نام و نام خانوادگی : صالح زارع زاده

شماره دانشجویی : ۶۱۰۳۹۶۱۰۹

پروژه داده کاوی

کد اصلی بنده که قسمت grid based clustering را انجام میدهد متاسفانه مقداری مشکل دارد که باعث میشود خروجی را به درستی ندهد.

در این قسمت توابع و کلاس های مهم کد را توضیح میدهم:

توسط csvToDictArray تابع که در parseCSV نوشته شده فایل های ورودی خوانده شده و به صورت یک لیست که شامل مقادیر هر ویژگی و اسامی ستون ها و مقادیر کل ورودی در میاید:

```
    return [data_set, attributes , valuesPerAttr]
```

در main در این قسمت تابع اصلی برای اجرای الگوریتم بین هر دو ویژگی اجرا میشود:

```
for i in range(len(attributes)-2):
    for j in range(i+1, len(attributes)-2):
        clusterTwoColumns(attributes[i], attributes[j], parsedData, min_den, grid
Size)
```

در Clusterdata در این قسمت تعداد پارتیشن ها و رنج ان ها تعیین میشود:

```
def partitionAttributes(values, partitionSize=200):
    """Divide values equally according to the specified partition size."""

    rangeDistance = (max(values) - min(values)) / (partitionSize + 1)

    ranges = [round(min(values) + (x * rangeDistance), 2)
              for x in range(partitionSize)]
    ranges.append(max(values))

    return ranges
```

در این قسمت خروجی ها چاپ میشود در فایل و توابع مهمی که برای الگوریتم نوشته شده استفاده میشوند:

```
def clusterTwoColumns(columnOneIdentifier, columnTwoIdentifier, parsedData, min_d
en, gridSize):
```

در این قسمت از تابع پاتیشن برای تعین مقادیر اولیه استفاده میشود:

```
xAxisRange = partitionAttributes(valuesPerAttr[columnOneIdentifier])
    yAxisRange = partitionAttributes(valuesPerAttr[columnTwoIdentifier])
```

و در این قسمت grid ساخته شده و نقطه های مربوط به دو ویژگی خاص به آن اضافه میشود:

```
data = [{columnOneIdentifier: item[columnOneIdentifier], columnTwoIdentifier:
            item[columnTwoIdentifier], "species": item['label_ <=50K'] } for ite
m in data_set]

    grid = Grid(gridSize, xAxisRange, yAxisRange)
    grid.buildGrid(min_den)
    grid.addPoints(data, columnOneIdentifier, columnTwoIdentifier)
```

در این قسمت هم سلول های چگال به دست امده و سلول ها به ترتیب چگالی به صورت نزولی مرتب میشوند و سپس سلول های چگال که به هم نزدیک هستند را با هم ترکیب کرده و بعد از ان سلول های uncertain (که در واقع نمیدانیم قرار است چگال باشند یا خیر) را به هم نزدیک هستند را ترکیب میکنیم:

```
# Gather and sort dense cells.
    grid.getDenseCells()
    grid.sortDenseCells()

    # Build clusters.
    clusters = grid.mergeCells()
    clusters = grid.mergeUncertainCells()
```

در utils.py توابع مربوط به قاصله و میانگین کلاستر ها و فاصله بین میانگین های ورودی تعیین شده است برای مثال :

```
def clusterMeans(cluster):
    """Calculates cluster mean."""

    xVal = sum([ point[0] for point in cluster ]) / len(cluster)
    yVal = sum([point[1] for point in cluster]) / len(cluster)

    return [xVal, yVal]


def eucideanDistance(obj1, obj2):
    """ Euclidean distance calculation between two objects."""
    xDissimilarity = (float(obj1[0] - float(obj2[0])))**2
    yDissimilarity = (float(obj1[1] - float(obj2[1])))**2

    return math.sqrt(xDissimilarity + yDissimilarity)
```

در Grid.py توابع مربوط به grid ها تعریف شده است که عبارت اند از :

برای ساختن grid ها از آن استفاده شده:

```
def buildGrid(self, min_den):
```

برای اضافه کردن نقطه ها به grid ها :

```
def addPoints(self, data, xAttr, yAttr):
```

برای گرفتن سلول های چگال :

```
def getDenseCells(self):
```

برای گرفتن سلولی که در خانه [x,y] است استفاده شده:

```
def getCellAtPosition(self, x, y):
```

برای مرتب کردن سلول ها به ترتیب چگالی به صورت نزولی برا ی پیدا کردن سلول های چگال و نویز :

```
def sortDenseCells(self):
```

برای ترکیب کردن سلول های چگال نزدیک به هم :

```
def mergeCells(self):
```

برای ترکیب کردن سلول های Uncertain :

```
def mergeUncertainCells(self):
```

در GridCell.py توابع مربوط به هر سلول grid تعیین شده است که توابع زیادی هستند برای مثال :

برای اضافه کردن یک نمونه به سلول از این تابع استفاده میشود:

```
    def addItem(self, item, xAttr, yAttr):
        """Adds item to cell."""
        item["xVal"] = item[xAttr]
        item["yVal"] = item[yAttr]
        item[xAttr] = self.xBin
        item[yAttr] = self.yBin
        self.items.append(item)
```

حال در VariousClassification.py برای مقایسه این جواب های با الگوریتم های Classification دیگر از کتابخانه sklearn استفاده شده و الگوریتم های GaussianNB و LogisticRegression و KNeighborsClassifier و RandomForestClassifier و StandardScaler استفاده شده و نتایج آن ها در زیر قرار داده شده است:

Best K Neighbors : Training score = 0.8022113022113022 and Test score = 0.8008844122343692 for 26 Neighbors

Gaussian Naive Bayes : Training score = 0.7945945945945946 and Test score = 0.7962166809974205

LogisticRegression : Training score = 0.796969696969697 and Test score = 0.7995332268763051

Random Forests : Training score = 0.9999590499590499 and Test score = 0.8530893010686648

SVM : Training score = 0.8529484029484029 and Test score = 0.8494042500921263

{algorithm': 'K Neighbors', 'training_score': 0.8022113022113022, 'testing_score': 0.8008844122343692'}

{algorithm': 'Gaussian Naive Bayes', 'training_score': 0.7945945945945946, 'testing_score': 0.7962166809974205'}

{algorithm': 'LogisticRegression', 'training_score': 0.796969696969697, 'testing_score': 0.7995332268763051'}

{algorithm': 'Random Forests', 'training_score': 0.9999590499590499, 'testing_score': 0.8530893010686648'}

{algorithm': 'SVM', 'training_score': 0.8529484029484029, 'testing_score': 0.8494042500921263'}

برای KNN تعدادی مقادیر مختلفی امتحان شده است و همان طور که میبینید بهترین آن ها انتخاب شده است و در واقع با بالا رفتن تعداد همسایه ها test_score و train_score به هم دیگر نزدیک میشنوند که در نمودار نشان داده شده است:

KNN : Training score = 0.9999590499590499 and Test score = 0.7406952462842402 for 1 Neighbors

KNN : Training score = 0.8321867321867322 and Test score = 0.7768087458543177 for 5 Neighbors

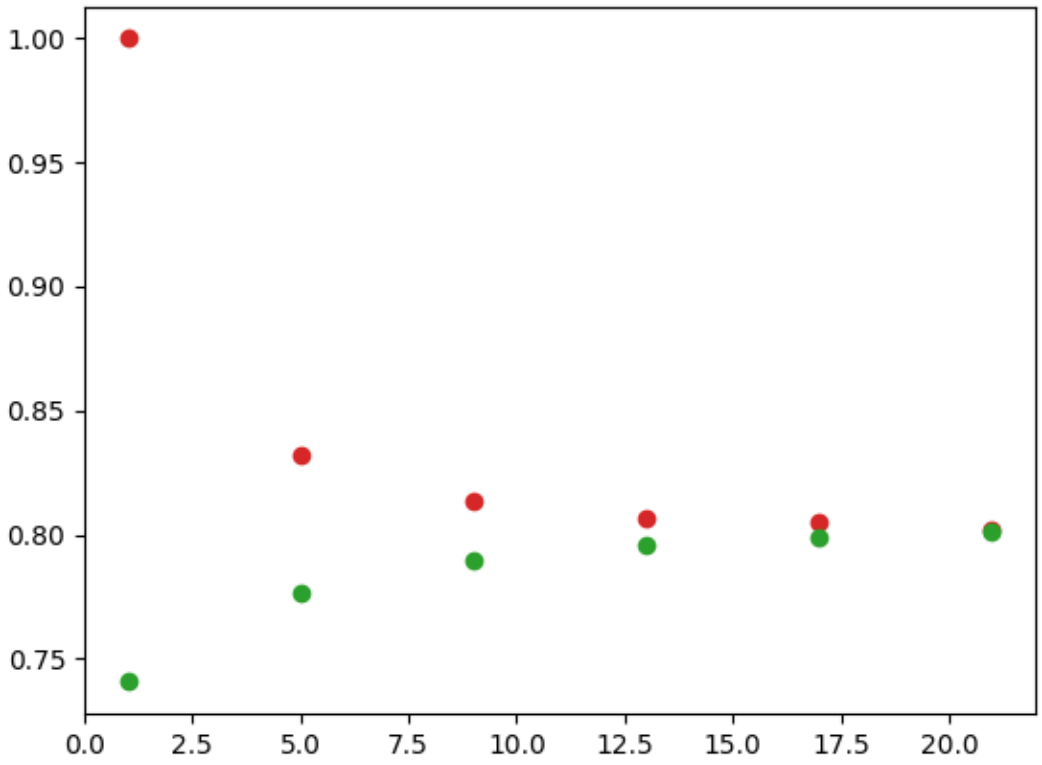
KNN : Training score = 0.8137182637182637 and Test score = 0.7898292593047537 for 9 Neighbors

KNN : Training score = 0.8062653562653562 and Test score = 0.7959710109323179 for 13 Neighbors

KNN : Training score = 0.8048730548730548 and Test score = 0.7989190517135487 for 17 Neighbors

KNN : Training score = 0.8022113022113022 and Test score = 0.8008844122343692 for 21 Neighbors

نمودار در صفحه بعدی قرار دارد :


1.00
0.95
0.90
0.85
0.80
0.75
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0