
돌체라떼

1

프로젝트 개요

2 프로젝트 개요

>> 팀장 : 박지용

•역할

- 프로젝트 전 총괄 및 검토
- 원 데이터 + 외부 데이터 병합
- 외부데이터 전 처리 및 특성 구하기 - 발전량 데이터
  - 외부 데이터 예측 모델 구현
- 단일 알고리즘 튜닝 및 성능 확인

>> 팀원 : 최두호

•역할

- 프로젝트 수행
- 외부데이터 전 처리 및 특성 구하기 - 기온 데이터
- Stacking 알고리즘 구현 및 성능 확인

2

프로젝트 수행 절차 및 방법

3 프로젝트 수행 절차 및 방법

>> ~ 중간 발표

- 각 팀원 당 한개의 외부 데이터를 전처리 해본다.
- Baseline에서 제공된 lgbm 모델로 외부데이터의 성능 일차적으로 확인해본다.

>> ~ 최종발표

- Feature 선택을 위해 Heatmap 시각화 작업
- PolynomialFeature과 스케일러, 파라미터 튜닝를 적용해본 단일 알고리즘의 성능 확인.
- CV Stacking을 활용한 모델 개선해보기.

목차 A table of contents

2 외부데이터

- 기온데이터

데이터프레임으로 하나씩 불러온뒤, 하나의 데이터 프레임으로 병합해주었음(이슈)

(숫자파일)
기온DF을 기본DF과 병합한뒤, 해당 데이터 프레임으로 기온예측모델생성

기온예측모델을 이용해서 test 데이터프레임에 붙어주었음

이제, 기온의 특성추가를 완료하였기때문에 구분, 년, 월, 일, 시간, 기온 데이터를 사용하여서
공급량 예측 모델 생성 후, 제출.
기본데이터만을 가지고 예측을 했을 때와 비교해서,

0.1617253368 → 0.1307686256
0.031점 정도 차이나는 유의미한 성과를 냈음.

2 외부데이터

- 화력 발전량 데이터

  - 공공데이터에서 17~18년 화력 발전소 시간데이터를 베이스로 데이터 구상을 시작했음.
  - 13년부터 15년 데이터는 에너지원별 발전량 데이터을 참고하여 총 생산량 비율로 17년 , 18년 기준으로 나눈 후 두개를 추합 후, 반으로 나눔.
  - 16년도 윤년 데이터는 예측하지 않고 제거하였음. → 이후, 새로운 특성 데이터와 합칠때 오류 발생 우려로 해결.

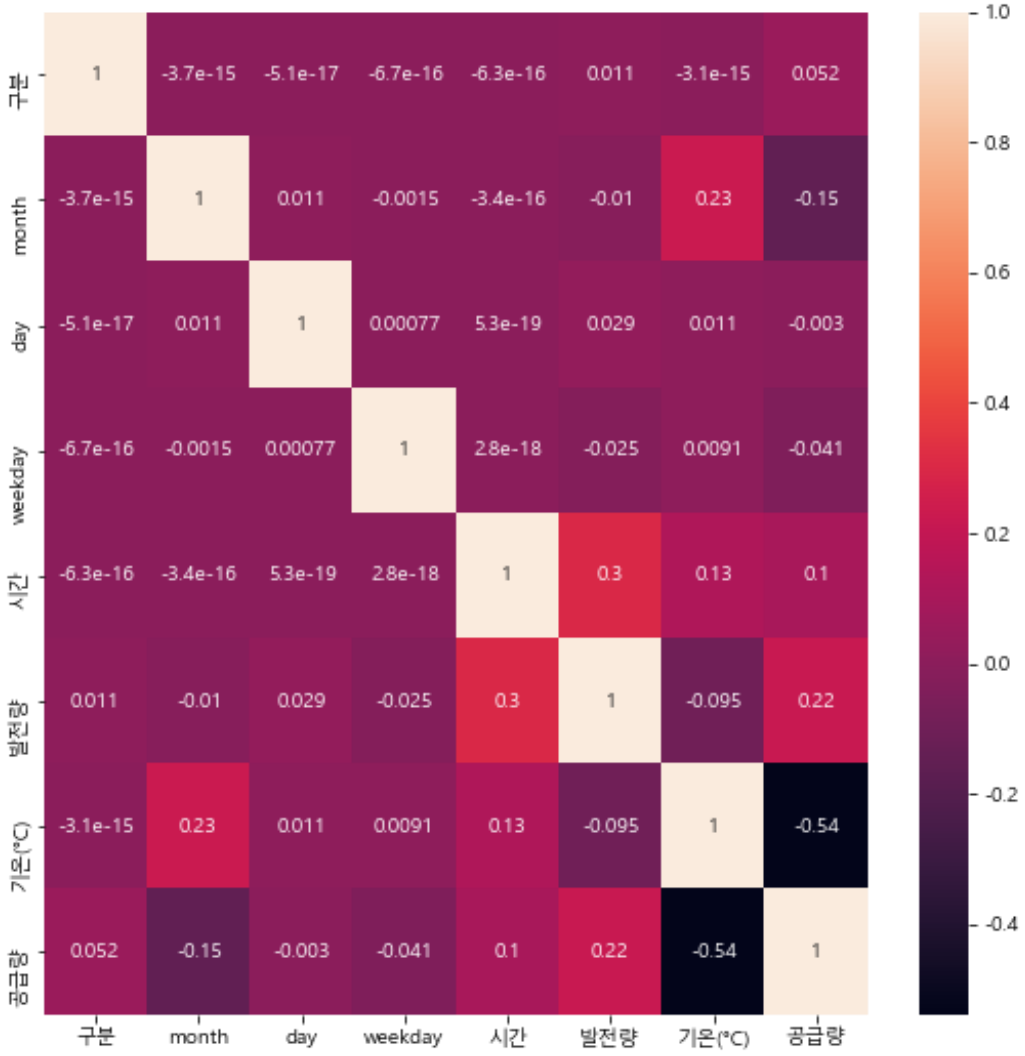
- 16년 2월 29일 데이터를 예측하기

  - pd.date_range를 이용해 시간데이터를 임의로 생성.
  - 16년 2월 29일에 해당되는 구간만 잘라내기.

  - 기존 발전량과 합치고, 잘라낸 구간 합치고. sort_index()로 다시 구간 맞추기.

  - Interpolate(method="time")을 사용하여, 시간별로 결측치 보간
  - 유의미한 결과를 가져오는 가에, baseline과 동일하게 lgbm을 사용해서 발전량의 예측량을 예측하고, 공급량 예측 모델을 생성
  - 베이스라인 점수와 비교했을 때

2 외부데이터와 가스공급량 상관계수



- 외부 데이터 **기온이 0.54 , 발전량이 0.22** 로 유의미한 특징을 찾았다.
  1차적으로 만든 lgbm 모델에서 발전량과 기온을 특징으로 선택하여 사용했을 때, 기온만을 사용한 것보다 점수가 낮게 측정되었다.

- Feature Selection으로 상관계수가 낮은 day와 weekday를 삭제하고 모델링을 하였을 때 오히려 더 점수가 떨어진 결과를 도출했다. - (적용하지 않을 예정.)

3 사용한 알고리즘에 따른 결과

•LGBM - Baseline params:

```python
params = {
'objective': 'regression',
'metric':'mae',
'seed':42
}
```

•LGBM - 튜닝한 파라미터:

```python
params = {'learning_rate': 0.01,
          'max_depth': 16,
          'objective': 'regression',
          'metric': 'mae',
          'is_training_metric': True,
          'num_leaves': 144,
          'feature_fraction': 0.9,
          'bagging_fraction': 0.7,
          'bagging_freq': 5,
          'seed':42
}
```

3 사용한 알고리즘에 따른 결과

- **Polynomial Feature (Degree = default) + StandardScaler - LGBM :**
  - 특성 개수 : 35개

```
Mean squared error:  24349.70591286092
R2 score:  0.9761128701805538
```

- **Polynomial Feature (Degree = 5) + StandardScaler - LGBM :**
  - 특성 개수: 791개

```
Mean squared error:  25284.566580018487
R2 score:  0.9751957692430968
```

- **StandardScaler - LGBM** 파라미터 튜닝했을때 **:**

```
Mean squared error:  24546.727737721496
R2 score:  0.9759195912175738
```

- **MinMaxScaler - LGBM** 파라미터 튜닝했을때 :

```
Mean squared error:  33363.985009238204
R2 score:  0.9672698370952896
```

- **Scaler** 없는 LGBM :

```
Mean squared error:  24798.451798518785
R2 score:  0.9756726492076595
```

- **Scaler** 없는 XGBoost :

```
Mean squared error:  24836.00095305641
R2 score:  0.9756358133817049
```

결론

- 1개의 알고리즘과 파라미터 튜닝으로는 마지막으로 제출한 모델과 큰 격차를 만들지 못한다.
- Scaler의 사용으로 크게 격차를 만들지는 못하였으나, StandardScaler가 의의가 있다.
- Stacking을 사용하여 차이를 본다.

3 사용한 알고리즘에 따른 결과

점수를 올리고자 **k-fold** 기반 스태킹 실행

Stacking에 사용한 모델

GradientBoosting	XGB (메타모델)
LGBM	RandomForest

💡 스태킹은 각 모델마다 파라미터 튜닝을 하는게 좋지만, 시간 관계상 하지못했다.

3 사용한 알고리즘에 따른 결과

(1)원본 train데이터를 위 4개의 모델이 학습한다.

(2)각 모델마다 test로 pred를 뽑아낸다.

(3)이 pred로 메타모델이 다시 학습데이터로 사용한다.

3 사용한 알고리즘에 따른 결과

결론

1. 메타모델을 xgboost 로 사용해서, 기존 기온데이터로 돌렸더니:
제출점수가 0.13 (80등) → 0.11대로 성능이 향상되었다. (60등)

2. 스태킹 후 생각보다 데이콘 제출점수가 높게 나오지 않았다.
    - 과대적합 되었다고 생각해서, 블랜드 기법을 사용했다.
        - 블랜딩 중 구글 Colab에서 모델생성할때 RAM이 부족해 다운되는 현상

→ 블랜딩에서 랜덤포레스트 삭제
        - 블랜딩 후 점수가 0.15대 로, 성능이 더 떨어졌다.

→ 과대적합 되지 않았다고 판단

→ 하이퍼 파라미터 튜닝을 하면 성능이 더 향상 될 것으로 예상

4 추후 개선 사항 및 자체 평가

원-핫 인코딩 - 구분(공급사)에 적용 (선형 모델을 사용하기 때문에)

Stacking - 하이퍼파라미터 조정

4 추후 개선 사항 및 자체 평가

>> 팀장 : 박지용

- 단일 알고리즘의 성능 한계로 어떤 방향으로 나아갈지 고민일 때 두호님께서 Stacking을 시도한다고 할 때 비교되는 두개의 군을 발표할 수 있는 좋은 결과를 예상했다.

- 특성을 제한적으로 사용했고, 파라미터를 완벽하게 적용하지 못한 Stacking이어서 점수의 개선이 예상된다.

>> 팀원 : 최두호

- 기온데이터와 스태킹을 사용했는데, 점수를 크게 향상할 것이라고 예상했고 실제로 점수가 올랐다. 파라미터 개선을 하면 더 높아지리라고 생각한다.

- 과정 하면서 부족함이 많았지만, 점수가 점점 오르다 보니 욕심이 생겼다. 공부를 많이 해야겠다고 느낀다.

5 시행착오

>> **~ 중간 발표**

- 데이터 결측치 처리
- 윤년이 껴있는 2016/2/29일 데이터 처리.

5 시행착오

>> ~ 최종발표

- 데이터 전처리에서 외부데이터 처리를 공급사와 같이 묶어 7번 반복된 데이터를 기반으로 예측치를 처리하였다. → 논리적으로 하루의 기온과 발전량은 1대 1 대응이므로 반복이 안된 데이터로 예측치를 새로 만들고 모델링하였다.

- 히트맵으로 보았을 때 공급사만큼 반복했을 때보다 기온에서 상관계수가 작게 나왔다.
  - 발전량 데이터에서 일정 구간이 겹치는 구간을 찾아서 재설정하였는데 변동이 없었고, 오히려 기온의 데이터의 변동으로 7번 반복된 기존의 방식의 예측 모델로 진행하기로 결정하였다.


Thanks!