오픈소스 SW 프로젝트

- 최종 보고서 -

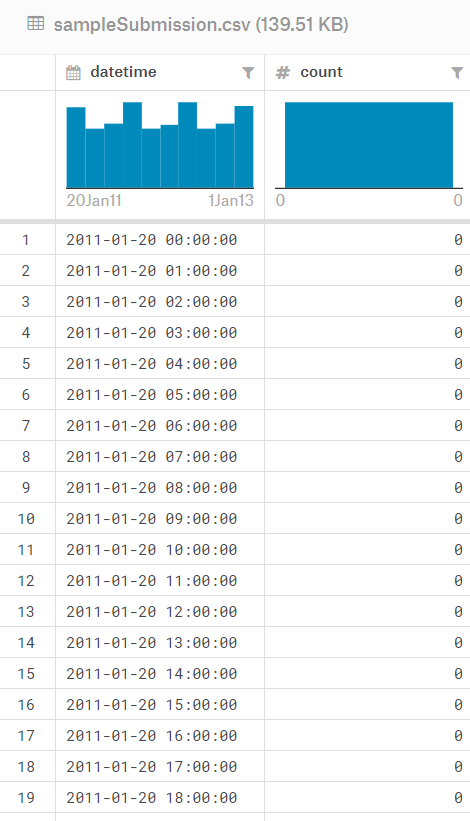
컴퓨터 공학과

20164244 전연지

**1. 주제**

자전거 대여 수요 예측하기 ( Predict Bike Sharing Demand )

<https://www.kaggle.com/c/bike-sharing-demand>

 <- kaggle에서 제공한 제출 sample

위의 사진에서 보여지는 것처럼, Datetime이 주어지면 그에 따른 Count(자전거 대여 수)를 예측하면 된다.

**2. 필요성 및 기대효과**

자전거를 대여하고 반납하는 자전거 대여 시스템은 전세계에 500개가 넘게 도입 돼있다. 실제로, 우리나라 또한 자전거 대여 시스템을 시행하고 있다. 이 시스템은 교통체증, 대기오염, 고유가 문제가 대두되는 현 문제들의 해결방안이 되어 줄 수 있다.

실제 자전거 대여 시스템에서 발생되는 데이터는 사용 기간, 출발 위치, 도착 위치 및 소요 시간등이 기록 돼있다. [1] 이 프로젝트에서 제공하는 데이터는 사용 시간과 날씨, 주말 여부 등을 제공하지만, 이 간단한 정보들을 분석함으로써 도시의 인구 유동성 연구, 또는 시스템 수 증설 여부 등에 도움을 줄 수 있다.

**3. 기존 연구**

대부분 Python 또는 R로 구현하였다. 하지만, R은 구현해 본 경험이 적어 Python으로 구현한 연구들을 중심으로 보았다.

|  |  |  |
| --- | --- | --- |
| EDA & Ensemble Model (Top 10 Percentile)  https://www.kaggle.com/viveksrinivasan/eda-ensemble-model-top-10-percentile | 모델 / 라이선스 | 앙상블 모델 / Apache License 2.0 |
| 특징 | 선형회귀(Linear Regression) 모델, 릿지(Rigde) 모델, 라쏘(Lasso) 모델, 랜덤 포레스트(Random Forest), 그라디언트 부스트(Gradient Boost) **모두를 적용**하여 **각각의 RMSLE을 알아 보고**, 적은 RMSLE를 가지는 모델(앙상블 모델)로 적용하였다. |
| 장점 | 시각화 도구를 다양하고 적합하게 사용하여, 결과를 이해하기 쉽게 볼 수 있다.  또한, 여러 모델을 사용하였기 때문에 가장 정확한 결과가 나타났다. |
| GBR + EDA  https://www.kaggle.com/delimixx/gbr-eda | 모델 / 라이선스 | 그라디언트 부스트 / Apache License 2.0 |
| 특징 | **그리드 서치(GridSearchCV)**를 통해 최고 점수를 낸 파라미터(best\_params\_)와 최고 점수(best\_score)를 출력하고, 알아낸 정보를 파라미터로 한 **그라디언트 부스트(Gradient Boost)** 를 모델로 사용하였다. |
| 장점 | heatmap을 사용하여 컬럼 사이의 연관관계를 보여주었는데, 연관관계가 현저히 낮은 것은 볼 필요가 없기 때문에 논리 연산자를 조건으로 주어 일부만 출력한 것이 보기에 편해 좋았다. |
| Python Machine Learning hands-on  https://www.kaggle.com/nandobr/python-machine-learning-hands-on | 모델 / 라이선스 | 랜덤 포레스트 / Apache License 2.0 |
| 특징 | 하지만 포루투칼어로 설명이 쓰여 있기 때문에 해석하기 어려울 수 있다.  count를 count+1에 log를 씌운 값으로 하여 count값이 최저 값과 최고 값의 낙폭이 크지 않도록 한다. |
| 장점 | 다른 커널들이 사용하지 않은 방식을 많이 사용하여 결과는 비슷하지만 이렇게도 표현할 수 있다는 것을 배울 수 있다. |
| Bike rental predictions using LR, RF, GBR  https://www.kaggle.com/yaroshevskiy/bike-rental-predictions-using-lr-rf-gbr | 모델 / 라이선스 | 선형회귀 모델, 랜덤 포레스트, 그라디언트 부스트 / Apache License 2.0 |
| 특징 | 다른 커널들과는 다르게 피쳐 엔지니어링 부분이 많이 적고, datetime을 파싱하는 기본적인 작업만 하였다. |
| 장점 | 선형회귀 모델, 랜덤 포레스트, 그라디언트 부스트 중 scatter를 사용하여 RMSLE가 적은 모델을 선택하는 과정에서 scatter를 이용해 시각화 하여 비교 하기에 한 눈에 보여 쉽게 느껴졌다. |
| BIKE SHARING DEMAND [ RMSLE:: 0.3194] https://www.kaggle.com/rajmehra03/bike-sharing-demand-rmsle-0-3194 | 모델 / 라이선스 | 랜덤 포레스트 / Apache License 2.0 |
| 특징 | 모든 모델들을 구현하여 RMSLE를 비교하지 않고, 사이킷런(scikit-learn)을 사용한 **머신 러닝 분류기(classfier)**로 가장 적은 RMSLE이 무엇인지 비교 해 본 후, 적은 모델을 적용하였다.  하지만 다른 커널들이 공통적으로 수행한 피처 엔지니어링(풍속이 0인 것에 대한 엔지니어링)을 않아 부족함이 있었다. |
| 장점 | 모든 모델에 대해 구현하지 않고도 어느 모델의 RMSLE가 가장 적게 나오는 지를 쉽고 간단하게 알 수 있다. |

**4. 제안 방법**

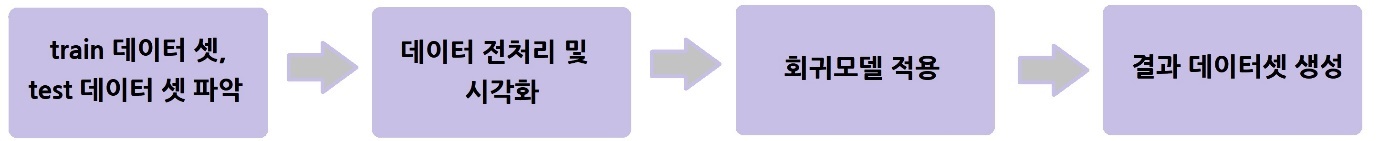
**1) 문제 파악**

미국 워싱턴 D.C.의 2011년부터 2012년까지 모든 월의 1일~19일 까지의 자전거 대여 데이터셋 (train set)을 가지고, 그 나머지 날짜 (20일~말일)의 자전거 대여 수를 예측한다.

즉, 시계열 데이터에따라 특정 기후 상황이 주어졌을 때 얼마나 자전거의 대여가 되는 지를 예측하는 것이기 때문에 **회귀(regression)**에 해당된다.

또한 지도 학습과 비지도 학습 중 **지도 학습**으로 풀어나가야 한다. train 데이터 셋에 대여량 데이터가 있다. 그 데이터를 가지고 모델을 만들어 학습을 시키고, 만든 모델에 우리가 알고 싶은 test 데이터를 넣어 대여량을 예측해야 한다. [2]

**2) 진행 단계**

****

① 몇개의 데이터, 어떠한 형태로 구성 되어있는지, 컬럼의 속성 데이터 값 파악

② train 데이터 셋을 예측 모델에 맞게 **가공**하고, **시각화** 함. test 데이터셋 또한 train 데이터 셋과 동일한 전처리 과정을 진행함

③ 피처를 갖는 **train 데이터셋과 train 레이블**로 **학습**을 시킨 모델을 만들고, 그 모델에 테스트 데이터 셋을 넣어 **예측**시킴

④ kaggle에 제출 할 결과 데이터셋을 생성함

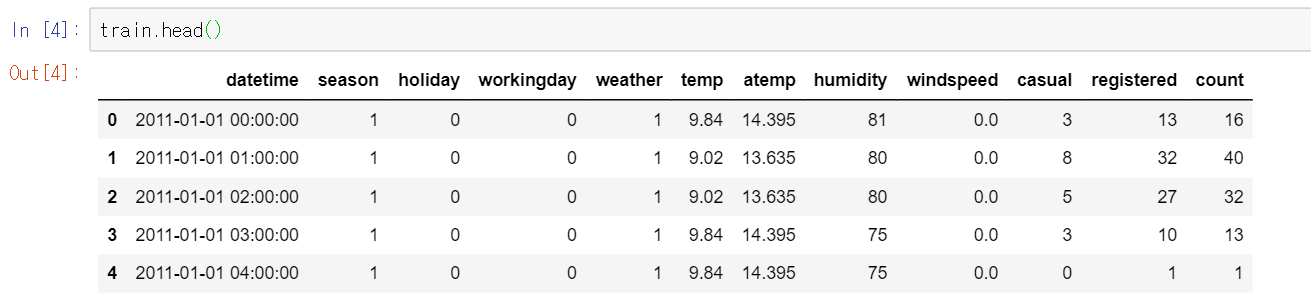
**3) 데이터베이스 다운로드 (자료 수집)**

- **train set** :

10886개의 데이터를 가짐

datetime, season, holiday, workingday, weather, temp, atemp, humidity, windspeed, casual, registered, count 를 데이터 필드로 가짐

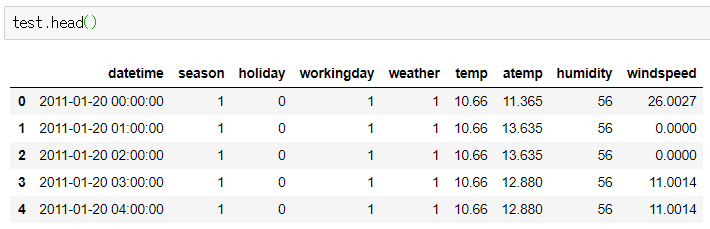
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 범주형 데이터를 갖는 데이터 필드 | | 연속형 데이터를 갖는 데이터 필드 | | |
| datetime | 연도-월-일-시간 | | **atemp** | 체감 온도 |
| season | 1: 봄 | | **humidity** | 습도 |
| 2: 여름 | |
| 3: 가을 | |
| 4: 겨울 | |
| holiday | 0: 공휴일 | | **windspeed** | 풍속 |
| 1: 비공휴일 | |
| workingday | 0: 근무일(주말 또는 휴일 아님) | | **casual** | 등록되지 않은 비회원의 대여량 |
| 1: 근무일 아님 | |
| weather | 1: 맑음 + 구름 거의 없음 + 구름 약간 | | **registered** | 등록된 회원의 대여량 |
| 2: 옅은 안개 + 구름 + 구름 약간과 옅은 안개 | |
| 3: 가벼운 눈 또는 비 + 가벼운 비 후 개임 + 흐트러진 구름(scattered clouds) | |
| 4: 호우+ 빙판 + 뇌우 + 짙은 안개와 눈 + 짙은 안개 | |
| temp | 섭씨 온도 | | **count** | 총 대여량 |



- **test set** :

6493개의 데이터를 가짐

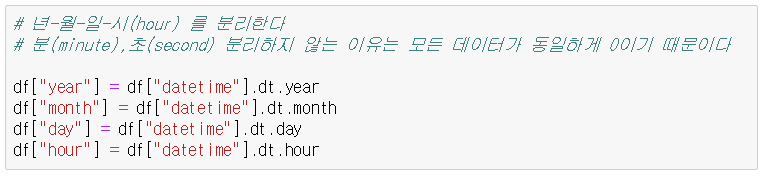
test set에서는 **casual, registered, count** 는 주어이지 않으며 (이를 예측 해야한다.), datetime은 test set의 datetime과 겹치지 않는다.

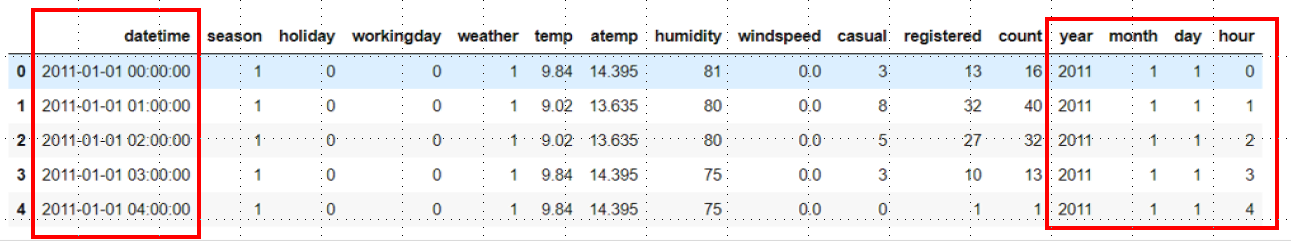


**4) 예측(prediction) 모델에 맞게 데이터 변형(데이터 가공)**

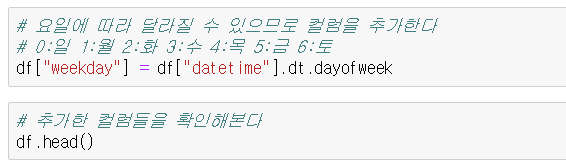
**① datetime을 파싱하기**

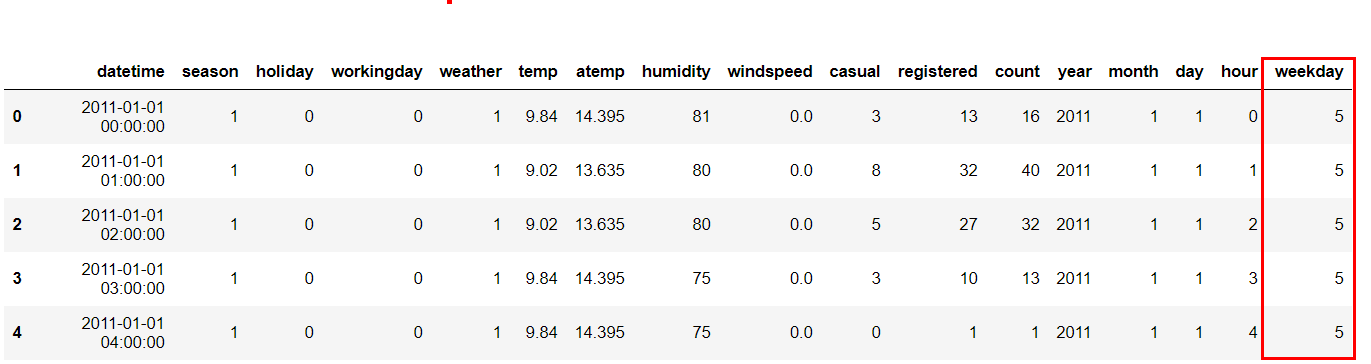
원래의 datetime 컬럼에는 2019년 1월 19일이라면, 2019-01-19 라는 형식의 데이터를 갖는다. 이들 각각을 분리해준다면 **연도, 월, 일, 시간에 따라 대여량을 비교**할 수 있으므로 datetime을 파싱하여 컬럼을 추가하였다.





**② 요일 지정하기**



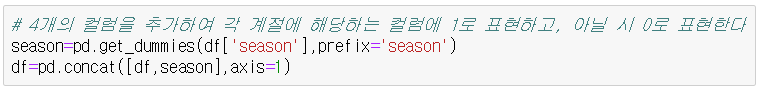


datetime을 통해, 무슨 요일인지 알아낸다. 알아낸 결과를 weekday라는 컬럼에 **각각 지정한 요일에 해당하는 숫자(0:일 1:월 2:화 3:수 4:목 5:금 6:토)** 를 넣어준다. 이 또한, **요일에 따른 대여량**을 알아낼 수 있게 해준다.

**③ season 데이터 별로 컬럼 추가하기**

season이라는 계절 컬럼에 1,2,3,4로 값을 표현하는 것에서, 한 계절 당 한 컬럼을 만들어 0과 1로 값을 바꾸었다. 각 계절과 대여량과의 연관관계를 알 수 있다.

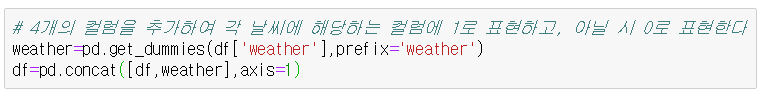
봄,여름,가을,겨울에 해당하는 **season\_1, season\_2, season\_3, season\_4** 컬럼이 생긴다.



**④ weather 데이터 별로 컬럼 추가하기**

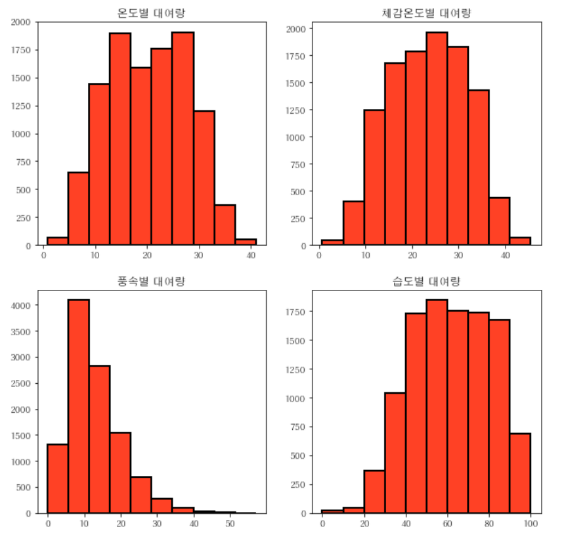
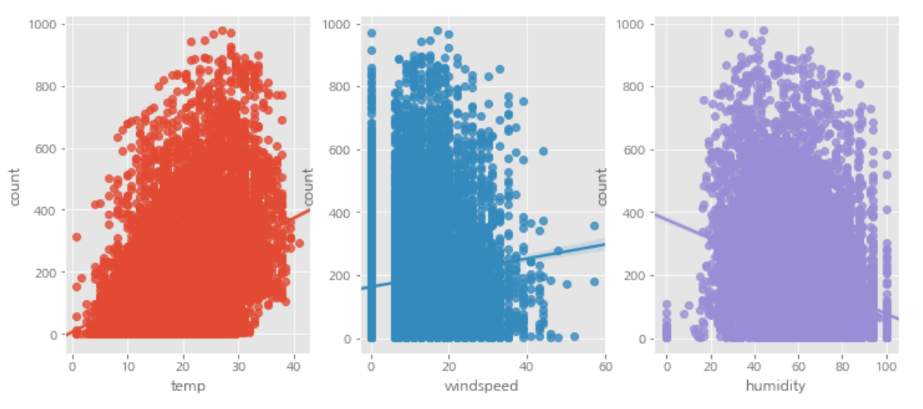
③번과 같이, weather도 4개로 나누어진 값들을 나누어 각각 하나의 컬럼을 만들어 표현한다.

weather 값 1,2,3,4에 해당하는 **weather\_1, weather\_2, weather\_3, weather\_4** 컬럼이 생긴다.



**⑤ windspeed가 0인 데이터 처리하기**

아래의 그래프를 보면 풍속 그래프에 다른 그래프와는 달리 0에 가까운 데이터가 많이 몰려있는 것을 볼 수 있다. 이는 아마도 관측되지 않은 수치에 대해 0으로 기록된 것이 아닐까 추측해 본다.



따라서 이를 머신 러닝(랜덤 포레스트 모델)으로 예측하여 해결한다.

1. 풍속이 0이 아닌 데이터 프레임(Wind0)과, 0인 데이터 프레임(WindNot0)을 나눈다

2. 풍속이 0이 아닌 데이터 프레임에서 풍속만을 데이터로 갖는 데이터 프레임(WindNot0\_series)을 만든다

3. 랜덤포레스트 모델에 WindNot0, WindNot0\_series를 학습시킨다

4. 그 학습된 모델에 풍속을 도출하고, 그 값을 Wind0 에 넣는다.



**5) 필요 데이터 선별(불필요 데이터 제외)**

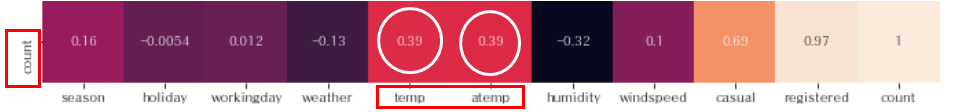


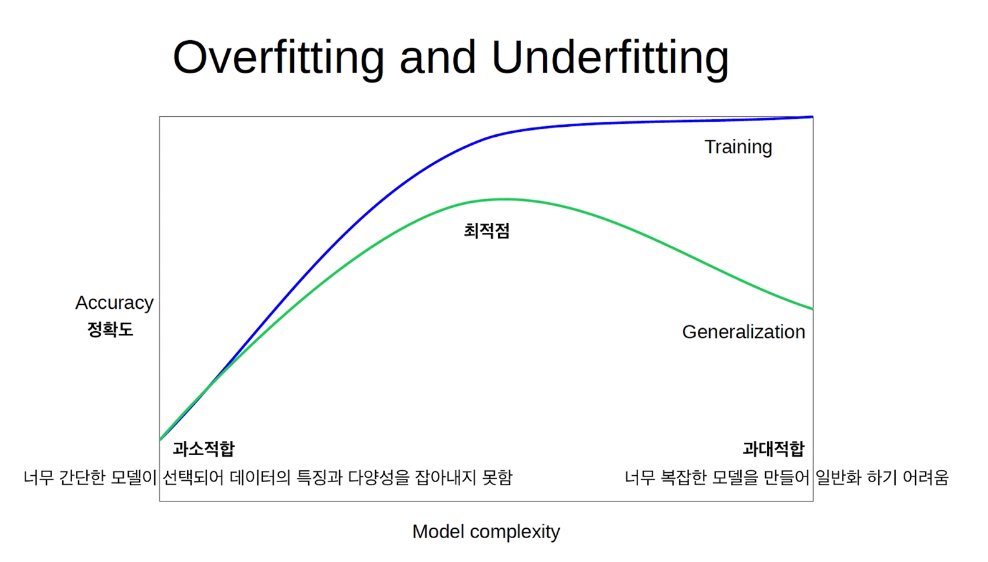
계절, 날씨, 요일, 체감온도, 습도, 풍속, 연, 월, 일, 시, 근무일 을 피처로 선정하였다.

피처로 선정되지 못한 컬럼은 **temp(온도), holiday(공휴일)** 이다.

① temp(온도)를 택하지 않은 이유

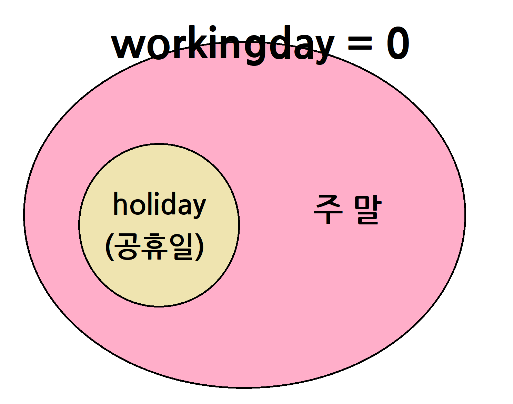
temp(온도)와 atemp(체감온도)는 count(대여량)과의 **연관관계에서 0.39라는 같은 값**을 가지고 있다.

****

****

따라서, 처음에는 temp와 atemp 둘 다 피처로 선정하려 했는데, **피처가 너무 많아 복잡한 모델을 만들게 되면 과대적합(Overfitting)이 되어 오히려 점수가 낮게 나올 수 있다**하여 하나만 택하였다.[2] 수치상의 heatmap 결과로는 온도와 체감온도 둘 다 연관관계는 같았지만, 주관적으로 생각하였을 때, 아무래도 몸이 느끼는 온도가 더 연관이 있을 거란 생각이 들어 체감온도인 atemp를 temp 대신에 피처로 선정하였다.

**②** holiday(공휴일)을 택하지 않은 이유

****

근무일(workingday)이 아닌 날에는 공휴일(holiday)과 주말이 있다. 공휴일이 세부 개념이고, 포함관계에 있다. 따라서, 공휴일 여부 보다는 근무일 여부를 피처로 선정하는 것이 **과대적합을 피하면서 대여량과의 연관관계가 더 높아** 적합하다.

**5. 실험 및 결과 (비교실험)**

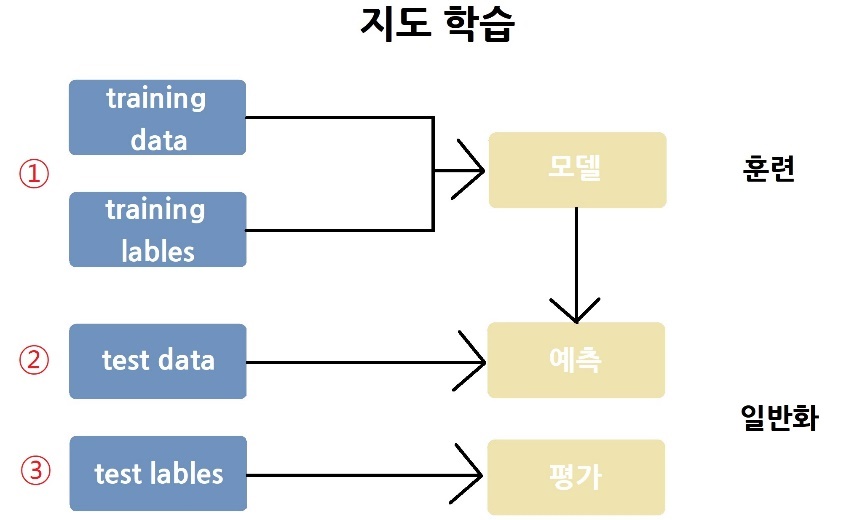
**1) 모델 학습**

**- 개발환경(H/W 스펙, SW 버전)**

|  |  |
| --- | --- |
| 운영체제 | Windows 10 |
| H/W | Intel® Core™ i5-8250U CPU @ 1.60GHz 1.80GHz  64비트 기반 프로세서 |
| 사용 프로그램 | Anaconda Navigator (버전 1.92)  ->Jupyter (버전 5.5.0) |
| 사용 언어 | Python |
| 사용 라이브러리 | Scikit-learn |
| 사용 라이선스 | Apache License 2.0 |

**- 모델 학습 과정**

데이터 셋들을 지도 학습을 하기 위한 형식으로 바꿔 주어야한다. 아래의 사진의 ①번과 ②번 과정을 진행한다.



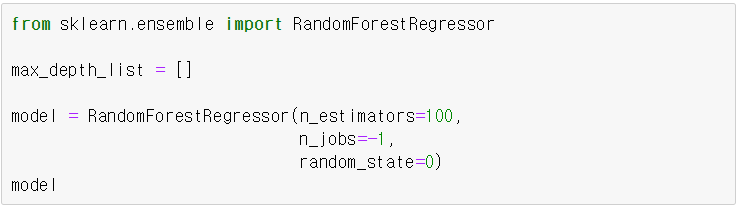
**① 훈련**

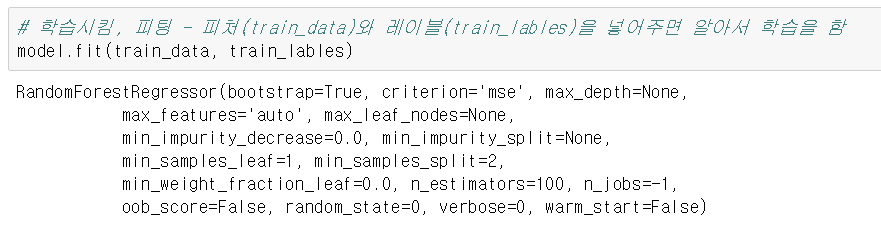
train\_data는 피쳐를 갖는 데이터 셋을 가진다. (형태는 행렬이다.)

train\_labels는 대여량을 갖는 데이터를 가진다. (형태는 벡터이다.)



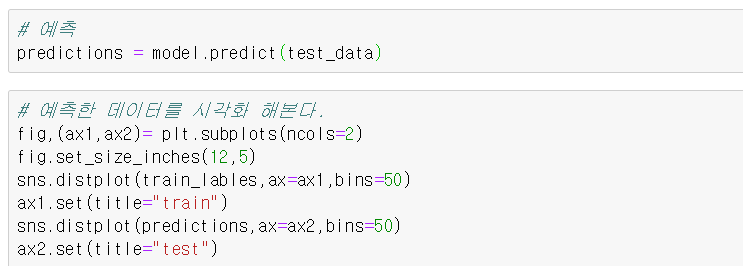
**랜덤 포레스트** 모델을 만들어 **train\_data와 train\_labels를 넣어주어 학습**시킨다.

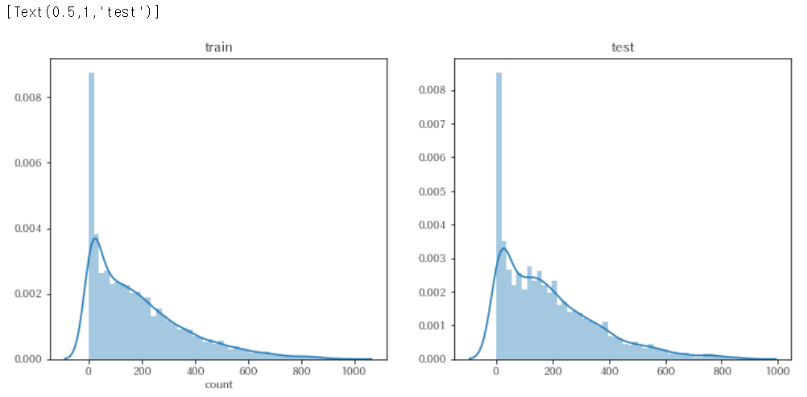




**② 예측**

랜덤 포레스트 모델에 테스트 데이터셋에서 피처 컬럼만을 갖는 test\_data를 넣어 예측하고, 이를 train\_labels와 쉽게 비교할 수 있게 시각화한다.

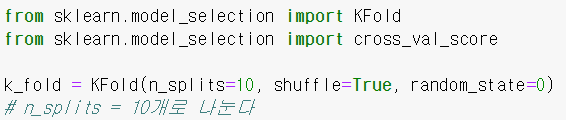




**2) 모델 테스트**

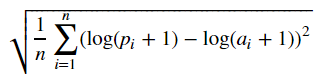
**③ 평가**

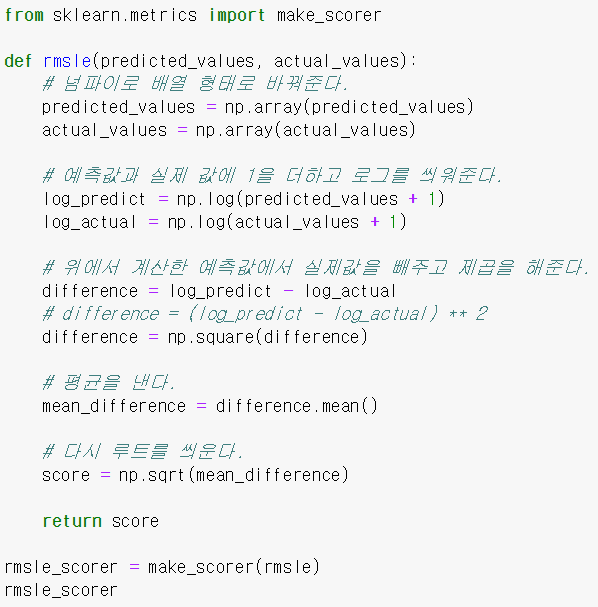
KFold 교차검증 : 만든 모델의 적합성을 보다 객관적으로 평가하기 위해 필요한 방법으로 교차 검증이 있다.[4] 교차 검증 중, KFold 교차 검증은 데이터셋을 여러 개로 나누어 하나씩 test 데이터 셋으로 사용하고 나머지를 모두 학습용 데이터 셋으로 사용하는 방법이다.



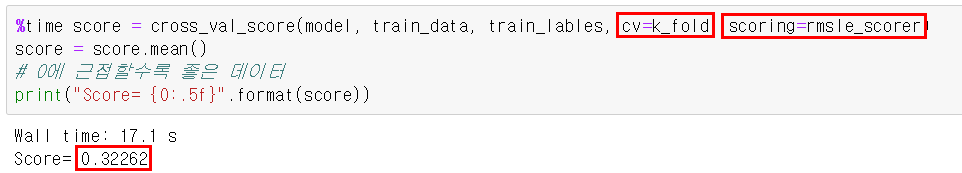
*n\_splits=10 이므로, 데이터 셋을 10개의 그룹으로 나눠 교차 검증을 한다.*

RMSLE 방식 : 과대평가된 항목보다는, 과소평가된 항목에 패널티를 주는 평가 방식이다. 값이 적을수록 오류가 적다는 것을 의미한다.[3] 아래는 RMSLE의 공식이다.



****

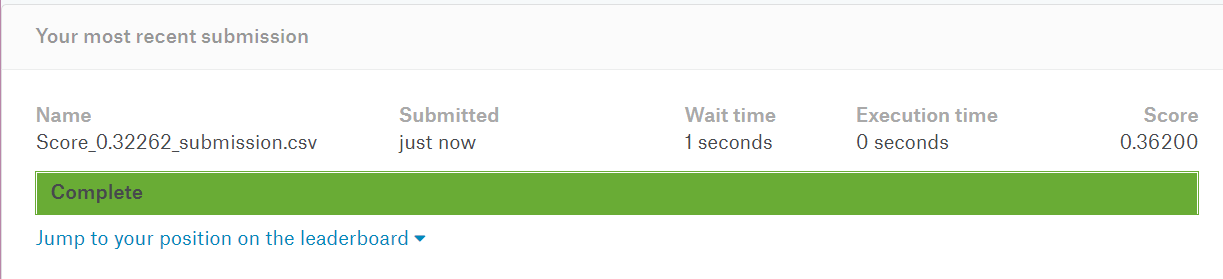
KFold를 통해 단순히 데이터 셋을 나누었다. 실제로 이 나누어진 데이터셋을 사용하여 **평가를 반복**해야한다. 이 과정을 자동화하는 명령인 **cross\_val\_score()**를 이용한다.

****

*cv=k\_fold : 위에서 진행한 k\_fold를 교차 검증하라*

*scoring=rmsle\_scorer : 점수는 위에서 정의한 rmsle방식으로 하여라*

print로 점수를 출력하였고, 나의 점수는 0.32262 이 나왔다.

****

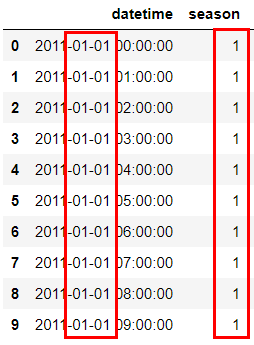
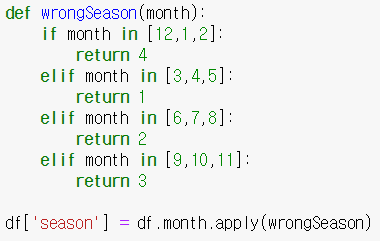
그리고 이를 kaggle에 제출하였다.

**3) 모델 개선**

다른 커널들이 시도하지 않은 나만의 아이디어를 넣어 개선하였다.

① 계절 오류 수정

데이터들은 **미국 워싱턴 D.C.**의 자전거 대여 프로그램에서 나온 데이터이다. **우리나라와 계절변화가 비슷**한데, 1월의 데이터에서 season 칼럼이 1, 즉 봄으로 나타나 있다. (왼쪽 사진)

따라서, 12월,1월,2월은 season이 4(겨울)로, 3월,4월,5월은 season이 1(봄)로, 6월,7월,8월은 season이 2(여름)로, 9월,10월,11월은 season이 3(가을)로 변경해준다. (오른쪽사진)

다른 커널 : 이를 수정하지않고 season 데이터 그대로를 사용하였다.

② 한 데이터 값 당, 컬럼 추가하기 (season, weather 컬럼)

4.제안방법 - 4) 예측(prediction) 모델에 맞게 데이터 변형(데이터 가공)의 내용이다. 1부터 4까지로 표현된 값을 갖는 컬럼에서 0과 1로 값을 표현하는 컬럼을 만들어, 대여량과의 상관관계를 잘 파악할 수 있게 만들어준다.

다른 커널 : season과 weather 컬럼 모두 값을 나누지 않고 사용하였다.

③ 피처 선정

4.제안방법 - 5)필요 데이터 선별(불필요 데이터 제외)의 내용이다. 온도(temp)와 체감온도(atemp)가 상관관계가 같기 때문에 체감온도를 피처로 선정한다.

다른 커널 : temp와 atemp 두 컬럼 모두를 피처로 선정하였다.

**6. 고찰(토론)**

피처를 너무 적게 선정할 시 일어나는 과소 적합과 너무 많이 선정할 시 일어나는 과대 적합을 피하고자 그 사이의 최적점을 찾는 것에 힘들었다. 그 과정에서 예상한대로 결과가 나온 케이스와, 예상과 다른 결과가 나와 실패한 케이스가 있었다.

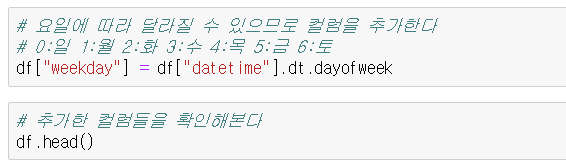
**1) 예상과 같아 성공한 케이스**

과대 적합을 피하기 위해, 대여량과의 연관관계가 서로 같은 temp와 atemp 두 컬럼을 모두 피처로 선정 할 필요가 없다고 느꼈다. 주관적으로 체감온도가 더 적합하다 생각하여 atemp를 선정하였는데, 실제로 temp를 선정하였을 때보다 점수가 좋았다.

|  |  |  |
| --- | --- | --- |
| 경우 | RMSLE | 알 수 있는 사실 |
| temp, atemp 둘 다 선정 | 0.32477 | 둘 다 선정하는 것은 좋지않다. |
| temp를 선정 (atemp를 제외함) | 0.32280 | 작은 차이지만,  atemp의 점수가 더 좋다 |
| atemp를 선정 (temp를 제외함) | 0.32267 |

**2) 예상과 달라 실패한 케이스**

데이터 전처리 과정에서 season컬럼과 weather컬럼에서 한 값마다 해당하는 컬럼을 만들어 1과 0으로 값을 표현했던 것처럼, 요일을 나타내는 0,1,2,3,4,5,6 의 데이터를 갖는 weekday 컬럼 또한 weekday\_0, weekday\_1, weekday\_2, weekday\_3, weekday\_4. weekday\_5. weekday\_6 을 만들어 진행했었다.



하지만, 오히려 점수는 낮게 나왔었다. 과대 적합으로 나타난 것이다. 컬럼을 나누어서 나타내면 요일별로의 연관관계를 더 잘 보여줄 수 있을거라 생각하여 나눴던 건데(실제로 season과 weather은 나눈 것이 더 점수가 좋았다), 아니였다. 따라서 그 진행은 지워버리고, weekday만을 선정하였더니 점수에 큰 차이를 보였다.

|  |  |  |
| --- | --- | --- |
| 경우 | RMSLE | 알 수 있는 사실 |
| weekday\_0 ~ weekday\_6를 선정 | 0.36515 | 가장 좋지 않은 점수  => 피쳐가 너무 많아 **과대적합** |
| weekday\_0 ~ weekday\_6를 제외,  weekday를 선정 | 0.32262 | 가장 적은 점수 |

**7. 참고 문헌**

[1] VG, “[데이터과학] 자전거 수요 예측”, <https://blog.naver.com/vangarang/221131945351>, 2017

[2] 오늘 코드,“자전거 수요 예측[2/4] 캐글 머신러닝 사이킷런 랜덤포레스트 RMSLE 소개”, [https://www.youtube.com/watch?v=95fCw-n5uW](https://www.youtube.com/watch?v=95fCw-n5uWM), 2017

[3] 보스턴테리어,”New\_York\_City\_Taxi\_Trip\_Duration[1]”, https://blog.naver.com/apples629/221242967685, 2018

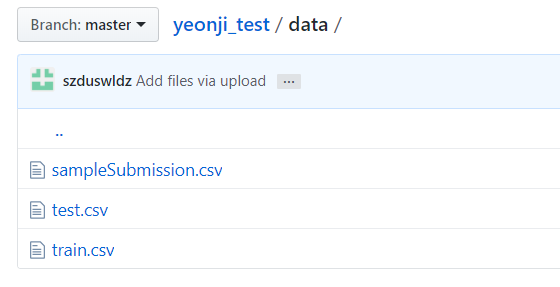
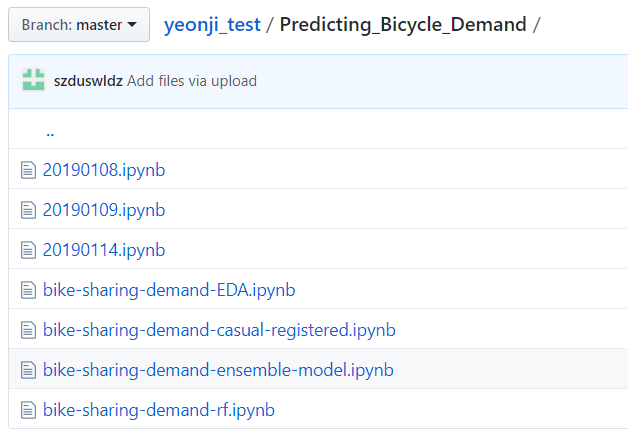
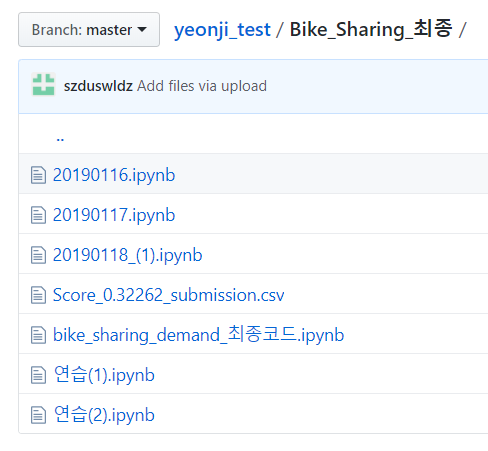
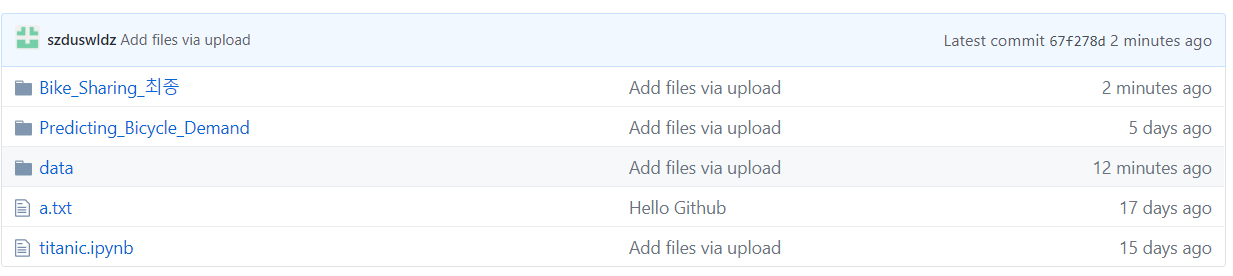
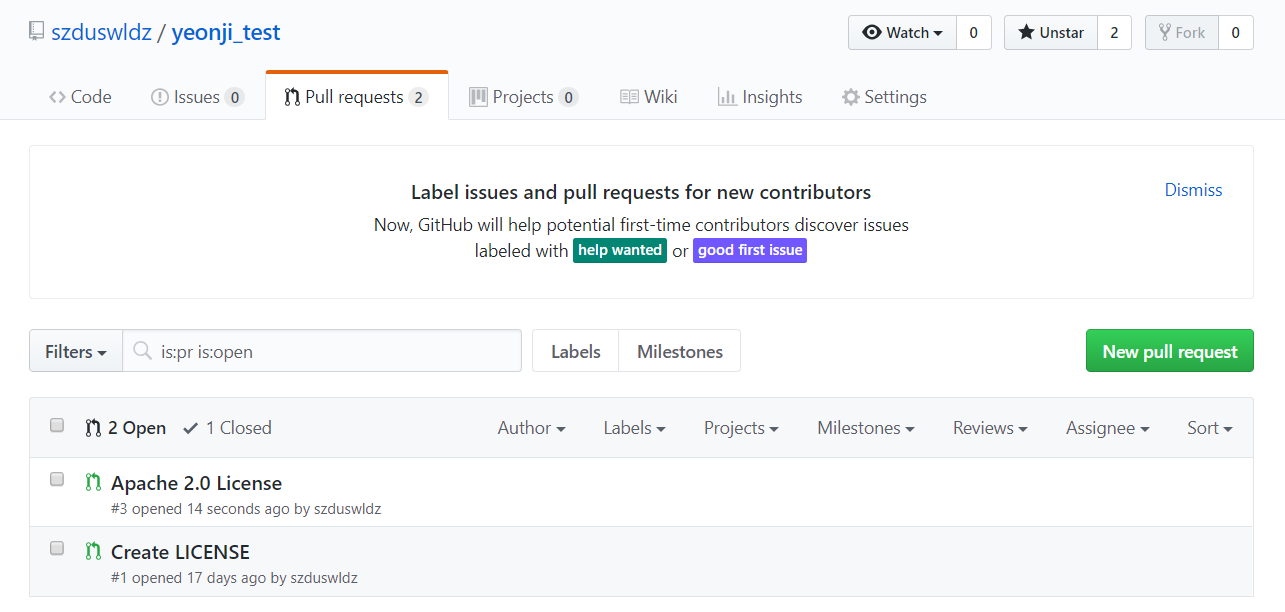
[4] 앨리스, “컴공생의 AI 스쿨 필기 노트 ④ 교차 검증(Cross Validation)과 정규화(regularization)”, <https://elicer.blog.me/221301996981>

**8. 느낀점**

저번 학기에 ‘빅데이터 시스템’ 수업을 들어서 다행히 파이썬을 사용하는 것에 어색함을 느끼지 못했다. 그리고, 종강을 했는데도 수업 때 배운 교재를 자주 펼쳐 보며 복습도 할 수 있었다.

캐글은 처음 접하는 것이었기 때문에 주제를 찾느라고 많이 애를 먹었고, 구현해 나가는 과정에서도 그러하였다. 심지어 머신러닝에 대해서는 학교에서 한번도 배워 본 적 없었기 때문에, 유투브 검색이나 구글링을 통해 독학하고 모방도 하며 힘들게 만들어 나갔다.

다른 학생들 모두가 그러했겠지만 역시나 성능 개선이 가장 힘들었다. 좋은 건지, 좋지 않은 건지 모르겠지만 다른 competition에 비해, 내가 선택한 competition은 참여한 팀도 많고 데이터의 개수가 적고 단순한 편이였다. 따라서 참고할 수 있는 커널이 많았고, 학습시키고 예측하는데 오랜 시간이 걸리지 않았다. 하지만, 아쉬운 점은 그러했기 때문에 남들과 다른 방식의 성능 개선에 어려움이 있었고, 데이터가 간단하기 때문에 좀 더 복잡한 결과를 내고 싶은 욕심이 생기기도 하였다.

이번 프로젝트를 통해 캐글 입문자가 되었기 때문에, 진행중인 competition도 참여해보고 github에 완벽하지는 못하더라도 나만의 코드를 올리면서 활동해볼까 한다. 이제서야 접하게 되어 아쉽지만, 지금이라도 접할 수 있었으니 다행이라 생각하며 부족한 실력이지만 캐글에 영어로 설명하는 나만의 커널을 올리고 싶다. 

https://github.com/szduswldz/yeonji\_test/tree/master/Predicting\_Bicycle\_Demand