INDUSTRIAL AI RESEARCH CENTER

어프렌티스 프로젝트

기말 프로젝트

충북대학교 산업인공지능학과 사수진

Dept. of Industrial AI

차례

1. 프로젝트 개요

INDUSTRIAL AI RESEARCH CENTER

- 프로젝트명: 서울시 공유 자전거 수요 예측
- 발표자 이름: 사수진

1. 프로젝트 개요

INDUSTRIAL AI RESEARCH CENTER

프로젝트 개요 및 데이터 소개

- 프로젝트 목표 및 문제 정의
- 자전거 대여 데이터와 날씨 정보를 활용하여 시간, 강우량, 습도 등과 같은 날씨 변수를 기반으로 특정 요일과 시간대에서의 공유 자전거 수요를 정확하게 예측하여 해당 지역에 적절한 자전거 공급을 유지하고 부족분을 최소화하는 것이 목표입니다.

- 사용한 데이터셋 소개
- 캐글의 “Seoul Bike Sharing Demand Prediction” 데이터 세트에는 날씨 정보(온도, 습도, 풍속, 가시성, 이슬점, 일사량, 강설량, 강우량), 시간당 대여 자전거 수 및 날짜 정보가 포함되어 있습니다.
- 데이터 다운로드 : https://www.kaggle.com/datasets/saurabhshahane/seoul-bike-sharing-demand-prediction?select=SeoulBikeData.csv

1. 프로젝트 개요

INDUSTRIAL AI RESEARCH CENTER

데이터 세부 정보

	Date	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	Seasons	Holiday	Functioning Day
0	01/12/2017	254	0	-5.2	37	2.2	2000	-17.6	0.0	0.0	0.0	Winter	No Holiday	Yes
1	01/12/2017	204	1	-5.5	38	0.8	2000	-17.6	0.0	0.0	0.0	Winter	No Holiday	Yes
2	01/12/2017	173	2	-6.0	39	1.0	2000	-17.7	0.0	0.0	0.0	Winter	No Holiday	Yes
3	01/12/2017	107	3	-6.2	40	0.9	2000	-17.6	0.0	0.0	0.0	Winter	No Holiday	Yes
4	01/12/2017	78	4	-6.0	36	2.3	2000	-18.6	0.0	0.0	0.0	Winter	No Holiday	Yes

- 8760 row의 데이터가 있는 14가지 열로 구성이 되어 있습니다.
- 4개의 범주형 열과 10개의 숫자형 열이 있습니다.
- '날짜', '계절', '휴일', ' 기능하는 날' 열은 $object$ 데이터 유형입니다.
- '임대된 자전거 수', '시간', '습도(%)' 및 '가시성(m)' 열은 $int64$ 수치 데이터 유형입니다.
- '온도(℃)', '풍속(m/s)', '이슬점 온도(℃)', '태양 복사$(M\ J/m2)$', '비낙하(mm)', '설우(cm)' 열은 $float64$ 수치 데이터 유형입니다
- 어떤 열에도 null 값은 없습니다

2. 프로젝트 단계

INDUSTRIAL AI RESEARCH CENTER

Step 1: 훈련 세트와 테스트 세트 만들기

```
#X 및 y 변수 또는 독립 및 종속 변수 정의
X = label_df.drop('Rented Bike Count',axis=1)
y = np.sqrt(label_df['Rented Bike Count'])
#데이터 분할
X_train, X_test, y_train, y_test = train_test_split( X,y , test_size = 0.2, random_state = 42)
```

- train_test_split함수를 이용하여 데이터셋 분리를 하였으며 test_size=0.20 데이터의 20%를 테스트에 사용하고 나머지 80%를 훈련에 사용하도록 지정하였습니다.
- random_state=42 재현성을 무작위 시드로 설정하였습니다.

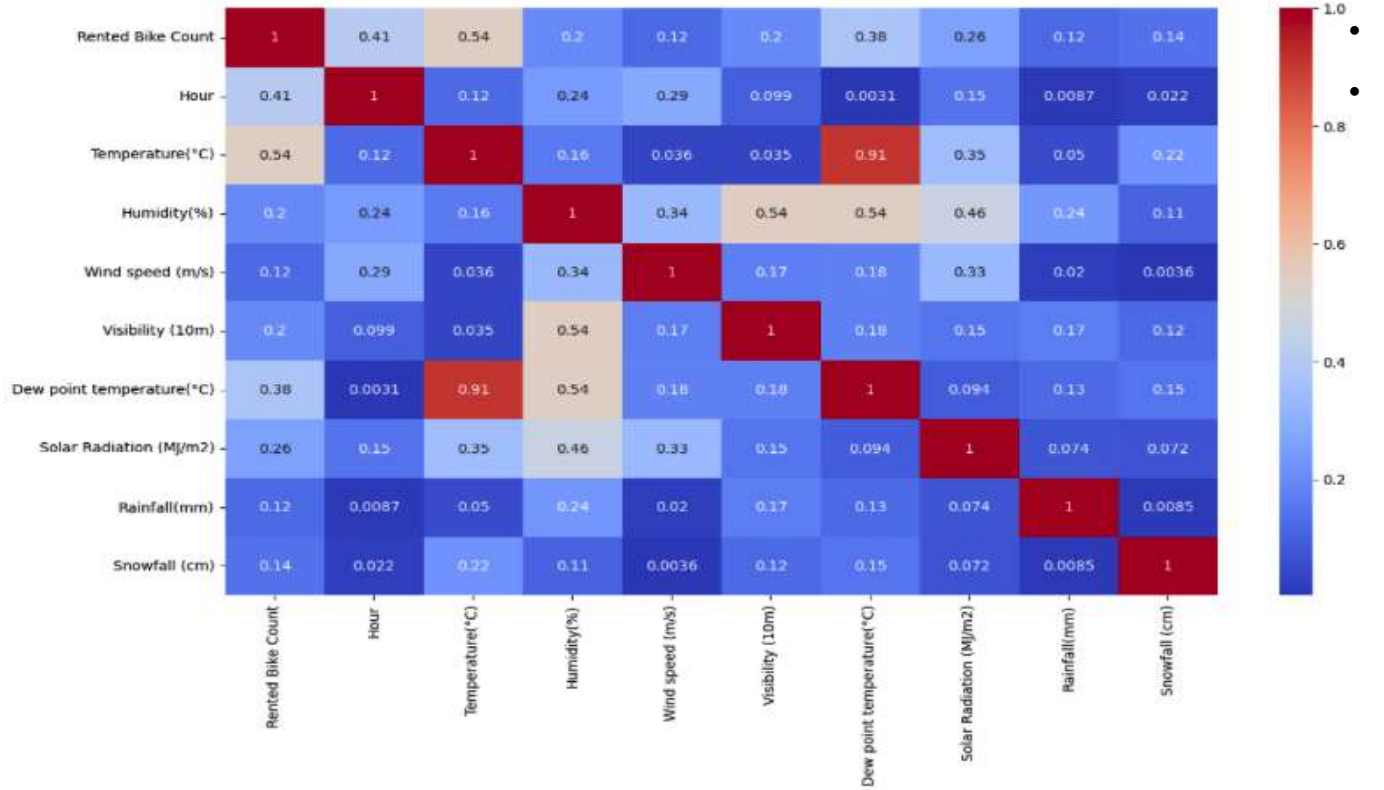
```
print(len(X_train))
print(len(y_train))
print(len(X_test))
print(len(y_test))
```

```
6772
6772
1693
1693
```

2. 프로젝트 단계

INDUSTRIAL AI RESEARCH CENTER

Step 2: 데이터 탐색 및 시각화

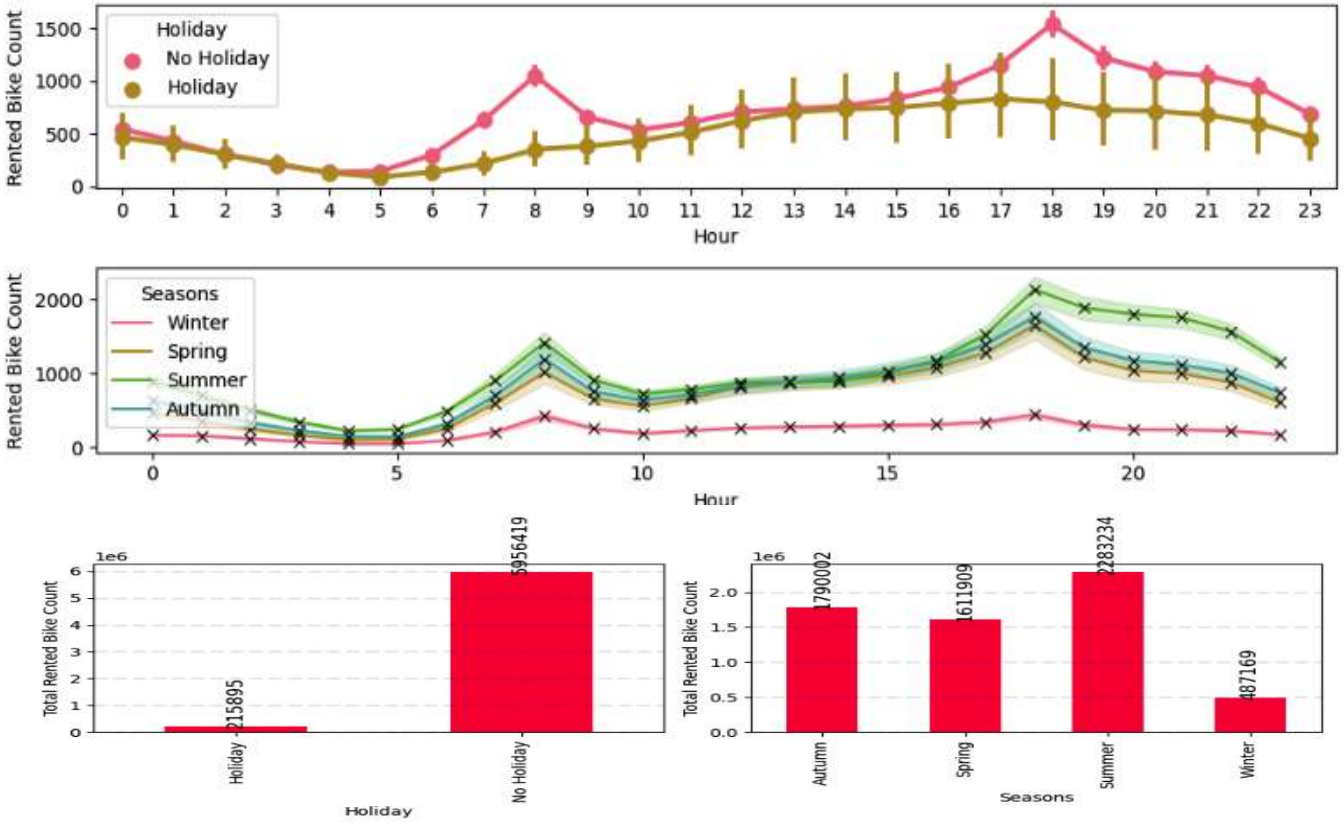


- **피처 간 상관 관계 시각화 (히트맵)**
- 온도와 이슬점 온도는 거의 0.91로 상관관계가 있기 때문에 다중 공선성 문제가 발생하지 안하도록 이슬점 온도("Dew point temperature(°C)" 컬럼을 삭제하였습니다.

2. 프로젝트 단계

INDUSTRIAL AI RESEARCH CENTER

Step 2: 데이터 탐색 및 시각화

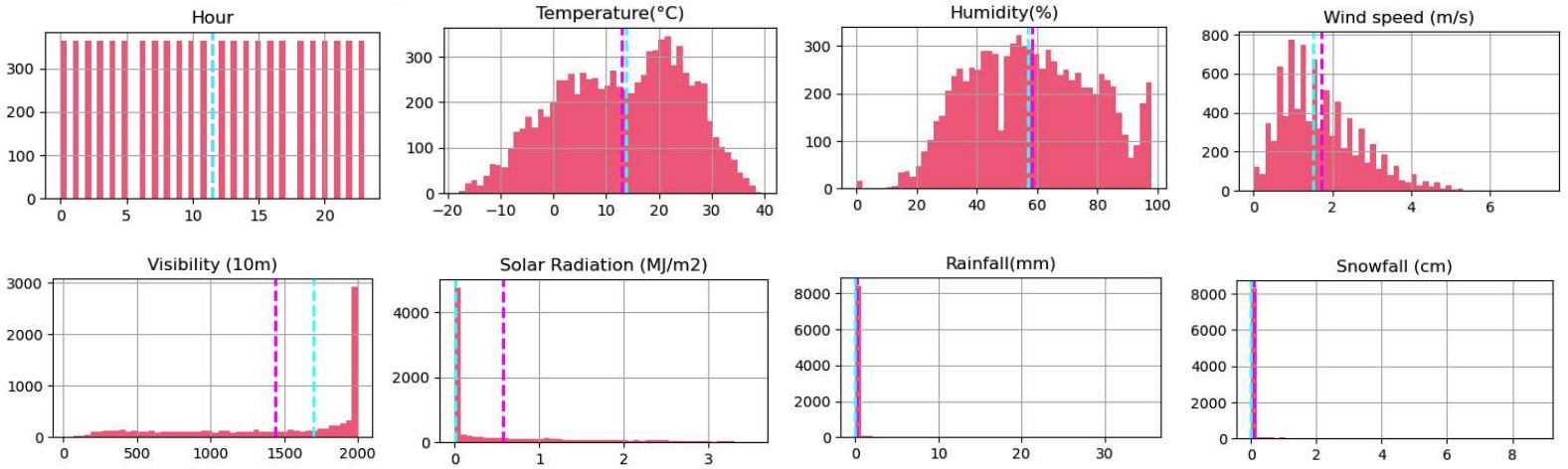


- 공휴일에는 비공휴일에 비해 대여 자전거의 수가 적습니다.
- 겨울에는 대여 자전거 수가 매우 적습니다.

2. 프로젝트 단계

INDUSTRIAL AI RESEARCH CENTER

Step 2: 데이터 탐색 및 시각화



- 위의 수치 속성 히스토그램을 보면 몇 가지 특징은 다음과 같습니다

1.일부 히스토그램은 꼬리가 무겁습니다.

2.모든 속성은 스케일이 다릅니다.

- 두 결과 모두 속성을 더 정규화를 해야 함을 확인하였습니다.

2. 프로젝트 단계

INDUSTRIAL AI RESEARCH CENTER

Step 3: 데이터 전처리

- 기능일 데이터 확인시 대여 자전거는 기능하는 날에만 제공되므로 해당 열 제거

```
# 기능일별 그룹화 및 총 대여 자전거 수 계산
Bike_data.groupby('Functioning Day').sum()['Rented Bike Count'].sort_values(ascending = Fals
```

	Functioning Day	Rented Bike Count
0	Yes	6172314
1	No	0

```
#비기능일 데이터 제거
Bike_data = Bike_data.drop(Bike_data[Bike_data['Functioning Day'] == 'No'].index)
```

```
# "Functioning Day" 컬럼 삭제
Bike_data =Bike_data.drop(['Functioning Day'], axis = 1)

print("자전거 대여일만 있는 필터링된 데이터 프레임 :",Bike_data.shape,"\n")
```

```
자전거 대여일만 있는 필터링된 데이터 프레임 : (8465, 13)
```

2. 프로젝트 단계

INDUSTRIAL AI RESEARCH CENTER

Step 3: 데이터 전처리

- 결측치는 존재하지 않았으며 범주형 데이터 라벨 인코딩 처리하였습니다.

```
#라벨 인코딩 적용
label_df= Bike_data.apply(LabelEncoder().fit_transform)
label_df
```

	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	Seasons	Holiday	Day	Month	Year
0	252	0	111	28	22	1779	0	0	0	3	1	11	0	0
1	202	1	108	29	8	1779	0	0	0	3	1	11	0	0
2	171	2	103	30	10	1779	0	0	0	3	1	11	0	0
3	105	3	101	31	9	1779	0	0	0	3	1	11	0	0
4	76	4	103	27	23	1779	0	0	0	3	1	11	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8755	990	19	205	25	26	1673	0	0	0	0	1	29	10	1
8756	754	20	197	28	23	1779	0	0	0	0	1	29	10	1
8757	685	21	189	30	3	1747	0	0	0	0	1	29	10	1
8758	703	22	184	32	10	1639	0	0	0	0	1	29	10	1
8759	579	23	182	34	13	1688	0	0	0	0	1	29	10	1

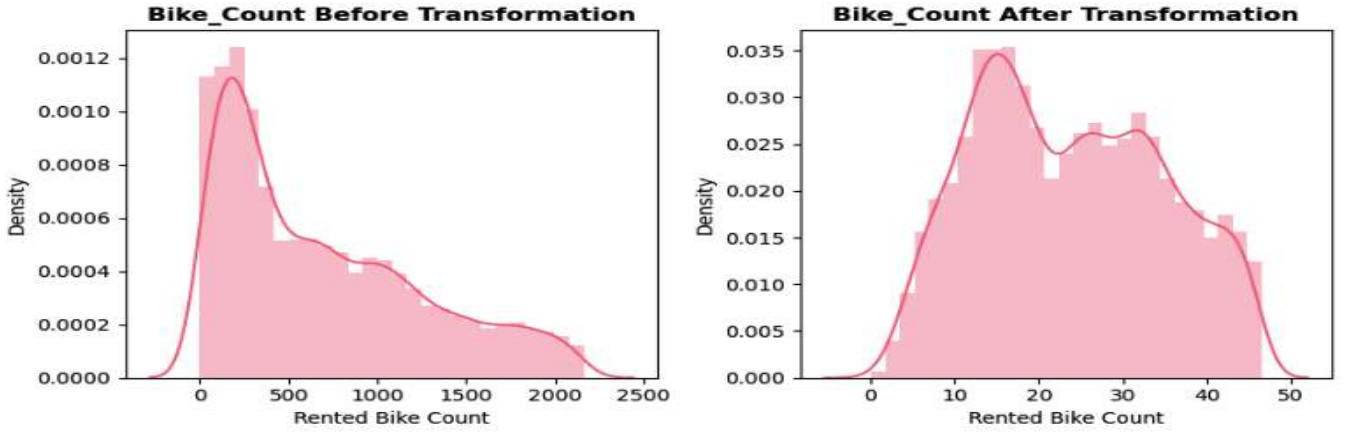
2. 프로젝트 단계

INDUSTRIAL AI RESEARCH CENTER

Step 4: 데이터 스케일링

- 정규화(Min-Max Scaling, Normalization) 스케일링 방법을 사용하였습니다.
- scaler = MinMaxScaler()
- 제곱근 변환 스케일링 이후의 데이터 분포 시각화

```
plt.figure(figsize=(9,4))
plot = plt.subplot(1,2,1)
sns.distplot(label_df['Rented Bike Count']).set_title('Bike_Count Before Transformation',wei
plot = plt.subplot(1,2,2)
sns.distplot(np.sqrt(label_df['Rented Bike Count'])).set_title('Bike_Count After Transformat
plt.tight_layout()
```



2. 프로젝트 단계

Step 5: 파이프라인 구성

- Pipeline 클래스 생성 및 Pipeline 컨스트럭트로 튜플 목록을 시퀀스 스텝 정의

```
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy="median")

num_pipeline = Pipeline([
    ("impute", SimpleImputer(strategy="median")),
    ("minmax", MinMaxScaler()),
])
```

- make_pipeline 함수에 결측치 발생시 중앙값으로 대체 및 정규화로 변환 정의

```
from sklearn.pipeline import make_pipeline

num_pipeline = make_pipeline(SimpleImputer(strategy="median"), MinMaxScaler())

from sklearn import set_config

set_config(display='diagram')

num_pipeline
```

2. 프로젝트 단계

INDUSTRIAL AI RESEARCH CENTER

Step 5: 파이프라인 구성

- ColumTransformer 클래스 생성하여 열 이름 목록 정의와 범주형 열 파이프라인 생성

```
from sklearn.compose import ColumnTransformer

num_attribs = ["Rented Bike Count", "Hour", "Temperature(°C)", "Humidity(%)",
               "Wind speed (m/s)", "Visibility (10m)", "Dew point temperature(°C)",
               "Solar Radiation (MJ/m2)", "Rainfall(mm)", "Snowfall (cm)"]
cat_attribs = ["Seasons", "Holiday"]

cat_pipeline = make_pipeline(
    SimpleImputer(strategy="most_frequent"),
    OneHotEncoder(handle_unknown="ignore"))

preprocessing = ColumnTransformer([
    ("num", num_pipeline, num_attribs),
    ("cat", cat_pipeline, cat_attribs),
])
```

- make_column_selector 함수로 숫자형, 범주형을 열 선택 후 make_column_transformer 열에 대한 변환을 정의

```
from sklearn.compose import make_column_selector, make_column_transformer

preprocessing = make_column_transformer(
    (num_pipeline, make_column_selector(dtype_include=np.number)),
    (cat_pipeline, make_column_selector(dtype_include=object)),
)

bike_prepared = preprocessing.fit_transform(data)
```

2. 프로젝트 단계

INDUSTRIAL AI RESEARCH CENTER

Step 6: 모델 선택 및 훈련

- LinearRegression, DecisionTreeRegressor, RandomForestRegressor
- 3가지 모델 선택하였으며 훈련을 위하여 모델 분석 함수 생성

```
# 모델 분석 함수 추가
def analyse_model(model, X_train, X_test, y_train, y_test):

  # Fitting the model
  model.fit(X_train,y_train)
  y_pred = model.predict(X_test)
  a,p = y_test**2,y_pred**2

  # Calculating Evaluation Matrix
  mse = mean_squared_error(a,p)
  rmse = np.sqrt(mse)
  r2 = r2_score(a,p)
  try:
    importance = model.feature_importances_
    feature = features
  except:
    importance = np.abs(model.coef_)
    feature = independent_variables
  indices = np.argsort(importance)
  indices = indices[::-1]

  # Printing Evaluation Matrix
  print("MSE :" , mse)
  print("RMSE :" ,rmse)
  print("MAE :" ,mean_absolute_error(a,p))
  print("Train R2 :", r2_score(y_train**2,model.predict(X_train)**2))
  print("Test R2 :" ,r2)
  print("Adjusted R2 : ", 1-(1-r2)*((len(X_test)-1)/(len(X_test)-X_test.shape[1]-1)))

  return model
```

2. 프로젝트 단계

Step 6: 모델 선택 및 훈련

- LinearRegression

```
# Linear Regression Model
lr = LinearRegression()

analyse_model(lr, X_train, X_test, y_train, y_test)
```

- DecisionTreeRegressor

```
# Decision Tree Regressor
regressor = DecisionTreeRegressor(random_state=42)

analyse_model(regressor, X_train, X_test, y_train, y_test)
```

- RandomForestRegressor

```
# Random Forest Regressor
regressor = RandomForestRegressor(random_state=42)

analyse_model(regressor, X_train, X_test, y_train, y_test)
```

2. 프로젝트 단계

INDUSTRIAL AI RESEARCH CENTER

Step 6: 모델 선택 및 훈련

- LinearRegression

```
MSE : 114516.28182418182
RMSE : 338.4025440568995
MAE : 243.02495571545168
Train R2 : 0.6100784578188265
Test R2 : 0.6337243464837449
Adjusted R2 :  0.6308883825196523
```

- DecisionTreeRegressor

```
MSE : 65647.49143532192
RMSE : 256.2176641750563
MAE : 155.05197873597166
Train R2 : 1.0
Test R2 : 0.7900291779985313
Adjusted R2 :  0.7884034360771381
```

- RandomForestRegressor

```
MSE : 38456.002608657145
RMSE : 196.1020209193601
MAE : 118.35738363692326
Train R2 : 0.9838576633595375
Test R2 : 0.8770000452098652
Adjusted R2 :  0.8760476929690839
```

- 모델 성능 평가 및 결과 해석

- 모델 성능이 중간 정도로 보입니다.
- Train R2와 Test R2 간의 차이가 적은 편입니다.
- 평균 제곱 오차 (MSE), 평균 제곱근 오차 (RMSE), 평균 절대 오차 (MAE)가 큰 편입니다.

- Train R2가 1.0으로 완벽한 모델로 보입니다. 그러나 Test R2는 상대적으로 낮은 편입니다.
- 모델이 훈련 데이터에 너무 맞춰져서 테스트 데이터에서 성능이 떨어질 수 있음을 시사합니다.
- 평가 지표에서 좋은 성능을 보이지만, 과적합 가능성이 있습니다.

- 모델 성능이 상대적으로 높아 보입니다.
- Train R2와 Test R2 간의 차이가 작고, 둘 다 높은 값을 가집니다.
- 평가 지표에서 우수한 성능을 보이며, 모델이 훈련 및 테스트 데이터에서 일반적으로 잘 되었음을 시사합니다.

2. 프로젝트 단계

INDUSTRIAL AI RESEARCH CENTER

Step 7: 모델 세부 튜닝

- 그리드 탐색 GridSearchCV 클래스를 이용한 최적 하이퍼파라미터값 조합 얻기

```
# 트리 수와 같은 하이퍼파라미터의 값 범위 제공:
n_estimators = [50,100,150]

# 트리의 최대 깊이:
max_depth = [6,8,10]

# 노드를 분할하는 데 필요한 최소 샘플 수:
min_samples_split = [50,100,150]

# 각 리프 노드에 필요한 최소 샘플 수:
min_samples_leaf = [40,50]

# 학습률:
eta = [0.05,0.08,0.1]
```

```
# Decision Tree Regressor
regressor = DecisionTreeRegressor(random_state=1)

# HYperparameter Grid
grid = {'max_depth' : max_depth,
        'min_samples_split' : min_samples_split,
        'min_samples_leaf' : min_samples_leaf}

# GridSearch to find the best parameters
dt = GridSearchCV(regressor, param_grid = grid, scoring = 'neg_mean_squared_error', cv=5)
dt.fit(X_train, y_train)

# Analysing the model with best set of parametes
analyse_model(dt.best_estimator_, X_train, X_test, y_train, y_test)
```

```
▾                          DecisionTreeRegressor
DecisionTreeRegressor(max_depth=10, min_samples_leaf=40, min_samples_split=50,
                      random_state=1)
```

2. 프로젝트 단계

INDUSTRIAL AI RESEARCH CENTER

Step 7: 모델 세부 튜닝

- 그리드 탐색 GridSearchCV 클래스를 이용한 최적 하이퍼파라미터값 조합 얻기

```
# 트리 수와 같은 하이퍼파라미터의 값 범위 제공:
n_estimators = [50,100,150]

# 트리의 최대 깊이:
max_depth = [6,8,10]

# 노드를 분할하는 데 필요한 최소 샘플 수:
min_samples_split = [50,100,150]

# 각 리프 노드에 필요한 최소 샘플 수:
min_samples_leaf = [40,50]

# 학습률:
eta = [0.05,0.08,0.1]
```

```
# Random Forest Regressor
regressor = RandomForestRegressor(random_state=42)

# HYperparameter Grid
grid = {'n_estimators' : n_estimators,
        'max_depth' : max_depth,
        'min_samples_split' : min_samples_split,
        'min_samples_leaf' : min_samples_leaf}

# GridSearch to find the best parameters
rf = GridSearchCV(regressor, param_grid = grid, scoring = 'neg_mean_squared_error', cv=5)
rf.fit(X_train, y_train)

# Analysing the model with best set of parametes
analyse_model(rf.best_estimator_, X_train, X_test, y_train, y_test)
```

```
▾ RandomForestRegressor
RandomForestRegressor(max_depth=10, min_samples_leaf=40, min_samples_split=50,
                      n_estimators=50, random_state=42)
```

2. 프로젝트 단계

INDUSTRIAL AI RESEARCH CENTER

Step 8: 결과와 결론

- DecisionTreeRegressor

```
MSE : 65647.49143532192
RMSE : 256.2176641750563
MAE : 155.05197873597166
Train R2 : 1.0
Test R2 : 0.7900291779985313
Adjusted R2 :  0.7884034360771381
```

- RandomForestRegressor

```
MSE : 38456.002608657145
RMSE : 196.1020209193601
MAE : 118.35738363692326
Train R2 : 0.9838576633595375
Test R2 : 0.8770000452098652
Adjusted R2 :  0.8760476929690839
```

튜닝후 결과

DecisionTreeRegressor

```
MSE : 56972.53496920022
RMSE : 238.6892016183393
MAE : 159.3716771829107
Train R2 : 0.838256062844053
Test R2 : 0.8177756721934104
Adjusted R2 :  0.8163647631633415
```

RandomForestRegressor

```
MSE : 59343.17073132673
RMSE : 243.60453758361467
MAE : 162.55325173186003
Train R2 : 0.8289167753125422
Test R2 : 0.8101932904640157
Adjusted R2 :  0.8087236732966734
```

- DecisionTreeRegressor의 경우 튜닝후 Train R2와 Test R2간의 차이가 작고 과적합의 가능성이 낮고, 더 균형이 잡힌 모델로 보입니다.
- RandomForestRegressor의 경우 튜닝후 튜닝전 결과보다 전체적으로 결과 수치가 낮게 나온것으로 보이며
- 종합적으로 3가지 모델중에 **RandomForestRegressor** 모델 성능이 상대적으로 높아 보이며 평가 지표에서 가장 우수한 성능을 보이며 대여 자전거의 수요를 예측하는데 도움이 될것으로 생각이 됩니다.

Thank You!