논문 2020-15-35

머신러닝 자동화를 위한 개발 환경에 관한 연구

(A Study on Development Environments for Machine Learning)

김 동 길, 박 용 순, 박 래 정, 정 태 윤*
(Dong Gil Kim, Yong-Soon Park, Lae-Jeong Park, Tae-Yun Chung)

Abstract : Machine learning model data is highly affected by performance. preprocessing is needed to enable analysis of various types of data, such as letters, numbers, and special characters. This paper proposes a development environment that aims to process categorical and continuous data according to the type of missing values in stage 1, implementing the function of selecting the best performing algorithm in stage 2 and automating the process of checking model performance in stage 3. Using this model, machine learning models can be created without prior knowledge of data preprocessing.

Keywords : Machine Learning, Supervised Learning, Feature Engineering, Data Preprocessing, Algorithm

Ⅰ. 서 론

인공지능의 기술이 발전함에 따라 빅데이터 스케일의 자연어와 영상 데이터를 분석하고, 처리하는 등의 업무를 자동화 할 수 있는 수준에 다다랐으며, 인공지능은 사물인터넷 기술과 더불어 4차 산업혁명 시대의 핵심 기술로 평가받고 있다 [1].

머신러닝 (Machine Learning, ML)은 이러한 인공지능의 한 분야로서 프로그래머의 하드 코딩 과정 없이 기계가 스스로 학습하여 데이터에 담긴 정보나 패턴을 분석해 판단과 예측을 가능하게 한다. 학습 단계는, 학습에 사용되는 데이터의 타겟 값 (Label)이 존재하는 지도 (Supervised) 학습과 타겟 값이 없는 비지도 (Unsupervised) 학습으로 구분된다 [2, 3].

본 연구는 입력 데이터 X_i와 타겟 값 Y_i의 형태로 되어 있을 때, X_i에 대해 Y_i를 예측 (Prediction)하는 지도 학습 모델의 개발 환경에 초점을 두었다. 예측 모델에서 일반적인 개발 과정을 상세히 설명하면, 학습 데이터의 입력 (Input)에 대해 전처리를 진행하고, 적절한 알고리즘을 선택해서 모델을 학습한 이후 모델 검증과 학습된 모델을 사용해서 테스트 데이터의 정답을 예측하는 3단계로 구분된다.

각 단계는 다음과 같은 요구사항을 필요로 하는데, ① 특징 추출, 변환 등 데이터 전처리에 관한 요구사항으로, 범주형 (Categorical) 데이터를 수치형 (Numerical)으로 변환하거나, 데이터가 누락된 경우 이를 대체하는 방법이 필요하다. ② 모델 학습에 사용되는 머신러닝 모델 선택에 관한 요구사항으로, Decision Tree, Random Forest, Linear Regression 등 주어진 문제에 가장 적합한 분류나 회귀 모델의 선택이 필요하다. ③ 모델 성능 평가와 관련된 요구사항으로, 예측 결과를 수치화하기 위해 교차검증 (K-Fold Cross Validation), 혼동 행렬 (Confusion Matrix)을 적절하게 사용해야 한다.

이처럼 머신러닝 모델의 구축 과정에는 선택해

*Corresponding Author (tychung@gwnu.ac.kr)
Received: Jul. 13, 2020, Revised: Jul. 24, 2020, Accepted: Aug. 6, 2020.
D.G. Kim: Gangwon EMbedded Software Cooperative Research Center (Ph.D)
Y.S. Park: Gangwon EMbedded Software Cooperative Research Center (Ph.D. Student)
L.J. Park: Gangneung-Wonju National University (Prof.)
T.Y. Chung: Gangneung-Wonju National University (Prof.)
※ 이 논문은 2020년도 정부(산업통상자원부)의 재원으로 한국산업기술진흥원의 지원을 받아 수행된 연구임(P0006706, 통합 홈케어 서비스플랫폼 개발 및 임상지원)

ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2020.15.6.307

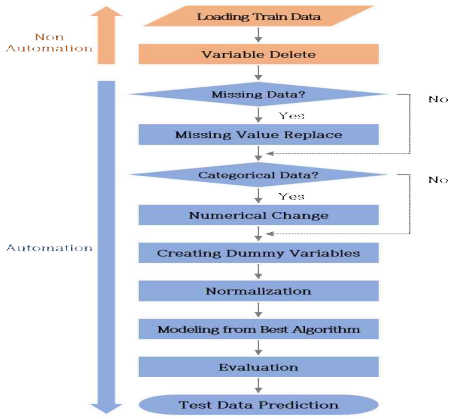

그림 1. 머신러닝 자동화 시스템 구조
Fig. 1 Machine Learning Automation System Architecture

그림 2. 매직 쿼드런트 보고서
Fig. 2 Magic Quadrant for Data Science and Machine Learning Platforms

야 하는 많은 조건과 변수 등이 존재하므로 모델의 생성부터 최종 평가 단계까지의 개발 과정을 효과적으로 보조할 수 있는 소프트웨어 개발 환경(Software Development Environment, SDE)이 필요하다.

본 연구의 목적은 지도 학습 기반의 머신러닝 모델을 빠르고 쉽게 구축할 수 있도록 그림 1에서와 같이 데이터 입력과 변수 삭제를 제외한 나머지 과정을 자동화 시키는 것이다. 이를 위해 ① 모델 구축 단계에서는 데이터의 특징을 추출하기 위해 결측값 결과에 따라 데이터를 분류하고, 분류된 데이터의 수치 변환 및 상관관계에 따른 결측값(Missing Value) 대체, 그리고 새로운 변수 생성과 왜도 (Skewness), 첨도 (Kurtosis) 등 정규화에 필요한 코드를, ② 모델 생성 단계에서는 사용된 머신러닝 모델의 성능을 확인하고 가장 우수한 성능을 선택할 수 있는 코드를, ③ 모델 평가 및 예측 단계에서는 성능 평가 지표에 따라 결과를 검증하고, 테스트 데이터의 예측 결과를 파일로 출력할 수 있는 코드를 적용하였다.

본 연구의 구성은 다음과 같다. 2장에서 본 연구와 관련된 문헌을 고찰하고, 3장에서 개발 환경에 필요한 기술들을 상세히 기술하였으며, 마지막으로 결론 및 향후 계획을 4장에서 제시한다.

Ⅱ. 선행연구 고찰

가트너에서 발표한 그림 2의 Magic Quadrant 보고서를 살펴보면 자동화된 머신러닝 시스템을 제공하는 소프트웨어 업체들이 세계 시장에서 강세를 보이고 있다 [4].

구글의 AutoML 서비스는 AutoML Tables, AutoML Natural Language, AutoML Translation, AutoML Vision, AutoML Video Intelligence가 있다. 구글은 비전문가들도 AutoML 서비스를 쉽게 사용할 수 있도록 UI에 사용자의 니즈를 반영하고, GCP (Google Cloud Platform)와 호환이 가능하며, 코드 사용 없이 클릭만으로 머신러닝 모델 학습이 가능하다는 장점이 있다. 하지만 대부분의 AutoML 서비스들이 클라우드 기반이기 때문에 계정 가입이 필요하며, AutoML Tables와 같은 유료 서비스가 있다. 그리고 AutoML에서 사용할 데이터를 준비하는 과정에서 특성에 따라 결측값을 다양한 값으로 대체하거나 알고리즘을 최적화하는 등 모델 성능을 높이기 위해서는 전문가 (데이터 사이언스, 데이터 분석 전문가 등)의 도움을 통해 분석에 적합한 데이터를 미리 준비해야 한다 [5].

마이크로소프트의 Azure Machine Learning은 데이터 셋과 모듈을 시각적으로 연결하여 Drag, Drop, Connect 등의 기능을 통해서 모델을 훈련하고 예측 분석 모델을 구성하고, 다양한 머신러닝 라이브러리가 제공되며, 계정 가입 없이 10GB 내에서 30일까지 무료로 이용할 수 있는 장점이 있다. 반면 데이터 업로드 속도가 느리고, 동시에 3개 이상의 모델 성능을 비교하는데 어려움이 있다. 또한 구글의 AutoML과 동일하게 머신러닝 모델에 적합한 데이터 준비가 필요하다 [6].

다음 표 1은 머신러닝의 자동화를 위한 기존 환경과 신규 환경의 비교를 보여준다.

표 1. 머신러닝 자동화를 위한 환경 비교
Table 1. Comparison Between the Environments for Machine Learning Automation

	New SDE	AutoML	Azure ML
Automation	●	◐	◐
Missing Value Type Check	●	○	○
Estimating a Missing Value from Correlation	●	○	○
Creating Dummy Variables	●	○	○
Normalization : Skewness, Kurtosis	●	○	○
Modeling from Best Algorithm	●	◐	◐
Visualization	●	◐	●

● Automation ◐ Semi-Automation ○ Non-Automation

표 2. 타겟 입력 및 데이터 삭제
Table 2. Label Check and Delete Data

Label Column	Survived
▼	
Delete? 1(Yes) 2(No)	1
How Many? (1~5)	1
Variable Name?	Ticket

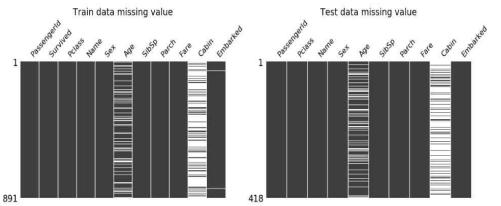

그림 3. 결측값 시각화
Fig. 3 Visualization of Missing Value

Ⅲ. 자동화를 위한 개발 환경 설계

개발 환경 설계에 앞서 최우선적으로 고려한 부분은 데이터 전처리부터 모델 예측까지 머신러닝 모델의 개발 경험이 부족한 개발자가 갖는 부담감과 많은 시행착오 과정을 줄이는 것이다. 이를 위해 본 연구에서 추구하고자 하는 개발 방향을 정리하면 다음과 같다.

① 데이터를 준비하고 삭제하는 등의 비자동화 과정을 최소화하고 이후 과정을 자동화한다.

② 텐서플로우 튜토리얼 방식의 머신러닝 신규 모델에서 사용한 코드를 참고하여 모델 구축, 생성, 평가 및 예측 과정을 자동화한다 [7].

③ 자동화 과정에서 결측값을 추정하기 위해 상관관계를 이용한 결측값 처리 코드를 참고하고 [8], 문자 형태의 데이터를 숫자로 변경할 때 해당 변수를 확인하지 않고 다음 단계로 진행이 될 수 있도록 정규 표현식을 사용하여 자동화한다.

④ 머신러닝 지도학습 기반의 분류, 회귀, 앙상블 알고리즘을 다양화하여 성능이 높은 모델이 한 개 선택될 수 있도록 자동화한다.

1. 비자동화

1.1 데이터 셋

지도 학습 기반의 머신러닝 모델을 학습하기 위해서는 훈련 데이터와 테스트 데이터가 필요한데, 캐글에서 공개된 타이타닉 데이터가 이에 해당되며 train.csv, test.csv 파일을 학습에 사용하였다 [9].

1.2 정답 확인 및 삭제

데이터 준비가 완료되면 타겟 값을 입력하고 특징 추출이 쉽지 않은 데이터를 삭제할 수 있는 창을 표 2에 나타냈다.

2. 자동화

2.1 모델 구축

머신러닝에서는 결측값이 존재할 경우 분석이 불가능하기 때문에 이를 먼저 파악하는 것이 중요하다. 그림 3은 결측값을 쉽게 파악할 수 있도록 시각화한 것으로 이를 통해 어떤 구간에서 결측값이 발생했는지 확인할 수 있다.

결측값이 확인되면 범주형과 수치형을 확인하는 과정이 진행되는데, select_dtypes 설정 값인 'int', 'float', 'object'에 따라 분리된 결측값 결과를 표 3에서 확인할 수 있다.

표 3. 결측값 유형
Table 3. Missing Value Types

Missing Value >>> Categorical Data	
Train Data	['Cabin', 'Embarked']
Test Data	['Cabin']
Missing Value >>> Numerical Data	
Train Data	['Age']
Test Data	['Age', 'Fare']
No Missing Value >>> Categorical Data	
Train Data	['Name', 'Sex']
Test Data	['Name', 'Sex', 'Embarked']
No Missing Value >>> Numerical Data	
Train Data	['Fare']
Test Data	[]

표 4. 정규 표현식
Table 4. Regular Expression

	Description	Ex
[]	a set of characters	[a-m]
₩	signals a special sequence	₩d
.	any character	he..o
^	starts with	^hello
$	ends with	world$
*	zero or more occurrences	aix*
+	one or more occurrences	aix+
{}	exactly the specified number of occurrences	al{2}
|	either or	a|b
()	capture and group	

▼ Stage 1

```
train = train.select_dtypes(include = 'object')
train = train.fillna('missing_value')

def transform_categorical(name):
    pattern = re.compile(r'([a-zA-Z]+)\.')

    for i in range(len(train.columns)):
        if pattern.search(name):
            train[train.columns[i]] =
train[train.columns[i]].str.extract(pattern)
            train[train.columns[i]] =
train[train.columns[i]].replace(train[train.columns[i]].value_c
ounts().iloc[4:].index, 'Other')
            break
        else:
            print('No Transformation is Required')
```

▼ Stage 2

```
train_list = [train[train.columns[c]].tolist()[0] for c in
range(len(train.columns))]

for i in range(len(train.columns)):
    transform_categorical(train_list[i])
    train = train.replace('missing_value', np.NaN)
```

그림 4. 범주형 데이터 전환 과정
Fig. 4 Categorical Data Change Process

▼ Stage 1

```
corr_0 = lambda x:
x.corrwith(x['Pclass']).sort_values().head(5)
corr_1 = lambda x: x.corrwith(x['Sex']).sort_values().head(5)
                                    ⋮
corr_11 = lambda x:
x.corrwith(x['Cabin']).sort_values().head(5)
```

▼ Stage 2

```
corr_0_df = corr_0(train)
corr_1_df = corr_1(train)
                                    ⋮
corr_11_df = corr_11(train)
```

▼ Stage 3

```
train['Pclass'] =
train['Pclass'].fillna(train[corr_1_df.index[0]].median())
train['Sex'] =
train['Sex'].fillna(train[corr_2_df.index[0]].median())
                                    ⋮
train['Cabin'] = train['Cabin'].fillna(train[corr_11_df.index[0]].
median())
```

그림 5. 결측값 대체 과정
Fig. 5 Process of Replace Missing Value

범주형 데이터는 수치 전환과 결측값 대체가 동시에 진행된다. 수치 전환은 파이썬 패키지인 re의 compile과 search 모듈을 통해 자동화하였으며, 예시로 그림 4에 이름 형태의 단어를 수치화하는 과정을 정규 표현식 코드로 나타냈다.

re.compile()에서 표 4의 기본적인 정규 표현식을 응용하여 이름 외에도 한 글자, 앞뒤 특수문자, 이메일 등 다양한 형태의 단어가 변경될 수 있도록 코드에 정규 표현식을 추가하였다 [10].

수치 전환이 완료되면 상관관계를 파악하여 결측값이 있는 데이터에 영향이 가장 많은 변수의 중앙값으로 결측값 대체가 진행되며, 이는 그림 5에서 확인할 수 있다.

하지만 이 방법은 결측값 데이터를 수동으로 파악하고 코드에 직접 표현해야하는 단점이 있기 때문에 그림 6의 방법으로 4단계에서 상관관계 정보를 데이터화하고 5단계에서 기존 1~3단계의 단점을 개선하여 자동화하였다.

결측값 대체가 완료되면 파이썬 패키지인 Scikit-Learn의 LabelEncoder 모듈을 통해 범주형 자료를 수치화할 수 있다.

범주형 자료는 연속된 값으로 나타나지 않는 특

▼ Stage 4

```
corr_list = []
for col in train.columns:
    corr = lambda x: x.corrwith(train[col]).abs()
    corr_ = corr(train)
    corr_list.append(corr_)
    corr_df = pd.DataFrame(corr_list).T

corr_df.columns = corr_.columns.tolist()
```

▼ Stage 5

```
train_corr = [corr_df[col].reset_index() for col in
train.columns]
top = [train_corr[i]['index'][0] for i in range(len(train_corr))]
def corr_top(df):
    for col_ in train.columns:
    df[col_] = [df[col_].fillna(df[top[i_]].median()) for i_ in
range(len(train.columns))]

train = corr_top(train)
```

그림 6. 결측값 대체 과정 개선

Fig. 6 Improving the Process of Replace Missing Value

표 5. 가변수 생성

Table 5. Create of Dummy Variables

Sex	▶	Sex_male	Sex_female
male		1	0
female		0	1
⋮		⋮	⋮

징을 갖는다. ‘Sex’를 예로 들면 ‘male’와 ‘female’ 사이에는 중간값이 없고 순서가 없기 때문에, ‘Sex’에 가장 적합한 데이터 표현을 찾는 기법이 필요하며, 가변수 (Dummy Variable)는 이러한 범주형 변수를 0 또는 1 값을 가지는 전처리 방법이다 [11-13].

파이썬 패키지인 Pandas의 get_dummies 모듈을 사용하면 표 5에서와 같이 가변수 변환이 가능하다.

훈련 데이터에만 과도하게 최적화되어 테스트 데이터에 적용했을 때 성능이 제대로 나타나지 않는 과적합 (Overfitting) 문제를 해결하기 위해 왜도와 첨도를 조절하고, 이를 그림 7에서 확인할 수 있다 [14, 15].

데이터 분포에서 일반적인 패턴과 매우 다른, 이상한 패턴을 가진 데이터를 제거하기 위해 그림 8에서와 같이 Scikit-Learn의 RobustScaler 모듈을 사용해서 이상치 (Outlier) 변화를 확인하였다 [16].

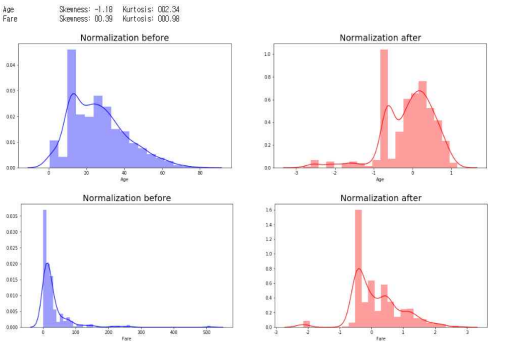

그림 7. 왜도, 첨도 시각화

Fig. 7 Visualization of Skewness, Kurtosis

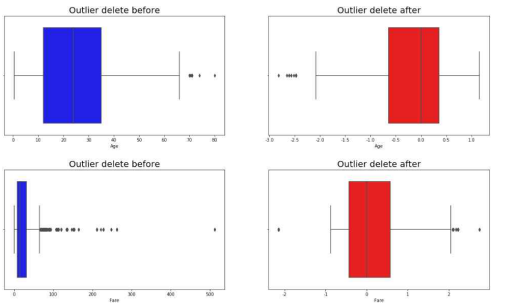

그림 8. 아웃라이어 시각화

Fig. 8 Visualization of Outlier

2.2 모델 생성

지도학습에서 사용되는 머신러닝 알고리즘은 다음과 같다.

KNN (K-Nearest Neighbor)는 데이터로부터 거리가 가까운 k개의 다른 데이터의 타겟 값을 참조하여 분류하는 모델이다 [17].

SVM (Support Vector Machine)은 분류 (결정경계)를 위한 기준 선을 정의하는 모델이다 [18].

DT (Decision Tree)는 각 데이터들이 가진 속성으로부터 패턴을 파악하여 분류 과제를 수행할 수 있도록 도와주는 모델이다 [19].

LogisticR (Logistic Regression)은 데이터가 어떤 범주에 속할 확률을 0에서 1사이의 값으로 예측하고 그 확률에 따라 가능성이 높은 범주에 속하는 것으로 분류하는 모델이다 [20].

LinearR (Linear Regression)은 데이터를 가장 잘 설명할 수 있는 선을 찾는 모델로 오차의 평균을 최소화할 수 있는 최적의 기울기와 절편을 파악하는 것이다 [21].

▼ Stage 1

```
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
knn_score = knn.score(X_val, y_val)

svm = SVC()
svm.fit(X_train, y_train)
svm_score = svm.score(X_val, y_val)

                    ⋮

lgb = LGBMClassifier()
lgb.fit(X_train, y_train)
lgb_score = lgb.score(X_val, y_val)
```

▼ Stage 2

```
Models = pd.DataFrame({
  'Model' : ['KNN', 'SVM', ···, 'LightGBM'],
  'Score' : [knn_score, svm_score, ···, lgb_score]
})
Models.sort_values(by = 'Score', ascending = False)
```

	Model	Score
7	XGBoost	0.82151
6	GBM	0.81192
⋮	⋮	⋮
1	Linear Regression	0.69175

▼ Stage 3

```
best = xgb.predict(test)
```

그림 9. 머신러닝 학습 과정
Fig. 9 Machine Learning Process

GBM (Gradient Boosting Machine)은 경사하강법을 사용해서 트리를 만들어가며 이전 트리의 오차를 보완하는 모델이다 [22].

XGBoost (eXtreme Gradient Boosting)는 트리를 만들 때 Classification and Regression Tree (CART)를 사용하여 각 분류 기간의 가중치를 최적화하는 모델이다 [23].

LightGBM은 Tree가 수직적인 Leaf-Wise 모델로 대량의 데이터를 병렬로 빠르게 학습하는데 효과적이다 [24].

일반적으로 머신러닝 모델을 생성하기 위해서는 앞에서 언급한 알고리즘의 성능을 먼저 파악하는 것이 중요하다. 기존에는 각각의 알고리즘을 개별로 학습하여 이를 모두 비교하고 우수한 성능을 보이는 알고리즘에서 모델을 다시 학습하는 과정이 그림 9에서와 같이 필요하다.

본 연구는 이러한 반복 과정을 제거하고, 분석의 효율성과 코드의 간결성을 높이기 위해 그림 10의 방법으로 기존 코드를 개선하였다.

2.3 모델 평가 및 예측

모델 학습에 사용되는 훈련 데이터는 K-Fold 교차 검증을 통해 k 등분 ($k=5$)하고, $k-n$은 훈련용 n은 검증용으로 분할하여 데이터를 여러 번 교

▼ Stage 4

```
best = []
for model_ in [KNeighborsClassifier(), SVC(), ··· ,
LGBMClassifier()]:
    model = model_
    model.fit(X_train, y_train)
    accuracy = model.score(X_val, y_val)
    best.append(type(model_).__name__ + " : " + str(accuracy)
+ "%")
    best_ = pd.DataFrame(best, columns = ['accuracy'])
    best_['algorithm'] = best_['accuracy'].str.split(':').str[0]
    best_['accuracy'] = best_['accuracy'].str.split(':').str[1]
    best_ = best_[['algorithm', 'accuracy']].reset_index(drop
= 'index')
    best_1 = best_['algorithm'][0].strip()
    exec('best_model = ' + best_1 + '()')
```

▼ Stage 5

```
best = best_model(test)
```

그림 10. 머신러닝 알고리즘 코드 개선
Fig. 10 Improved in Machine Learning Algorithm Code

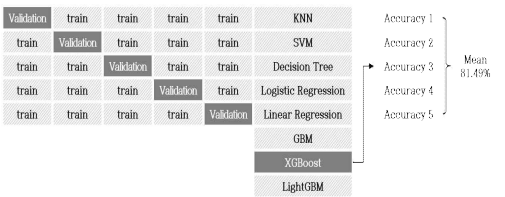

그림 11. 교차 검증 결과
Fig. 11 Cross Validation Result

차하면서 성능을 파악하였으며 [25], 파이썬 패키지인 Scikit-Learn의 KFold 모듈을 통해 나타난 결과는 그림 11에서와 같다.

이전 모델 생성 단계에서 성능이 가장 우수한 모델은 XGBoost 알고리즘으로 교차 검증 결과 $k=5$일 때 80% 이상의 성능을 보이는 것으로 나타났다. 이는 모델 학습에 사용된 훈련 데이터의 분류가 잘되어 테스트 데이터의 예측 결과에도 좋은 영향을 줄 수 있다고 판단할 수 있다.

훈련 데이터의 성능 검증이 완료되면 모델의 예측력을 검증하고 평가하는 과정이 필요한데, 정확도 (Accuracy), 정밀도 (Precision), 재현율 (Recall), F1 Score 점수를 통해 이를 판단할 수 있다 [26-28].

정확도는 예측된 값이 실제 값과 얼마나 정확하게 예측했는지를, 정밀도는 예측된 값 중 실제 값과 얼마나 관련이 있는지를, 재현율은 실제 값 중에서 예측된 값이 얼마나 관련이 있는지를, F1 score는 정밀도와 재현율의 조화 평균을 의미한다. 각각의 계산 방법은 다음과 같다.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F1\,score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4)$$

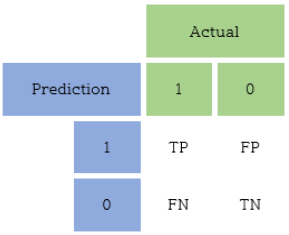

그림 12. 혼동 행렬
Fig. 12 Confusion Matrix

각 식에서 사용된 TP (True Positive)는 True를 True로, TN (True Negative)는 False를 True로, FP (False Positive)는 True를 False로, FN (False Negative)는 False를 False로 예측하는 것을 의미한다. Scikit-Learn의 confusion_matrix 모듈을 통해 위의 조건을 모두 확인할 수 있는 혼동 행렬을 그림 12에 나타냈다.

위의 조건에 따라 학습 모델의 성능을 측정한 결과 모든 평가 지표에서 80% 이상을 보이는 것으로 나타났다. 만약 테스트 데이터에 이 모델을 적용한다면 결과 또한 평가 지표 점수와 비슷할 것으로 판단할 수 있다. 하지만 하이퍼 파라미터 최적화 과정이 없기 때문에 실제 결과에서는 차이가 발생할 수 있다는 점을 염두에 두어야 한다.

3. 결과

완성된 자동화 모델은 그림 13의 방법으로 훈련 데이터와 테스트 데이터를 변경하고, 변경된 훈련 데이터로 모델을 학습한 뒤 테스트 데이터를 예측하고자 한다.

비자동화 단계에서는 그림 14에서와 같이 생존(Survived) 값을 타겟으로 설정하고, 티켓 (Ticket) 값을 삭제하였다.

자동화 단계는 그림 15에서 하단의 모델 성능을 확인할 수 있는 Classification Report가 나타나면 자동화가 정상적으로 진행된다고 판단할 수 있다.

```
▼ Datetime

def random_dates(start, end, n = len(train)):
    start_ = start.value // 10 ** 9
    end_ = end.value // 10 ** 9
    return  pd.to_datetime(np.random.randint(start_,  end_,
n), unit = 's')

start = pd.to_datetime('2020-01-01')
end = pd.to_datetime('2020-06-30')
train['Datetime'] = random_dates1(start, end)

▼ Email

i_col = []
for i in range(1, len(train) + 1):
    i_col.append(str(i) + '_test@test.com')

email_df_1 = pd.DataFrame(i_col, columns = ['Email'])
train = pd.concat([train, email_df_1], axis = 1)
```

그림 13. 데이터 변경
Fig. 13 Process for Changing data

```
Target(Label) : Survived
Variable to Delete : 1(Yes), 2(No) : 1
The Number of Variable to Delete(1~5) : 1
Which Variable do You Want to Delete : Ticket
```

그림 14. 자동화 진행 전 요구되는 설정
Fig. 14 Setting Requesting Before Automation Progress

```
(Step 1) Missing Value (Yes) >>> Categorical
============================================================
Train data: ['Cabin', 'Embarked']
Test data: ['Cabin']

(Step 2) Missing Value (Yes) >>> Numerical
============================================================
Train data: ['Age']
Test data: ['Age', 'Fare']

(Step 3) Missing Value (No) >>> Categorical
============================================================
Train data: ['Name', 'Sex', 'Email']
Test data: ['Name', 'Sex', 'Embarked', 'Email']

(Step 4) Missing Value (No) >>> Numerical
============================================================
Train data: ['PassengerId', 'Survived', 'Pclass', 'SibSp', 'Parch', 'Fare', 'Datetime']
Test data: ['PassengerId', 'Pclass', 'SibSp', 'Parch', 'Datetime']


Data Preprocessing Done···


Best Algorithm? XGBoost
Cross Validation Mean: 0.81931

Classification Report

              precision    recall  f1-score   support

           0       0.83      0.92      0.87       159
           1       0.86      0.73      0.79       109
```

그림 15. 머신러닝 자동화 모델 분석 결과
Fig. 15 Result of ML Automation Model Analysis

표 6. 결측값 대체 방법 비교
Table 6. Comparing Method of Replacing Missing Value

Model	Delete	Mean	Correlation
LinearR	0.76364	0.73881	0.75000
XGBoost	0.76364	0.79851	0.82463
LightGBM	0.74546	0.81716	0.83955
KNN	0.72727	0.65672	0.70149
DT	0.69091	0.64926	0.70896
GBM	0.65455	0.76119	0.76493
Logistic R	0.34546	0.65672	0.69776
SVM	0.59056	0.66212	0.68817

Ⅳ. 결 론

본 연구는 지도 학습 기반의 머신러닝 분석이 가능한 자동화 모델로 다음과 같은 특징이 있다.

첫 번째, 머신러닝 모델은 문자 형태가 있는 데이터의 경우 전처리 이후 분석이 가능하다. 기존 환경은 전처리에 필요한 데이터를 수동으로 확인하고 코드화해야하는 단점이 존재하기 때문에 정규 표현식을 패턴화하여 코드화하고, 5 이상일 경우 기타(Other) 값으로 통일하여 범주형의 수치 전환 과정을 자동화하였다.

두 번째, 결측값이 있을 경우 모델 성능이 달라지는 문제가 발생한다. 기존 환경은 값을 삭제하거나 평균 등의 대푯값으로 값을 통일하는데 표 6에서와 같이 상관관계를 이용한 방법에서 모델 성능이 좋아진 것을 확인할 수 있다. 따라서 상관관계의 수치화가 가능한 코드를 작성하고, 절댓값 기준으로 수치가 가장 높은 변수의 중앙값으로 결측값이 대체되도록 자동화하였다.

세 번째, 머신러닝 모델을 학습하기 위해서는 성능이 우수한 알고리즘이 필요하다. 기존 환경은 사용되는 알고리즘을 모두 비교하여 성능이 높은 모델을 확인하는 과정이 필요하다. 이를 해결하기 위해 파이썬의 내장함수인 exec()를 사용하여 모델을 학습하는 과정에서 문자열로 표현된 model.fit 문을 인수로 받아 다음 단계로 진행되도록 코드화하여 모델 생성 과정을 자동화하였다.

이와 같은 과정들을 통해 앞에서 언급한 분석에 적합한 데이터 준비 과정이 불필요하며, 모델 구축부터 생성, 평가 및 예측까지 추가적인 코드 보완 없이 대응할 수 있음을 확인하였다.

본 연구의 결과는 머신러닝에 관한 개념과 데이터 구조에 대한 사전 지식 없이 모델을 자동으로 생성하기 위한 기술로서 활용될 수 있을 것으로 기대된다. 추후에는 모델 성능 향상을 위해 학습에 사용된 데이터의 중요도가 높은 변수를 모델에 적용하는 변수 선택 및 중요도와 딥러닝 모델을 추가하는 연구를 과제로 남겨두고자 한다.

References

[1] P. S. Jang, “[EU] 2016 Davos Forum,” Science and Technology Policy Institute, Vol. 26, No. 2, pp. 12-15, 2016 (in Korean).

[2] A. L. Samuel, “Some Studies in Machine Learning Using the Game of Checkers,” IBM Journal of Research and Development, Vol. 3, No. 3, pp. 201-229, 1959.

[3] S. Z. Cho, S. H. Kang, “Industrial Application of Machine Learning (Artificial Intelligence,” Industrial Engineering Magazine, Vol. 23, No. 2, pp. 34-38, 2016 (in Korean).

[4] https://www.gartner.com/en/search?keywords=

[5] https://cloud.google.com/automl/docs?hl=ko

[6] https://docs.microsoft.com/ko-kr/azure/machine-learning/

[7] D.G. Kim, Y.S. Park, L.J. Park, T.Y. Chung, “Developing of New a Tensorflow Tutorial Model on Machine Learning : Focusing on the Kaggle Titanic Dataset,” IEMEK J. Embed. Sys. Appl., Vol. 14, No. 4, pp. 207-218, 2019 (in Korean).

[8] D.G. Kim, Y.S. Park, T.Y. Chung, “Development of Processing Missing Value for the Code Using Correlation,” Journal of the Institute of Embedded Engineering of Korea, 2019 (in Korean).

[9] https://www.kaggle.com/datasets

[10] https://en.wikipedia.org/wiki/Regular_expression

[11] https://en.wikipedia.org/wiki/Dummy_variable_(statistics)

[12] K.H. Kim, B.H. Chang, H.K. Choi, “Deep Learning Based Short-Term Electric Load Forecasting Models Using One-Hot Encoding,” Journal of IEEE Korea Council, Vol. 23, No. 3,

pp. 852-857, 2019 (in Korean).
[13] Y.S. Tak, J.H. Cheol, J.Y. Jung, "Performance Comparison Among Deep Neural Networks Consisting of Various Constituents," Journal of The Korean Data Analysis Society, Vol. 21, No. 5, pp. 2289-2301, 2019 (in Korean).
[14] S.H. Ryu, J.B. Yoon, "The Effect of Regularization and Identity Mapping on the Performance of Activation Functions," Journal of the Korea Academia-Industrial cooperation Society, Vol. 18, No. 10, pp. 75-80, 2017 (in Korean).
[15] C.S. Hong, J.H. Sung, "Multivariate Skewness and Kurtosis," Journal of the Korean data & information science society, Vol. 29. No. 1, pp. 71-81, 2018 (in Korean).
[16] H.K. Jong, K.W. Lee, E.S. Cho, "Detecting Outliers on Training Dataset for Better Quality of Estimation on Vessel Traces," KIISE Transactions on Computing Practices, Vol. 25, No. 12, pp. 594-601, 2019 (in Korean).
[17] Y.G. Lee, "A Study on Book Categorization in Social Sciences Using kNN Classifiers and Table of Contents Text," Journal of the Korean society for information management, Vol. 37, No. 1, pp. 1-21, 2020 (in Korean).
[18] S.W. Rhee, H.J. Cho, C.J. Chae, "EEG Signal Classification based on SVM Algorithm," Journal of the Korea Convergence Society, Vol. 11, No. 2, pp. 17-22, 2020 (in Korean).
[19] Y.S. Lee, J.S. Yi, S.H. Kim, "Segmentation of Performing Arts Market: An Application of Latent Class Analysis and Decision Tree Analysis to Infrequent Attendees," Journal of consumer studies, Vol. 31, No. 3, pp. 245-267, 2020 (in Korean).
[20] J.H. Lee, "Machine Learning Applications to Households Insolvency with Imbalanced Data," Journal of consumer studies, Vol. 30, No. 6, pp. 97-118, 2019 (in Korean).
[21] C.G. Park, K.E. Lee, "A Linearity Test Statistic in a Simple Linear Regression," Journal of the Korean data & information science society, Vol. 25, No. 2, pp. 305-315, 2014 (in Korean).
[22] C.Y. Seo, Y.J. Suh, D.J. Kim, "Study on Fault Detection of a Gas Pressure Regulator Based on Machine Learning Algorithms," Journal of the Korea society of computer and information, Vol, 25, No. 4, pp. 19-27, 2020 (in Korean).
[23] D.W. Hah, Y.M. Kim, J.J. Ahn, "A Study on KOSPI 200 Direction Forecasting Using XGBoost Model," Journal of the Korean Data And Information Science Society, Vol. 30, No. 3, pp. 655-669, 2019 (in Korean).
[24] W.W, Nam, N.W. Kim, "Deep Learning Based Depression Classification Using Environmental Factor Selection," The transactions of The Korean Institute of Electrical Engineers, Vol. 69, No. 7, pp. 1102-1110, 2020 (in Korean).
[25] J.C. Jeong, H.J. Youn, "Region of Interest (ROI) Selection of Land Cover Using SVM Cross Validation," Journal of Cadastre & Land InformatiX, Vol. 50, No. 1, pp.75-85, 2020 (in Korean).
[26] https://en.wikipedia.org/wiki/Accuracy_and_precision
[27] https://en.wikipedia.org/wiki/Precision_and_recall
[28] https://en.wikipedia.org/wiki/F1_score

Dong Gil Kim (김 동 길)

He received Ph.D. He is an Data Analysis Group Senior Researcher at he since joining Gangwon Embedded Software Research Center in November 2018. He received the BA, MA, and PhD degrees in Industrial Engineering from Gangneung-Wonju University, Gangneung, Korea. His Current Research interests are Engineering Economics, Decision Making, Statistical Analysis and Information Systems, Machine Learning, Deep Learning, Data Analysis.

Email: dgkim1108@gwnu.ac.kr

Yong-Soon Park (박 용 순)

He is currently a Ph.D. candidate in Electrical Engineering at Gangneung, Korea. He received his M.S. degree in Electronic Engineering from Gangneung-Wonju National University. His current research interests include Data Analysis. Currently, He is working in Gangwon Embedded Software Research Center in Korea.

Email: ee9415@gwnu.ac.kr

Lae-Jeong Park (박 래 정)

He received the B.S. degree in Elec. Eng. from Seoul National University, Seoul, Korea, in 1991, and the MS and Ph.D. degrees in Elec. Eng. from KAIST, Taejon, Korea, in 1993 and 1997, respectively. From 1997 to 1999, he was with the Information Technology Lab. at LG CIT, Seoul, Korea. He is currently a professor in the Department of Elec. Eng. at Gangneung-Wonju National University, Gangneung, Korea. His current research interests are machine learning, deep learning, and pattern recognition.

Email: ljpark87@gmail.com

Tae-Yun Chung (정 태 윤)

He received Ph.D. degree in Electronics Engineering from Yonsei University, Seoul, Korea, in 2000. From 1989 to 2001 he was a senior research engineer of Samsung Electronics, Korea. Since 2001 he has been with Gangneung-Wonju National university, where he is a professor of school of electronic engineering. He is a director of Gang- won Embedded Software Research Center. His research interests include Embedded System and Sensor Network.

Email: tychung@gwnu.ac.kr