Университет ИТМО

Кафедра информатики и прикладной математики

Машинное обучение

Лабораторная работа №2

Деревья решений

Выполнили: Иппо Вера, группа P4117

Преподаватель:

Санкт-Петербург
2017

1. Цель работы: получить практические навыки работы с методом деревьев решений на практических примерах с использованием языка программирования python.

2. Исходные данные

Датасет: https://archive.ics.uci.edu/ml/datasets/Letter+Recognition

Предметная область: буквы латинского алфавиита

Задача: определить, какой из букв латинского алфавита соответствует набор характеристик ее написания.

Количество записей: 20000

Количество атрибутов: 16

Атрибуты:

1. lettr capital letter (26 values from A to Z)
2. x-box horizontal position of box (integer)
3. y-box vertical position of box (integer)
4. width width of box (integer)
5. high height of box (integer)
6. onpix total # on pixels (integer)
7. x-bar mean x of on pixels in box (integer)
8. y-bar mean y of on pixels in box (integer)
9. x2bar mean x variance (integer)
10. y2bar mean y variance (integer)
11. xybar mean x y correlation (integer)
12. x2ybr mean of x * x * y (integer)
13. xy2br mean of x * y * y (integer)
14. x-ege mean edge count left to right (integer)
15. xegvy correlation of x-ege with y (integer)
16. y-ege mean edge count bottom to top (integer)
17. yegvx correlation of y-ege with x (integer)

3. Ход работы

Код программы:

```
import pandas
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier

#загрузка датасета
def load_dataset(filename):
    csv_dataset = pandas.read_csv(filename, header=None).values
    dataset=csv_dataset
    return dataset

#разбиение на обучающую и тестовую выборки
def split_dataset(dataset,test_size):
    letter_attr = dataset[:,1:] # список атрибутов (признаков) для каждой буквы
    letter_class = dataset[:,0] # классы букв
    data_train, data_test, class_train, class_test = train_test_split(letter_attr, letter_class, test_size=test_size)
    return data_train, class_train, data_test, class_test

#обучение дерева и расчет точности
def train_and_score_tree(dataset,forest,size):
    data_train, class_train, data_test, class_test = split_dataset(dataset, size)
    forest = forest.fit( data_train, class_train )
    return forest.score(data_test, class_test)
```

```
def main():
    dataset=load_dataset("letter-recognition.csv")
    random_forest = RandomForestClassifier(n_estimators=100)
    decision_tree = DecisionTreeClassifier(random_state=100)
    # Получение средней точности классификации на тестовых данных
    print("Size of datasets \t Random forest \t Decision tree ")
    print( "60% train, 40% test:\t", train_and_score_tree(dataset,random_forest, 0.4),"\t", train_and_score_tree(dataset,decision_tree,0.4))
    print( "70% train, 30% test:\t", train_and_score_tree(dataset,random_forest, 0.3),"\t", train_and_score_tree(dataset,decision_tree,0.3))
    print( "80% train, 20% test:\t", train_and_score_tree(dataset,random_forest, 0.2),"\t", train_and_score_tree(dataset,decision_tree,0.2))
    print( "90% train, 10% test:\t", train_and_score_tree(dataset,random_forest, 0.1),"\t", train_and_score_tree(dataset,decision_tree,0.1))
main()
```

Результаты работы программы:

Size of datasets	Random forest	Decision tree
60% train, 40% test:	0.957375	0.86075
70% train, 30% test:	0.9545	0.868666666667
80% train, 20% test:	0.9615	0.87375
90% train, 10% test:	0.9615	0.887

4. Выводы

В ходе лабораторной работы были проведены эксперименты по классификации датасета с использованием двух алгоритмов: Decision tree и Random Aorest. В каждом эксперименте перераспределялись обучающая и тестовая выборки. В результате, можно сделать вывод, что алгоритм Random Forest более точен, чем дерево решений. При этом увеличение размера обучающей выборки после достижения какого-то порога, не дает существенного увеличения точность классификации.