Assignment 3

2015313254 노인호

November 3rd 2019

1 Progress

1.1 Problem

binary classification problem 으로 문제에서 주어진 dataset은 x1 x7 까지 총 7개의 feature를 가지고 있으며 target으로 0또는 1의 값을 가지고 총 1000개의 데이터가 있다. supervised learning중 neural network를 이용하여 학습을 진행한다.

1.2 Data Load

다운받은 csv파일을 dataframe type으로 저장하였다. 먼저 데이터에 결측치가 있는지 조사하였다. df.isnull().sum()으로 조사를 해 본 결과 결측치는 하나도 없음을 알 수 있었다. 또한 x1은 성별로 2가지 (male, female), x2는 group name으로 5가지 (Group A,B,C,D,E), x3는 학위로 6가지, x4는 2가지, x5는 2가지의 범주형 변수임을 알 수 있었고, x6, x7은 연속형 범수임을 알 수 있었다.

1.3 Data preprocessing

(1) data를 학습시키기 위한 입력변수와 종속변수는 다음과 같게 된다.

- input variables(x) : x1, x2, x3, x4, x5, x6, x7
- output variables(y) : y

(2) scikit-learn의 train_test_split함수를 이용해서 먼저 train(train and valid)과 test set을 8:2로 나누고 validation을 위해 train(train and valid) set을 train과 valid set으로 8:2로 나눈다. random_state=0으로 두었다. 또한 데이터의 target값이 class1은 875개, class0은 125개로 imbalance하므로 데이터를 고르게 split하기 위해 stratify옵션을 넣어 주었다.
그래서 data set의 비율은 train : valid : test = 0.64 : 0.16 : 0.2 가 된다.

(3) Learning algorithm을 neural network로 설정하였기 때문에 input variables는 input layer가 되며, output variables 는 output layer로 간주된다. 범주형 데이터가 있기 때문에 data scaling이 필요한데, x1-x5는 범주형 데이터로 one-hot encoding, x6-x7은 연속형 데이터로 standardscaler를 이용하였다. 이 2개를 한번에 적용시키기 위해 scikit-learn의 ColumnTransformer 함수를 이용하였다. 그래서 전처리후 총 input variables의 fearture는 19개(2(x1)+5(x2)+6(x3)+2(x4)+2(x5)+1(x6)+1(x7)=19)가 됨을 알 수 있었다.

1.4 Learning Algorithm(neural network)

scikit-learn의 classification을 위한 MLPClassifier 함수를 사용하였다. 변화를 줄 main parameter는 다음과 같다.

- hidden_layer_sizes ((a,b,c, ...) a is number of neurons in the 1th hidden layers, b 2th, c 3rd, ...)
- activation (activation function for hidden layer (identity, logistic, tanh, relu))
- solver (the solver for weight optimization (sgd, adam))

1.5 Hyperparameter

hidden_layer_sizes에서 히든층의 개수는 한개로 유지한다.(2개이상이면 딥러닝) (i,)에서 i를 1에서 20까지, activation(identity, logistic, tanh, relu), solver(sgd, adam)로 각각을 변화시켜준다. solver중 sgd와 adam은 max_iter가 number of epochs로 결정된다.(수렴 조건으로 max_iter=10000, learning_rate_init=0.001 으로 둠.)
위의 조건으로 hyperparameter를 변경시켜가며 valid accuracy가 가장 높게 나오는 hyperparameter를 찾아보았다. valid accuracy가 가장 높게 나오는 sweetspot은(학습을 시킬때마다 다른 값이 나왔지만 공통적으로 나온 값) activation='identity', solver='sgd', hidden_layer_sizes=(2,), valid accuracy=0.9625 였다.

2 Conclusion

이 1.5에서 나온 hyperparameter로 계산해본 test accuracy=0.95가 나왔다. 또한 AUROC=0.978이 나왔다.

3 Python code in jupyter notebook

웹사이트 https://nbviewer.jupyter.org/ 에서 다음의 gist 주소를 입력하면

- https://gist.github.com/nosy0411/759e6440d450cfb571a3a3d074a44149

안깨진 assignment3.ipynb 파일 실행결과와 코드를 볼수 있다.

```
#1.2 Data load

df=pd.read_csv('dataset.csv')
df.head(10)
df.isnull().sum()
print(df['x1'].value_counts())
print(df['x2'].value_counts())
print(df['x3'].value_counts())
print(df['x4'].value_counts())
print(df['x5'].value_counts())

#1.3 Data preprocessing

X=df.iloc[:, :-1]
y=df.iloc[:,-1]
# X=X_df.values
# y=y_df.values
print("Type of X:", type(X))
print("Shape of X:", X.shape)
print("Type of y:", type(y))
print("Shape of y:", y.shape)

from sklearn.model_selection import train_test_split
X_train_and_valid, X_test, y_train_and_valid, y_test = train_test_split(
X, y, test_size=0.2, random_state=0, stratify=y)
print("X_train_and_valid shape:", X_train_and_valid.shape)
print("y_train_and_valid shape:", y_train_and_valid.shape)
print("==============================")
print("X_test shape:", X_test.shape)
print("y_test shape:", y_test.shape)
print("==============================")

X_train, X_valid, y_train, y_valid = train_test_split(
X_train_and_valid, y_train_and_valid, test_size=0.2, random_state=0, stratify=y_train_and_valid)
print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
print("==============================")
print("X_valid shape:", X_valid.shape)
print("y_valid shape:", y_valid.shape)

from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder

ct = ColumnTransformer(
[("scaling", StandardScaler(), ['x6', 'x7']),
("onehot", OneHotEncoder(sparse=False), ['x1','x2','x3','x4','x5'])])

ct.fit(X_train)
X_train_scale=ct.transform(X_train)
X_valid_scale=ct.transform(X_valid)
X_test_scale=ct.transform(X_test)
print(X_train_scale.shape)
print(X_valid_scale.shape)
print(X_test_scale.shape)
y_train=y_train.values
y_valid=y_valid.values
y_test=y_test.values

#1.4 Learning Algorithm(neural network) and #1.5 hyperparameter

from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score

hidden_layer_set=list(range(1,21))
activation_set = ['identity','logistic','tanh','relu']
solver_set=['sgd','adam']
data_dict=dict()

for a in activation_set:
  solver_dict=dict()
  for s in solver_set:
    training_accuracy=[]
```

```
    valid_accuracy=[]
    for n in hidden_layer_set:
      # build the model
      mlp=MLPClassifier(hidden_layer_sizes=(n,),
                        activation=a,
                        solver=s,
                        max_iter=10000,
                        learning_rate_init=0.001)
      mlp.fit(X_train_scale,y_train)

      y_train_hat = mlp.predict(X_train_scale)
      training_accuracy.append(accuracy_score(y_train, y_train_hat))

      y_valid_hat = mlp.predict(X_valid_scale)
      valid_accuracy.append(accuracy_score(y_valid, y_valid_hat))

    solver_dict[s]=dict(training=training_accuracy, valid=valid_accuracy)
  data_dict[a]=solver_dict

import pandas as pd

data=dict()

for a in activation_set:
  for s in solver_set:
    for n in hidden_layer_set:
      if data.get('activation'):
        data['activation'].append(a)
        data['solver'].append(s)
        data['hidden_layer_unit'].append(n)
        data['training_accuracy'].append(data_dict[a][s]['training'][n-1])
        data['valid_accuracy'].append(data_dict[a][s]['valid'][n-1])
      else:
        data['activation']=[a]
        data['solver']=[s]
        data['hidden_layer_unit']=[n]
        data['training_accuracy']=[data_dict[a][s]['training'][n-1]]
        data['valid_accuracy']=[data_dict[a][s]['valid'][n-1]]

df=pd.DataFrame(data, index=list(range(0,len(hidden_layer_set)*len(activation_set)*len(solver_set))),
columns=['activation','solver','hidden_layer_unit','training_accuracy','valid_accuracy'])

df

m=df['valid_accuracy'].max()
max_index=[i for i, j in enumerate(list(df['valid_accuracy'])) if j == m]
print(max_index, m)
print("==========================")
df.loc[max_index]

#2. Conclusion

mlp=MLPClassifier(hidden_layer_sizes=(2,),
                  activation='identity',
                  solver='sgd',
                  max_iter=10000,
                  learning_rate_init=0.001)
mlp.fit(X_train_scale,y_train)
y_test_hat = mlp.predict(X_test_scale)
print("test accuracy : ", accuracy_score(y_test, y_test_hat))

from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

y_test_score = mlp.predict_proba(X_test_scale)

fpr, tpr, _ = roc_curve(y_test, y_test_score[:, 1])
roc_auc = auc(fpr, tpr)

plt.figure()
plt.title('Receiver Operating Characteristic (ROC)')
plt.plot(fpr, tpr, 'b', label = 'ROC curve (area = %0.3f)' %roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1], [0,1], 'r--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.ylabel('True Positive Rate (sensitivity)')
plt.xlabel('False Positive Rate (1-specificity)')
plt.show()
```