따릉이 데이콘

머신러닝 3조

Table of Contents.

결측치 처리 방법

Original Method (시간별 평균)

```python
ori_train = train.copy()

fill_dict = {}
idxs = ori_train[ori_train['hour_bef_windspeed'].isna()].index
att_by_hr = ori_train.groupby('hour').mean()['hour_bef_windspeed']
for idx in idxs:
    hr_mean = att_by_hr[ori_train['hour'][idx]]
    fill_dict[idx] = hr_mean
ori_train['hour_bef_windspeed'].fillna(fill_dict, inplace=True)

fill_dict = {}
idxs = ori_train[ori_train['hour_bef_ozone'].isna()].index
att_by_hr = ori_train.groupby('hour').mean()['hour_bef_ozone']
for idx in idxs:
    hr_mean = att_by_hr[ori_train['hour'][idx]]
    fill_dict[idx] = hr_mean
    if ori_train['hour'][idx] == 1:
        fill_dict[idx] = (0.033763 + 0.030492) / 2
ori_train['hour_bef_ozone'].fillna(fill_dict, inplace=True)

imputation_compare['Original'] = ori_train.loc[miss_idx, 'hour_bef_windspeed']
ori_train.isna().sum()
```

결측치 처리 방법

KNN Imputer

```
[ ] from sklearn.impute import KNNImputer

    knn_train = train.copy()

    imputer = KNNImputer(n_neighbors=20)
    knn_train = imputer.fit_transform(knn_train)

    knnImp = pd.DataFrame(knn_train)
    knnImp.columns = column_names

    imputation_compare['KNN Imputer'] = knnImp.loc[miss_idx, 'hour_bef_windspeed']
    knnImp.isna().sum()
```

```
hour                      0
hour_bef_temperature      0
hour_bef_precipitation    0
hour_bef_windspeed        0
hour_bef_humidity         0
hour_bef_visibility       0
hour_bef_ozone            0
hour_bef_pm10             0
hour_bef_pm2.5            0
count                     0
dtype: int64
```

결측치 처리 방법

MICE Imputer

https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779
Mice Explained: https://stats.stackexchange.com/questions/421545/multiple-imputation-by-chained-equations-mice-explained

+ 코드 + 텍스트

```
[ ] pip install impyute
```

```
Requirement already satisfied: impyute in c:\users\retepmil\anaconda3\lib\site-packages (0.0.8)
Requirement already satisfied: scikit-learn in c:\users\retepmil\anaconda3\lib\site-packages (from impyute) (0.24.2)
Requirement already satisfied: numpy in c:\users\retepmil\anaconda3\lib\site-packages (from impyute) (1.19.2)
Requirement already satisfied: scipy in c:\users\retepmil\anaconda3\lib\site-packages (from impyute) (1.5.2)
Requirement already satisfied: joblib>=0.11 in c:\users\retepmil\anaconda3\lib\site-packages (from scikit-learn->impyute) (0.17.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\retepmil\anaconda3\lib\site-packages (from scikit-learn->impyute) (2.1.0)
Note: you may need to restart the kernel to use updated packages.
```

```
[ ] from impyute import mice

mice_train = train.to_numpy()
mice_train = mice(mice_train)

miceImp = pd.DataFrame(mice_train)
miceImp.columns = column_names

imputation_compare['MICE Imputer'] = miceImp.loc[miss_idx, 'hour_bef_windspeed']
miceImp.isna().sum()
```

결측치 처리 방법

Iterative Imputer

가장 결과 좋게 나와서 이걸로 선정

Bayesian Ridge 사용

```
[ ]  from sklearn.experimental import enable_iterative_imputer
     from sklearn.impute import IterativeImputer

     it_train = train.copy()

     it_train = IterativeImputer(random_state=2021).fit_transform(it_train)

     itImp = pd.DataFrame(it_train)
     itImp.columns = column_names

     imputation_compare['Iterative Imputer'] = itImp.loc[miss_idx, 'hour_bef_windspeed']
     itImp.isna().sum()
```

```
hour                      0
hour_bef_temperature      0
hour_bef_precipitation    0
hour_bef_windspeed        0
hour_bef_humidity         0
hour_bef_visibility       0
hour_bef_ozone            0
hour_bef_pm10             0
hour_bef_pm2.5            0
count                     0
dtype: int64
```

이상치 처리 방법

Isolation Forest

```
# 참고: https://donghwa-kim.github.io/iforest.html
# Regression Tree 기반의 Split으로 모든 데이터 관측치를 고립시키는 방법(?)
# 비정상 데이터가 고립되려면, root node와 가까운 depth를 가짐
# 정상 데이터가 고립되려면, tree의 말단노드에 가까운 depth를 가짐
# 특정 한 개체가 isolation 되는 leaf 노드(terminal node)까지의 거리를 outlier score로 정의하며,
# 그 평균거리(depth)가 짧을 수록 outlier score는 높아짐

from sklearn.ensemble import IsolationForest
iso = IsolationForest(n_jobs=-1)

#Visilbility의 density shape 때문에 제외시킴
itImp_IF_mid = itImp.drop(['hour_bef_precipitation', 'hour_bef_visibility', 'count'], axis=1)

yhat = iso.fit_predict(itImp_IF_mid)
mask = yhat != -1
itImp_IF = itImp[mask]

itImp_IF[['hour', 'hour_bef_temperature', 'hour_bef_windspeed', 'hour_bef_humidity', 'hour_bef_visibility', 'hour_bef_ozone']].info()
#알고리즘이 Stochastic한 것 같다: 수가 계속 변함
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1225 entries, 0 to 1456
Data columns (total 6 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   hour                  1225 non-null   float64
 1   hour_bef_temperature  1225 non-null   float64
 2   hour_bef_windspeed    1225 non-null   float64
 3   hour_bef_humidity     1225 non-null   float64
 4   hour_bef_visibility   1225 non-null   float64
 5   hour_bef_ozone        1225 non-null   float64
```

이상치 처리 방법

Minimum Covariance Determinant

```python
# 참고: https://scikit-learn.org/stable/modules/generated/sklearn.covariance.EllipticEnvelope.html
# 만약 사용하고자 하는 변수가 가우시안 분포를 가지고 있다면, 이 특성을 이용해 Elliptic하게 묶에서 외곽의 데이터는
# 버리는 방법으로 Outlier 제거가 가능하다(는 알고리즘)

from sklearn.covariance import EllipticEnvelope
ee = EllipticEnvelope(contamination=0.1)

#가우시안 분포 아닌 변수들 제거
itImp_MCD_mid = itImp.drop(['hour', 'hour_bef_precipitation', 'hour_bef_visibility', 'count'], axis=1)

yhat = ee.fit_predict(itImp_MCD_mid)
mask = yhat != -1
itImp_MCD = itImp[mask]

itImp_MCD[['hour', 'hour_bef_temperature', 'hour_bef_windspeed', 'hour_bef_humidity', 'hour_bef_visibility', 'hour_bef_ozone']].info()
# 얘는 일정함
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1311 entries, 0 to 1456
Data columns (total 6 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   hour                  1311 non-null   float64
 1   hour_bef_temperature  1311 non-null   float64
 2   hour_bef_windspeed    1311 non-null   float64
 3   hour_bef_humidity     1311 non-null   float64
 4   hour_bef_visibility   1311 non-null   float64
 5   hour_bef_ozone        1311 non-null   float64
dtypes: float64(6)
memory usage: 71.7 KB
```

이상치 처리 방법

Original Method (IQR)

가장 결과 좋게 나와서 이걸로 선정

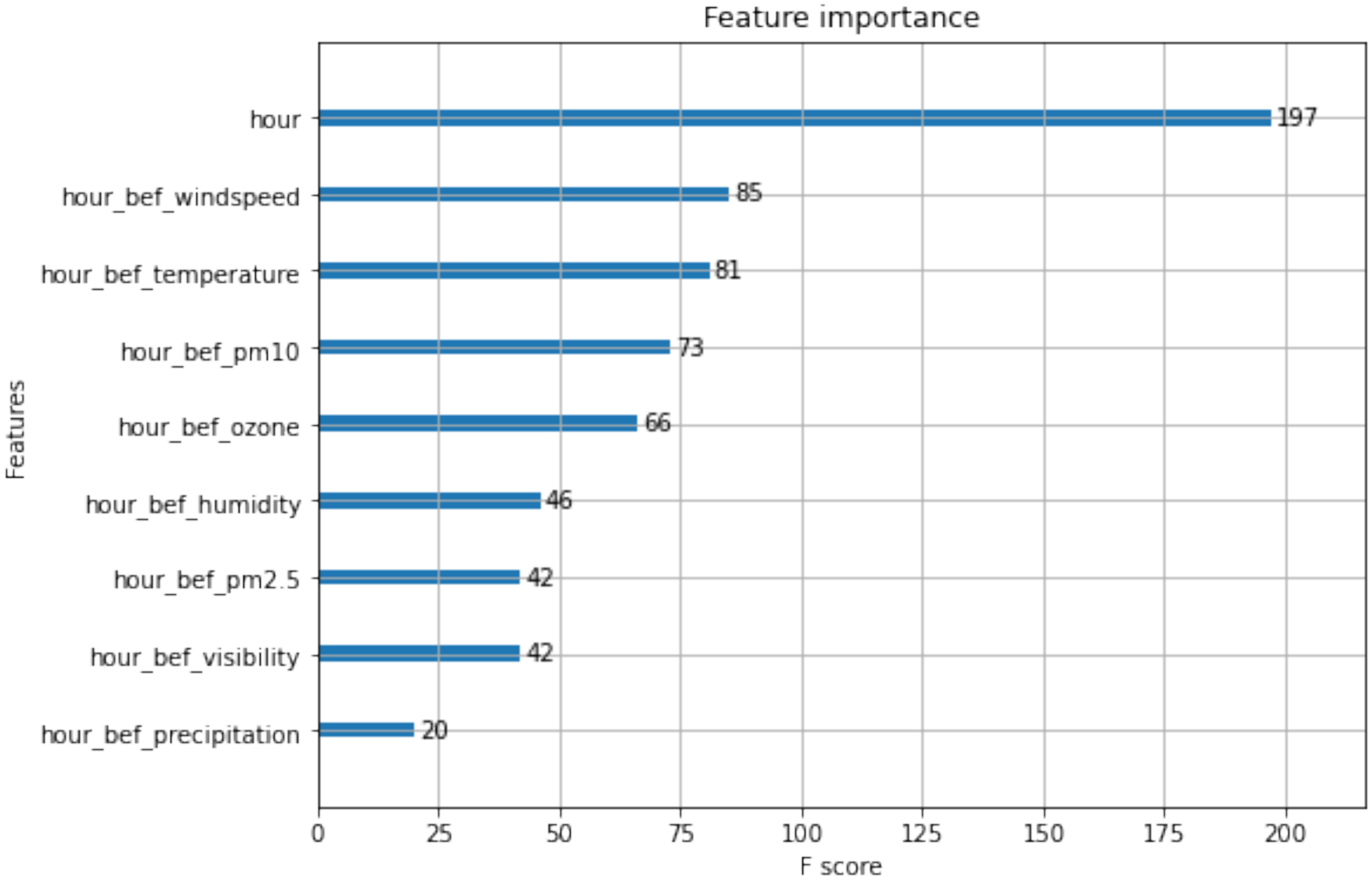
```
col_name=['hour', 'hour_bef_temperature', 'hour_bef_windspeed', 'hour_bef_humidity', 'hour_bef_visibility', 'hour_bef_ozone']

itImp_mid = itImp.copy()

for iIt in col_name:
    Q1=itImp_mid[iIt].quantile(0.25)
    Q3=itImp_mid[iIt].quantile(0.75)
    IQR=Q3-Q1
    train_delout=itImp_mid[(itImp_mid[iIt]<(Q1 - 1.5*IQR)) | (itImp_mid[iIt]>(Q3+1.5*IQR))]
    itImp_mid=itImp_mid.drop(train_delout.index, axis=0)
itImp_mid[col_name].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1433 entries, 0 to 1456
Data columns (total 6 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   hour                  1433 non-null   float64
 1   hour_bef_temperature  1433 non-null   float64
 2   hour_bef_windspeed    1433 non-null   float64
 3   hour_bef_humidity     1433 non-null   float64
 4   hour_bef_visibility   1433 non-null   float64
 5   hour_bef_ozone        1433 non-null   float64
dtypes: float64(6)
memory usage: 78.4 KB
```

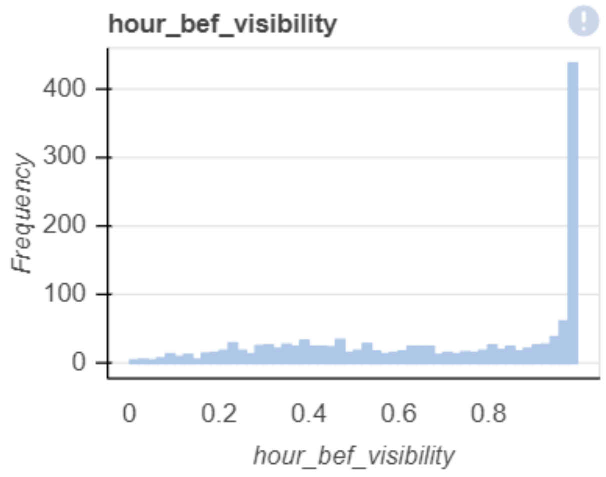
전체 변수 중요도


Feature importance
hour
hour_bef_windspeed
hour_bef_temperature
hour_bef_pm10
hour_bef_ozone
hour_bef_humidity
hour_bef_pm2.5
hour_bef_visibility
hour_bef_precipitation
197
85
81
73
66
46
42
42
20
Features
0
25
50
75
100
125
150
175
200
F score

변수 선택 – 제외한 변수

hour_bef_visibility
numerical
Show Details

Approximate Distinct Count	782	Mean	0.6903
Approximate Unique (%)	53.6%	Minimum	0
Missing	0	Maximum	1
Missing (%)	0.0%	Zeros	1
Infinite	0	Zeros (%)	0.1%
Infinite (%)	0.0%	Negatives	0
Memory Size	22.8 KB	Negatives (%)	0.0%

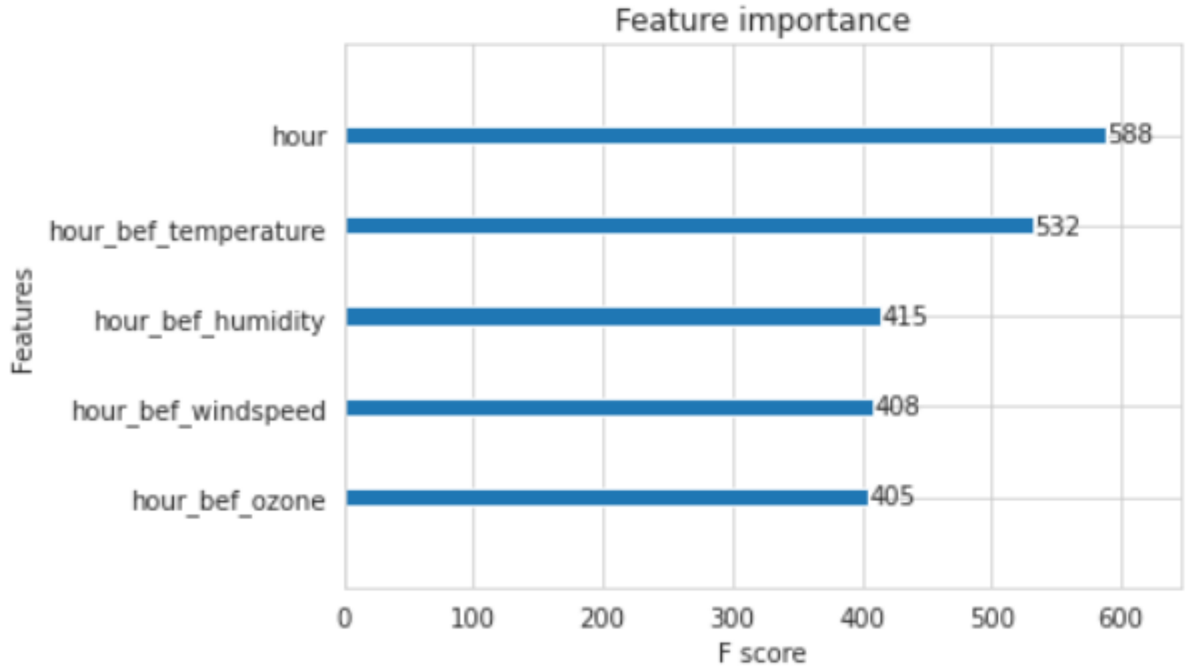


상관계수 높은 상위 5개 변수 중요도

```
!pip install graphviz
!conda install graphviz
import xgboost as xgb

xgb.plot_importance(my_model)
```

```
Requirement already satisfied: graphviz in /usr/local/lib/python3.7/dist-packages (0.10.1)
/bin/bash: conda: command not found
<matplotlib.axes._subplots.AxesSubplot at 0x7f06eda26110>
```



모델링 방법 part1

Modeling

1. Linear Regression
2. Ridge
3. Lasso
4. ElasticNet
5. Decision Tree
6. Randomforest
7. XGBoostRegressor
8. LightGBMRegressor

Stacking Regressor

```
[ ] X=train.drop('count',axis=1)
    y=train['count']
    X_train, X_test, y_train, y_test= train_test_split(X,y,test_size=0.3, random_state=2021)
```

```
[ ] lr_reg_alone=LinearRegression()
    lr_reg_alone.fit(X_train, y_train)
    y_lr_pred=lr_reg_alone.predict(X_test)
    rmse= np.sqrt(mean_squared_error(y_test, y_lr_pred))
    print(rmse)
```

```
59.43058667824046
```

```
[ ] from sklearn.ensemble import StackingRegressor

    lr_final=LinearRegression()

    estimators= [('knn_reg',KNeighborsRegressor(n_neighbors=10)),('rf_reg',RandomForestRegressor(max_depth=3, n_jobs=-1, n_estimators=300, random_state=2021)),
    ('aba_reg',AdaBoostRegressor(n_estimators=300,learning_rate=0.3,random_state=2021,loss='square')),
    ('dt_reg',DecisionTreeRegressor(max_depth=5,random_state=2021))]

    reg= StackingRegressor(estimators=estimators, final_estimator=lr_final)
    reg.fit(X_train, y_train)
```

Stacking Regressor

```
[ ]  y_pred=reg.predict(X_test)
     rmse = np.sqrt(mean_squared_error(y_test, y_pred))
     print(rmse)
```

```
44.07484855369241
```

```
[ ]  lr_reg_alone=LinearRegression()
     lr_reg_alone.fit(X_train, y_train)
     y_lr_pred=lr_reg_alone.predict(X_test)
     rmse= np.sqrt(mean_squared_error(y_test, y_lr_pred))
     print(rmse)
```

```
59.43058667824046
```

모델링 방법 part1

```
from sklearn.tree import DecisionTreeRegressor
param_grid = {'min_impurity_decrease' : np.arange(0.0001, 0.001, 0.0001),
              'max_depth' : range(5, 20, 1),
              'min_samples_split' : range(2, 100, 10),
              'min_samples_leaf': [1, 3, 5]}
grid_search = GridSearchCV(DecisionTreeRegressor(random_state = 42), param_grid, cv = 3, scoring = 'neg_mean_squared_error',
                           verbose = 0, n_jobs = -1)
grid_search.fit(train_poly, y_train)
```

```
[ ] grid_search.best_params_
```

```
{'max_depth': 6,
 'min_impurity_decrease': 0.0001,
 'min_samples_leaf': 5,
 'min_samples_split': 52}
```

```
[ ] print('Best rmse of decision tree: {}'.format(np.sqrt(-grid_search.best_score_)))
```

```
Best rmse of decision tree: 48.45553277579874
```

Decisiontree가 가장 rmse가 작게 나온 모델

모델링 방법 part2

Modeling

```
[ ] col = ['hour', 'hour_bef_temperature', 'hour_bef_windspeed', 'hour_bef_humidity', 'hour_bef_ozone']

    '''X = itImp_IF[col]
    y = itImp_IF[['count']]
    X_train_IF, X_val_IF, y_train_IF, y_val_IF = train_test_split(X, y, test_size=0.33, random_state=2021)

    X = itImp_MCD[col]
    y = itImp_MCD[['count']]
    X_train_MCD, X_val_MCD, y_train_MCD, y_val_MCD = train_test_split(X, y, test_size=0.33, random_state=2021)'''

    X = itImp_mid[col]
    y = itImp_mid[['count']]
    X_train_ori, X_val_ori, y_train_ori, y_val_ori = train_test_split(X, y, test_size=0.33, random_state=2021)
```

Isolation Forest

[ ] 숨겨진 셀 1개

MCD

[ ] 숨겨진 셀 1개

IQR

모델링 방법 결론

IQR

```python
from sklearn.ensemble import RandomForestRegressor

ori_RF = RandomForestRegressor(max_depth=10, random_state=2021)
ori_RF.fit(X_train_ori, y_train_ori)
y_preds = ori_RF.predict(X_val_ori)

rmse_ori = np.sqrt(mean_squared_error(y_val_ori, y_preds))
print(rmse_ori)
```

```
37.64945833929329
```

```python
from sklearn.ensemble import ExtraTreesRegressor

best_reg = ExtraTreesRegressor(n_estimators=100, max_depth=10, random_state=2021)
best_reg.fit(X_train_ori, y_train_ori)
y_preds = best_reg.predict(X_val_ori)

rmse = np.sqrt(mean_squared_error(y_val_ori, y_preds))
print(rmse)
```

```
37.47143608933198
```

결론: IQR 방식이 제일 정확함 X max_depth 늘리니까 MCD IF 방식의 rmse가 엄청 줄음

Q / A

Thank You ☺