**Upstage AI Lab 3기 ML 프로젝트 보충자료**

아래 보충자료 가이드를 드리니 참고하시고 해당 내용으로 작성해주시길 바랍니다.

진행하지 않은 내용은 ‘진행하지 않음’이라고 표기해주시길 바랍니다.

1. **협업방식 - 실제사용한 깃헙 컨벤션, 데이터셋 관리방법, 의사소통 및 실험기록 방법 등**

* git repo 에 개인 파일 공유
* https://github.com/fastcamp-ml-project-3/ml\_project\_3
* 10시 zoom 미팅 후 개인별 작업 진행, slack에서 중요사항 공유

1. **대회 및 데이터 개요**

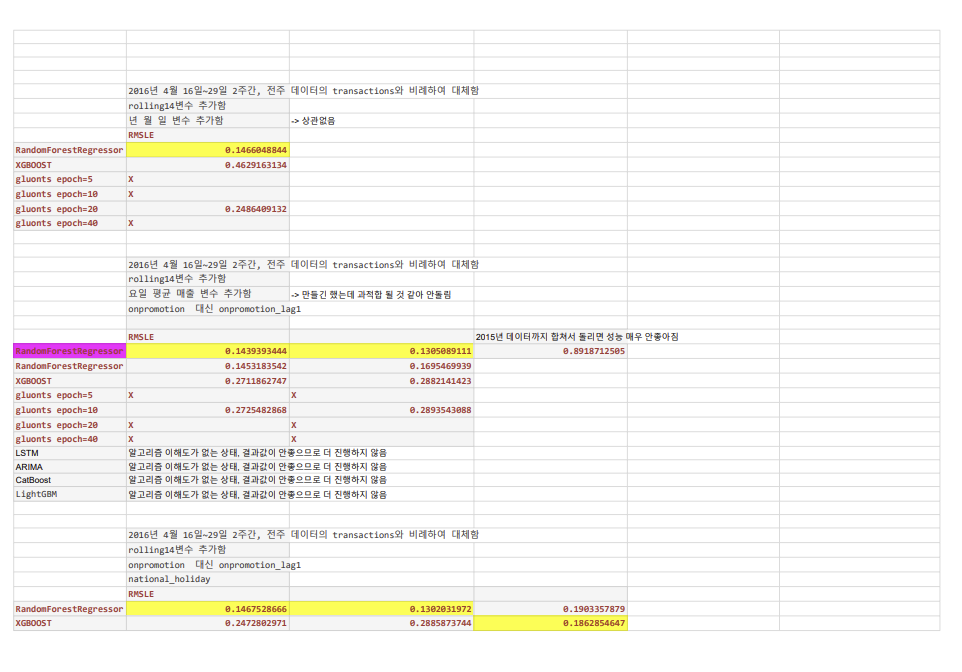
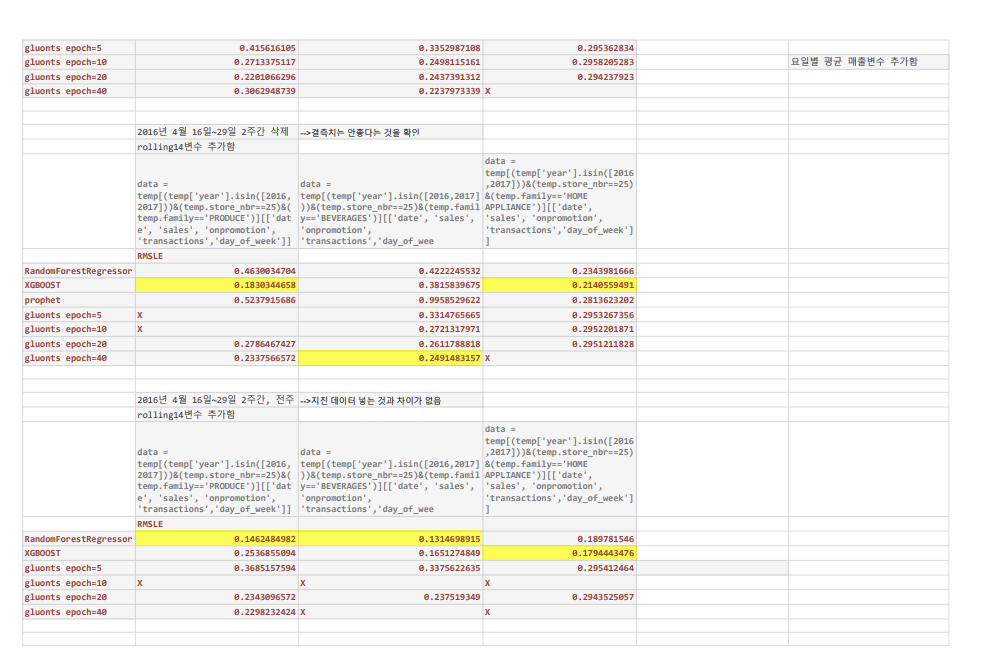
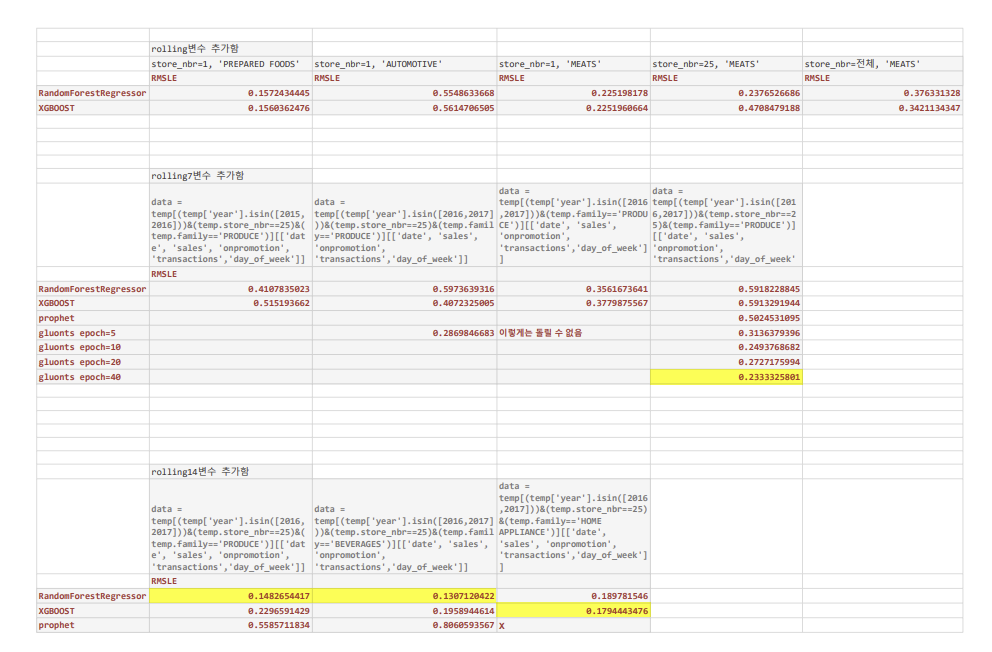
* <https://www.kaggle.com/competitions/store-sales-time-series-forecasting/overview>
* train, stores, oil, holidays\_events, transactions, test, sample\_submission 총 7개 file
* 2013년 1월 1일~2017년 8월 15일 데이터를 활용하여 8월 16일~31일 sales를 예측하는 것이 목표

1. **데이터의 이해**
   1. **EDA를 통한 인사이트**
      1. 기술 통계 및 시각화: 데이터의 분포, 중앙값, 평균, 최대값, 최소값 등의 기술 통계를 계산하고 히스토그램, 박스 플롯, 산점도, 히트맵 등의 시각화 도구를 사용하여 변수 간의 관계를 시각적으로 확인하였습니다.

* store, family별 sales 양상이 매우 다르다
* 16년 4월 16일 대지진 이후로 며칠 간 transaction에 비해 sales 가 비정상적으로 튄 곳이 많았다. 2주까지도 sales 영향이 있었다.
* 주중에서는 주말이, 연중에서는 연말에 sales 값이 크다(판매량이 많다).
* oil 가격과 영향이 있는 곳도 있었으나 그렇지 않은 곳이 더 많았다.
* wage와 sales의 영향이 크게 없었다.
  + 1. 결측치 분석: 데이터셋 내의 결측치를 확인하고, 결측치가 있는 변수와 이의 비율을 파악하였습니다. 결측치가 많은 변수는 제거하거나 적절한 대체 방법을 적용하였습니다. 다만 최종 test 의 transactions 값이 없이 예측하거나 평균값으로 대치해서 적용합니다.
    2. 변수 변환: 특정 변수의 스케일을 맞추거나 로그 변환을 통해 데이터의 분포를 정규화하는 등의 변환을 수행하였습니다.
  1. **데이터 전처리 근거와 방법**
* 이상치: 16년 4월 16일 대지진 이후 2주 간의 sales 데이터를 과거 2주간의 transaction과 비례하여 대체하였다.
* 결측치: sales가 존재하는데 transactions이 존재하지 않는 경우 과거 데이터에 비례하여 대체하였다.
* oil의 경우 인터폴레이트 방식으로 결측치를 처리하였습니다.
* **범주형 변수 인코딩**: 원-핫 인코딩(one-hot encoding)으로 변환하였습니다.
* **스케일링**: 변수들 간의 스케일 차이를 맞추기 위해 표준화(Standardization)를 적용 하였습니다.
  1. **Feature Engineering 근거와 방법**
* seasonal: year, month, day, day\_of\_week, season 등을 만들어 효과가 좋은 day\_of\_week를 선택
* rolling 변수 세팅 방법 I : sales의 주간 변화를 반영하기 위해 skew, kurt, mean, min, max등의 값을 적용 window(7, 14)
* rolling 변수 세팅 방법 II : sales의 이동평균선을 14일 28일로 평균값과 분산값을 적용 window(14, 28)
* lag: onpromotion의 경우 앞 뒤로 10일 lagging한 결과 하루 지난 경우가 sales에 영향을 미쳤다고 판단하여 1일 뒤로 셋팅
* corr heatmap 을 통해 transactions, onpromotion의 영향도가 높은 것을 확인하였다.

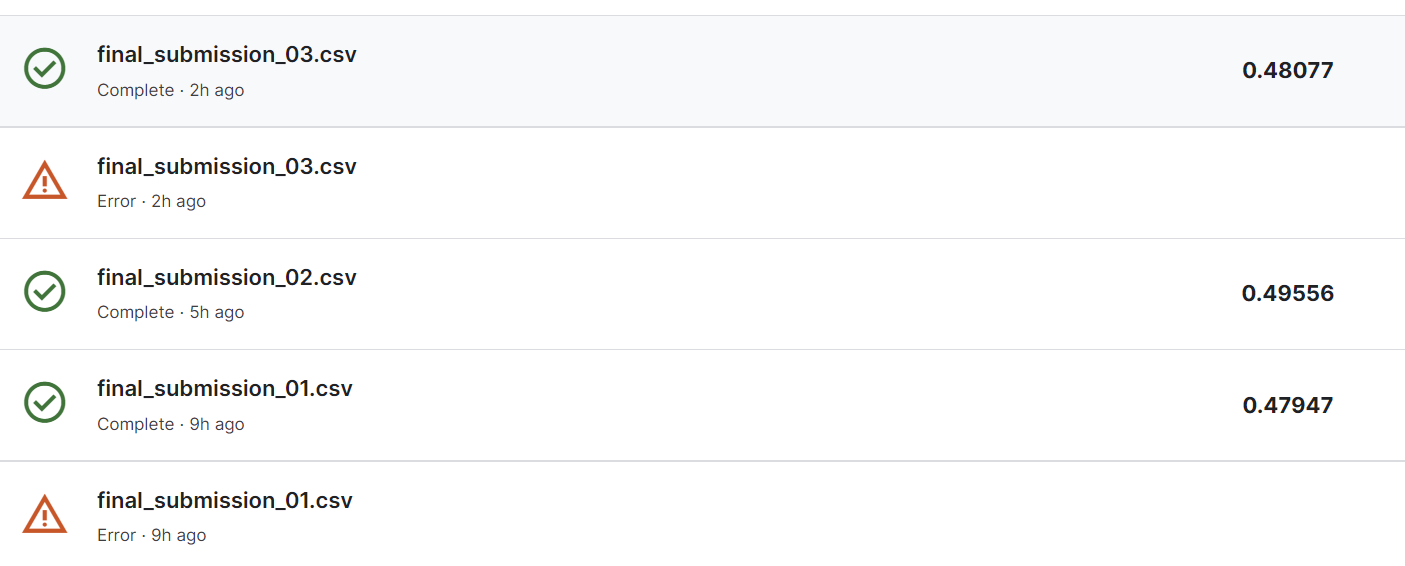
1. **모델링**
   1. **선택한 모델과 선택한 이유**

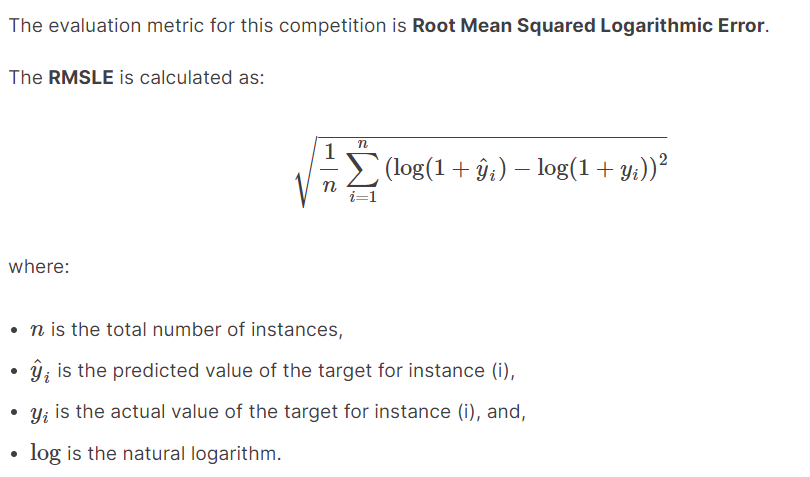
* randomforestregressor은 가장 낮은 0.14 대의 결과값을 보여줬다.
* xgboost은 randomforestregressor 보다 높은 결과값을 보여줬다.
* gluonts은 어떠한 모델 변화에도 안정적인 0.2대의 결과값을 보여줬다.
* prophet은 결과값의 변동이 심했다.

기타 다른 모델의 결과값이 좋지 않아 randomforestregressor로 결정하였다.

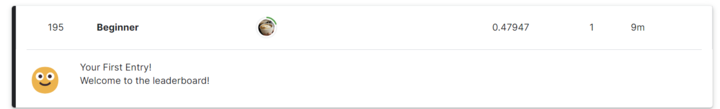
* 1. **Hyper Parameter Tuning 방법**
     1. **이번 분석에서는 GridSearchCV를 사용하여 최적의 하이퍼파라미터를 탐색하였습니다. GridSearchCV는 하이퍼파라미터의 조합을 체계적으로 탐색하여 교차 검증을 통해 최적의 파라미터를 선택합니다. 주요 튜닝 파라미터는 다음과 같습니다:**
        1. **MODEL 1: n\_estimators: 결정 트리의 개수(50부터 120까지 차등적으로 적용하여 제일 성능이 좋은 100-120 설정하였습니다.**
* 각 모델들마다 변경할 수 있는 파라미터의 값을 증감하며 테스트를 진행하였다.
* 대회 평가 산식인 rmsle값이 가장 낮은 것으로 선택하였다.

1. **평가**
   1. **Insight 1 : 이윤재**
      1. **해당 대회의 평가 산식과 실제 대회를 진행한 결과에서 나온 인사이트를 작성해주세요.**

* **[sales예측이니까 음수값은 전부 0으로 대체해야 되는거 아니야?]**
* RMSLE는 예측값과 실제값의 차이를 로그 변환하여 계산하는 지표로, 예측값과 실제값 모두의 로그를 취한다. 로그의 특성상 0의 로그는 정의되지 않으므로, 일반적으로 1과 같은 작은 상수를 더하여 계산한다. 예를 들어, log⁡(1+0)\log(1 + 0)log(1+0)와 log⁡(1+0.0005)\log(1 + 0.0005)log(1+0.0005)는 큰 차이가 있다.
* 예측값을 0으로 대체하면 실제 판매량이 0이 아닌 경우 로그 차이가 크게 증가한다.
* RMSLE는 과소 예측에 대해 더 큰 페널티를 부과한다. 예를 들어, 실제 판매량이 0.1인 경우 0으로 예측하면 큰 페널티를 받는다. 반면, 0.0005로 예측하면 페널티가 상대적으로 적다.
* 따라서 실제 sales에서는 0보다 작은 값은 있을 수 없기에 음수 값을 0으로 대체하거나 threshold를 0.1 값 등으로 대체하는 경우 submission결과가 더 낮아졌다. 0.49447 → 0.49556 → 0.48077
* 



* **[Colab GPU 에서 코드 돌리면 그냥 엄청 빨라지는 거 아니였어?]**
* 각 store, family마다 sales 양상이 다르기 때문에 전체를 두고 모델을 만들면 안된다고 생각하여 54 x 34 = 1836 번을 모델을 돌려야 했다. 그러나 Intel i7, 16GB RAM, 64-bit OS 에서 총 54시간 이상 걸리는 상황이라 colab 결제하고 T4 GPU로 돌려보았다. 그러나 gluonts의 경우 해결하기 어려운 에러를 만났고, colab cpu환경에서도 같은 코드임에도 로컬에서 만나지 못한 에러를 만나게 되었다.
* TypeError: cannot pickle 'generator' object
* ValueError: Input X contains NaN. RandomForestRegressor does not accept missing values encoded as NaN natively.
* colab의 경우 버젼셋팅이 로컬과 다른 경우가 많고 버젼들끼리 충돌하는 경우도 많았다. 또한 GPU셋팅 환경은 녹록치 않았다. 결국 $11달러를 결제했지만 GPU를 활용하지 못했다.
* **[모델 만들기 전에 test셋 컬럼 확인 안하면 X된다.]**
* 우리는 train 데이터를 학습하여 모델을 만들고 test 데이터에 merge하여 제출하면 끝이라고 생각했는데, 실제 test 데이터에는 sales를 예측할 수 있는 가장 중요한 transactions값이 없었다. 우리는 transactions를 정확하게 예측하는 것이 이 대회의 관건이라는 것을 프로젝트 마감기한 며칠 전에 알게 되었다. 어쩔 수 없이 transactions 를 쉽게 예측해주는 gluonts, prophet 같은 모델을 사용하여 간단하게 transaction을 예측하였다.
* 그러나 또 하나의 문제는 rolling 변수였다. rolling변수는 sales를 토대로 산출해내는데 sales를 예측하는 상황에서 rolling변수를 만들 수는 없었다. 따라서 sales 값 역시 gluonts로 빠르고 편리하게 예측하여 rolling변수를 만들었다. 이렇게 transactions, sales 변수를 만든 상태에서 rolling변수를 만들고 randomforestregressor 로 모델을 산출하였다.
* **[최종 결과 및 향후 개선 방안]**
* 결과는 0.47947 이다. 1등이 0.37768 이니까 26.94%의 차이가 있다. 이는 상당히 큰 차이로 아직 모델 개선이 필요해 보인다. 현재 모델에서 수정해야 할 부분은 transactions, sales예측 부분으로 이 부분을 고도화한 뒤 submit해 볼 계획이다.



* 1. **Insight 2 : 조용중**
     1. **해당 대회의 평가 산식과 실제 대회를 진행한 결과에서 나온 인사이트를 작성해주세요.**

대회 평가 산식 으로는 RMSLE(Root Mean Squared Log Error)가 사용되었습니다. RMSLE는 예측값과 실제값의 로그 차이를 기반으로 한 오차 측정 방법으로, 예측값이 실제값과 얼마나 비슷한지를 평가합니다. RMSLE의 장점은 예측값이 실제값보다 작을 때보다 클 때 더 큰 페널티를 부과하여 모델의 예측이 과도하게 높은 것을 방지하는 데 유용할것으로 생각됩니다.

대회를 통해 얻은 주요 인사이트는 다음과 같습니다:

1. **데이터 전처리의 중요성**: 결측치 처리, 스케일링, 변수 인코딩 등 전처리 과정이 모델의 성능에 큰 영향을 미친다는 것을 확인할 수 있었습니다.
2. **Feature Engineering의 효과**: 새로운 변수를 생성하고 불필요한 변수를 제거하는 Feature Engineering 과정이 모델의 성능을 크게 향상시키며, 이것을 반복하는 것이 필요합니다.
3. **하이퍼파라미터 튜닝의 필요성**: 적절한 하이퍼파라미터 튜닝을 통해 모델의 성능을 최적화할 수 있었습니다.
4. 가장 중요하게 초기부터 test 와 sample\_submission을 확인하는 것이 중요하다고 생각합니다.
5. 초기에 결과를 내는데 급급하여 많은 부분을 미스한것 같아 별로 좋은 결과를 얻지 못함
6. 향후 추가로 차원축소나 DL 알고리즘으로 추가하여 지속적으로 등수를 올려볼 생각입니다.

