Софийски университет "Св. Климент Охридски"
Факултетът по математика и информатика

ДОКУМЕНТАЦИЯ

на
КУРСОВ ПРОЕКТ
по

Препоръчващи системи

от
Симеон Бонев, ф.н. M-24610
Петър Динев, ф.н. M-24913
Ганди Пирков, ф.н. M-24658
магистърска програма: Изкуствен интелект

юни, 2015

1. Задача

Избраната задача е от състезанието RecSys Challenge 2015. Входните данни представляват сървърни логове analytics от уеб приложение и включват кликове върху потребителския интерфейс на стоки от онлайн магазин. Всеки клик се характеризира със сесия, време, категория на стоката. Някои от сесиите водят до успешни покупки. Целта е да се предвиди дали ще се осъществи продажба и ако да кави стоки ще се закупят. От тази отправна точка се определя формула от състезанието за оценка доколко е успешна една система, използвайки тестово множество.

Web site: **http://2015.recsyschallenge.com/challenge.html**

2. Етапи на обработка

Предварителен анализ на данните. Dataset-а включва около 33 000 000 редове лог с кликове, които водят до около 9 000 000 уникални сесии.

Обработка на данните. За да усъществим пропоръчваща система използвайки Mahout трябваше да обратим данните в подходящ формат за да извлечем възможно на информативните атрибути за дадена сесия. За целта излозвахме MongoDB. С помощта му лесно успяхме да групираме и агрегиране данните.

Изследване на данните за зависимостти. Едно от основните цели беше да изледваме данните зависимостти в процеса на първоначална обработка. Главно изследване на корелации между атрибути, както и намиране на средно и дисперсия по атрибут. Идеята ни беше да разгледаме обучаващото множество по атрибути и да изразим статистическото им разпределение чрез boxplots.

Коефициент Корелация:

```
def correlation_coefficient(x, y):
        x_ = sum(x)/ float(len(x))
        y_ = sum(y)/ float(len(y))
        li = zip(x, y)

        return sum(map(lambda a: (a[0] - x_)*(a[1]- y_), li))/ sqrt(
                sum(map(lambda xi: pow(xi - x_, 2) ,x)) *
                sum(map(lambda yi: pow(yi - y_, 2), y))
```

Медиана:

```
def median_calc(groupBy, cumulative_dict, sorted_attr, length, inteval_length,
percentile):
        if not (percentile < 1 and percentile > 0):
        raise ValueError('Invalide percentile:' + str(percentile))
        median_unit = (length + 1) * percentile
        group_median = -1
        for cur in sorted_attr:
        index =sorted_attr.index(cur, )
        if index < len(sorted_attr) - 1 and cumulative_dict[cur] < median_unit and
median_unit <= cumulative_dict[sorted_attr[index + 1]]:
        group_median = sorted_attr[index + 1]
        lower_limit = lower_limit_calc(group_median, inteval_length)
        cumulative_frequency =
cumulative_dict[sorted_attr[sorted_attr.index(group_median) - 1]]
        return lower_limit + (median_unit - cumulative_frequency) * ( inteval_length /
float(groupBy[group_median]) )
```

Мода:

```
def mode_calc(groupBy, sorted_attr, inteval_length):
        group_mode = reduce(lambda x, y: x if groupBy[x] >= groupBy[y] else y,
sorted_attr)
        group_mode_index = sorted_attr.index(group_mode)
        group_mode_fr = groupBy[group_mode]
        group_mode_fr_before = groupBy[sorted_attr[group_mode_index-1]]
        group_mode_fr_after = groupBy[sorted_attr[group_mode_index+1]]
        lower_limit_mode = lower_limit_calc(group_mode, inteval_length)
        return lower_limit_mode + (
                        ((group_mode_fr - group_mode_fr_after)* INTERVAL_LENGTH) /
                        float( (group_mode_fr - group_mode_fr_after) +  (group_mode_fr -
group_mode_fr_before) )
                        )
```

Внасяне. Данните се внасят в нерелационна документна база от данни MongoDB.

mongoImportClicks.sh

```
        mongoimport -d challenge -c clicks --type csv --file yoochoose-clicks.dat --fields
sessionId,timestamp,itemId,category
```

mongoImportBuys.sh

mongoimport -d challenge -c buys --type csv --file yoochoose-buys.dat --fields sessionId,timestamp,itemId,price,quantity

Агрегиране. Понеже търсим свойствата на една сесия, които могат да доведат до успешна покупка. Първоначално агрегираме по сесии, така разполагаме с всички кликове.

```
function aggregateClicks() {
  print('Aggregate clicks');
  var a = db.sampleclicks.aggregate(
      [
       {
         $group :
           {
             _id : "$sessionId",
             clicks: {
               $push: {
                 timestamp: "$timestamp",
                 itemId: "$itemId",
                 category: "$category"
               }
             },
             clicksCount: {$sum: 1},
             distinctItems: {$addToSet: "$itemId"},
             distinctCategories: {$addToSet: "$category"},
             timestampArray: {$push: "$timestamp"},
             sessionStart: {$min: "$timestamp"},
             sessionEnd: {$max: "$timestamp"}
           }
       },
       {
         $out: "sampleClicksAggregated"
       }
      ],
       {
         allowDiskUse: true
       }
    );
  printjson(a);
}
```

Обогатяване. По време на агрегиране обогатяваме информацията с данни за брой кликс на сесия, брой уникални категории, брой уникални стоки, информация за началото и края на сесията.

```
function addMoreFields() {
  var a = db.clicksAggregated.aggregate(
    [
      {
```

```
            $project: {
                clicks: 1,
                clicksCount: 1,
                distinctItems: 1,
                distinctCategories: 1,
                timestampArray: 1,
                sessionStart: 1,
                sessionEnd: 1,
                numCat: {$size: "$distinctCategories"},
                numItems: {$size: "$distinctItems"},
            }
        },
        {
            $out: "clicksAggregatedMore"
        }
    ],
        {
            allowDiskUse: true
        }
);
printjson(a);
}
```

Добавяне на етикети. На целия dataset добавяме етикета bought: false, както и boughtNumber: 0. Чрез forEach заявка се добавят bought: true и се обновява boughtNumber от файла yoochoose-buys.dat

```
function addBuyStatus() {
    print('Adding sessionDuration');
    var p = 0;
    db.buys.find().forEach(
        function (elem) {
            p++;
            if(p%1000 === 0) {
                print("PROGRESS: ", p);
            }
            db.clicksAllFields.update(
                {
                    _id: elem.sessionId
                },
                {
                    $set: {
                        bought: true
                    },
                    $inc: {
                        boughtNumber: 1
                    }
                }
```

```
        );
      }
    );
}
```

Експорт. Колекцията се експортира до csv файл.

```
mongoexport --db challenge --collection clicksAllFields --csv --out mongoProcessedData.csv --fields
_id,clicksCount,sessionStart,sessionEnd,distinctCategories,numCat,numItems,bought,bought
```

Финална обработка. В Java приложение се обработва докрай dataset-а, като му се добавят дължина на сесията, ден от седмицата, час, месец, дали е в работно време, дали е след работа, дали е през уикенд, дали е през нощта, дали е кликвана празна категория, дали е кликвана основна категория, дали е кликвана специфична категория и дали е кликвана промоция.

CSVProcessor.java

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.util.Calendar;
import java.util.Date;

import javax.xml.bind.DatatypeConverter;

import org.json.JSONArray;
import org.json.JSONException;

public class CSVProcessor {
        static int sessionId = 0;
        static int clicksCount = 1;
        static int sessionStart = 2;
        static int sessionEnd = 3;
        static int distinctCategories = 4;
        static int numCat = 5;
        static int numItems = 6;
        static int bought = 7;
        static int boughtNumber = 8;

        public static void main(String[] args) throws IOException {
```

```
		BufferedReader bufferedReader = null;
		BufferedWriter bufferedWriter = null;
		Calendar calendar = Calendar.getInstance();
		try {
			bufferedReader = new BufferedReader(
					new FileReader(

"/Users/monkeybusiness/Documents/RecSys/workspace/mongoProcessedData.csv"));
			File fout = new File(

"/Users/monkeybusiness/Documents/RecSys/workspace/testNotBroken.csv");
			FileOutputStream fos = new FileOutputStream(fout);

			bufferedWriter = new BufferedWriter(new OutputStreamWriter(fos));

			String line = "";
			String splitter = ",";
			String header = "";
			boolean skipOne = true;
			while ((line = bufferedReader.readLine()) != null) {
				if (skipOne) {
					header = line;
					skipOne = false;
					bufferedWriter
							.write(line
									+
",duration,dayOfWeek,month,hour,isWeekend,isDuringWork,isAfterWork,isAtNight,hasNoCat,ha
sGroupCat,hasSpecificCat,hasPromoCat");
					bufferedWriter.newLine();
					continue;
				}
				// use comma as separator
				String[] entity = line.split(splitter);
				// System.out.println("sessionId" + " " + entity[sessionId]);
				// System.out.println("clicksCount" + " " +
				// entity[clicksCount]);
				// System.out.println("sessionStart" + " "
				// + entity[sessionStart].replaceAll("^\"|\"$", ""));
				// System.out.println("sessionEnd" + " " + entity[sessionEnd]);
				// System.out.println("distinctCategories" + " "
				// + entity[distinctCategories]);
				// System.out.println("numCat" + " " + entity[numCat]);
				// System.out.println("numItems" + " " + entity[numItems]);
```

```
                        // System.out.println("bought" + " " + entity[bought]);
                        // System.out.println("boughtNumber" + " " +
                        // entity[boughtNumber]);

                        Date sStart = DatatypeConverter.parseDateTime(
                                        entity[sessionStart].replaceAll("^\"|\"$", ""))
                                        .getTime();
                        Date sEnd = DatatypeConverter.parseDateTime(
                                        entity[sessionEnd].replaceAll("^\"|\"$",
"")).getTime();
                        long duration = (sEnd.getTime() - sStart.getTime()) / 1000;

                        calendar.setTime(sStart);
                        int dayOfWeek = calendar.get(Calendar.DAY_OF_WEEK);
                        int month = calendar.get(Calendar.MONTH);
                        int hour = calendar.get(Calendar.HOUR);
                        boolean isWeekend = dayOfWeek == 1 || dayOfWeek == 7
                                        || (dayOfWeek == 6 && hour > 18);
                        boolean isDuringWork = dayOfWeek > 1 && dayOfWeek < 7
                                        && hour > 9 && hour < 17;
                        boolean isAfterWork = dayOfWeek > 1 && dayOfWeek < 7
                                        && hour > 17 && hour < 23;
                        boolean isAtNight = hour > 0 && hour < 7;

                        boolean hasNoCat = false;
                        boolean hasGroupCat = false;
                        boolean hasSpecificCat = false;
                        boolean hasPromoCat = false;

                        JSONArray jsonArray = null;
                        String jaString = "";
                        try {
                                jaString = entity[distinctCategories]
                                                .replaceAll("^\"|\"$", "").replaceAll("\"\"S\"\"",
                                                                "-1").replaceAll("\"\"S\"",
"-1").replaceAll("\"S\"\"", "-1").replaceAll("\"S\"", "-1");
                                jsonArray = new JSONArray(jaString);
                        } catch (JSONException je) {
                                System.out.println("Broken line: "
                                                + line
                                                + "###"
```

```
                                                + entity[distinctCategories] + "&&&" +
entity[distinctCategories].replaceAll("^\"|\"$", "") + "%%%" +
entity[distinctCategories].replaceAll("^\"|\"$", ""));
                                        continue;
                                }
                                int[] categories = new int[jsonArray.length()];
                                for (int i = 0; i < jsonArray.length(); i++) {

                                        categories[i] = jsonArray.optInt(i);
                                        if (categories[i] < 0) {
                                                hasPromoCat = true;
                                        }
                                        if (categories[i] == 0) {
                                                hasNoCat = true;
                                        }

                                        if (categories[i] >= 1 && categories[i] <= 12) {
                                                hasGroupCat = true;
                                        }

                                        if (categories[i] > 12) {
                                                hasSpecificCat = true;
                                        }
                                }

                                bufferedWriter.write(line + "," + duration + "," + dayOfWeek
                                                + "," + month + "," + hour + "," + isWeekend + ","
                                                + isDuringWork + "," + isAfterWork + "," + isAtNight
                                                + "," + hasNoCat + "," + hasGroupCat + ","
                                                + hasSpecificCat + "," + hasPromoCat);
                                bufferedWriter.newLine();

                        }
                } catch (Exception e) {
                        e.printStackTrace();
                } finally {
                        if (bufferedReader != null) {
                                bufferedReader.close();
                        }

                        if (bufferedWriter != null) {
                                bufferedWriter.close();
                        }
```

```
                System.out.println("Done");
            }
        }
}
```

3. Създаване на модел

Подходи -

- **Decision Tree** - Избрахме едни от подходите с които да разботим да бъде Decision Tree. Тъй като mahout сам дискретизира атрибуте си, пропуснахме тази стъпка. Решихме да изпозваме този подход тъй като дава доста добри резултати. А и предположихме, че ще е добър при работа с clicks, а и освен това дава много добри резултати по принцип.

- **Същина -** Decision tree представлява модел за взимане на решения като се отговаря на въпроси свързани с атрибутите на разглежданата същност (entity). След успешно трениране, то придобива по оптимална форма като намира взаимовръзка между отделни атрибути на изходното множество.

- **Random Forest** и **Decision Forest** - са разширени подходи, които използват възможностите на decision tree и надграждат над тях, като водят до по-леки изчислителни задачи, или избягват ефекти като overfitting. Използва се ансамблово обучение, като задачата е класификационна. Класифицираме в два класа “покупка” и “без покупка”. Използват се наблюденията от научния труд на **Leo Breiman.** Използват се методи като bagging (случайното избиране на атрибути), което е част от Random Subspace Method или случайната проекция на атрибути от датасет.
- **Зареждане на данните в Apache Mahout.**

```
String descriptor = "I N I I I N N L I N N N N C C C C C C C C";
                String[] trainDataValues =
fileAsStringArray("/Users/monkeybusiness/Documents/RecSys/workspace/testNotBroken.csv");

                Data data = DataLoader.loadData(
                        DataLoader.generateDataset(descriptor, false, trainDataValues),
                        trainDataValues);
```

- **Изграждане на дървета**

```
int numberOfTrees = 100;
                DecisionForest forest = buildForest(numberOfTrees, data);

                String[] testDataValues = testFileAsStringArray("data/test.csv");
                Data test = DataLoader.loadData(data.getDataset(), testDataValues);
```

```
Random rng = RandomUtils.getRandom();

int m = (int) Math.floor(Maths.log(2, data.getDataset().nbAttributes()) + 1);

DefaultTreeBuilder treeBuilder = new DefaultTreeBuilder();
treeBuilder.setM(m);

return new SequentialBuilder(RandomUtils.getRandom(), treeBuilder,
        data.clone()).build(numberOfTrees);
```

- **Класификация**

```
for (int i = 0; i < test.size(); i++) {
    Instance oneSample = test.get(i);

    double classify = forest
            .classify(test.getDataset(), rng, oneSample);
    int label = data.getDataset().valueOf(0,
            String.valueOf((int) classify));

    System.out.println("label: " + label);
```

4. Библиография

Breiman, L. (2001). Random Forests. Statistics Department, University of California Berkeley, CA 94720

Needham, M (2012). http://www.markhneedham.com/blog/2012/10/27/kaggle-digit-recognizer-mahout-random-forest-attempt/