딥러닝을 이용한 소프트웨어 결함 예측 모델 제작 및 선행 연구 결과와 비교

성신여자대학교 컴퓨터공학과 20180998 임수정

지도 교수 : 홍의석 교수님

1. 연구 배경

다양한 분야에서 뛰어난 성능을 보여주고 있는 딥러닝 기술이 소프트웨어 결함 예측 모델에 적용했을 때에도 유의미한 성능 향상을 보이는지 알아보기 위해 연구를 진행했다.

연구의 목적은 소프트웨어 결함 예측 분야에서 흔히 사용되는 SVM(Support Vector Machine), naive bayes, decision tree 와 같은 머신러닝 기술을 소프트웨어 결함 예측 모델에 적용했을 때와 MLP, CNN 과 같은 딥러닝 기술을 적용했을 때의 예측 성능을 비교하기 위함이다.

2. 사용 데이터

실험에는 소프트웨어 결함 예측 모델의 성능 실험에 많이 사용되는 NASA MDP 데이터 집합을 사용했다. NASA MDP 데이터에 포함된 5 개의 프로젝트(CM1, PC1, KC1, KC2, JM1)를 모두 사용하여 실험했다.

각 프로젝트의 데이터 속성은 아래 표와 같다.

dataset	instance 개수	attribute 개수	결함 있는 모듈 수	결함 없는 모듈 수
CM1	498	22	49	449
PC1	1109	22	77	1032
KC1	2109	22	326	1783
KC2	522	22	105	415
JM1	10880*	22	2106	8779

*JM1 총 10885개의 인스턴스 중 5개 인스턴스에서 값이 '?'인 경우가 있어 제외 (10880개로 수정)

다섯개의 데이터 집합은 모두 같은 attribute를 가진다. attribute는 다음과 같다.

attributes : loc, v(g), ev(g), iv(g), n, v, l, d, i, e, b, t, lOCode, lOcomment, lOBlank, lOCodeAndComment, uniq_Op, uniq_Opnd, total_Op, total_Opnd, branchCount, defect

소프트웨어 결함 예측 모델에 머신러닝을 적용하는 실험에서는 feature selection을 하지 않은 데이터와 feature selection을 한 데이터를 모두 사용하여 실험했다. 이때 feature selection 방법은 CSE(CfsSubsetEval)을 사용했다. feature selection을 수행한 attribute는 다음과 같다.

attributes : loc, v(g), ev(g), iv(g), i, lOcomment, lOBlank, lOCodeAndComment, defect

3. 머신러닝 적용 실험

3.1. 실험 방법

모든 실험은 같은 환경에서 진행하였다. 언어는 python 을 사용하였다. python 코드를 구글 클라우드 서버에서 동작시키는 구글의 Colaboratory 를 사용했다. 구글 클라우드 서버를 이용하면 GPU 나 TPU 등 구글의 하드웨어를 사용할 수 있다는 이점이 있다.

실험에서 사용한 라이브러리는 다음과 같다.

라이브러리	**사용 목적**
sklearn	머신러닝 모델 라이브러리 사용

numpy	데이터 형식
sklearn.model_selection.train_test_split	훈련 데이터, 시험 데이터 분류
google.colab.drive	구글 드라이브 파일 연동

사이킷런(sklearn)은 python 을 위한 기계 학습 라이브러리로, 머신러닝 라이브러리 중 가장 많이 사용되는 라이브러리이다. 이 실험에서는 사이킷런이 제공하는 머신러닝 API 인 sklearn.svm, sklearn.naive_bayes.GaussianNB, sklearn.tree 와 예측 성능 측정 도구로 사용하는 sklearn.metrics.roc_curve, sklearn.metrics.auc, sklearn.metrics.confusion_matrix 모듈을 사용했다.

3.2. 모델 구조

머신러닝 모델 중 SVM, naive bayes, decision tree 모델을 적용했다.

3.2.1. SVM(Support Vector Machine)

서포트 벡터 머신(SVM)은 분류 문제에 적용할 수 있는 머신러닝 지도학습 모델이다. 서포트 벡터 머신은 분류를 위한 기준선을 정의하는 모델이다. 기준선은 선형 혹은 비선형 등으로 정의할 수 있다. 이 실험에서는 선형의 기준선을 이용했다. 사이킷런이 제공하는 서포트 벡터 머신의 API 는 아래와 같다.

class sklearn.svm.SVC(*, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None)

매개변수에서 기준선의 형태를 결정하는 kernel의 값을 'linear'(선형)으로 설정하고 그 외의 값은 모두 default 값을 사용했다.

3.2.2. 나이브 베이즈(naive bayes)

나이브 베이즈(naive bayes)는 통계적 분류 기법에 기반한 머신러닝 지도학습 모델이다. 나이브 베이즈는 데이터의 feature의 독립성이 보장되어야 한다는 조건이 있다. 사이킷런이 제공하는 나이브 베이즈 API 는 다음과 같다.

class sklearn.naive_bayes.GaussianNB(*, priors=None, var_smoothing=1e-09)

3.2.3. 의사결정트리(decision tree)

의사결정트리는 분류와 회귀 문제 모두에서 사용될 수 있는 머신러닝 지도학습 모델이다. 의사결정트리는 데이터를 분석하여 데이터 의 패턴을 예측 가능한 규칙들의 조합으로 나타내는 모델이다. 사이킷런이 제공하는 의사결정트리의 API 는 다음과 같다.

class sklearn.tree.DecisionTreeClassifier(*, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, ccp_alpha=0.0)

3.3. 실험 결과

모든 실험의 결과를 평가하는 지표로 Accuracy(정확도)와 AUC(Area Under the ROC Curve)를 사용했다.

3.3.1. SVM 적용 실험 결과

SVM 모델을 feature selection 을 하지 않은 5 개의 데이터 집합에 모두 적용한 결과 각각의 Accuracy 와 AUC 는 다음 표와 같다.

dataset	Accuracy	AUC
CM1	0.94	0.85
PC1	0.90	0.59
KC1	0.84	0.51
KC2	0.81	0.63

JM1	0.73	0.54

SVM 모델을 feature selection 을 수행한 5 개의 데이터 집합에 모두 적용한 결과 각각의 Accuracy 와 AUC 는 다음 표와 같다.

dataset	Accuracy	AUC
CM1	0.89	0.55
PC1	0.92	0.53
KC1	0.85	0.54
KC2	0.81	0.54
JM1	0.74	0.48

실험 결과 Accuracy 와 AUC 를 종합하여 판단했을 때 feature selection 을 수행하지 않은 데이터가 더 높은 정확도를 보였다.

3.3.2. 나이브 베이즈 적용 실험 결과

나이브 베이즈 모델을 feature selection 을 하지 않은 5 개의 데이터 집합에 모두 적용한 결과 각각의 Accuracy 와 AUC 는 다음 표와 같다.

dataset	Accuracy	AUC
CM1	0.82	0.61
PC1	0.91	0.72
KC1	0.82	0.75
KC2	0.83	0.83
JM1	0.81	0.68

나이브 베이즈 모델을 feature selection 을 수행한 5 개의 데이터 집합에 모두 적용한 결과 각각의 Accuracy 와 AUC 는 다음 표와 같다.

dataset	Accuracy	AUC
CM1	0.84	0.65
PC1	0.93	0.73
KC1	0.82	0.72
KC2	0.85	0.82
JM1	0.81	0.67

나이브 베이즈 모델에서는 feature selection 을 수행한 데이터의 정확도가 미세하게 더 높았다.

3.3.3. 의사결정트리 적용 실험 결과

의사결정트리 모델을 feature selection 을 하지 않은 5 개의 데이터 집합에 모두 적용한 결과 각각의 Accuracy 와 AUC 는 다음 표와 같다.

dataset	Accuracy	AUC
CM1	0.86	0.53
PC1	0.92	0.67
KC1	0.81	0.61
KC2	0.83	0.66
JM1	0.76	0.60

의사결정트리 모델을 feature selection 을 수행한 5 개의 데이터 집합에 모두 적용한 결과 각각의 Accuracy 와 AUC 는 다음 표와 같다.

dataset	Accuracy	AUC
CM1	0.84	0.52
PC1	0.90	0.66
KC1	0.81	0.60
KC2	0.85	0.63
JM1	0.74	0.58

실험 결과 대체적으로 나이브 베이즈 모델을 적용했을 때 가장 우수한 예측 성능을 보였다.

4. 딥러닝 적용 실험

딥러닝은 단순히 은닉층의 개수를 여러개를 사용하는 MLP 모델과 CNN 모델을 사용했다.

4.1. 실험 방법

코드 실행 환경은 머신러닝 적용 실험과 동일하다. 딥러닝 모델에서는 모델의

라이브러리로 사이킷런이 아닌 tensorflow.keras 를 사용한다.

4.2. MLP 적용 실험

MLP 적용 실험에서는 테스트 요소를 조절하면서 실험했다. 레이어 개수와 레이어의 노드 수, 학습률, 레이어의 구성, 최적화 알고리즘을 조절하면서 가장 좋은 결과가 나오는 경우로 실험했다.

4.3.1. 모델 구조

실험에서 사용한 MLP 의 기본 구조는 다음과 같다. 은닉층은 64 개의 노드를 갖고 활성화 함수로 Relu 함수를 사용하는 층과 Drop out 층으로 구성하였으며 테스트 케이스마다 층의 개수를 다르게 하였다. 또한 출력층은 노드 수를 1 개 갖고 sigmoid 함수를 활성화 함수로 사용하는 층으로 구성했다. 최적화 알고리즘은 Adagrad 와 Nadam, Adam 을 모두 사용해본 결과 Adam 의 성능이 가장 좋아 Adam 을 사용했다. 손실함수로는 이진 분류에서 가장 보편적으로 사용되는 binary crossentropy 를 사용했다.

조율한 요소 값들 중 학습률을 기본값인 0.001 에서 0.005, 0.0001 로 조절하면서 실험해 본 결과 학습 속도에서 유의미한 차이를 보이지만 최종 정확도에서 큰 차이를 보이지 않았기 때문에 기본 값인 0.001 을 사용했다.

4.3.2. 실험 결과

은닉층 1층

dataset	loss	accuracy	auc
cm1	0.399	0.93	0.57
pc1	0.52	0.936	0.69
kc1	0.46	0.836	0.746
kc2	2.11	0.828	0.786
jm1	0.46	0.815	0.59

은닉층 2 개(drop out layer 사용)

dataset	loss	accuracy	auc
cm1	0.28	0.93	0.59
pc1	0.47	0.936	0.74
kc1	0.46	0.838	0.74
kc2	0.54	0.83	0.81
jm1	0.43	0.81	0.71

은닉층 3 개(drop out layer 사용)

dataset	loss	accuracy	auc
cm1	0.56	0.93	0.65
pc1	0.25	0.927	0.5
kc1	0.57	0.836	0.69
kc2	0.45	0.79	0.758
jm1	0.47	0.815	0.54

은닉층 3 개

dataset	loss	accuracy	auc
cm1	1.57	0.93	0.55
pc1	0.48	0.927	0.69
kc1	1.5	0.83	0.729
kc2	1.79	0.85	0.77
jm1	0.64	0.815	0.63

은닉층 5 개(drop out layer 사용)

dataset	loss	accuracy	auc
cm1	0.26	0.93	0.5
pc1	0.25	0.927	0.51
kc1	1.90	0.836	0.49
kc2	0.41	0.79	0.78
jm1	0.477	0.815	0.5

은닉층 10 개(drop out layer 사용)

dataset	loss	accuracy	auc
cm1	0.26	0.93	0.5
pc1	0.25	0.927	0.51
kc1	0.44	0.836	0.5

kc2	0.51	0.79	0.5
jm1	0.478	0.815	0.5

실험 결과 은닉층이 1 개인 MLP 에서 대체적으로 가장 높은 예측 성능을 보였다.

4.4. CNN 적용 실험

CNN 적용 실험 또한 앞의 딥러닝 MLP 적용 실험 환경과 동일한 환경에서 실행하였고 동일한 데이터를 사용했다.

4.4.1. 모델 구조

CNN 모델 구성은 다음과 같다. convolution층과 max pooling층을 사용하여 모델을 구성했다.

```
_________________________________________________________________
Layer.    (type)     Output Shape
Param
=================================================================
conv2d_4(Conv2D)(None, 21, 1, 64)
 128
_________________________________________________________________
conv2d_5(Conv2D) (None, 21, 1, 64)
 4160
_________________________________________________________________
max_pooling2d_2 (MaxPooling2) (Non
e, 10, 1, 64) 0
_________________________________________________________________
conv2d_6(Conv2D)(None, 10, 1, 128)
 8320
_________________________________________________________________
conv2d_7(Conv2D)(None, 10, 1, 128)
 16512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2) (Non
e, 5, 1, 128) 0
_________________________________________________________________
flatten_1 (Flatten) (None, 640)
 0
_________________________________________________________________
dense_2 (Dense) (None, 128)
82048
_________________________________________________________________
dropout_1 (Dropout) (None, 128)
0
_________________________________________________________________
dense_3 (Dense) (None, 1)
129
=================================================================
```

4.4.2. 실험 결과

dataset	loss	accuracy	auc
cm1	0.264	0.93	0.5
pc1	0.25	0.927	0.5
kc1	0.44	0.836	0.5
kc2	0.51	0.79	0.5
jm1	0.478	0.815	0.5

CNN 적용 실험 결과 머신러닝, MLP 에서의 결과와 비교했을 때 예측 정확도 면에서 유의미한 성능 향상은 나타나지 않았다. 하지만 손실함수의 값이 적은 학습에서도 빠르게 줄어드는 양상을 보였다. 최종 예측 정확성은 큰 차이가 없었지만 학습 속도 향상에는 CNN 모델이 더 나은 성능을 보였다.

5. 결론

소프트웨어 결함 예측 모델에 머신러닝, 딥러닝(MLP, CNN)을 모두 적용하여 실험하고 비교해보았다. 실험 결과 머신러닝과 딥러닝을 적용했을 때 최종 정확도에 유의미한 성능 차이가 있지는 않았다. 하지만 학습 속도나 오차율은 머신러닝 모델 보다는 딥러닝에서 더 나은 성과를 보였다. 소프트웨어 결함 예측 분야에서 사용되는 데이터가 수치 데이터이면서 정형 데이터라는 점과 분야의 특성상 많은 양의 데이터가 수집되기 어려운 점을 고려했을 때, 다량의 비정형 데이터, 구조가 복잡한 데이터에 적합한 딥러닝 모델이 소프트웨어 결함 예측의 모델로 적합하지 않을 수 있다. 딥러닝 모델을 적용한 실험에서 이전 연구와 같은 예측 모델의 성능 저하는 발견되지 않았지만 머신러닝을 적용한 실험 결과와 비교했을 때 유의미한 성능 향상을 보이지 못했다. 따라서 소프트웨어 결함 예측 분야의 데이터에 적합하지 않은 딥러닝 모델보다는 머신러닝 모델 중 소프트웨어 결함 예측 분야에 더 적합하다고 판단되는 다른 모델을 적용해보는 실험이나 SVM, 나이브 베이즈와 같은 머신러닝 모델의 매개변수를

조정하여 소프트웨어 결함 예측에 최적화된 모델을 구축하는 연구가 보다 더 고무적이라고 생각한다.