Assignment 2

2015313254 노인호

October 13th 2019

1 Progress

1.1 Problem

Dataset 1을 선택해서 분석을 진행하였다. 1열은 GRE Scores, 2열은 TOEFL Scores, 3열은 University Rating, 4열은 SOP, 5열은 LOR, 6열은 CGPA, 7열은 Research, 그리고 마지막 열 8열은 Chance of Admit 이다. 1-7열의 feature값들을 이용하여 target이 되는 chance of Admit을 예측해보는 것이 이번 과제의 목표라고 볼 수 있다. 따라서 Regression 문제로 분류하고 다음 과정을 진행하였다.

1.2 Data Load

다운받은 csv파일을 dataframe type으로 저장하였다. 또한 데이터가 어떤 특성을 가지고 있는지 알아보기 위해 EDA(Exploratory Data Analysis)를 실시하였다.

먼저 데이터에 결측치가 있는지 조사하였다. df.isnull().sum()으로 조사를 해 본 결과 결측치는 하나도 없음을 알 수 있었다. 또한 feature들 간의 상관관계를 확인해보기 위해 df_corr과 시각적으로 확인하기 위해 seaborn 패키지를 이용하여 산점도 그래프를 출력해보았다. 그 결과 feature들 간 상관계수 값은 모두 높은 편이었고, feature중 3열의 University Rating(5가지), 4열의 SOP(5가지), 7열의 Research(2가지)는 범주형 변수임을 알 수 있었다.

1.3 Data preprocessing

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

- input variables(x) : GRE Scores, TOEFL Scores, University Rating, SOP, LOR, CGPA, Research
- output variables(y) : Chance of Admit

(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. 그래서 data set의 비율은 train : valid : test = 0.64 : 0.16 : 0.2 가 된다.

(3) Learning algorithm을 random forest로 설정하여서 data scaling이 필요 없기 때문에 진행하지 않았다. RF를 선택한 이유는 다음 항목에 설명하였다.

1.4 Learning Algorithm(Random Forest)

feature간 상관계수가 높고, 범주형 자료가 포함이 되어 있어 Linear Regression을 쓰기는 힘들다고 생각되었고 Decision Tree 단독으로 쓰기에는 overfitting의 문제가 있기 때문에 최종적으로 Random Forest방식을 쓰기로 결정하였다. Regression문제이므로 Scikit-learn의 RandomForestRegressor을 사용하였다. main parameter은 다음과 같다.

- n_estimators (the number of trees in the forest)
- max_features (the number of features to consider when looking for the best split)
- main hyperparameters of decision trees for pre-pruning(max_depth, min_samples_leaf)

1.5 Hyperparameter

n_estimators 값을 64-128개 까지 바꾸고, max_features 값을 2-7개, hyperparameters for pre-prunning 각각을 2-10개로 변화시켜 주었다. valid rmse가 가장 낮게 나오는 sweetspot은 n_estimators=78 max_features=2, max_depth =6, min_samples_leaf=2 이었다.

2 Conclusion

1.5에서 나온 hyperparameter로 계산해본 test rmse=0.07234가 나왔다. 또한 feature의 중요도를 feautre Importance로 계산해본 결과 CGPA, GRE Score, TOEFL Score, University Rating, LOR, SOP, Research의 순으로 중요함을 알 수 있었다.

3 Python code in jupyter notebook

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

- https://gist.github.com/nosy0411/08257cfc8ccf6f1bd0c6678ef6b182e9

안깨진 assignment2.ipynb 파일 실행결과와 코드를 볼수 있다.

```
#data load

df=pd.read_csv('dataset_1.csv')
df

#check missing figure

df.isnull().sum()

#check correlation

df.corr()

#scatter plot

import matplotlib.pyplot as plt
import seaborn as sns
cols1=["Chance of Admit ","GRE Score","TOEFL Score","University Rating",
"SOP","LOR ","CGPA","Research"]
sns.pairplot(df[cols1])
plt.show()

#train, valid, test dividing

X_df=df.iloc[:, :-1]
y_df=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)
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)

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)
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)

#Random Forest learing using each hyperparmeter and save rmse, r2 value

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

estimators_settings=list(range(64,129))
max_feature_settings = list(range(2, 8))
pruning_settings=list(range(2,11))
data_dict=dict()

for n in estimators_settings:
 feature_dict=dict()
  for f in max_feature_settings:
        p1_dict=dict()
        for p1 in pruning_settings:
                training_r2=[]
                training_rmse=[]
                valid_r2=[]
                valid_rmse=[]
                for p2 in pruning_settings:
                # build the model
                RFR=RandomForestRegressor(n_estimators=n,
```

```
                                        max_features=f,
                                        max_depth=p1,
                                        max_leaf_nodes=None,
                                        min_samples_leaf=p2,
                                        n_jobs=3,
                                        random_state=0)
                RFR.fit(X_train, y_train)

                y_train_hat = RFR.predict(X_train)
                training_rmse.append(mean_squared_error(y_train, y_train_hat)**0.5)
                training_r2.append(r2_score(y_train, y_train_hat))

                y_valid_hat = RFR.predict(X_valid)
                valid_rmse.append(mean_squared_error(y_valid, y_valid_hat)**0.5)
                valid_r2.append(r2_score(y_valid, y_valid_hat))
        p1_dict[p1]=dict(TRMSE=training_rmse, VRMSE=valid_rmse, TR2=training_r2, VR2=valid_r2)
  feature_dict[f]=p1_dict
 data_dict[n]=feature_dict

print(data_dict)

#organize in dataframe form

import pandas as pd

data=dict()


for n in estimators_settings:
 for f in max_feature_settings:
  for p1 in pruning_settings:
        for p2 in pruning_settings:
                if data.get('estimators'):
                        data['estimators'].append(n)
                        data['max_feature'].append(f)
                        data['max_depth'].append(p1)
                        data['min_samlples_leaf'].append(p2)

                        data['training_rmse'].append(data_dict[n][f][p1]['TRMSE'][p2-2])
                        data['valid_rmse'].append(data_dict[n][f][p1]['VRMSE'][p2-2])
                        data['training_r2'].append(data_dict[n][f][p1]['TR2'][p2-2])
                        data['valid_r2'].append(data_dict[n][f][p1]['VR2'][p2-2])
                else:
                        data['estimators']=[n]
                        data['max_feature']=[f]
                        data['max_depth']=[p1]
                        data['min_samlples_leaf']=[p2]

                        data['training_rmse']=[data_dict[n][f][p1]['TRMSE'][p2-2]]
                        data['valid_rmse']=[data_dict[n][f][p1]['VRMSE'][p2-2]]
                        data['training_r2']=[data_dict[n][f][p1]['TR2'][p2-2]]
                        data['valid_r2']=[data_dict[n][f][p1]['VR2'][p2-2]]

df=pd.DataFrame(data, index=list(range(0,len(estimators_settings)*6*9*9)),
columns=['estimators','max_feature','max_depth','min_samlples_leaf',
'training_rmse','valid_rmse','training_r2','valid_r2'])
df

#information about minimu value of valid RMSE and hyperparameter

m=df['valid_rmse'].min()
min_index=[i for i, j in enumerate(list(df['valid_rmse'])) if j == m]
print(min_index, m)

df.loc[min_index]

#test using upper hyperparameters

rfr=RandomForestRegressor(n_estimators=78,
max_features=2,
max_depth=6,
max_leaf_nodes=None,
min_samples_leaf=2,
n_jobs=3,
random_state=0)
rfr.fit(X_train, y_train)
y_test_hat = rfr.predict(X_test)
print("test_root_mean_squared_error_:_", mean_squared_error(y_test, y_test_hat)**0.5)


#check feature importance
```

```
import numpy as np
import matplotlib.pyplot as plt

featureImportance = rfr.feature_importances_

ynames=np.array(["GRE Score","TOEFL Score","University Rating","SOP","LOR ","CGPA","Research"])

featureImportance = featureImportance/featureImportance.max()
sorted_idx = np.argsort(featureImportance)
barPos = np.arange(sorted_idx.shape[0])+.5
plt.barh(barPos, featureImportance[sorted_idx], align='center')
plt.yticks(barPos, ynames[sorted_idx])
plt.xlabel('Variable Importance')
plt.show()
```