

주차수요 예측 AI 경진대회

AutoML 라이브러리 pycaret을 이용한 머신러닝

- AutoML이란?

머신러닝 모델 개발, 배포등을 자동으로 처리할 수 있게 도와주는 프로세스
구글의 vertex ai, sagemaker autopilo 등의 클라우드 서비스와 파이썬 라이브러리가 있음.

- Pycaret

python의 AutoML 라이브러리

1. 데이터 준비 (setup)

데이터프레임으로 로드된 데이터를 머신러닝에 사용할 수 있도록 로드 및 전처리 하는 기능.
파라미터로 전처리 적용 기법 변경 가능.

2. 모델 생성 및 비교

setup 이후, 머신러닝 모델을 선언해서 사용하거나, 전처리한 데이터셋에 맞는 모델을 비교해 볼 수 있음.

- model(): 각 머신러닝 기법에 따라 구현된 모델을 나열
- compare_models(): setup된 데이터를 각각 머신러닝 모델에 적용 후 비교.
- create_model(): model()에 적힌 머신러닝 모델을 선택해서 생성

3. 모델 최적화

- tune_model(): 모델의 하이퍼파라미터를 최적화하는 모듈. 하이퍼파라미터 반복 횟수나 최적화할 매트릭을 선택할 수 있음.
- ensemble_model(): ensemble 기법을 구현한 모듈. bagging과 boosting을 파라미터에서 선택할 수 있음.
- blend_models(): voting 알고리즘을 구현한 모듈. compare_models()에서 성능이 잘 나온 모델들을 선택하는 파라미터를 적용(n_select)시켜서 사용할 수 있다.
- stack_models(): staking ensemble 방법을 구현한 모듈. compare_models()에서 성능이 잘 나온 모델들을 선택하는 파라미터를 적용시켜서 사용할 수 있음.

4. 학습된 모델 분석

- interpret_model(): 모델이 예측한 결과에 대해서 각 파라미터들이 얼마나 영향을 줬는지 시각화해서 보여줌.

- `assign_model()`: unsupervised 류의 머신러닝 기법에 대한 머신러닝 결과값을 데이터셋에 부여해줌.
- `evaluate_model()`: 모델 분석 후 각 플롯을 볼 수 있도록 사용자 인터페이스 제공
- `get_leaderboard()`: setup 이후 훈련된 모든 모델 출력
- `eda()`: `auto_viz`를 사용해 데이터셋 분석 결과를 제공.

수상자 코드

```
reg = setup(data=train_data,
            target='주차면수대비등록확률',
            session_id = 201,
            numeric_imputation = 'mean',
            fold_shuffle = True,
            numeric_features=list(train_data.drop(columns = ['주차면수대비등록확률']).columns),
            ignore_low_variance = True,
            combine_rare_levels = True, rare_level_threshold = 0.05,
            remove_multicollinearity = True, multicollinearity_threshold = 0.90,
            normalize = True,
            silent= True)

blended_l = blend_models(estimator_list = best_5_l, fold = 5, optimize = 'MAE')
pred_holdout = predict_model(blended_l)
final_model_l = finalize_model(blended_l)
pre_esb_l = predict_model(final_model_l, test_data)
```

AutoGluon-Tabular

1. 테이블 데이터를 자동으로 학습하여 높은 성능을 발휘하는 AutoML 프레임워크
2. 기존 AutoML 프레임워크 모델이 하이퍼파라미터의 선택을 중시하는 반면, 여러 레이어를 사용해 모델의 앙상블과 스택킹을 실시

fit API

train.csv의 예측 라벨이 class 열에 있는 경우

```
from autogluon import TabularPrediction as task #AutoGluon의 로딩
predictor = task.fit("train.csv", label="class") #학습
predictions = predictor.predict("test.csv") #테스트 데이터에 대한 예측
```

위의 코드만으로 태스크가 분류인지 회귀인지, 분류라면 어떤 클래스의 분류인지를 자동으로 판정하고 다양한 모델을 앙상블 시켜 강력한 모델을 학습시킨다.

데이터 전처리

먼저 학습 데이터가 어떤 작업을 위한 것인지 자동으로 판별한다. target 변수가 포함된 열을 참조하고 그 값이 문자열이나 이산 값이면 분류 작업으로, 연속 값이면 회귀 작업이라고 판단한다.

이후, 모든 모델에 공통되는 처리를 실시한 후 각 모델에 적합한 처리를 적용한다. 분류 모델인 경우 카테고리, 숫자, 문자, 시간 등의 데이터가 모두 범주형 변수로 변환된다. 또한 결손치는 unknown으로 라벨이 부여되기에 학습 시 관측되지 않은 변수값이 나타나도 오류가 발생하지 않는다.

모델

AutoGluon-Tabular는 처음부터 이용할 모델을 정해두고 사용함.

→ 불필요한 모델 탐색 x, 제한된 시간 내 효율적인 학습 가능

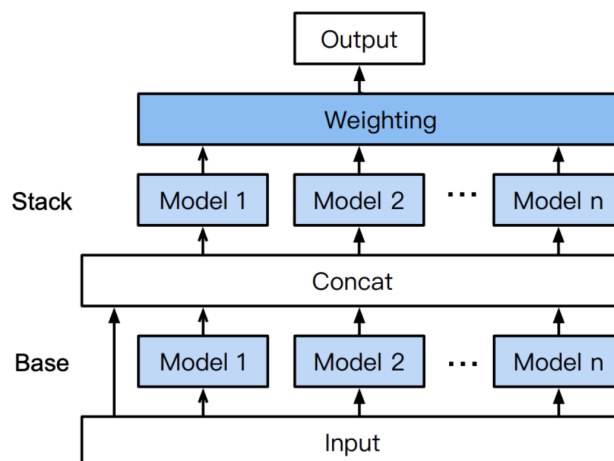
사용하는 알고리즘은 다음과 같음.

- 신경망
- LightGBM(부스팅 트리 일종)
- CatBoost(부스팅 트리 일종)
- 랜덤 포레스트
- ERT(랜덤 포레스트의 변종)
- k nearest 방법

Multi-Layer Stak Ensembling

AutoGluon-Tabular는 스택킹 앙상블 사용.

→ 여러 모델의 출력의 가중값을 최종 출력하는 것.



1. n개의 독립적인 모델(6개의 알고리즘에서 선택)에 전처리를 한 데이터(input)를 입력
2. 1의 각 모델의 출력과 input을 결합시켜 1과 같은 알고리즘, 같은 하이퍼매개 변수 모델에 입력
3. 2의 각 모델의 출력의 가중값을 최종 출력으로 하여, 오차를 계산하여 학습

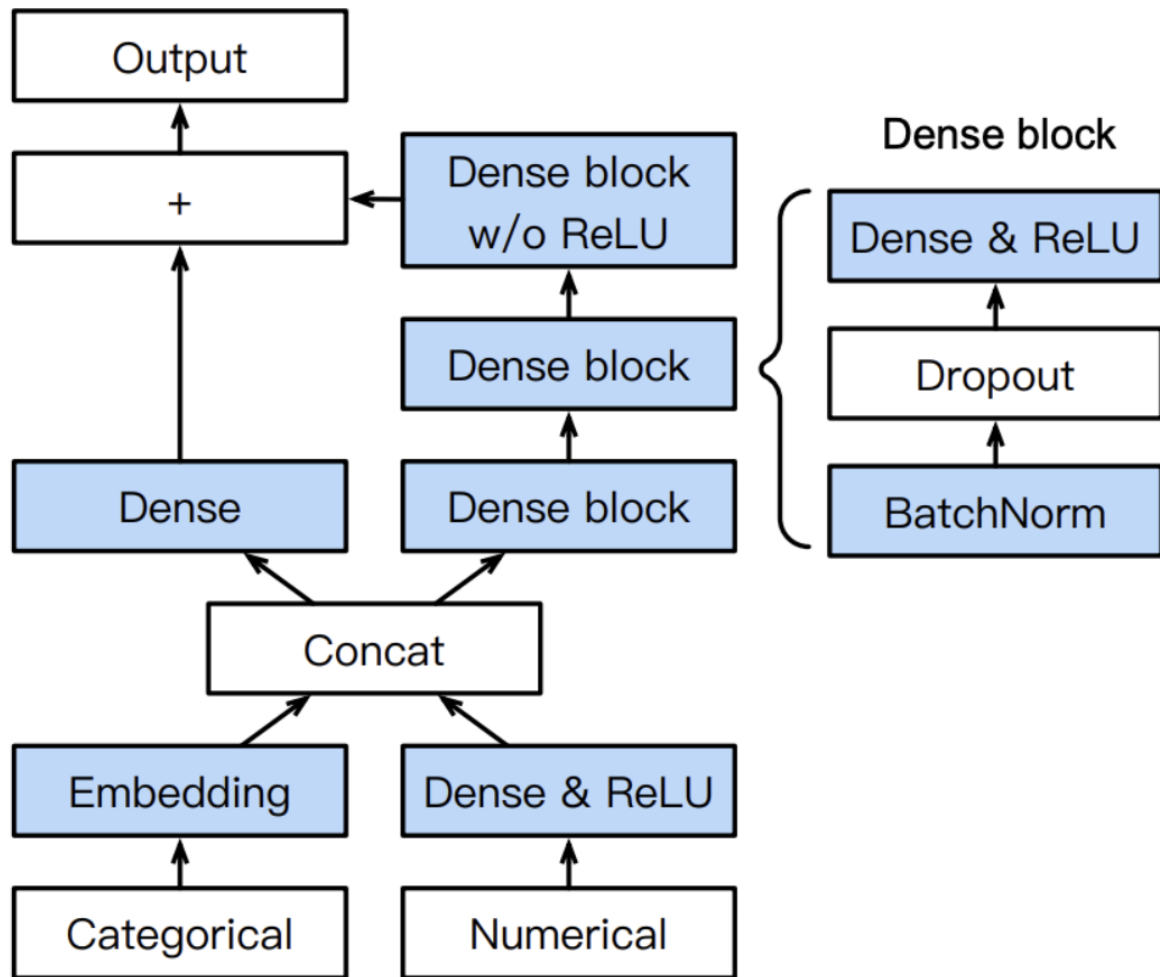
기존 AutoML의 스택킹은 단순히 다른 모델 출력의 가중값을 계산 (단층 스택킹)

AutoGluon-Tabular에서는 각 모델의 출력과 원래의 입력을 스킵 커넥션으로 결합하고 그것을 각의 모델에 입력하여 스택킹 (다층 스택킹)

→ 이미지 인식 작업에서 고성능을 발휘하고 있는 ResNet과 MobileNet 등의 구조(심플한 구조층 단방향 쌓기 + 스킵 커넥션)의 응용

신경망

테이블 데이터는 각각의 열에 성별 · 연령 · 체중 · 신장 등 의미 있는(=각각의 성질이 다른) 변수가 포함되어 있기 때문에 **다양한 변수를 선형합으로 혼합해 버리는 신경망보다 결정 트리계의 알고리즘이 더 테이블 데이터의 패턴 인식에 적합할 수 있음.**



카테고리 데이터와 수치 데이터를 정리해 Dense 층에 넣는 것이 아니라, 카테고리 데이터는 Embedding 층에 채우고, 수치 데이터는 Dense 층을 통과시켜 ReLU에서 활성화한 것을 결합함.

→ 카테고리 수치 데이터가 선형합으로 혼합되기 전, 두 특징량을 신경망이 별도로 학습할 수 있음!

→ 스킵 커넥션을 이용하여 원래 테이블 데이터의 성질을 잊지 않게 하는(각 변수를 혼합해 버리는 것을 방지하는) 효과

Repeated k-fold Bagging

k-fold를 지정한 횟수(n회)만큼 반복하는 방법

Algorithm 1 AutoGluon-Tabular Training Strategy
(multi-layer stack ensembling + n -repeated k -fold bagging).

Require: data (X, Y) , family of models \mathcal{M} , # of layers L

```

1: Preprocess data to extract features
2: for  $l = 1$  to  $L$  do {Stacking}
3:   for  $i = 1$  to  $n$  do { $n$ -repeated}
4:     Randomly split data into  $k$  chunks  $\{X^j, Y^j\}_{j=1}^k$ 
5:     for  $j = 1$  to  $k$  do { $k$ -fold bagging}
6:       for each model type  $m$  in  $\mathcal{M}$  do
7:         Train a type- $m$  model on  $X^{-j}, Y^{-j}$ 
8:         Make predictions  $\hat{Y}_{m,i}^j$  on OOF data  $X^j$ 
9:       end for
10:    end for
11:  end for
12:  Average OOF predictions  $\hat{Y}_m = \{\frac{1}{n} \sum_i \hat{Y}_{m,i}^j\}_{j=1}^k$ 
13:   $X \leftarrow \text{concatenate}(X, \{\hat{Y}_m\}_{m \in \mathcal{M}})$ 
14: end for

```

셋째 줄에서 시작되는 for문을 보면, 다층 적층 모델의 각 계층의 학습을 시작할 때마다 무작위로 데이터를 k 분할 하도록 작성되어 있음. 결국 하나의 층의 출력은 각각의 fold 출력의 평균이 되고 있음.

→ Repeated k -fold Bagging을 이용하여 fold에 포함된 데이터가 적은 경우에도 **모델의 과학습을 방지**

AutoML 프레임 워크의 비교

- Auto-WEKA : AutoML의 원조. Java의 기계 학습 라이브러리인 WEKA에 대한 CASH를 베이스 최적화를 통해 해결하고 있습니다.
- Auto-Sklearn : 익숙한 Python의 기계 학습 라이브러리 scikit-learn에 대한 CASH를 해결합니다. 하이퍼 매개 변수의 탐색에 메타 학습을 이용하고 있는 것이 특징입니다.
- TPOT : 진화 알고리즘을 사용하여 기계 학습 과정을 최적화하는 도구입니다.
- H2O AutoML : 특히 Kaggle 공모전에서 자주 사용되는 라이브러리입니다. 하이퍼 매개 변수 탐색에 무작위 검색을 사용하고 있음에도 불구하고 종종 다른 라이브러리보다 좋은 성적을 내는 점이 흥미롭습니다.
- GCP-Tables : GCP(Google Cloud Platform)에 발표된 AutoML 도구에서 원시 데이터를 업로드하면 쉽게 기계 학습이 가능합니다. 그러나 Google Cloud에서만 사용할 수 있으며, 오픈 소스가 아니기 때문에 내부는 알 수 없는 것이 많습니다.
- 기타(오픈 소스) : auto-egboost, GAMA, hyperopt-sklearn, OBOE, Auto-Keras
- 기타(상용 소프트웨어) : Sagemaker, AutoPilot, Azure ML, H2O Driverless AI, DataRobot, Darwin AutoML

key point

- 다양한 변수를 포함한 테이블 데이터를 처리할 수 있는 견고한 데이터 처리 기술.
- 선진적인 신경망을 이용
- 다층 스택킹과 repeated k-fold bagging에 의한 강력한 앙상블 학습


Reference

Deploy

MLOps and deployment related functions in PyCaret

 PyCaret Official

Deploy

 <https://pycaret.gitbook.io/docs/get-started/functions/deploy>

 Powered By GitBook

<https://velog.io/@gyounghwan1002/python-AutoML라이브러리-pycaret-사용법>

<https://velog.io/@devseunggwan/Machine-Learning-pycaret을-사용한-데이터-분석>

<https://dooooob.tistory.com/110>