Лабораторна Робота №5

Тема: Дослідження методів ансамблевого навчання

Посилання на гітхаб: https://github.com/ViMIMercurysMight/python-ai.git

Завдання 2.1. Створення класифікаторів на основі випадкових та гранично випадкових лісів

*Лістинг програми*

import argparse  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier  
from sklearn.metrics import classification\_report  
from sklearn.model\_selection import train\_test\_split  
  
from utilities import visualize\_classifier  
  
  
def build\_arg\_parser():  
 parser = argparse.ArgumentParser(description='Classify \ data using Ensemble Learning techniques')  
 parser.add\_argument('--classifier-type',  
 dest='classifier\_type',  
 required=True,  
 choices=['rf', 'erf'],  
 help="Type of \ classifier to use; can be either 'rf' of 'erf'")  
 return parser  
if \_\_name\_\_ == '\_\_main\_\_':  
 args = build\_arg\_parser().parse\_args()  
 classifier\_type = args.classifier\_type  
  
 input\_file = 'data\_random\_forests.txt'  
 data = np.loadtxt(input\_file, delimiter=',')  
 X, y = data[:, :-1], data[:, -1]  
  
 class\_0 = np.array(X[y == 0])  
 class\_1 = np.array(X[y == 1])  
 class\_2 = np.array(X[y == 2])  
  
 plt.figure()  
 plt.scatter(class\_0[:, 0], class\_0[:, 1], s=75, facecolors='white', edgecolors='black', linewidth=1, marker='s')  
 plt.scatter(class\_0[:, 0], class\_1[:, 1], s=75, facecolors='white', edgecolors='black', linewidth=1, marker='o')  
 plt.scatter(class\_0[:, 0], class\_2[:, 1], s=75, facecolors='white', edgecolors='black', linewidth=1, marker='^')

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=5)  
  
 params = {'n\_estimators': 100, 'max\_depth': 4, 'random\_state': 0}  
  
 if classifier\_type == 'rf':  
 classifier = RandomForestClassifier(\*\*params)  
 else:  
 classifier = ExtraTreesClassifier(\*\*params)  
  
 classifier.fit(X\_train, y\_train)  
 visualize\_classifier(classifier, X\_train, y\_train)  
  
 y\_test\_pred = classifier.predict(X\_test)  
 visualize\_classifier(classifier, X\_test, y\_test)  
  
 class\_names = ['Class-0', 'Class-1', 'Class-2']  
 print("n" + "#" \* 40)  
 print('\nClassifier performance on traininig dataset\n')  
 print(classification\_report(y\_test, y\_test\_pred, target\_names=class\_names))  
 print("#" \* 40 + "\n")  
  
 test\_datapoints = np.array([5, 5], [3, 6], [6, 4], [7, 2], [4, 4], [5, 2])  
 print("Confidence measure:")  
 for datapoint in test\_datapoints:  
 probabilities = classifier.predict\_proba([datapoint])[0]  
 predicted\_class = 'Class-' + str(np.argmax(probabilities))  
 print('\nDatapoint:', datapoint)  
 print('Predicted class:', predicted\_class)  
  
 visualize\_classifier(classifier, test\_datapoints, [0] \* len(test\_datapoints))  
 plt.show()

Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generatedCalendar

Description automatically generatedCalendar

Description automatically generated

Рисунок 1. Результат виконання (rf, edge rf, erf, edge rf, rf консоль, erf консоль)

Завдання 2.2. Обробка дисбалансу класів

*Лістинг програми*

import sys  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.ensemble import ExtraTreesClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import classification\_report  
from utilities import visualize\_classifier  
  
# Завантаження вхідних даних  
input\_file = 'data\_imbalance.txt'  
data = np.loadtxt(input\_file, delimiter=',')  
X, y = data[:, :-1], data[:, -1]  
  
# Поділ вхідних даних на два класи на підставі міток  
class\_0 = np.array(X[y == 0])  
class\_1 = np.array(X[y == 1])  
  
# Візуалізація вхідних даних  
plt.figure()  
plt.scatter(class\_0[:, 0], class\_0[:, 1], s=75, facecolors='black',  
 edgecolors='black', linewidth=1, marker='x')  
plt.scatter(class\_1[:, 0], class\_1[:, 1], s=75, facecolors='white',  
 edgecolors='black', linewidth=1, marker='o')  
plt.title('Input data')  
# Розбиття даних на навчальний та тестовий набори  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(  
 X, y, test\_size=0.25, random\_state=5)  
  
# Класифікатор на основі гранично випадкових лісів  
params = {'n\_estimators': 100, 'max\_depth': 4, 'random\_state': 0}  
if len(sys.argv) > 1:  
 if sys.argv[1] == 'balance':  
 params = {'n\_estimators': 100, 'max\_depth': 4, 'random\_state': 0, 'class\_weight': 'balanced'}  
 else:  
 raise TypeError("Invalid input argument; should be 'balance'")  
  
classifier = ExtraTreesClassifier(\*\*params)  
classifier.fit(X\_train, y\_train)  
visualize\_classifier(classifier, X\_train, y\_train)  
  
y\_test\_pred = classifier.predict(X\_test)  
visualize\_classifier(classifier, X\_test, y\_test)  
  
# Обчислення показників ефективності класифікатора  
class\_names = ['Class-0', 'Class-1']  
print("\n" + "#" \* 40)  
print("\nClassifier performance on training dataset\n")  
print(classification\_report(y\_train, classifier.predict(X\_train), target\_names=class\_names))  
print("#" \* 40 + "\n")  
  
print("#" \* 40)  
print("\nClassifier performance on test dataset\n")  
print(classification\_report(y\_test, y\_test\_pred, target\_names=class\_names))  
print("#" \* 40 + "\n")  
  
plt.show()

Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated

Graphical user interface

Description automatically generatedChart, scatter chart

Description automatically generated

Calendar

Description automatically generated

Рисунок 2. Резльтат виконання

Завдання 3. Знаходження оптимальних навчальних параметрів за допомогою сіткового пошуку

*Лістинг програми*

import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from sklearn.model\_selection import GridSearchCV  
from sklearn.ensemble import ExtraTreesClassifier  
from sklearn.metrics import classification\_report  
input\_file = 'data\_random\_forests.txt'  
data = np.loadtxt(input\_file, delimiter=',')  
X, y = data[:, :-1], data[:, -1]  
  
# Розбиття даних на три класи  
class\_0 = np.array(X[y == 0])  
class\_1 = np.array(X[y == 1])  
class\_2 = np.array(X[y == 2])  
  
# Розбиття даних на навчальний та тестовий набори  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(  
 X, y, test\_size=0.25, random\_state=5)  
  
# Визначення сітки значень параметрів  
parameter\_grid = [{'n\_estimators': [100], 'max\_depth': [2, 4, 7, 12, 16]},  
 {'max\_depth': [4], 'n\_estimators': [25, 50, 100, 250]}  
 ]  
metrics = ['precision\_weighted', 'recall\_weighted']  
  
for metric in metrics:  
 print("\n##### Searching optimal parameters for", metric)  
  
 classifier = GridSearchCV(  
 ExtraTreesClassifier(random\_state=0),  
 parameter\_grid, cv=5, scoring=metric)  
 classifier.fit(X\_train, y\_train)  
  
 print("\nGrid scores for the parameter grid:")  
 for results in classifier.cv\_results\_:  
 print(results)  
 print("\nBest parameters:", classifier.best\_params\_)  
 y\_pred = classifier.predict(X\_test)  
 print("\nPerformance report:\n")  
 print(classification\_report(y\_test, y\_pred))

Text

Description automatically generatedText

Description automatically generated

Рисунок 3. Результат виконання (оптимуми precision\_weighted / recall\_weighted)

Завдання 4 Обчислення відносної важливості ознак

Під час виконання завдання, виникла проблема, а саме функцію було видалено з необхідного модуля. Згідно з рекомендаціями до виконання отриманими на лекції було вирішино пропустити дане завдання

Завдання 5. Прогнозування інтенсивності дорожнього руху за допомогою класифікатора на основі гранично випадкових лісів

*Лістинг програми*

import numpy as np  
from sklearn.metrics import mean\_absolute\_error  
from sklearn.model\_selection import train\_test\_split  
from sklearn import preprocessing  
from sklearn.ensemble import ExtraTreesRegressor  
  
input\_file = 'traffic\_data.txt'  
data = []  
with open(input\_file, 'r') as f:  
 for line in f.readlines():  
 items = line[:-1].split(',')  
 data.append(items)  
data = np.array(data)  
  
# Перетворення рядкових даних на числові  
label\_encoder = []  
X\_encoded = np.empty(data.shape)  
for i, item in enumerate(data[0]):  
 if item.isdigit():  
 X\_encoded[:, i] = data[:, i]  
 else:  
 label\_encoder.append(preprocessing.LabelEncoder())  
 X\_encoded[:, i] = label\_encoder[-1].fit\_transform(data[:, i])  
X = X\_encoded[:, :-1].astype(int)  
y = X\_encoded[:, -1].astype(int)  
  
# Розбиття даних на навчальний та тестовий набори  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(  
 X, y, test\_size=0.25, random\_state=5)  
  
# Регресор на основі гранично випадкових лісів  
params = {'n\_estimators': 100, 'max\_depth': 4, 'random\_state': 0}  
regressor = ExtraTreesRegressor(\*\*params)  
regressor.fit(X\_train, y\_train)  
# Обчислення характеристик ефективності регресора на тестових даних  
y\_pred = regressor.predict(X\_test)  
print("Mean absolute error:", round(mean\_absolute\_error(y\_test, y\_pred), 2))  
# Тестування кодування на одиночному прикладі  
test\_datapoint = ['Saturday', '10:20', 'Atlanta', 'no']  
test\_datapoint\_encoded = [-1] \* len(test\_datapoint)  
  
# Прогнозування результату для тестової точки даних  
print("Predicted traffic:", int(regressor.predict([test\_datapoint\_encoded])[0]))



Рисунок 4. Результат виконання

**Висновок:** Під час виконання лабораторної роботи дослідили методи ансамблів у машинному навчанні.