

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ
ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
Обнинский институт атомной энергетики –
филиал федерального государственного автономного образовательного учреждения высшего
образования «Национальный исследовательский ядерный университет «МИФИ»
(ИАТЭ НИЯУ МИФИ)

Отделение интеллектуальных кибернетических систем

ЛАБОРАТОРНАЯ РАБОТА №3
«Анализ набора данных»
по дисциплине
«Большие данные»

Выполнил студент 1 курса
группы ИВТ-М20
Лискунов Р. Г.

Проверил:
кандидат технических наук
Грицюк С. В.

Цель работы

Проанализировать набор данных (популярные детские имена) с помощью Spark SQL (работа на одной машине с несколькими потоками без полноценной среды Hadoop) и визуализировать при помощи библиотеки Vegas.

Краткая теория

Spark SQL – это модуль Apache Spark, интегрирующий реляционную обработку данных и процедурный API Spark. Spark SQL является частью ядра Spark с версии 1.0. Он может работать совместно с Hive (HiveQL/SQL) или замещать его.

Благодаря Spark SQL, функционал фреймворка получает два ключевых дополнения. Во-первых, модуль обеспечивает тесную интеграцию между реляционной и процедурной обработкой данных посредством интеграции декларативного DataFrame API и процедурного API Spark. Во-вторых, он включает в себя расширяемый оптимизатор, созданный на языке Scala, обладающем широкими возможностями сопоставления с образцом (pattern matching), что позволяет легко формировать правила, управлять генерацией кода и создавать расширения.

Spark SQL и DataFrame

DataFrame – это распределенная коллекция данных, организованных посредством именованных столбцов. Данная абстракция предназначена для выборки, фильтрации, агрегации и визуализации структурированных данных.

DataFrame поддерживает глубокую реляционную/процедурную интеграцию в рамках программ Spark и позволяет манипулировать данными как с помощью процедурного API Spark, так и посредством нового реляционного API, обеспечивающего более эффективную оптимизацию. DataFrame может быть создан непосредственно из RDD, что обеспечивает возможность реляционной обработки уже имеющихся данных.

DataFrame предоставляет более удобные и эффективные средства обработки данных, чем процедурный API Spark. В частности, можно вычислить несколько агрегаций за один проход с мощностью SQL-инструкции, что достаточно сложно реализовать посредством традиционного процедурного API.

В отличие от RDD, DataFrame отслеживает свою схему и поддерживает различные реляционные операции, что обеспечивает более оптимизированное выполнение. DataFrame формирует схему посредством отражения (reflection).

DataFrame является «ленивой» структурой данных, то есть содержит логический план для вычисления набора данных, при этом вычисления не выполняются до тех пор, пока пользователь не

запросит специальную «операцию вывода», например, сохранение. Такой подход обеспечивает эффективную оптимизацию всех операций.

Концепция DataFrame расширяет модель RDD. В результате, благодаря упрощенным методам фильтрации и агрегации, Spark-разработчики получают возможность быстрее и эффективнее работать с большими наборами структурированных данных.

Ход работы

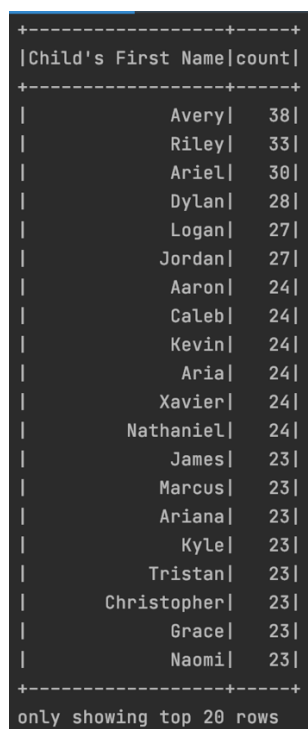
В процессе работы мы рассмотрим набор данных, состоящий из популярных детских имен. Прежде всего мы ознакомимся с структурой данной информации. Набор данных из 19418 строк представляет из себя файл с расширением CSV (Comma-Separated Values — значения, разделённые запятыми). Его заголовок содержит следующие поля:

- | | |
|------------------|-----------------------|
| 1. Year of Birth | 4. Child's First Name |
| 2. Gender | 5. Count |
| 3. Ethnicity | 6. Rank |

Набор данных имеет следующую структуру:

```
StructType(StructField(Year of Birth,StringType,true), StructField(Gender,StringType,true),  
StructField(Ethnicity,StringType,true), StructField(Child's First Name,StringType,true),  
StructField(Count,StringType,true), StructField(Rank,StringType,true))
```

Приведём пример на рисунке 1 топ 20 самых детских популярных имен в этом наборе данных.



Child's First Name	count
Avery	38
Riley	33
Ariel	30
Dylan	28
Logan	27
Jordan	27
Aaron	24
Caleb	24
Kevin	24
Aria	24
Xavier	24
Nathaniel	24
James	23
Marcus	23
Ariana	23
Kyle	23
Tristan	23
Christopher	23
Grace	23
Naomi	23

Рисунок 1. Топ 20 детских имён

Остановимся поподробнее на столбце №1, который называется «Год рождения». Исследуем его корреляцию с количественным показателем числа имен, которые встречаются в данном наборе. Взаимосвязь представлена на рисунке 2.

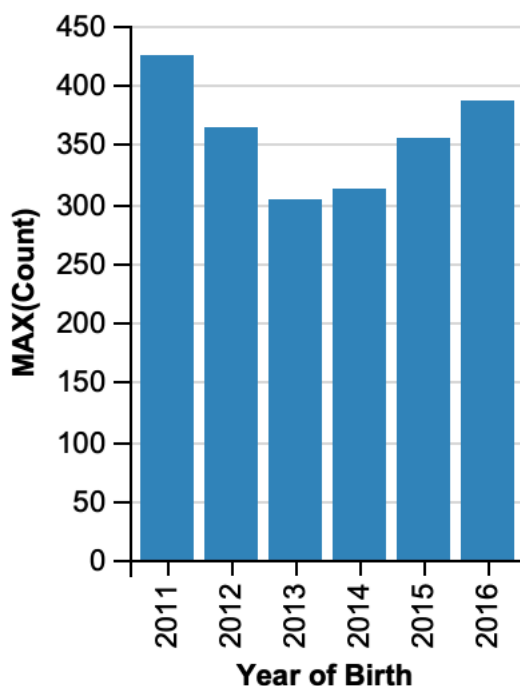


Рисунок 2. Взаимосвязь года и числа имён

Данное наблюдение демонстрирует нам, что с 2011 год число детей, появившихся на свет, начало уменьшаться вплоть до 2013. Можем предположить, что с 2013 и далее (2016 год – последний в данном наборе) положение демографии в городе Нью-Йорк начало улучшаться.

Для того, чтобы начать дополнительный анализ данной взаимосвязи, совершим общий подсчёт числа этнических происхождений в наборе данных (рисунок 3).

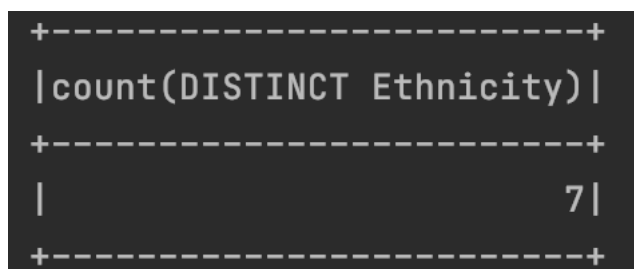


Рисунок 3. Число уникальных этнических происхождений

Как известно, в мире существует четыре основных этноса: WHITE, BLACK, ASIAN, HISPANIC, но исследование, проведенное над текущими данными, показывает нам, что составители набора выделили несколько больше групп этнических происхождений. Рассмотрим подробнее: построим

график связи между значением этноса и ранга на рисунке 4, чтобы определить смысловую нагрузку выделения большего числа происхождений.

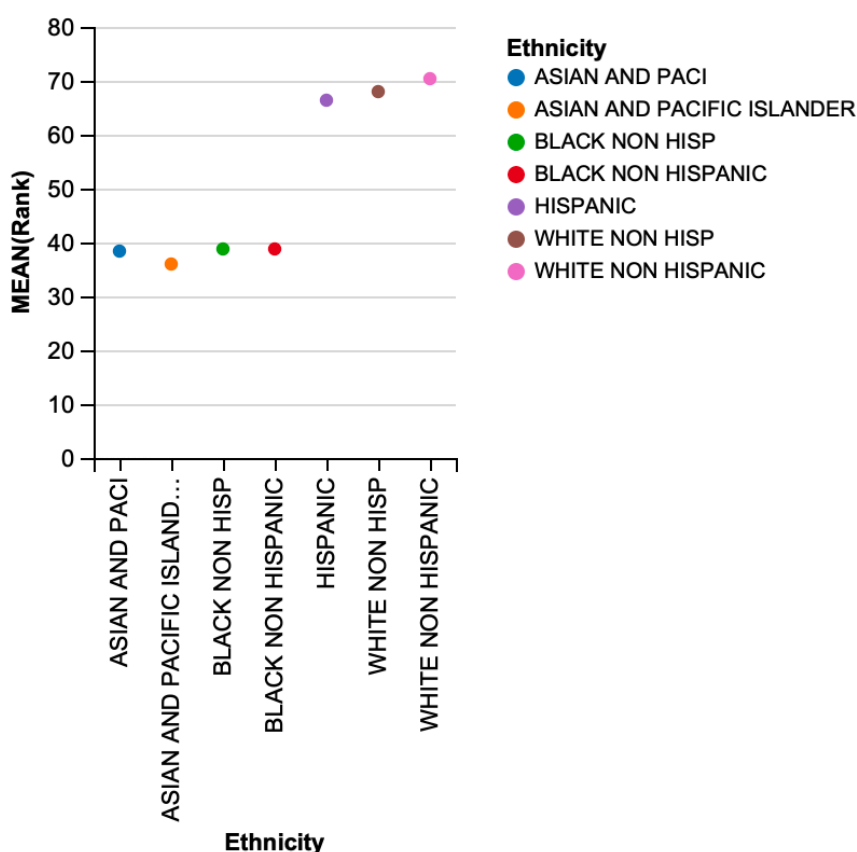


Рисунок 4. Взаимосвязь этноса и ранга

Данный график демонстрирует, что набор данных разделяет классические четыре типа этноса на более мелкие этнические группы, выделяя латиноамериканских граждан. Поскольку метаданные описывают этот набор как информацию, переданную от федеральной службы города Нью-Йорк, то можем предположить, что данная выборка является более точной с точки зрения представления числа качественной информации. Сжатие данных может привести к потере учётной информации числа определенных групп с непохожим этническим происхождением.

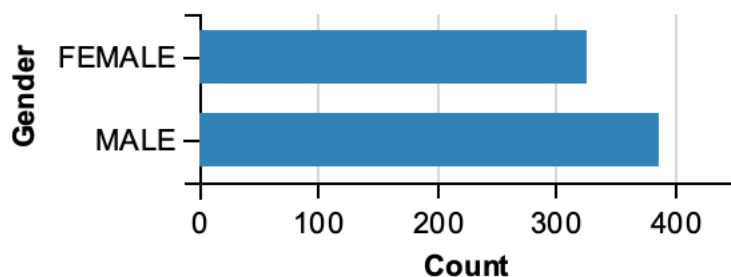


Рисунок 5. Взаимосвязь гендера и числа детей

Для того, чтобы завершить исследование рассмотрим разницу между числом родившихся девочек и мальчиков. Как известно, в мире больше людей женского пола, но данный график для города Нью-Йорк за 2011-2016 является исключением для этого статистического правила.

Листинг кода

```
package LabThree

import org.apache.log4j.Level.WARN
import org.apache.log4j.LogManager
import org.apache.spark.sql.functions.{countDistinct, _}
import org.apache.spark.sql.{DataFrame, SparkSession}
import vegas._
import vegas.sparkExt._

object LabThree {
  val PATH: String = "src/main/data"
  val NODES: Int = 3

  def main(args: Array[String]): Unit = {
    val spark: SparkSession = SparkSession.builder()
      .appName("Lab3")
      .master(s"local[$NODES]")
      .getOrCreate
    LogManager.getRootLogger.setLevel(WARN)

    val dataframe: DataFrame = spark.read
      .format("csv")
      .option("header", "true")
      .option("delimiter", ",")
      .load(s"$PATH/var.csv")

    dataframe.show(false)

    Vegas("Children_Info")
      .withDataFrame(dataframe)
      .encodeX(field = "Year of Birth", dataType = Nominal)
      .encodeY(field = "Count", dataType = Quantitative, aggregate = AggOps.Max)
      .mark(Bar)
      .show

    Vegas("Children_Info")
      .withDataFrame(dataframe)
      .encodeX(field = "Ethnicity", dataType = Nominal)
      .encodeY(field = "Rank", dataType = Quantitative, aggregate = AggOps.Mean)
      .encodeColor(field = "Ethnicity", dataType = Nominal)
      .mark(Circle)
      .show
  }
}
```

```
Vegas("Children_Info")
  .withDataFrame(dataframe)
  .encodeX(field = "Count", dataType = Quantitative, sortOrder = SortOrder.Asc)
  .encodeY(field = "Gender", dataType = Nominal)
  .mark(Bar)
  .show
```

```
Vegas("Children_Info")
  .withDataFrame(dataframe)
  .encodeX(field = "Rank", dataType = Quantitative, sortOrder = SortOrder.Asc)
  .encodeY(field = "Gender", dataType = Nominal)
  .mark(Bar)
  .show
```

```
// Number of unique ethnic groups
dataframe.agg(countDistinct("Year of Birth")).show()
// Top 20 most popular names
dataframe.groupBy("Child's First Name").count().sort(col("count").desc).show()
}
}
```

Вывод

В ходе данной лабораторной работы я изучил особенности и возможности способов работы со Spark SQL и Vegas. Для этого я использовал набор данных популярных детских имен. Подробно остановился на взаимосвязи между этническим происхождением и некоторыми другими параметрами. Исследование показало, что самым популярным именем было Avery из этноса WHITE NON HISPANIC.