머신러닝과 딥러닝을 활용한

국내 병원 개/폐업 예측

인천대학교

과목	데이터 사이언스
담당 교수	성미영
학번	201600779 201700255
학과	김영민: ㈜경제학과, (복)컴퓨터공학부 강민정: ㈜수학과, (복)컴퓨터공학부
이름	김영민,강민정

ABSTRACT

This project forecasts the opening/closing of domestic hospitals through accounting data for hospitals. The prediction process is performed according to the (1) KDD analysis procedure, and the characteristics of the data are identified through the visualization of the EDA, and (2) the performance of the machine learning model and the artificial neural network model is compared, thereby obtaining higher accuracy.

본 프로젝트는 병원에 대한 회계 데이터를 통해 국내 병원 개/폐업 예측한다. 예측하는 과정은 (1) KDD 분석 절차에 따라 진행되며 EDA의 시각화를 통해 데이터의 특징을 파악하고 (2) 머신러닝 모델과 인공 신경망 모델의 성능을 비교하고 이를 통해 보다 높은 정확도를 얻어낸다.

Ⅰ. 서론

2020년 현재 코로나가 사회 전반에 흐름을 망가뜨리면서 소상공인의 경제가 심각해지고 있다. 이에 따라 모든 산업 전반에 생산량이 감소하고 있으며 그들의 생계에도 어려움이 생기는 상황이다. 이 상황은 병원도 예외가 아니다. 전염병이라는 특성 때문에 개인 병원에 가기 꺼려하는 상황이 발생하여 병원의 폐업도 늘어나고 있다.

이러한 문제 인식을 인식하고 본 팀은 MOUDA가 데이터 분석 대회인 DAICON에 제공하는 데이터를 활용해 병원의 개/폐업을 예측하려 한다. 병원의 개/폐업 예측을 통해 다른 산업 분야에도 적용하여 사업의 존망을 예측할 수 있다. 또한 폐업의 중요 변수를 고려하여 기존에 사업을 하는 사람들에게 유의점을 줄 수 있다.

Ⅱ. 관련 연구 및 참고 문헌

본 연구는 딥러닝을 활용하기 때문에 한빛미디어 출판의 파이토치 첫걸음(저: 최건호)책 [1] 을 참고하였다.

Ⅲ. 본론

1) 데이터

먼저 데이터의 구성요소는 train data, test data로 나뉘어져있다. Train data는 301개의 행과 58개의 열로 구성되어있고, test data는 127개의 행과 58개의 열로 이루어져있다. 데이터의 타입은 float type 64개, object type 4개, int64 type 4개로 구성되어있다. 또한 이 데이터는 주로 회계데이터가 구성되어 있다.

데이터의 전처리는 결측치 처리, 파생변수 생성, 변수변환, 변수 삭제의 순서로 진행하였다.

(1) 결측치처리

openYear = 결측치는 중위값으로 대체하였다.

instkind와 bedcount의 결측치는 각 병원 종류 별로 bedcount의 이상치가 많아서 instkind로 그룹을 묶은 다음에 bedcount의 중위값을 선택하여 대체. 두 변수 모두 결측치인 경우는 다른 변수를 고려하여 모델링하여 대체하였다.

employee 직원 수와 상관관계가 높은 변수들로 모델링을 하여 결측치 처리하였다.

회계 데이터도 모델링으로 예측하여 결측치 처리하였다.

(2) 파생변수 생성

Profit_diff, debt_diff, surplus_diff, netasset_diff 컬럼을 생성해 2016년과 2017년의 회계 데이터의 변화를 나타내는 컬럼을 생성하였다.

Old 변수는 2018년 기준으로 오픈한지 얼마나 지났는지를 알 수 있게 하였다.

직원수의 변화 차이도 알기 위해 employee_change 컬럼을 생성하였다.

(3) 변수 변환

Sido 컬럼은 도시의 특성에 맞게 변화시켜서 유니크 값을 줄인 후 원핫인코딩을 하였다. 또한, inst_kind 변수도 원핫 인코딩을 하였다. ownerChange는 같으면 1, 다르면 0으로 라벨 인코딩 하였다. 또한, 부채 같은 변수들은 음수 처리 하였다.

(4) 변수 삭제

Index나 오픈 연도, 시군구 코드는 의미 없는 변수라 판단하여 삭제하였다.

2) 실험

위와 같은 전처리 과정을 마치고 모델링을 하였다. 모델링은 연구의 목적인 머신러닝과 딥러닝 중 일반 인공신경망의 예측 정확도 차이를 알기 위해서 같은 전처리를 한 데이터에 머신러닝과 인공신경망 모델을 적용시켰다.

처음으로 머신러닝 모델 중 앙상블 모델인 RandomForestClassifer 와 부스팅 모델인 XGBoostClassifier를 선택하여 모델링을 하였다. 주 모델이 머신러닝 모델이 아니므로 하이퍼파라미터 조절은 하지 않았다. 그 결과 RandomForestClassifier 모델은 약 84퍼센트의 정확도가 측정되었고, XGBoostClassifier모델은 약 85퍼센트의 정확도가 측정되었다.

다음으로 인공신경망 모델은 기본으로 epoch = 10000, hidden_layer = 4 로 설정하고 모델을 적용시켰다. 이는 valid data를 생성하여 즉, train data의 20퍼센트를 검정용 데이터로 생성하여 검정을 먼저 실시하고 최선의 모델을 선택하였다. 최선의 모델은 learning_rate와 activation function의 변화를 주어서 정확도가 가장 높은 모델을 선택하는 방식으로 하였다.

다음은 인공신경망 모델이다.

① ReLU - ReLU - ReLU – LeakyReLU

```
[106]  1 model = model_RRRLR()
       2 print(model)
```

```
Sequential(
  (0): Linear(in_features=73, out_features=438, bias=True)
  (1): ReLU()
  (2): Linear(in_features=438, out_features=730, bias=True)
  (3): ReLU()
  (4): Linear(in_features=730, out_features=438, bias=True)
  (5): ReLU()
  (6): Linear(in_features=438, out_features=73, bias=True)
  (7): LeakyReLU(negative_slope=0.01)
  (8): Linear(in_features=73, out_features=2, bias=True)
)
```

② ReLU - ReLU - ReLU - ReLU

```
[107]  1 model = model_RRRR()
       2 print(model)
```

```
Sequential(
  (0): Linear(in_features=73, out_features=438, bias=True)
  (1): ReLU()
  (2): Linear(in_features=438, out_features=730, bias=True)
  (3): ReLU()
  (4): Linear(in_features=730, out_features=438, bias=True)
  (5): ReLU()
  (6): Linear(in_features=438, out_features=73, bias=True)
  (7): ReLU()
  (8): Linear(in_features=73, out_features=2, bias=True)
)
```

③ ReLU - ReLU - ReLU – Tanh

```
[108]  1 model = model_RRRT()
       2 print(model)
```

```
Sequential(
  (0): Linear(in_features=73, out_features=438, bias=True)
  (1): ReLU()
  (2): Linear(in_features=438, out_features=730, bias=True)
  (3): ReLU()
  (4): Linear(in_features=730, out_features=438, bias=True)
  (5): ReLU()
  (6): Linear(in_features=438, out_features=73, bias=True)
  (7): Tanh()
  (8): Linear(in_features=73, out_features=2, bias=True)
)
```

④ Sigmoid - Sigmoid - Sigmoid - Sigmoid

* sigmoid 함수를 사용할 때에는 학습률을 조금 더 크게 설정하였다.

```
[109]  1 model = model_SSSS()
       2 print(model)
```

```
Sequential(
  (0): Linear(in_features=73, out_features=438, bias=True)
  (1): Sigmoid()
  (2): Linear(in_features=438, out_features=730, bias=True)
  (3): Sigmoid()
  (4): Linear(in_features=730, out_features=438, bias=True)
  (5): Sigmoid()
  (6): Linear(in_features=438, out_features=73, bias=True)
  (7): Sigmoid()
  (8): Linear(in_features=73, out_features=2, bias=True)
)
```

⑤ Tanh – Tanh – Tanh- Tanh

```
[110]  1 model = model_TTTT()
       2 print(model)
```

```
Sequential(
  (0): Linear(in_features=73, out_features=438, bias=True)
  (1): Tanh()
  (2): Linear(in_features=438, out_features=730, bias=True)
  (3): Tanh()
  (4): Linear(in_features=730, out_features=438, bias=True)
  (5): Tanh()
  (6): Linear(in_features=438, out_features=73, bias=True)
  (7): Tanh()
  (8): Linear(in_features=73, out_features=2, bias=True)
)
```

⑥ LeakyReLU- LeakyReLU- LeakyReLU- LeakyReLU

```
[111]  1 model = model_LRLRLRLR()
       2 print(model)
```

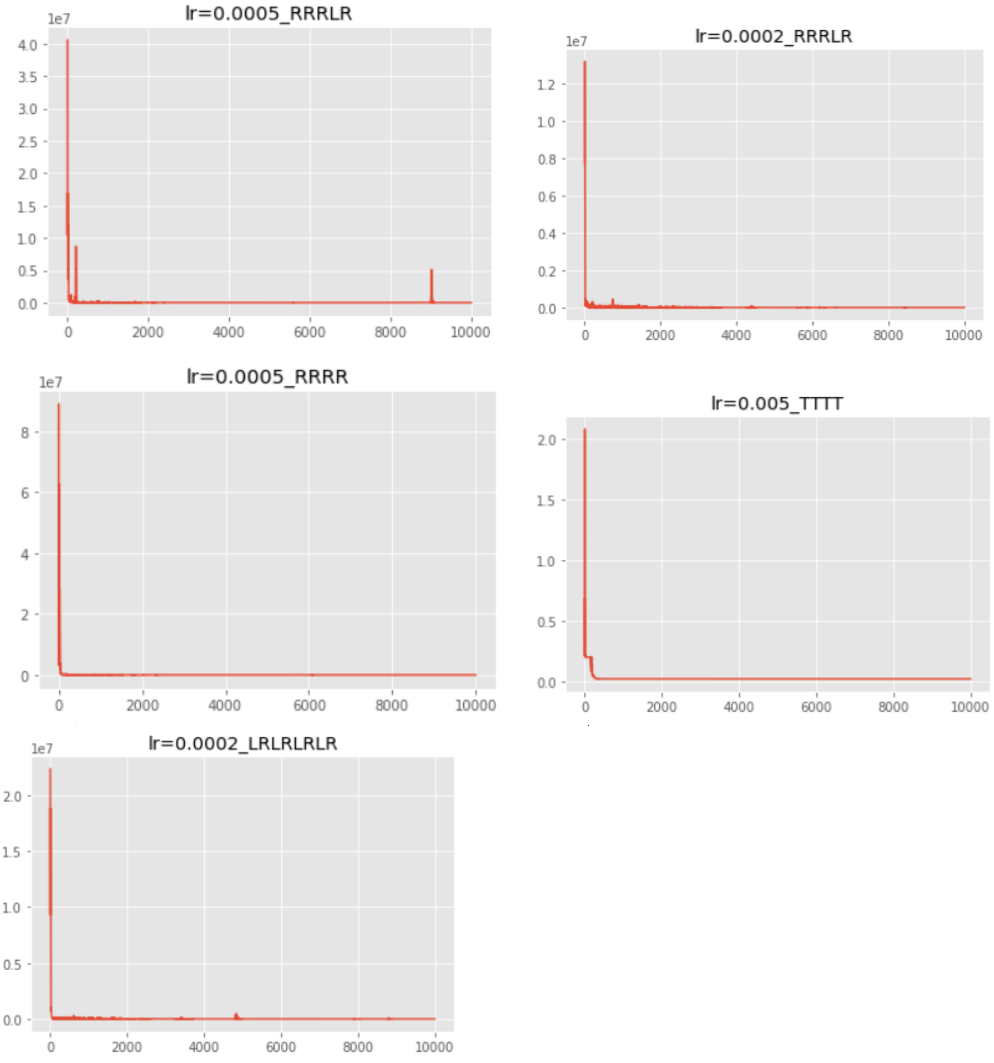
```
Sequential(
  (0): Linear(in_features=73, out_features=438, bias=True)
  (1): LeakyReLU(negative_slope=0.01)
  (2): Linear(in_features=438, out_features=730, bias=True)
  (3): LeakyReLU(negative_slope=0.01)
  (4): Linear(in_features=730, out_features=438, bias=True)
  (5): LeakyReLU(negative_slope=0.01)
  (6): Linear(in_features=438, out_features=73, bias=True)
  (7): LeakyReLU(negative_slope=0.01)
  (8): Linear(in_features=73, out_features=2, bias=True)
)
```

위와 같은 6 개의 모델을 학습률을 조정하여 최적의 모델을 선택할 것이다.

	RRRLR	RRRR	RRRT	TTTT	LRLRLRLR		SSSS
0.005	0.93	0.95	0.86	0.967	0.93	0.05	0.91
0.002	0.95	0.91	0.901	0.934	0.93	0.02	0.91
0.0005	0.967	0.967	0.901	0.95	0.95	0.005	0.81
0.0002	0.967	0.95	0.868	0.93	0.967	0.002	0.91

위는 모델링을 한 후 검정용 데이터로 정확도를 측정한 결과이다. 이와 같은 결과를 바탕으로 최적의 모델은 learning_rate = 0.0005 일 때 ReLU-ReLU-ReLU-LeakyReLU , ReLU- ReLU- ReLU-ReLU, learning_rate = 0.0002 일 때, ReLU-ReLU-ReLU-LeakyReLU, LeakyReLU - LeakyReLU - LeakyReLU - LeakyReLU , learning_rate = 0.005 일 때, Tanh – Tanh – Tanh – Tanh 일 때가 가장 최적의 모델이다. 따라서 위 5 개의 모델로 test를 시행하였다.

이 test로 적용한 손실그래프는 다음과 같다.



이 손실 그래프의 accuracy_score를 측정한 결과 학습률 0.005, 활성화 함수 하이퍼볼릭탄젠트 함수 4 번이 85%로 가장 좋았다.

Ⅳ. 결론

머신러닝(randomforest, xgboost)와 딥러닝의 정확도는 거의 비슷하였다. 머신러닝의 변수 중요도로 보아 병원이 얼마나 오래됐는지가 가장 중요한 변수로 선택되었다. 이어서 병원침대 개수, 당기순이익 변화율이 중요한 변수로 꼽혔다 이를 통해 머신러닝 모델도 딥러닝 모델과 아주 큰 차이는 없다. 다른 딥러닝 모델로 성능을 향상 시킬 수 있는 기대효과가 있는 것 같다. 변수 중요도를 파악한 결과 다른 산업에 적용할 때 중요한 변수는 당기 순이익 변화율과 고정비용이 중요한 것으로 파악되었다.

향후 연구로 더 좋은 모델을 통해 예측력으로 높이는 방법을 더 연구하거나 다른 산업의 데이터도 같은 방법으로 예측하여 전 산업에 적용할 수 있는 만드는 연구 과정이 필요하다.

Ⅴ. 참고문헌

[1] 파이토치 첫걸음, 2019, 최건호, 한빛출판사
[2] 인천대학교 데이터 사이언스 강의자료 (2020 년도 1 학기), 2020