자전거 수요 예측
스프린트 미션 14

분석 보고서

2팀고명진

목 차 Contents

Ⅰ 분석 배경

1. 미션 소개
- 가. 공유자전거 대여 수요 예측 5
- 나. 데이터 소개 5

2. 분석 방법
- 가. 분석 방법 소개 6

Ⅱ 분석 방법

1. 탐색적 데이터 분석
- 가. 데이터 기본 정보 확인 8
- 나. 데이터 통계 확인 8
- 다. 범주형 데이터 확인 9
- 라. 연속형 데이터 확인 9
- 마. 날짜와 시간대에 따른 대여량 확인 10

2. 데이터 전처리
- 가. 데이터 중복값 확인 11
- 나. 데이터 이상치 제거 11
- 다. 데이터 이상치 대체 12

3. 피쳐 엔지니어링
- 가. 상관관계 시각화 13
- 나. 범주형 변수 인코딩 14
- 다. 연속형(수치형) 데이터 피쳐 스케일링 14
- 라. 주성분 분석 15
- 마. 피처 선택 15

4. 예측 모델 선택
- 가. 다중 선형 회귀 분석 16
- 나. XGBoost 회귀 분석 17
- 다. 랜덤포레스트 회귀 분석 19

목 차 Contents

Ⅲ 모델 성능 향상

1. 하이퍼 파라미터 튜닝

가. GridSearchCV 22

2. 제출용 데이터셋 적용

I

분석 배경

1. 미션 소개
2. 분석 방법

1. 미션 소개

1.1 공유자전거 대여 수요 예측

스프린트 미션 14는 공유자전거 대여 패턴을 분석하여 자전거 배치 및 운영 전략을 최적화하고, 대여 수요 예측을 분석하는 미션입니다. 해당 데이터는 워싱턴 DC의 Capital Bikeshare 프로그램에서 자전거 대여 수요를 예측하기 위해 사용된 패턴과 날씨 데이터를 결합하여 제작되었습니다.

1.2 데이터 소개

해당 데이터는 2년의 기간동안 시간단위 데이터가 포함되어 있습니다. 제공되는 훈련 데이터를 통해 테스트 데이터에 존재하는 날짜에 맞게 자전거 대여량을 예측합니다. **훈련 데이터는 월별로 1일~19일** 까지의 데이터가 존재하며, **테스트 데이터에는 월별로 20일~말일** 까지의 데이터가 존재합니다.

데이터 컬럼 설명

컬럼명	데이터 타입	데이터 설명
datetime	datetime	자전거 대여 기록의 날짜 및 시간. 예시: 2011-01-01 00:00:00
season	int	계절 (1: 봄, 2: 여름, 3: 가을, 4: 겨울)
holiday	int	공휴일 여부 (0: 평일, 1: 공휴일)
workingday	int	근무일 여부 (0: 주말/공휴일, 1: 근무일)
weather	int	날씨 상황 (1: 맑음, 2: 구름낌/안개, 3: 약간의 비/눈, 4: 폭우/폭설)
temp	float	실측 온도 (섭씨)
atemp	float	체감 온도 (섭씨)
humidity	int	습도 (%)
windspeed	float	풍속 (m/s)
casual	int	등록되지 않은 사용자의 대여 수
registered	int	등록된 사용자의 대여 수
count	int	총 대여 수 (종속 변수)

※ casual, registered, count 데이터는 test 데이터에 미포함
※ 예측을 해야되는 종속변수는 count로 설정

2. 분석 방법

2.1 분석 방법 소개

제공되는 데이터의 각 피쳐들을 통해 자전거 대여량의 수요를 예측하는 문제로 회귀 분석 방법을 사용합니다. 회귀(Regression) 분석은 정답이 있는 데이터의 변수들로 학습하여 변수들과 목표값과의 관계를 찾고, 정답이 없는 데이터의 변수들을 통해 예측값을 찾는 분석 방법입니다.

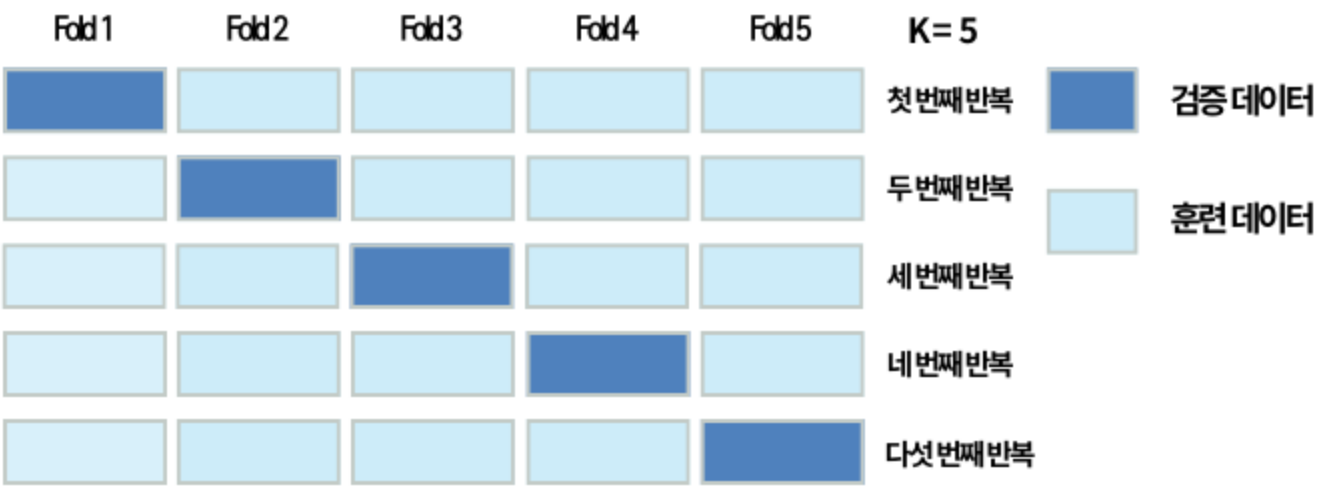
회귀 모델은 독립 변수인 X와 종속 변수인 Y의 관계를 수식으로 나타낸 것으로 대표적인 회귀 모델은 아래와 같습니다.

회귀 모델 종류

구분	세부 내용
단순 선형 회귀	X가 하나 일 때, y의 관계를 표현한 모델 (y=aX+b)
다중 선형 회귀	X가 여러 개일 때, y의 관계를 표현한 모델 (y=a1X1+a2X2+⋯+anXn+b)
비선형 회귀	관계식이 직선이 아닌 곡선인 회귀 모델
로지스틱 회귀	결과값이 0 또는 1로 결정되는 분류 모델 (해당 모델은 회귀분석이 아닌 분류 모델)

적용이 가능한 회귀 분석들을 사용하여, 최적의 결과를 도출하는 모델을 선정합니다. 선정된 모델의 성능을 더욱 향상시키기 위해 **모델 교차 검증**, **하이퍼 파라미터 튜닝**을 진행하여 더욱 정확한 예측성능을 보유한 모델을 제작하도록 합니다.

K-fold 모델 교차 검증



<그림 1-1> K-Fold 모델 교차검증 예시

Ⅱ

분석 방법

1. 탐색적 데이터 분석
2. 데이터 전처리
3. 피쳐 엔지니어링
4. 예측 모델 선택

1. 탐색적 데이터 분석

1.1 데이터 기본 정보 확인

제공받은 훈련 데이터셋에는 아래 <그림 2-1>과 같은 형태의 데이터 프레임으로 구성되어 있습니다. 대여량 수요 예측을 진행하는 테스트 데이터셋은 아래 <그림 2-2>와 같이 분석에 불필요한 casual, registered 컬럼과, 종속변수인 count 컬럼이 제외 되었습니다.

앞서 설명드린 내용처럼, 훈련 데이터의 날짜 범위는 매월 1일 부터 19일 까지의 데이터가 존재하며, 테스트 데이터에는 매월 20일 부터 말일 까지의 데이터가 존재합니다.

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0000	3	13	16
10885	2012-12-19 23:00:00	4	0	1	1	13.12	16.665	66	8.9981	4	84	88

<그림 2-1> train 데이터 기본 프레임

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed
0	2011-01-20 00:00:00	1	0	1	1	10.66	11.365	56	26.0027
6492	2012-12-31 23:00:00	1	0	1	1	10.66	13.635	65	8.9981

<그림 2-2> test 데이터 기본 프레임

1.2 데이터 통계 확인

제공받은 데이터의 기술통계량을 확인하여 데이터 타입과 결측치 수를 확인하는 과정을 진행합니다.

훈련 데이터 기술 통계량

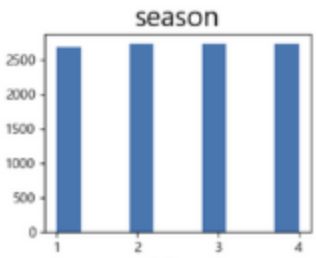
컬럼명	데이터 타입	null값의 수	컬럼명	데이터 타입	null 값의 수
datetime	object	0개	atemp	float64	0개
season	int64	0개	humidity	int64	0개
holiday	int64	0개	windspeed	float64	0개
workingday	int64	0개	casual	int64	0개
weather	int64	0개	resistered	int64	0개
temp	float64	0개	count	int64	0개

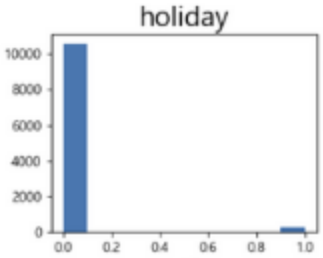
전체 훈련 데이터는 10,886개의 데이터가 존재하며, 결측치가 존재하는 컬럼은 없습니다. datetime 컬럼은 object 타입으로 제공되었으므로, 분석 시에는 datetime 형태로 변환하여 시계열 분포를 확인합니다.

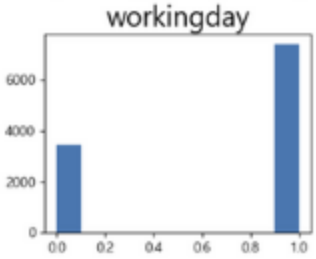
1.3 범주형 데이터 확인

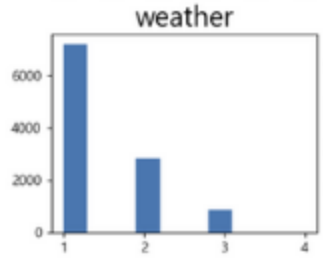
훈련 데이터와, 테스트 데이터의 범주형 데이터의 빈도수를 시각화한 결과는 다음과 같습니다.

훈련 데이터 범주형 변수 분포 시각화

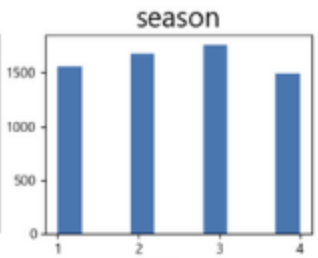


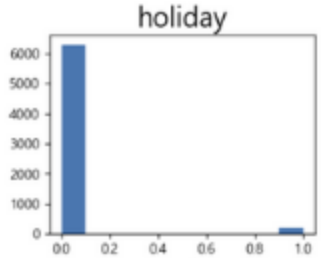


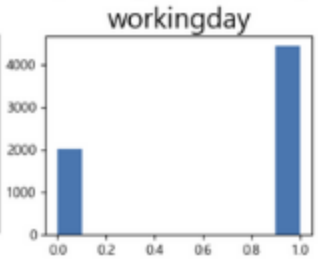


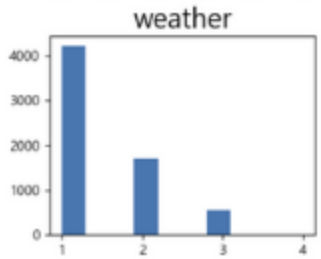


테스트 데이터 범주형 변수 분포 시각화







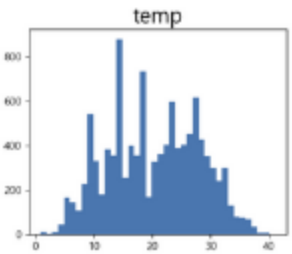


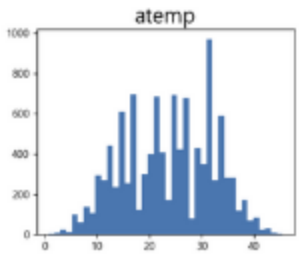
- season : 전체적으로 데이터가 매우 고르게 분포됨
- holiday : 1보다 0 값에 대부분의 데이터가 분포됨, 즉 공휴일이 아닌 날에 대여가 더 많이 발생함
- workingday : 쉬는 날 보다 일하는 날에 약 두 배 정도 더 많이 대여가 발생함
- weather : 날씨가 좋은 날에 대여량이 많으며, 날씨가 점점 안좋아질수록 대여량이 감소함

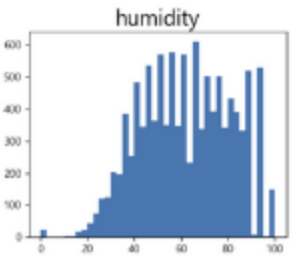
1.4 연속형 데이터 확인

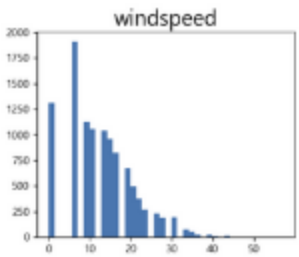
훈련 데이터와, 테스트 데이터에 존재하는 연속형 데이터를 확인한 결과 다음과 같습니다.

훈련 데이터 연속형 변수 분포 시각화

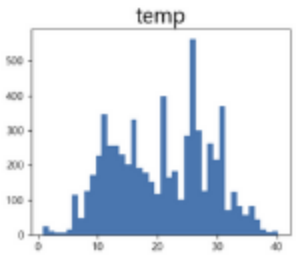


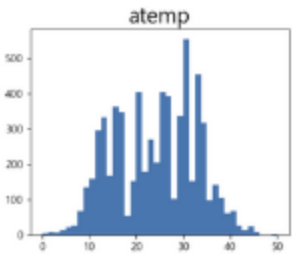


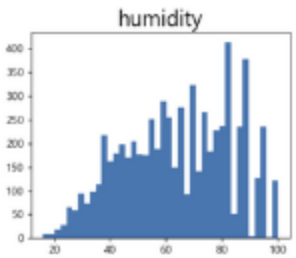


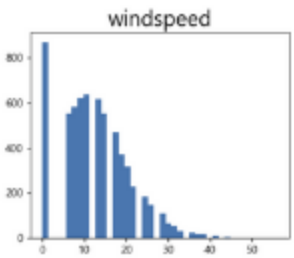


테스트 데이터 연속형 변수 분포 시각화







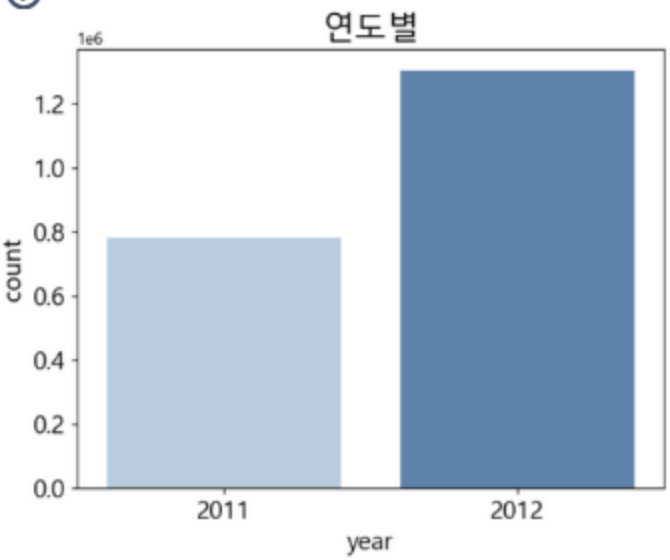


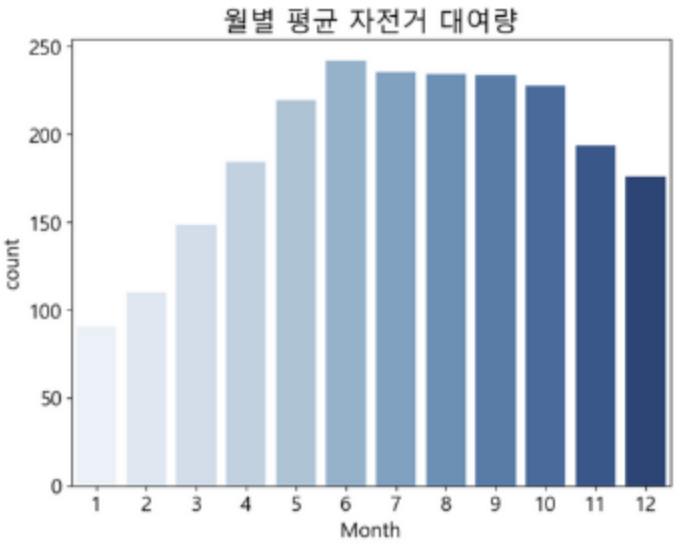
- temp : 온도가 너무 낮거나 높은 날을 제외하고는 데이터가 고르게 잘 분포됨
- atemp : 체감온도는 temp(온도) 데이터와 매우 비슷한 형상을 보여줌
- humidity : 습도가 너무 낮은 경우를 제외하고는 대부분의 구간에서 정규분포 형태를 보인다.
- windspeed : 바람이 강하게 불수록 대여량이 감소하는 추세를 보여줌

1.5 날짜와 시간대에 따른 대여량 확인

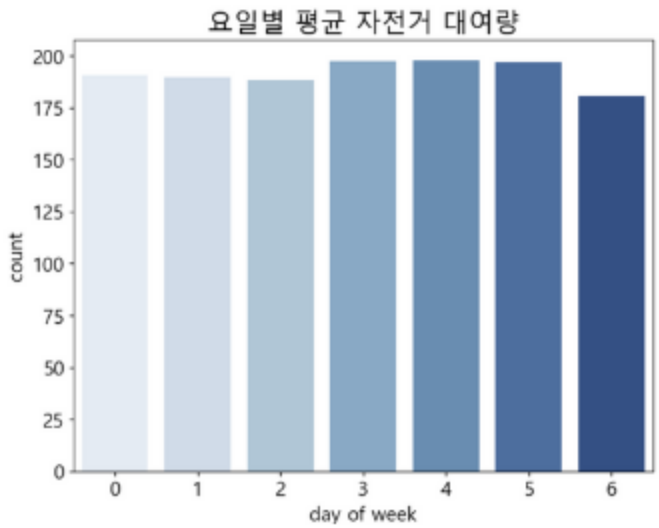
날짜와 시간대에 따른 자전거 평균 대여량을 확인하여 월별, 요일별, 시간대별로 자전거의 평균 대여량이 어떻게 변화되는지 파악한다. 시계열 변수로 인하여 대여량에 큰 차이가 발생한다면, 변수를 여러 컬럼으로 변환하여 모델 예측에 필요한 독립변수로 활용하도록 한다.

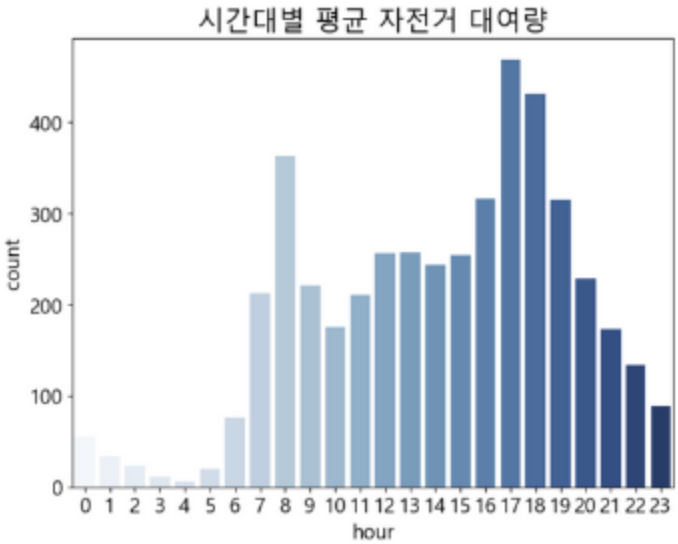
시계열 데이터 시각화





- 2011년보다 2012년에 더 많은 대여량이 존재함
  - 공유 자전거 시스템이 2012년에 더 많이 활성화 되었음을 알 수 있다.
- 1월부터 점차 대여량이 증가하고, 10월부터 감소세를 보임
  - 날씨가 추운 겨울에서 봄으로 갈수록 대여량이 증가하며, 가을부터 점차 감소함





- 모든 요일에 평균 대여량이 비슷함
  - 요일과 대여량에는 큰 상관이 없을 것으로 예상됨
- 밤에서 새벽사이 대여량이 매우 낮으며, 출퇴근 시간대에 급격하게 증가하고 이후 감소하는 추세
  - 출퇴근시에 이용하는 사용자들이 다수 존재하며, 특정 시간대에 쏠림 현상이 발생 할 것으로 예상됨

2. 데이터 전처리

2.1 데이터 중복값 확인

데이터 확인 결과 해당 데이터에는 중복값이 존재하지 않음을 확인

```
train_df.duplicated().sum()
```

```
0
```

<그림 23> 중복 데이터 확인

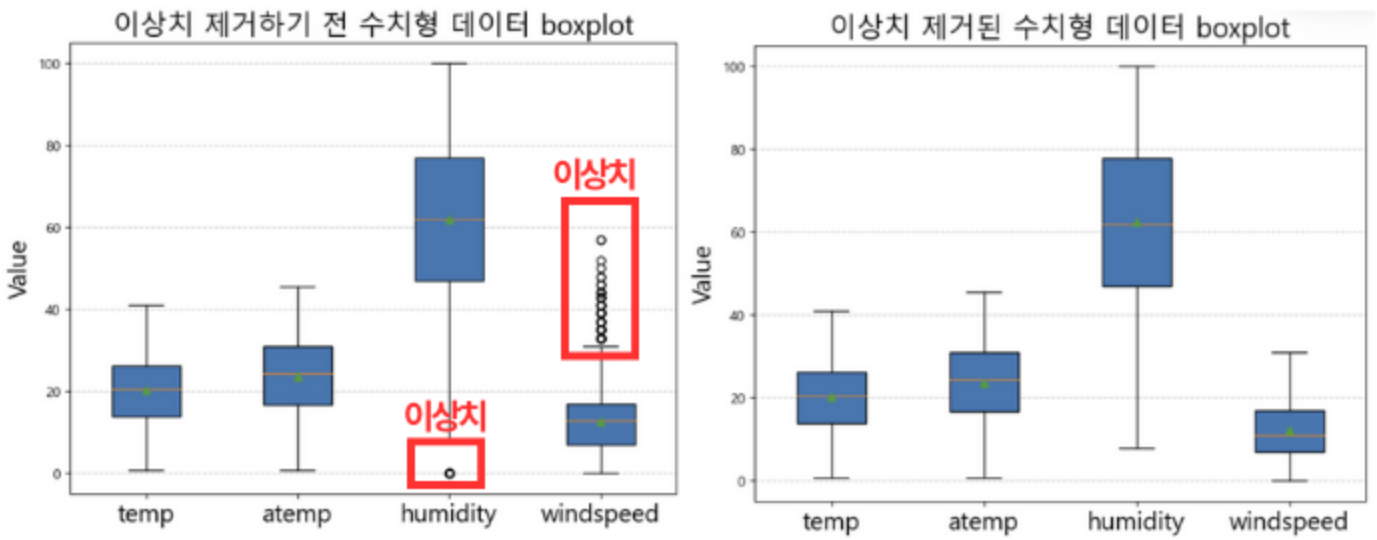
2.2 데이터 이상치 제거

예측 모델의 정확도 향상을 위해 train 데이터에 존재하는 연속형 변수들의 이상치를 제거한다. 이상치 제거 방법으로는 사분위수를 기준으로 1.5 * IQR 미만이거나 초과되는 데이터를 제거한다.

이상치 확인 대상 컬럼

컬럼명	설명	예시	컬럼명	데이터 타입	예시
temp	실제 온도(섭씨)	9.84	humidity	습도(%)	75
atemp	체감 온도(섭씨)	15.495	windspeed	풍속(m/s)	2.1

이상치 시각화 확인



이상치 제거 기준 정립

- 실제 온도와 체감 온도의 데이터는 분포된 범위가 넓지 않아서 이상치가 거의 존재하지 않음
- 습도 데이터는 0부터 100까지의 값을 가지는 정수형 데이터이며, 대부분의 데이터는 40~80% 사이에 분포하고 있다. 습도가 낮을수록 건조한 날씨라고 판단할 수 있으나, **매우 낮은 습도는 오히려 예측 모델 성능을 저하**할 수 있는 요인이 될 수 있으므로 습도 데이터의 이상치를 제거한다.

- 풍속 데이터의 값은 10~20m/s 사이에 분포하며, 이상치의 값들이 다수 존재한다. 풍력의 계급을 13개의 구간으로 나눠놓은 보퍼트 풍력 계급을 참조하는 경우 가장 강한 단계인 **12번째 싹쓸바람은 풍속이 32.7m/s 이상인 바람**을 의미한다. 이상치로 표현되는 데이터는 싹쓸바람의 데이터로 분류되며, 관측하기 힘들 정도로 강력한 바람으로 판단되므로 **해당 구간의 데이터를 제거하여 모델을 훈련시킨다.**

이상치 제거 후 데이터 개수

- 이상치 제거 전 데이터 개수 : 10,886개
- 이상치 제거 후 데이터 개수 : 10,638개
- 이상치로 판단되어 제거된 데이터의 개수 : 248개

2.2 데이터 이상치 대체

이상치 제거 후 windspeed(풍속)이 0인 데이터는 1,313개로 전체 데이터에서 약 13%를 차지하고 있다. 풍속 0 다음의 최소값은 6.0032m/s으로 0m/s 데이터는 풍속이 낮은 데이터를 전부 0으로 변경되거나, 정확한 측정이 이뤄지지 않았다고 볼 수 있다.

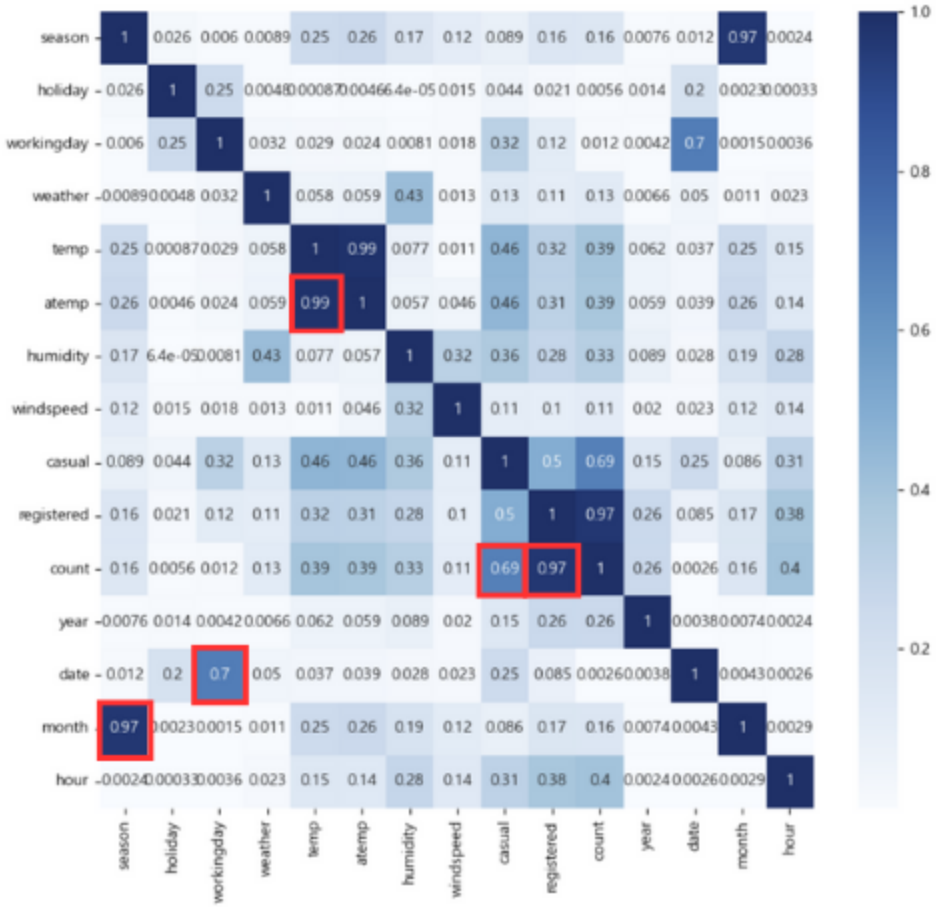
풍속이 0m/s인 지점에는 0 다음의 최소값인 6.0032m/s를 입력하여 0m/s인 지점을 삭제하지 않고 대체하도록 한다.

3. 피처 엔지니어링

3.1 상관관계 시각화

히트맵 시각화를 통해 변수들의 상관 관계를 파악하여 상관성이 강한 데이터 제거 또는 주성분 분석을 통해 상관성이 높은 변수를 하나의 성분으로 변환하도록 한다.

상관관계 시각화



※ 상관계수 상위 5개 변수

- atemp, temp : 0.99
- month, season : 0.97
- count, registered : 0.97
- date, workingday : 0.7
- count, casual : 0.69

상관관계 시각화 결과

- atemp, temp : 상관계수 0.99로 매우 높은 상관성을 보인다. 체감 온도는 실제 온도가 낮으면 더 낮게 측정되므로 **두 변수는 거의 동일한 변수로 판단할 수 있다.**
- month, season : 상관계수 0.97로 매우 높은 상관성을 보인다. season(계절)이 동일한 데이터는 month(월) 데이터도 동일한 값을 가지므로, **두 변수는 거의 동일한 변수로 판단할 수 있다.**
- count, registered : count의 값은 registered 변수와 casual 변수의 값이 더해져 생성되었다. 해당 예측모델에서는 count는 종속변수로 설정되며, registered, casual 변수는 테스트 데이터셋에 존재하지 않으므로 **독립 변수를 설정할 때 제거**하도록 한다.
- date, workingday : 요일과 공휴일 여부를 나타내는 변수들로 각각 범주형 데이터에 속한다.
- count, casual : registered와 마찬가지로 count의 값을 구성하는 데이터, **독립 변수를 설정할 때 registered 변수와 같이 제거**하도록 한다.

3.2 범주형 변수 인코딩

날씨 정보가 포함된 **weather 컬럼은 One-Hot Encoding을 진행**하여 모델의 독립변수로 설정한다.

반면, year, month, date, hour 컬럼은 시계열 주기성을 나타내는 변수로 회귀 예측 모델에 독립변수로 적용하기 위해 인코딩을 진행하지 않는다.

상관관계 시각화 결과 season 컬럼은 month 컬럼과 강한 상관성이 존재한다. 따라서 season 컬럼 또한 인코딩을 진행하지 않으며, 독립 변수를 설정할 때 season 컬럼을 제거하도록 한다.

weather 컬럼 one-hot encoding

humidity	windspeed	casual	registered	count	year	date	month	hour	weather_2	weather_3	weather_4
78	7.0015	15	33	48	2012	4	5	1	0	0	0
76	8.9981	40	98	138	2012	6	1	11	0	0	0

3.3 연속형(수치형) 데이터 피쳐 스케일링

연속형(수치형) 데이터의 범위를 동일하게 설정하기 위해 피쳐 스케일링을 진행한다. 피쳐 스케일링은 크게 표준화와 정규화 두 가지 방법을 사용한다.

- 표준화 : 데이터들의 평균을 0, 분산을 1로 변형
- 정규화 : 데이터의 범위를 0 ~ 1사이로 변형

분석에 사용되는 연속형 데이터들이 퍼져있는 범위가 넓지 않고, 0 ~ 100 사이에 분포하고 있으므로 **정규화 방법을 사용하여 연속형 데이터들의 범위를 0 ~ 1 사이로 변형**한다.

정규화 전/후 연속형 데이터

humidity	windspeed	humidity	windspeed
46	23.9994	0.53	0.254894
39	6.0032	0.34	0.254894
55	7.0015	0.65	0.058731
61	12.9980	0.44	0.000000

정규화 후 기술통계량

	humidity	windspeed
count	10886.000000	10886.000000
mean	0.618865	0.147474
std	0.192450	0.141338
min	0.000000	0.000000
25%	0.470000	0.019577
50%	0.620000	0.137170
75%	0.770000	0.215609
max	1.000000	1.000000

최소값 : 0

최대값 : 1

3.4 주성분 분석

온도 정보가 포함된 temp 컬럼과 atemp 컬럼의 상관계수는 0.99로 매우 높은 상관성을 보이므로, 해당 컬럼들은 PCA(주성분 분석)를 통해 하나의 성분으로 축소하여 예측 모델의 독립 변수로 설정한다.

온도 데이터 PCA

- temp, atemp 컬럼 표준화 전 주성분의 계수 : -0.6775, -0.7354
  - 표준화 전 주성분 분석을 진행하였을 때 두 변수가 비슷한 주성분 계수를 포함하고 있음
  - temp(실제온도) 계수(-0.6775)와 , atemp(체감온도) 계수(-0.7354) 으로 구성됨
- temp, atemp 컬럼 표준화 후 주성분의 계수 : -0.7071, -0.7071
  - 표준화 후 데이터로 PCA를 진행한 결과 새로운 변수에 두가지 변수(atemp, temp)의 분산이 동일하게 포함되고 있음을 확인
  - 따라서 표준화된 두 변수로 주성분이 담긴 새로운 변수(temp_pca)를 생성
- 주성분인 새로운 변수(temp_pca)의 전체 데이터 설명력 : 0.99247
  - 새로운 변수의 전체 데이터 설명력이 매우 높은값을 나타냄, 따라서 해당 변수(temp_pca)를 예측 모델의 학습에 필요한 독립 변수로 설정
- 주성분 분석에 사용된 atemp와 temp 컬럼 제거

3.5 피처 선택

현재까지 추가된 파생 변수들과, 이로 인해 불필요해진 변수들을 확인하여 불필요한 변수들을 제거하고 모델 제작에 필요한 변수들만 선택한다.

제거되는 변수 목록

['datetime', 'season', 'temp', 'atemp', 'casual', 'registered', 'count', 'scaled_atemp', 'scaled_temp']

최종 선정된 독립변수 및 종속변수

```
X = encoded_train_df.drop(columns='count')
X.sample(3)
```

holiday	workingday	humidity	windspeed	year	date	month	hour	weather_2	weather_3	weather_4	temp_pca
0	1	77	16.9979	2012	3	3	6	0	0	0	0.477279

```
y = encoded_train_df[['count']]
y.sample(3)
```

	count
5842	130

해당 변수들은 X라는 새로운 데이터 프레임으로 저장하고, 종속변수인 count는 y라는 새로운 데이터 프레임으로 저장한다.

4. 예측 모델 선택

4.1 다중 선형 회귀 분석

하나의 변수로 예측을 진행하는 단일 선형 회귀와는 다르게, 다양한 변수를 통해 예측을 진행하는 경우 다중 선형 회귀 분석을 통해 더 높은 예측 정확도를 확보할 수 있습니다.

✓ 모델 생성

```
from sklearn.linear_model import LinearRegression

# 다중 선형회귀 모델 정의
lr_model = LinearRegression()

# 모델 학습
lr_model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

✓ 예측값 확인

```
lr_preds = lr_model.predict(X_test)

lr_preds
```

```
array([[ 241.59912794],
       [  39.64142629],
       [ 201.98646686],
       ...,
       [ 207.57281013],
       [ 300.58625901],
       [-116.81594528]])
```

```
# 전체 데이터 수
print(f"전체 예측값 개수 : {len(lr_preds)}개")
```

```
전체 예측값 개수 : 3266개
```

```
# 음수값 개수 확인
print(f"음수 예측값 개수 : {(lr_preds <= 0).sum()}개")
```

```
전체 예측값 개수 : 3192개
음수 예측값 개수 : 151개
```

다중 선형 회귀 모델을 통해 생성된 예측값을 확인한 결과, 자전거 대여량 예측값에 음수 값이 존재한다는 것을 확인하였다. 이후 음수값의 개수를 파악한 결과 전체 데이터 3,266개 중에서 **음수로 예측한 값은 151개**로 확인되었다. 예측값에 음수가 발생한 원인은 다음과 같다.

- 선형회귀는 **예측값에 제약 조건이 없으므로**, 양수 또는 음수로 예측값이 도출될 수 있다.
- 데이터가 선형이 아닌 비선형의 모양으로 **비선형 문제에 적합한 회귀 모델을 사용**해야 한다.

✓ 해결 방안

예측값이 음수로 나오는 문제를 해결하기 위해서는 크게 두 가지 방법이 존재한다.

1. 음수 예측값을 대여량의 최저값인 1으로 대체
2. 비선형 문제에 적합한 예측 모델 사용
    - 포아송 회귀분석, 랜덤포레스트 회귀분석 등의 앙상블 기법

먼저 음수로 도출되는 예측값을 전부 1로 대체하고 오차값을 비교해보고, 앙상블 기법을 적용해본다.

RMSLE 예측 비교 및 결과 해석

	예측값 음수 모델	예측값 양수 모델
RMSE	141.01298	140.2606
RMSLE	Nan	1.2396

예측값 음수인 데이터와, 예측값 음수를 1로 변환한 데이터의 RMSE 및 RMSLE 비교하여 아래와 같은 결과를 도출

- RMSE에는 큰 차이가 존재하지 않으며, 둘 다 약 140대 정도 오차가 발생한다고 예측
- 예측하는 단위가 작을경우 RMSLE를 사용하여 예측값과 실제값을 비교하는 방법을 사용
- RMSLE 확인 결과 예측값이 음수로 나오는 경우 계산이 불가능하며, **예측값을 전부 양수로 변경 후 RMSLE를 구했을 경우 1.237 도출**

4.2 XGBoost 회귀 분석

여러 변수와 종속변수의 관계가 일차식이 아닌 다차식으로 이루어져 복잡한 비선형 관계가 형성되는 경우 비선형 문제를 해결할 수 있는 분석 방법을 사용한다. XGBoost는 여러 개의 약한 결정 트리 모델을 결합하여 점점 더 나은 예측을 할 수 있도록 만들어내는 **부스팅(Boosting)기법의 단점을 보완한 예측&분류 알고리즘**이다. 이 때, 각 단계는 이전 모델의 오차를 보완하는 방식으로 학습을 진행한다.

모델 생성

```
import xgboost as xgb
from sklearn.metrics import mean_squared_error, make_scorer

# XGBoost 모델 생성
xgb_model1 = xgb.XGBRegressor(objective='count:poisson', base_score=0.1, random_state=5)

# XGBoost 모델 학습
xgb_model1.fit(X_train, y_train)
```

```
XGBRegressor
        colsample_bylevel=None, colsample_bynode=None,
        colsample_bytree=None, device=None, early_stopping_rounds=None,
        enable_categorical=False, eval_metric=None, feature_types=None,
        gamma=None, grow_policy=None, importance_type=None,
        interaction_constraints=None, learning_rate=None, max_bin=None,
        max_cat_threshold=None, max_cat_to_onehot=None,
        max_delta_step=None, max_depth=None, max_leaves=None,
        min_child_weight=None, missing=nan, monotone_constraints=None,
        multi_strategy=None, n_estimators=None, n_jobs=None,
        num_parallel_tree=None, objective='count:poisson', ...)
```

XGBoost에 입력되는 파라미터는 기본값으로 하되, 음수 예측값을 방지하기 위해 objective 파라미터에 Poisson(포아송 분포), base_score 파라미터는 0.1로 설정하여 음수 예측을 줄이도록 한다.

예측값 확인

```
# XGBoost 예측
xgb_pred1 = xgb_model1.predict(X_test)

# 성능 평가
# RMSE
print(f"XGBoost 기본 파라미터 모델 RMSE : {mean_squared_error(y_test.iloc[:, 0], xgb_pred1)**0.5:.4f}")
print(f"XGBoost 기본 파라미터 모델 RMSLE : {rmsle(y_test.iloc[:, 0], xgb_pred1):.4f}")
```

```
XGBoost 기본 파라미터 모델 RMSE : 40.0079
XGBoost 기본 파라미터 모델 RMSLE : 0.3154
```

K - Fold 교차 검증

훈련 데이터와 테스트 데이터로 나누어진 데이터가 단순히 특별한 지점에서만 좋은 예측 결과를 도출할 수 있으므로, **데이터를 K개로 나누어서 교차 검증**을 진행하도록 한다.

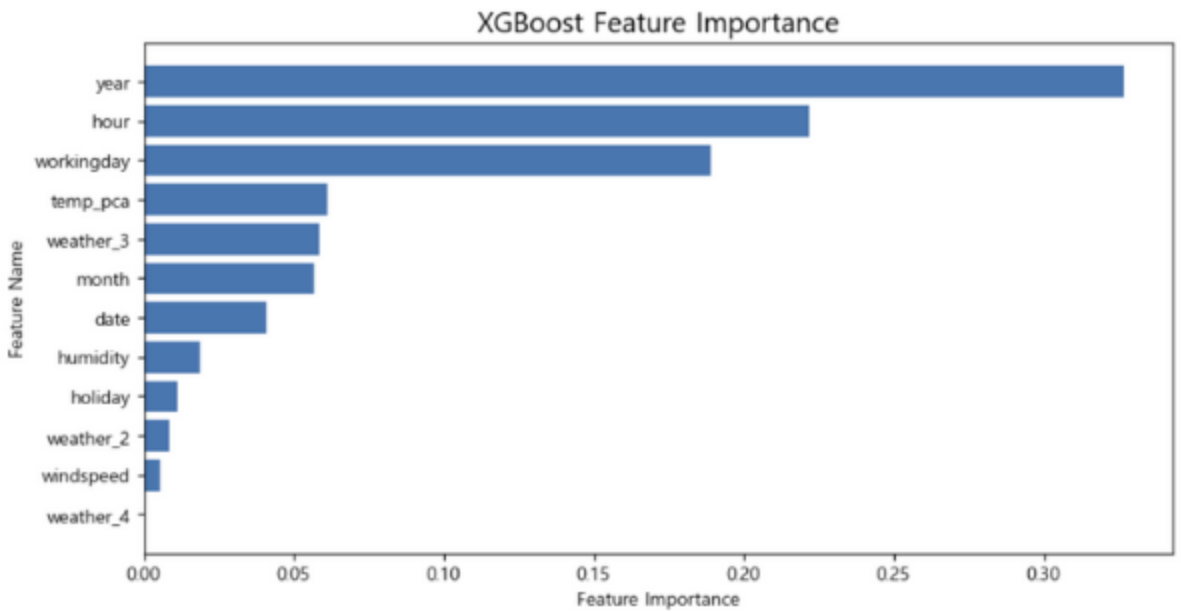
XGBoost K - Fold 교차 검증 결과

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
각 폴드의 RMSLE	0.3190	0.2832	0.3065	0.2956	0.2957
평균 RMSLE			**0.3001**		

다중 선형 회귀, XGBoost 모델 RMSLE 비교

	다중 선형 회귀 분석	XGBoost 회귀 분석
RMSE	140.2606	40.0079
RMSLE	1.2396	0.3001

변수 중요도 시각화



4.3 랜덤 포레스트 회귀 분석

여러 변수와 종속변수의 관계가 일차식이 아닌 다차식으로 이루어져 복잡한 비선형 관계가 형성되는 경우 비선형 문제를 해결할 수 있는 분석 방법을 사용한다. **여러 개의 모델을 결합하여 예측 성능을 향상시키는 앙상블 기법 중 대표 알고리즘인 랜덤 포레스트 방식**을 사용해서 복잡한 비선형 관계를 해석하고, 대여량을 예측하도록 한다.

⊘ 모델 생성

```
# 기본 파라미터값으로 모델 생성하여 할당
rf_model = RandomForestRegressor(random_state=5)

# 모델 학습
rf_model.fit(X_train, y_train.iloc[:, 0])
```

```
RandomForestRegressor
RandomForestRegressor(random_state=5)
```

⊘ 예측값 확인

```
# 예측값 생성
rf_preds = rf_model.predict(X_test)

# 예측값 확인
rf_preds
```

```
array([136.3 ,  20.77, 149.9 , ..., 232.34, 189.83,   2.31])
```

```
# 음수값 개수 확인
print(f'음수 예측값 개수 : {(rf_preds <= 0).sum()}개')
```

```
음수 예측값 개수 : 0개
```

랜덤 포레스트 회귀 분석을 통해 예측을 진행한 결과 음수값이 포함된 예측값은 생성되지 않았다.
앞서 구한 XGBoost의 RMSE와 RMSLE를 랜덤 포레스트 모델과 비교하여, 어떤 모델이 더 좋은 성능을 나타내는지 확인한다.

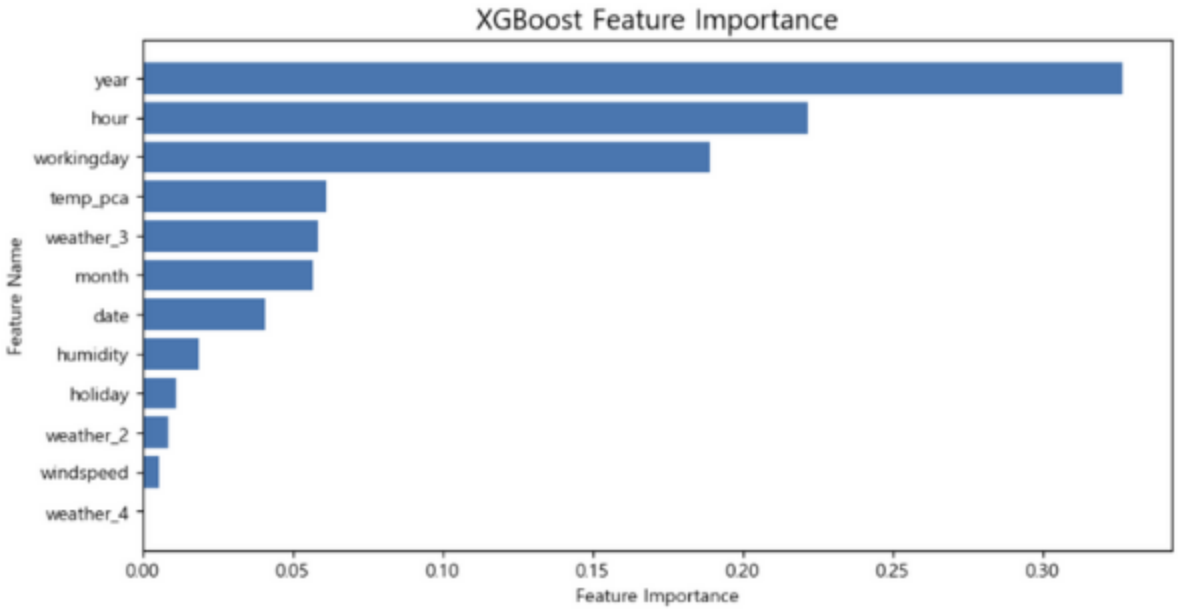
⊘ K - Fold 교차 검증 결과

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
각 폴드의 RMSLE	0.3416	0.3094	0.3240	0.3098	0.3279
평균 RMSLE			**0.3223**		

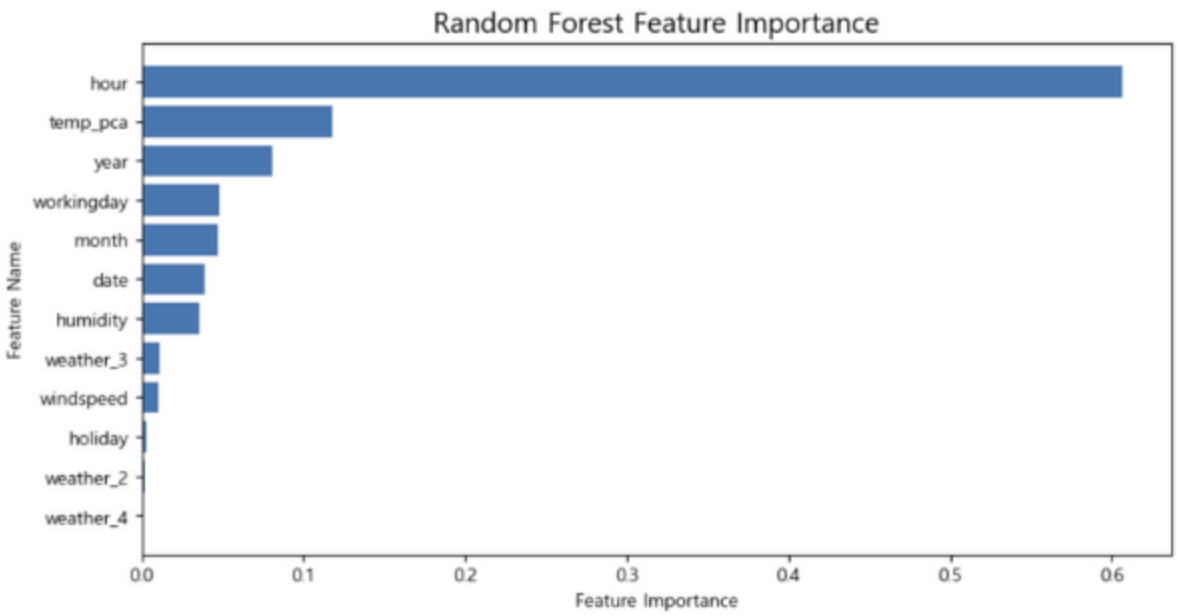
⊘ 다중 선형 회귀, XGBoost, 랜덤 포레스트 모델 RMSLE 비교

	다중 선형 회귀 분석	XGBoost 회귀 분석	랜덤 포레스트 회귀 분석
RMSE	140.2606	40.0079	41.8516
RMSLE	1.2396	0.3154	0.3223

XGBoost 모델 변수 중요도 시각화



랜덤포레스트 모델 변수 중요도 시각화



모델에 따른 변수 평가 방식이 다르기 때문에 변수 중요도의 순위가 다르게 표현된다. XGBoost는 year과 같이 데이터 패턴을 크게 결정하는 변수에 중요도를 높게 주기 때문에 year의 중요도가 높게 나타난 반면에, 랜덤포레스트는 hour와 같이 즉각적인 영향을 주는 변수의 중요도가 높게 표현됐다.

- XGboost 변수 중요도 : year > hour > workingday > temp_pca > weather_3 ....
- 랜덤포레스트 변수 중요도 : hour > temp_pca > year > workingday > month > date > .....

RMSLE는 XGBoost가 더 높게 나왔지만, 연도 변수의 중요도가 높은 모델을 선택할 경우 연도가 달라짐에 따라서 모델 성능 차이가 크게 나타날 수 있으므로, **시간 변수 중요도가 높은 랜덤포레스트 모델을 선정하여 분석을 진행하도록 한다.**

Ⅲ

모델 성능 향상

1. 하이퍼 파라미터 튜닝
2. 제출용 데이터셋 적용

1. 하이퍼 파라미터 튜닝

1.1 GridSearchCV

랜덤포레스트 모델의 파리미터 값을 찾기 위해 GridSearchCV를 활용하여 선정한 파라미터 값 중에서 최적의 파라미터 값을 도출한다.

GridSearch 파라미터 값 리스트

	grid1	grid2	grid3
n_estimators	50	100	200
max_depth	None	10	20
min_sample_split	2	5	10

GridSearch는 총 5번 교차 검증을 진행하고, 점수 평가는 RMSLE 를 평가하도록 한다.

최적의 파라미터 값

- max_depth : None
- min_sample_split : 2
- n_estimators : 200

최적의 파라미터 값 적용 후 RMSLE

- RMSLE : 0.3208 (하이퍼 파라미터 튜닝 전 RMSLE : 0.3223
- 하이퍼 파라미터 튜닝을 통해 약 0.0015 정도 RMSLE 값이 감소되었음을 확인

2. 제출용 데이터셋 적용

train 데이터로 생성된 랜덤포레스트 모델을 제출용인 test 데이터에 적용하여 예측값을 생성 후 csv 파일로 제출한다.

```
# 예측값 test 데이터에 입력하기
test_df['count'] = submission_predict_count
# 결과 확인
test_df.sample(2)
# csv 파일 저장
test_df.to_csv("test_df_2팀_고명진.csv")
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	year	date	month	hou	count
1166	2011-05-26 14:00:00	2	0	1	1	33.62	38.635	52	19.0012	2011	3	5	14	123.28
5806	2012-10-24 18:00:00	4	0	1	1	27.06	31.060	44	0.0000	2012	2	10	18	503.05