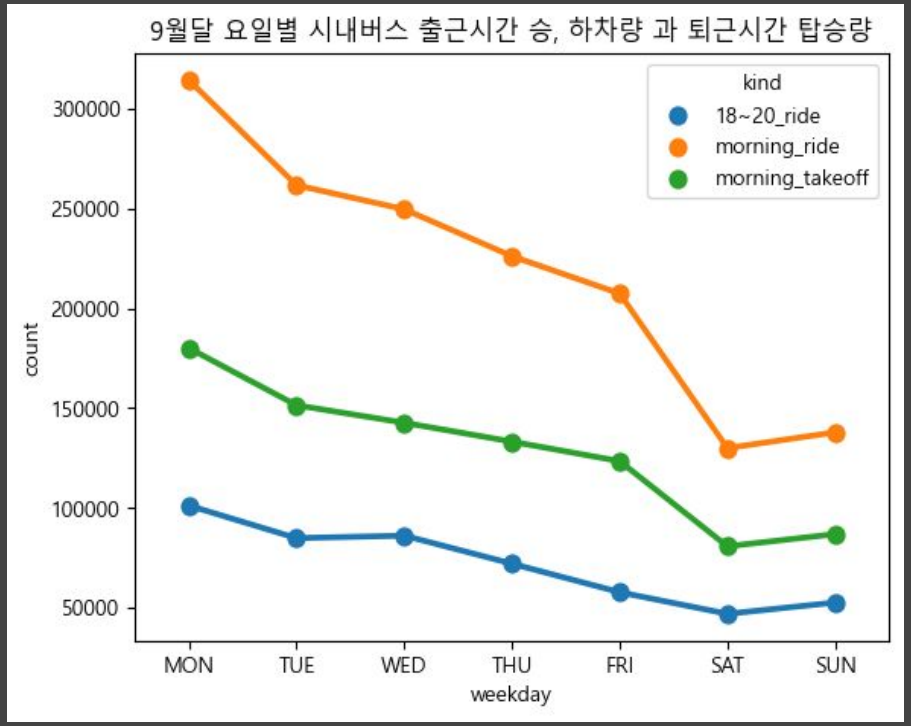

출근 시간 승하차 인원과 퇴근 시간 승차자 인원의 상관관계
6~8_ride
8~10_ride 0.55
10~12_ride 0.47 0.64
6~8_takeoff 0.17 0.17 0.2
8~10_takeoff 0.12 0.35 0.31 0.47
10~12_takeoff 0.13 0.28 0.42 0.4 0.64
18~20_ride 0.39 0.54 0.62 0.25 0.33 0.34
6~8_ride 8~10_ride 10~12_ride 6~8_takeoff 8~10_takeoff 10~12_takeoff 18~20_ride
1.0 0.8 0.6 0.4 0.2 0.0

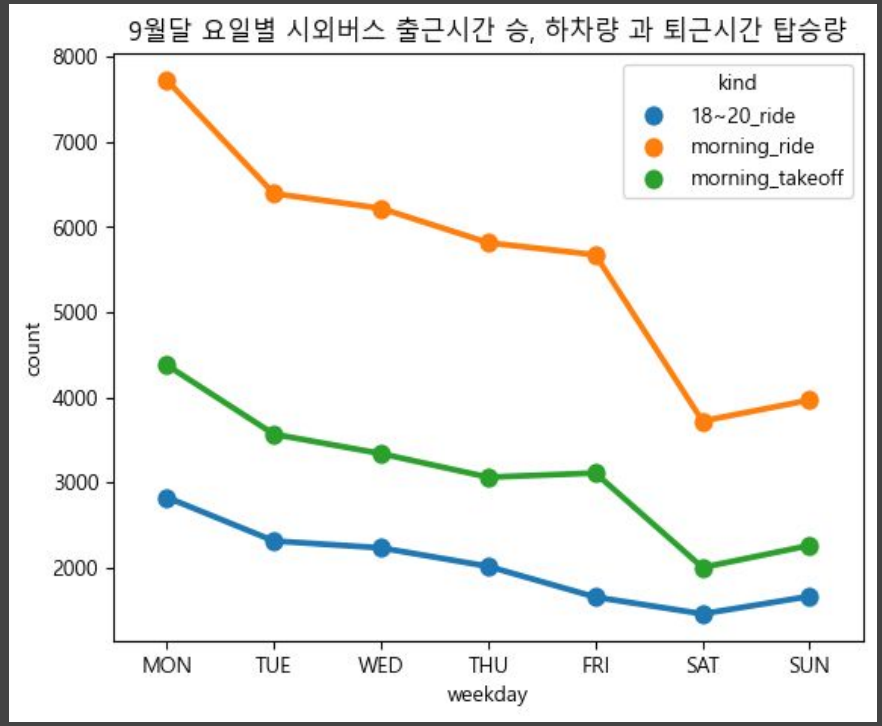
데이터 시각화


출근시간 승, 하차인원과 퇴근시간 승차인원 일별 추이
passenger_sum
25000
20000
15000
10000
5000
kind
8~10_ride
8~10_takeoff
18~20_ride
2019-09-01
2019-09-02
2019-09-03
2019-09-04
2019-09-05
2019-09-06
2019-09-07
2019-09-08
2019-09-09
2019-09-10
2019-09-11
2019-09-12
2019-09-13
2019-09-14
2019-09-15
2019-09-16
2019-09-17
2019-09-18
2019-09-19
2019-09-20
2019-09-21
2019-09-22
2019-09-23
2019-09-24
2019-09-25
2019-09-26
2019-09-27
2019-09-28
2019-09-29
2019-09-30
date

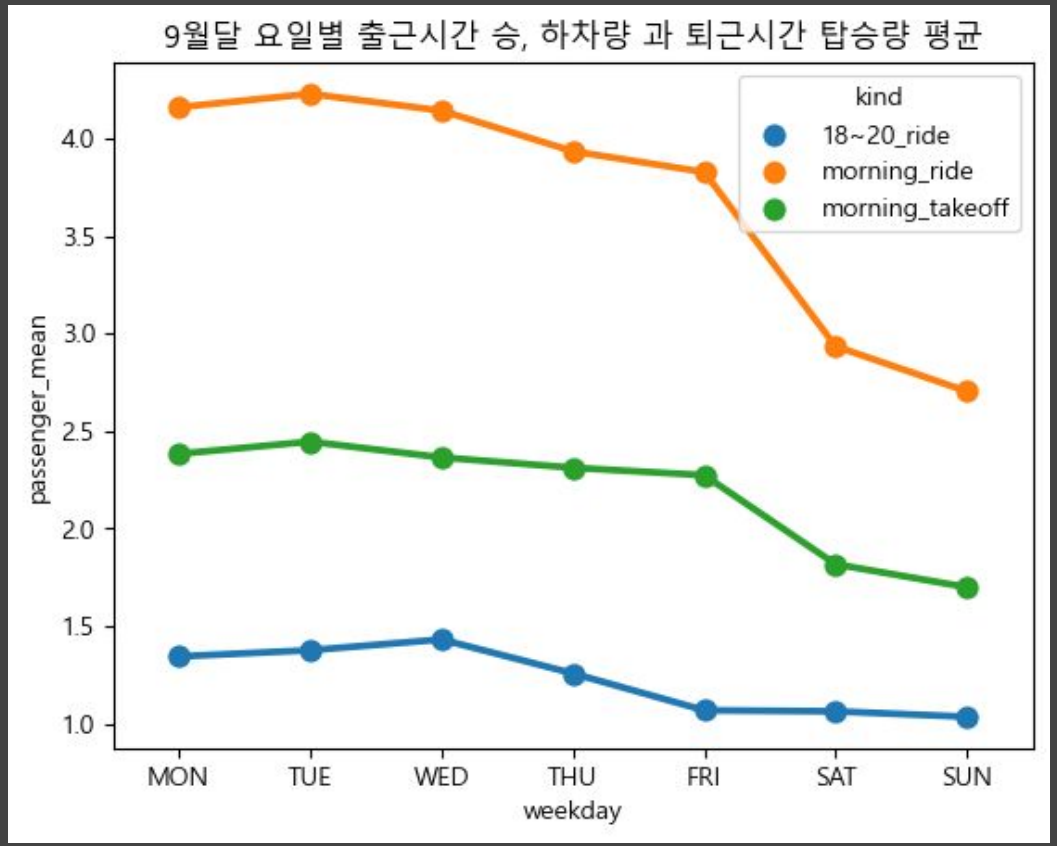

주일과 주말의 퇴근시간 탑승량 비교
passenger_sum
400000
350000
300000
250000
200000
150000
100000
50000
0
weekday
weekend
weekday

데이터 시각화


9월달 요일별 시내버스 출근시간 승, 하차량 과 퇴근시간 탑승량
kind
18~20_ride
morning_ride
morning_takeoff
count
300000
250000
200000
150000
100000
50000
MON
TUE
WED
THU
FRI
SAT
SUN
weekday


9월달 요일별 시외버스 출근시간 승, 하차량 과 퇴근시간 탑승량
kind
18~20_ride
morning_ride
morning_takeoff
count
8000
7000
6000
5000
4000
3000
2000
MON
TUE
WED
THU
FRI
SAT
SUN
weekday

데이터 시각화


9월달 요일별 출근시간 승, 하차량 과 퇴근시간 탑승량 평균
kind
18~20_ride
morning_ride
morning_takeoff
passenger_mean
4.0
3.5
3.0
2.5
2.0
1.5
1.0
MON
TUE
WED
THU
FRI
SAT
SUN
weekday

🍊 데이터 전처리

데이터 전처리

승하차 인원 전처리

```
# 2시간 간격으로 묶기
train['6~8_ride'] = train['6~7_ride'] + train['7~8_ride']
train['8~10_ride'] = train['8~9_ride'] + train['9~10_ride']
train['10~12_ride'] = train['10~11_ride'] + train['11~12_ride']

train['6~8_takeoff'] = train['6~7_takeoff'] + train['7~8_takeoff']
train['8~10_takeoff'] = train['8~9_takeoff'] + train['9~10_takeoff']
train['10~12_takeoff'] = train['10~11_takeoff'] + train['11~12_takeoff']
```

```
# test 동일 적용
test['6~8_ride'] = test['6~7_ride'] + test['7~8_ride']
test['8~10_ride'] = test['8~9_ride'] + test['9~10_ride']
test['10~12_ride'] = test['10~11_ride'] + test['11~12_ride']

test['6~8_takeoff'] = test['6~7_takeoff'] + test['7~8_takeoff']
test['8~10_takeoff'] = test['8~9_takeoff'] + test['9~10_takeoff']
test['10~12_takeoff'] = test['10~11_takeoff'] + test['11~12_takeoff']
```

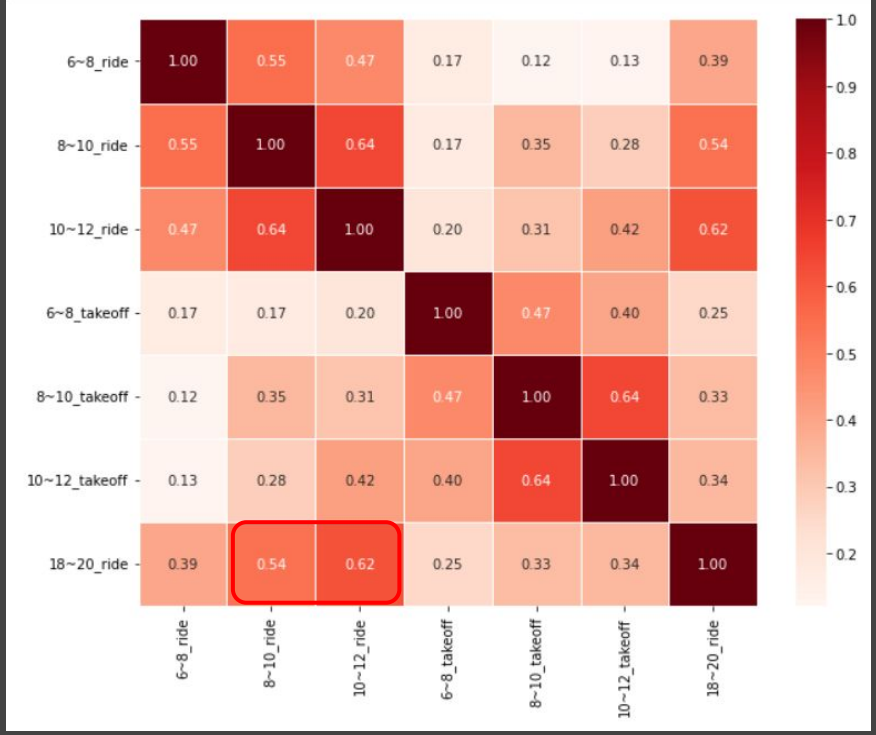
데이터 전처리

승하차 시간 상관 관계 알아보기

```
ride_df = train[['6~8_ride', '8~10_ride', '10~12_ride','6~8_takeoff',
                 '8~10_takeoff', '10~12_takeoff', '18~20_ride'
                ]]
ride_df
```

	6~8_ride	8~10_ride	10~12_ride	6~8_takeoff	8~10_takeoff	10~12_takeoff	18~20_ride
0	1.0	7.0	8.0	0.0	0.0	0.0	0.0
1	5.0	6.0	11.0	0.0	0.0	0.0	5.0
2	2.0	2.0	0.0	0.0	0.0	0.0	2.0
3	17.0	32.0	30.0	0.0	0.0	0.0	53.0
4	0.0	0.0	0.0	0.0	1.0	0.0	0.0
...	...	...	...	...	...	...	...
415418	4.0	0.0	0.0	0.0	0.0	0.0	0.0
415419	4.0	0.0	0.0	0.0	0.0	0.0	0.0
415420	0.0	0.0	0.0	1.0	0.0	0.0	0.0
415421	1.0	0.0	0.0	0.0	0.0	0.0	0.0
415422	0.0	0.0	0.0	0.0	4.0	0.0	0.0

```
corr = ride_df.corr()
plt.figure(figsize=(10,8))
sns.heatmap(corr, annot=True, fmt ='.2f', cmap='Reds', linewidth=0.5);
```



승하차 인원 전처리 추가 (ID 컬럼도 제거)

```
train = pd.read_csv('data/train_time.csv', index_col=0)
```

```
X = train.drop(['18~20_ride', 'id', 'date', 'in_out', 'station_name'], axis = 1)
y = train['18~20_ride']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

```
rf = RandomForestRegressor(random_state=42, n_estimators = 100)
neg_mse_scores = cross_val_score(rf, X, y, scoring = "neg_mean_squared_error", cv=3, n_jobs=-1)
rmse_scores = np.sqrt(-1 * neg_mse_scores)
avg_rmse = np.mean(rmse_scores)

print('Negative MSE scores: ', np.round(neg_mse_scores, 2))
print('개별 RMSE scores : ', np.round(rmse_scores, 2))
print('평균 RMSE : {0:.3f}'.format(avg_rmse))
```

```
Negative MSE scores:  [-7.86 -9.36 -7.73]
개별 RMSE scores :  [2.8  3.06 2.78]
평균 RMSE : 2.881
```

- 2.881

데이터 전처리

날짜 전처리 ¶

```
train['date'] = pd.to_datetime(train['date'])
```

```
test['date'] = pd.to_datetime(test['date'])
```

```
train['weekday'] = train['date'].dt.weekday
```

```
test['weekday'] = test['date'].dt.weekday
```

```
train = pd.get_dummies(train, columns = ['weekday'])
```

```
test = pd.get_dummies(test, columns = ['weekday'])
```

```
train.head()
```

longitude	6~7_ride	7~8_ride	...	10~11_takeoff	11~12_takeoff	18~20_ride	weekday_0	weekday_1	weekday_2	weekday_3	weekday_4	weekday_5	weekday_6
126.49373	0.0	1.0	...	0.0	0.0	0.0	0	0	0	0	0	0	1
126.48508	1.0	4.0	...	0.0	0.0	5.0	0	0	0	0	0	0	1
126.47352	1.0	1.0	...	0.0	0.0	2.0	0	0	0	0	0	0	1
126.49252	0.0	17.0	...	0.0	0.0	53.0	0	0	0	0	0	0	1
126.41260	0.0	0.0	...	0.0	0.0	0.0	0	0	0	0	0	0	1

데이터 전처리

시내/외 버스 전처리

```
train['in_out'].value_counts()
```

```
시내    408500
시외      6923
Name: in_out, dtype: int64
```

```
train['in_out'].replace({'시외': 1, '시내': 0}, inplace=True)
```

```
# test 동일 적용
test['in_out'].replace({'시외': 1, '시내': 0}, inplace=True)
```

```
train.head()
```

	id	date	bus_route_id	in_out	station_code	station_name	latitude
0	0	2019-09-01	4270000	1	344	제주썬호텔	33.48990
1	1	2019-09-01	4270000	1	357	한라병원	33.48944
2	2	2019-09-01	4270000	1	432	정존마을	33.48181
3	3	2019-09-01	4270000	0	1579	제주국제공항(600번)	33.50577
4	4	2019-09-01	4270000	0	1646	중문관광단지입구	33.25579

데이터 전처리

label encoding

```
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()

label = ['bus_route_id', 'station_code', 'station_name']

for i in label:
    le.fit(train[i])
    train[i] = le.transform(train[i])
```

```
le = LabelEncoder()

label = ['bus_route_id', 'station_code', 'station_name']

for i in label:
    le.fit(test[i])
    test[i] = le.transform(test[i])
```

```
train.head(2)
```

	id	date	bus_route_id	in_out	station_code	station_name	latitude	longitude	6~7_ride	7~8_ride	...	weekday_3	weekday_4	weekday_5	weekday_6	6
0	0	2019-09-01	0	1	321	1481	33.48990	126.49373	0.0	1.0	...	0	0	0	1	
1	1	2019-09-01	0	1	334	1822	33.48944	126.48508	1.0	4.0	...	0	0	0	1	

데이터 전처리

날짜, 승하차인원(시간별), 노선ID, 정류장ID 전처리 추가

```
train = pd.read_csv('data/base_model.csv',  index_col=0)
```

```
X = train.drop(['id', 'date', '18~20_ride','6~7_ride', '7~8_ride', '8~9_ride',
       '9~10_ride', '10~11_ride', '11~12_ride', '6~7_takeoff', '7~8_takeoff',
       '8~9_takeoff', '9~10_takeoff', '10~11_takeoff', '11~12_takeoff'], axis = 1)
y = train['18~20_ride']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

```
rf = RandomForestRegressor(random_state=42, n_estimators = 100)
neg_mse_scores = cross_val_score(rf, X, y, scoring = "neg_mean_squared_error", cv=3, n_jobs=-1)
rmse_scores = np.sqrt(-1 * neg_mse_scores)
avg_rmse = np.mean(rmse_scores)

print('Negative MSE scores: ', np.round(neg_mse_scores, 2))
print('개별 RMSE scores : ', np.round(rmse_scores, 2))
print('평균 RMSE : {0:.3f}'.format(avg_rmse))
```

```
Negative MSE scores:  [-7.32 -8.99 -7.32]
개별 RMSE scores :  [2.71 3.   2.71]
평균 RMSE : 2.803
```

- 2.803

🍊 머신러닝 성능 올리기

머신러닝 성능 올리기

map 시각화

```
import folium                              # 지도 시각화를 위한 folium 라이브러리
from folium.plugins import MarkerCluster # 지도 마커 표시를 위한 folium 플러그인
```

```
station_df = train[['latitude', 'longitude', 'station_name']].drop_duplicates(keep = 'first')
station_df
```

...

```
station_df2 = station_df.groupby(['station_name'])['latitude','longitude'].mean()
station_df2
```

...

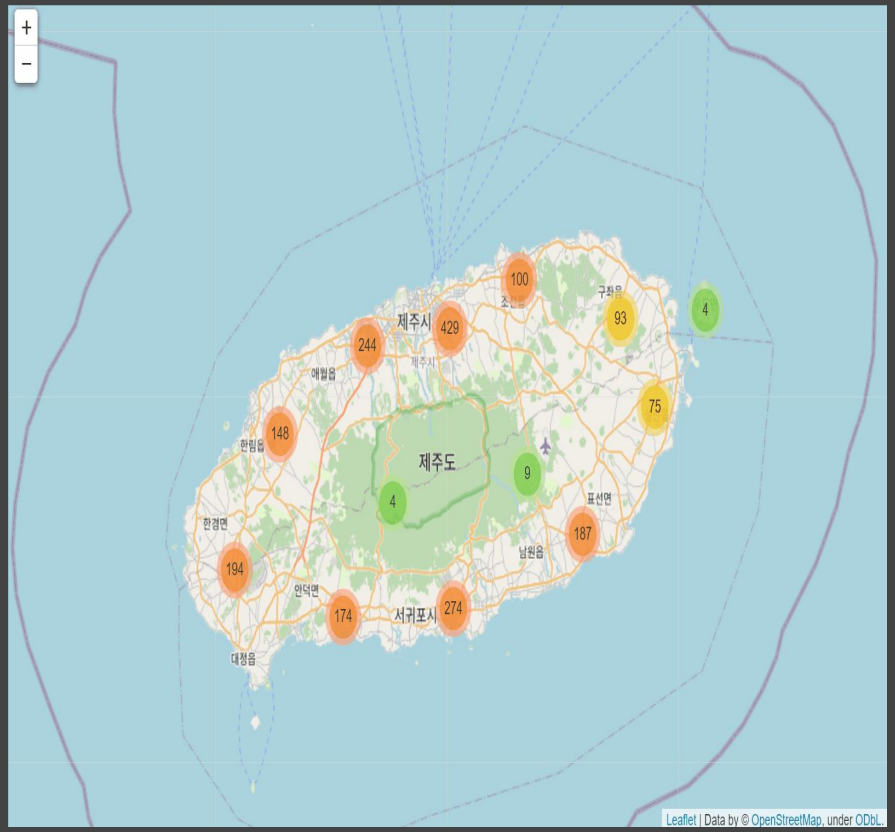
```
station_df2.to_csv('foilum.csv')
```

```
station_df2 = pd.read_csv('foilum.csv')
```

```
map_osm= folium.Map(location =[33.399835, 126.506031] ,zoom_start=9)
```

```
mc = MarkerCluster()
 # itertuples() : 이름이 있는 튜플 (인덱스, 행, 열) 에 대해 순환 반복
for row in station_df2.itertuples():
    mc.add_child(folium.Marker(location=[row.latitude, row.longitude], popup= row.station_name))
    map_osm.add_child(mc)

map_osm
```



위도, 경도 거리 좌표

```
import geopy.distance
from geopy import distance

jeju=(33.51411, 126.52969)
gosan=(33.29382, 126.16283)
seongsan=(33.38677, 126.8802)
po=(33.24616, 126.5653)

t1 = [geopy.distance.geodesic( (i,j), jeju).km for i,j in list( zip( train['latitude'],train['longitude'] )) ]
t2 = [geopy.distance.geodesic( (i,j), gosan).km for i,j in list( zip( train['latitude'],train['longitude'] )) ]
t3 = [geopy.distance.geodesic( (i,j), seongsan).km for i,j in list( zip( train['latitude'],train['longitude'] )) ]
t4 = [geopy.distance.geodesic( (i,j), po).km for i,j in list( zip( train['latitude'],train['longitude'] )) ]

train['dis_jeju']=t1
train['dis_gosan']=t2
train['dis_seongsan']=t3
train['dis_po']=t4
```

머신러닝 성능 올리기

위,경도 거리차이 정보 추가

```
train = pd.read_csv('data/train_lalo.csv',  index_col=0)
```

```
X = train.drop(['18~20_ride'], axis = 1)
y = train['18~20_ride']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

```
rf = RandomForestRegressor(random_state=42, n_estimators = 100)
neg_mse_scores = cross_val_score(rf, X, y, scoring = "neg_mean_squared_error", cv=3, n_jobs=-1)
rmse_scores = np.sqrt(-1 * neg_mse_scores)
avg_rmse = np.mean(rmse_scores)

print('Negative MSE scores: ', np.round(neg_mse_scores, 2))
print('개별 RMSE scores : ', np.round(rmse_scores, 2))
print('평균 RMSE : {0:.3f}'.format(avg_rmse))
```

```
Negative MSE scores:  [-7.1 -8.8 -7.2]
개별 RMSE scores :  [2.66 2.97 2.68]
평균 RMSE : 2.772
```

- 2.772

머신러닝 성능 올리기

요일별 18~20 탑승인원 평균, 합 ¶

```
train['date'] = pd.to_datetime(train['date'])
train['weekday'] = train['date'].dt.weekday
test['date'] = pd.to_datetime(test['date'])
test['weekday'] = test['date'].dt.weekday
```

```
train['bus_route_id_weekday'] = train['bus_route_id'].astype(str) + ',' + train['weekday'].astype(str)
test['bus_route_id_weekday'] = test['bus_route_id'].astype(str) + ',' + test['weekday'].astype(str)
```

```
f = train.groupby(['bus_route_id_weekday'])['18~20_ride'].agg([('mean_bus_weekday_ride','mean')]).reset_index()
tr = pd.merge(train, f, how='left', on='bus_route_id_weekday')
te = pd.merge(test, f, how='left', on='bus_route_id_weekday').fillna(f['mean_bus_weekday_ride'].mean())

def id_statistic(ID, col1, col2) :
# 요일, 평일
    # mean, sum
    rs_mean = train.groupby([ID])['18~20_ride'].agg([(col1, 'mean')]).reset_index()
    # 요일별 18~20 탑승인원 평균
    rs_sum = train.groupby([ID])['18~20_ride'].agg([(col2, 'sum')]).reset_index()
     # 요일별 18~20 탑승인원 합
    rs_mean_sum = pd.merge(rs_mean, rs_sum, on=ID)
    # merge
    tr = pd.merge(train, rs_mean_sum, how='left', on=ID)
    # train 데이터에 평균값 합치기
    te = pd.merge(test, rs_mean_sum, how='left', on=ID)
    # test 데이터에 탑승인원 합 합치기
    return tr, te
```

```
train, test = id_statistic('weekday', '1820_w_mean', '1820_w_sum')
```

```
train.to_csv('bus_train2.csv', index = False)
test.to_csv('bus_test2.csv', index = False)
```

머신러닝 성능 올리기

요일별 18~20시 탑승 평균, 합 정보 추가

```
train = pd.read_csv('data/train_1820.csv',  index_col=0)
```

```
X = train.drop(['18~20_ride'], axis = 1)
y = train['18~20_ride']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

```
rf = RandomForestRegressor(random_state=42, n_estimators = 100)
neg_mse_scores = cross_val_score(rf, X, y, scoring = "neg_mean_squared_error", cv=3, n_jobs=-1)
rmse_scores = np.sqrt(-1 * neg_mse_scores)
avg_rmse = np.mean(rmse_scores)

print('Negative MSE scores: ', np.round(neg_mse_scores, 2))
print('개별 RMSE scores : ', np.round(rmse_scores, 2))
print('평균 RMSE : {0:.3f}'.format(avg_rmse))
```

```
Negative MSE scores:  [-6.81 -8.7  -6.85]
개별 RMSE scores :  [2.61 2.95 2.62]
평균 RMSE : 2.726
```

- 2.726

최종예측과 성능평가

최종 예측과 성능평가

데이콘 제출

```
input_var=['bus_route_id', 'in_out', 'station_code', 'station_name', 'weekday_0', 'weekday_1', 'weekday_2', 'weekday_3', 'weekday_4',
           'weekday_5', 'weekday_6', '6~8_ride', '8~10_ride', '10~12_ride', '6~8_takeoff', '8~10_takeoff', '10~12_takeoff',
           'dis_jejusi', 'dis_seoquipo', 'dis_jeju', 'dis_gosan', 'dis_seongsan', 'dis_po', 'weekday', '1820_w_mean', '1820_w_sum']

target=['18~20_ride']
```

```
X_train=train[input_var]
random.seed(1217)
train_list=random.sample(list(range(X_train.shape[0])), int(round(X_train.shape[0]*0.01, 0)) )

X_train=train[input_var]
X_train=X_train.iloc[train_list,:]
y_train=train[target]
y_train=y_train.iloc[train_list,:]

X_test=test[input_var]

X_train.shape, y_train.shape
```

```
((4154, 26), (4154, 1))
```

```
param_grid = {
    'max_features': [2,3,5],
    'min_samples_leaf': [2,3],
    'min_samples_split': [2,4,6],
    'n_estimators': [100, 200,500]
}

rf = RandomForestRegressor(random_state=1217) # 랜덤포레스트 모델을 정의한다.

grid_search = GridSearchCV(estimator = rf, param_grid = param_grid) # GridSearchCV를 정의한다.

grid_search.fit(X_train, y_train)

grid_search.best_params_
```

```
{'max_features': 5,
 'min_samples_leaf': 2,
 'min_samples_split': 2,
 'n_estimators': 100}
```

최종 예측과 성능평가.

데이콘 제출

```
X_train=train[input_var]
y_train=train[target]

X_test=test[input_var]

X_train.shape, y_train.shape, X_test.shape
```

```
((415423, 26), (415423, 1), (228170, 26))
```

```
rf = RandomForestRegressor(max_features=5,min_samples_leaf=2,min_samples_split=2,n_estimators=500,random_state=1217)

rf.fit(X_train,y_train) #학습

test['18~20_ride'] = rf.predict(X_test) #예측값 생성 후, test['18~20_ride']에 집어 넣는다.

test[['id','18~20_ride']].to_csv("rf_test.csv",index=False)
```

	제목	제출 일시	public점수 private점수
768211	rf_test.csv edit	2022-12-08 00:15:13	2.4450196752 2.4704608123

Thank you