Trends in Police Traffic Stops in the United States

Rutuja Gurav, Krupa Hegde and Abhay Singh
Department of Computer Science and Engineering
University of California, Riverside
{rgura001,khegd001,asing073}@ucr.edu

***Abstract*— Across the United States, police officers make more than 50,000 traffic stops on a typical day. Data of these traffic stops is stored in a variety of formats and with high inconsistency across states. This renders the data inaccessible for cross-state and nation-wide statistical analysis. A significant effort has been spent in collecting and standardizing the data from several states, spanning approximately 60 million stops between 2010 and 2015. We analyzed this data to gain insights into factors affecting these stops. We use the Big-data Analysis platform- Apache Spark, for our analysis tasks.**

I. Introduction

During a traffic stop, a police officer ideally registers certain information about the stop consisting of stop date, stop time, stop location, driver's age, gender, race, type of violation committed by the driver, for example, drunken driving, speeding, etc. A stop may lead to a search of the driver's vehicle in which case the police officer also registers whether any contraband was found in the vehicle and whether the driver was arrested.

We seek to analyze the trends in the number of stops conducted over years and the factors affecting these stops like the age, race and gender of the driver. We also further investigate if any of the mentioned factors even weakly predict the likelihood of getting arrested, searched or being found in possession of a contraband.

We have used Apache Spark and spark.ml to perform the analysis tasks and visualized the data using tools like Microsoft Power maps. In the following sections we describe about the dataset we used, nature of the data, background of the systems ans tools used, analysis tasks and experimental results.

II. Police Traffic Stops Dataset

A. Data Collection

- Stanford Open Policing Project- The Stanford Open Policing Project, is an interdisciplinary team of researchers and journalists at Stanford University, committed to combining the academic rigor of statistical analysis with the explanatory power of data journalism. The Stanford Open Policing Project is collecting and standardizing data on vehicle and pedestrian stops from law enforcement departments across the country and were making that information freely available. The project has already gathered 130 million records from 31 state police agencies and have begun collecting data on stops from law enforcement agencies in major cities, as well.

B. Nature of the Data

To assemble a national dataset of traffic stops, they first identified which state law enforcement agencies electronically maintain traffic stop records that, at a minimum, include the race of the stopped driver. Of the the 50 state agencies, 7 did not respond to the Stanford group's request for information, 9 agencies did not compile stop records electronically; and 3 state agencies keep electronic records but do not track the race of stopped drivers. An additional 11 states provided data that was insufficient to assess racial disparities, and 19 states have not provided any data (including Hawaii and Alaska) on each stop conducted since 2005.

This shows that it is incredible hard to collect, clean and standardize this huge amount of data and highlights the characteristics of *Volume* and *Variety* of Big Data.

III. Background

A. Apache Spark for Big-Data Processing

Apache Spark is a fast and general-purpose cluster computing system designed to perform computations that need to iterate over distributed data. Although being written in Scala, It provides APIs that can be written in Java, Scala, Python and R, and an optimized engine that supports general execution graphs. Since it was developed, there have been multiple framework additions on top of the aforementioned features. It supports a rich set of higher-level tools including but not limited to Spark SQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming. Spark can run on top of any data distributed storage systems like HDFS, Cloudera and Google Big Query. We decided to run our operations with Spark because spark has RDDs that make performing read-only tasks at least 10x times faster according to paper[11].

1) Spark Architecture: Spark has a layered architecture with all components loosely coupled and integrated together with a different libraries and extensions.

The spark driver program runs on the master node of the spark cluster and its main job is to schedule the job execution and negotiate with the cluster manager. On the other hand the worker performs all of the data processing.

The most basic and most important abstractions in Spark are:

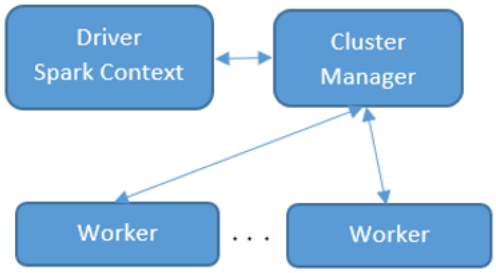

Fig. 1. Spark Architecture

- Resilient Distributed Datasets (RDD): We will discuss this in greater detail in the forthcoming section.
- Directed Acyclic Graph (DAG): Apache Spark like most distributed data processing frameworks, follows a master/slave architecture with two main daemons and a cluster manager namely, a Master Daemon (Master/Driver Process) and a Worker Daemon (Slave Process). A spark cluster comprises of a single Master and any given number of Slaves/Workers. The driver and the executors run their individual processes and users run them on the same horizontal spark cluster or on separate machines i.e. in a vertical spark cluster or in mixed machine configuration.

2) Spark RDD: RDDs basically are a partitioned collection of records that are read only. RDDs provide an interface based coarse grained transformations: map, filter and join to name a few. They highly inefficient for a broad array of applications where we need to reuse the data. RDDs can be created only by deterministic operations on stable storage data or on other RDDs. Users are only allowed to alter two aspects of RDDs i.e. persistence and partitioning. Spark allows users to use RDDs through an API.

3) Spark SQL: Spark SQL is a component of the spark distribution that sits on top of Spark Core introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data. It allows us to run SQL queries directly on the data. When running SQL from within another programming language the results will be returned as a Dataset/DataFrame. You can also interact with the SQL interface using the command-line or over JDBC/ODBC.

4) SparkSQL: DataFrame API: A Dataset is a distributed collection of data. Dataset interface was added to spark after version 1.6. It provides the benefits of RDDs (strong typing, ability to use powerful lambda functions) along with the benefits of Spark SQLs optimized execution engine. A DataFrame is a special Dataset that has been organized into named columns. It is conceptually equivalent to a table in terms of relational database terminology. DataFrames can be constructed from structured data files, tables in Hive, external databases, or existing RDDs. In the Scala API, DataFrame is simply a type alias of Dataset $< Row >$. While, in Java API, users need to use Dataset$< Row >$ to represent a DataFrame.

5) Spark MLlib: MLlib is apache sparks machine leaning library which was started at AMP lab at UC Berkley. It supports multiple languages like Python, Java and Scala. As machine learning tasks are iterative in nature, using spark is a better option. The traditional map reduce becomes slower because of serialization and deserialization overhead.Spark provides its machine learning API in two ways. It has RDD based library spark.MLlib and a Dataframe based library spark.ml. The recent versions of MLlib mentions Dataframe based API is the primary API. BY choosing Dataframes over RDD it becomes easier to use different components of spark like Spark SQL or spark GraphX. It is also helpful in writing the machine learning pipelines. Some of the tools provided by MLlib are listed below:

- Algorithms: common learning algorithms such as classification, regression, clustering, and collaborative filtering
- Featurization: feature extraction, transformation, dimensionality reduction, and selection.
- Pipelines: tools for constructing, evaluating, and tuning ML Pipelines. Persistence: saving and load algorithms, models, and Pipelines
- Utilities: linear algebra, statistics, data handling, etc.

IV. Analysis Tasks

A. Selected Tasks

This section describes the tasks performed on the dataset to gain insight about the data and find interesting pattern and results. Tasks are defined as follows.

- Perform aggregations to find Yearly number of stops at the national level or state levels to see increase or decrease in trend
- Number of stops conducted for different age groups, genders or race.
- Compare how various states fare in number of stops.
- Decision tree to predict likelihood of events:
- Build a logistic regression model to see if age, race or sex determines, even weakly the possibility of being arrested, being searched or having found a contra band
- Geo-coding the locations of the stops in each state using Google Maps API.

B. Implementation Details

This section describes the implementation and detailed explanations of the analysis tasks defined in the previous section.

1) Perform aggregations to find Yearly number of stops at the national level or state levels to see increase or decrease in trend: To know the yearly number of stops, simple group by queries are written in sparkSQL in Dataframe level. This data was joined with census data to check the percentage of population that was stopped for each state. Results showed number of people stopped are very less compared to the

population of the state. Results section of this paper shows more interesting results obtained by this data. The main task in obtaining these results was to write a Sparl SQL query.

2) Number of stops conducted for different age groups, genders or race: Analysis on trends in stops that took place by race, gender and age has been performed. The dataset of Traffic Stops for a given state was joined with census data for a particular year. To normalize our results, we aggregated percentage by population of the particular demographic in that state. We also aggregated percentage by cumulative stops in the state to see how a particular demographic was represented.

3) Decision tree to predict likelihood of events: Decision trees are built for individual states separately, to see if a person is stopped will he be searched or be arrested?, given the race,gender and age. The dataset we have is not consistent in representation and most of the columns are categorical. spark.ml supports binary and multi class classification. Input columns to the decision tree are label columns and feature column. For some of the cities, the age group is discrete and in some datasets the age is grouped. Other columns like race or search conducted are either in string format or in boolean. StringIndexer is used to encoded a string column of labels or features to a column of label indices. By using vectorAssembler feature columns are converted to vectors. By using Bucketizer, age column is transferred. By using bucketizer, age has been converted to groups based on the split we gave. Age is divided into below 20, 2-50 and above 50. features converted to vectors are then transformed to vector indices using VectorIndexerModel using spark.ml VectorIndexer. Data is split for training and testing by 0.7 : 0.3 ratio. A pipeline is constructed to chain the indexer, decision tree model and LabelConverter. Decision tree model is trained by using training set data and DecisionTreeClassifier tool given by the spark.ml. After building the model the test data is passed to the model and predictions are made.

The data is highly unbalanced. Percentage of people who were searched are very less when compared to those who were stopped but not searched. It is difficult to get good results with the imbalance data. Decision tree built on this data shows false label for all the test cases. A sample if else ladder built on Colorado state data is shown in fig-7. Results of the model are discussed in the result section of this paper. Decision trees discussed here are all built on dataframes.

4) Build a logistic regression model: see if age, race or sex determines, even weakly the possibility of being arrested, being searched or having found a contra band Logistic regression is usually used to predict a binary outcome which in our case is an answer if the driver was arrested or not, if the driver was searched or not or if contraband was found on the driver or not. We use age, sex and race of the driver as a predictor and build a model to see if any of the predictors can with a certainty predict Logistic regression models in spark are built by using binomial logistic regression, or it can be used to predict a multiclass outcome by using multinomial logistic regression. With the help of the family parameter we can select between these two algorithms, or leave it unset and Spark will infer the correct variant. The following figure is a short demonstration on how to build a logistic model. Most of other implementation code is simillar to that of building the decision tree.

```
// Load training data
val training = spark
    .read
    .format("libsvm")
    .load("file.txt")
// Object of the model
val lr = new LogisticRegression()
  .setMaxIter(10)
  .setRegParam(0.3)
  .setElasticNetParam(0.8)
//train on data
val lrModel = lr.fit(training)

// Print the coefficients and intercept
println("Coefficients: ${lrModel.coefficients}"
    +"Intercept: ${lrModel.intercept}")
```

Fig. 2. Building a Logistic regression model

5) Geo-coding the locations of the stops in each state using Google Maps API: The dataset shows the county and the raw location where the stop occurred. Attempt was made to use these raw locations to get latitude and longitude of the location by using Google API. However, the raw locations provided in the dataset are not accurate so the results we got from Google API were defined in a boundary which was large to make decision on the exact location. We took the raw location, county name and state name as input to the Google API.

V. Experimental Results

In this section, we describe some of the experimental results. Total number of stops conducted in each state were calculated and this data was joined with census information to see the percentage of population being stopped. Our experimental results show that the number of stops conducted are significantly less compared to the population of the cities. This suggests that only a small fraction of the population experiences traffic stops by police.

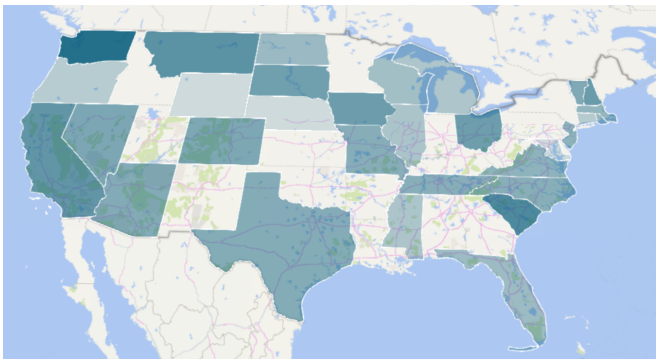

Fig. 3. Number of Stops in 2015

Figure 3 shows the heat-map of the United States for the number of stops conducted in 2015. Washington and South Carolina are the states where the maximum number of stops occurred in 2015.

For some of the states like California, stops have become less. Figure 4 shows the traffic stops in California from 2010-2015. The number of stops conducted over the span of 5 years shows a decreasing trend.

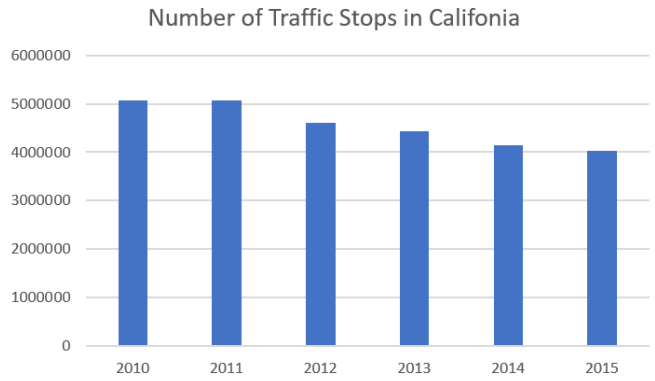

Fig. 4. Total Stops in California

We compared two states with large populations viz. California and Texas.

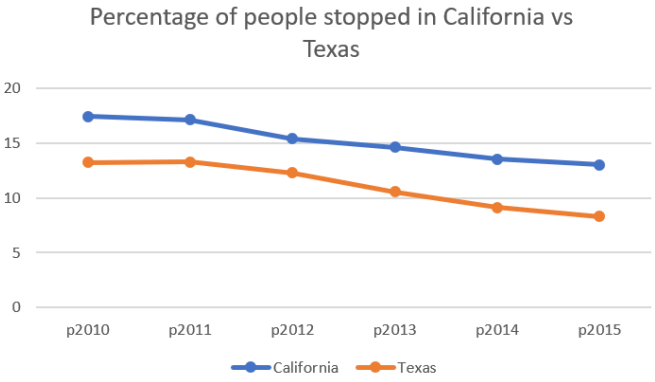

Fig. 5. Total Stops in California vs Texas

Figure 5 shows the trend for California and Texas. Percentage of stops in California are more than those in Texas. The interesting fact is, in both the states, there is defined decrease in the number of stops as shown in the trend line. We observed similar decrement in other states also. There are some states where there is not significant amount of change in the trend line.

In the year 2015 in California, Black and Hispanics were searched more among the other race group in the traffic stops. We can see the percentage of different race groups searched in 2015 in California in the figure 6.

Decision trees have been modeled for various states as explained in the Analysis of task section. Figure 7 shows the

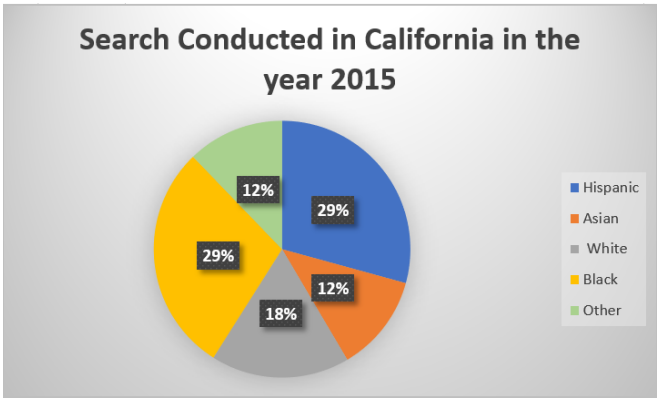

Fig. 6. Percentage of stops leading to vehicle search by race in California

if-else ladder constructed for the dataset of Colorado state. Note that the test error of the model for Colorado decision tree was 0.0039620514864309175. The weighted precision of 0.9920915948791192 was achieved by the model. Even though the Accuracy and precision are higher it doesn't justify the correctness of the model as recall is low. Some sampling has been done on the data to boost the minority label. However that is not included in this paper.

Thirty one logistic regression models were built for each state that the data was available. After the model converged, models for all the sates returned a coefficient matrix with only 0.0 as the elements for all the tests(arrested, searched and contraband found). Figure 8 shows the results for Ohio state for the reader's understanding. Data for Ohio state traffic stop did not have the column age so there are only two elements in the coefficient matrix. It can be seen that all the coefficients are 0.0 implying that they do not even weakly predict the outcomes for getting searched, Contraband found and getting arrested. The regression model built is a line parallel to the predictor axis. Thus the Logistic regression model could not predict an outcome using the given features: age, sex and race.

Analysis done by demographics features such as race, gender and age of the driver yielded some interesting insights into the nature of stops being conducted.

Figure 9 shows number of stops conducted in California by gender from 2010 till 2015. It can be seen that the number of stops conducted have decreased over the years and there are less number of stops conducted on female drivers than male drivers.

Figures 10, 11 and 12 are time series plots which depict the evolution of the nature of stops conducted over the years from 2010 to 2015 in three different states viz. California, Texas and Washington. These are the states which ranked amongst the highest in the number of stops conducted as compared to other states.

The time series plots provide an interesting insight into the way stops are conducted and the influence that the driver's race has on the stop event happening. Distinct repeated motifs can be observed in all three states which suggests

```
DecisionTreeClassificationModel (uid=dtc_98b99e9431c7) of depth 5 with 41 nodes
  If (feature 1 in {2.0})
   If (feature 0 in {1.0,2.0,3.0,4.0})
    If (feature 2 in {1.0})
     If (feature 0 in {3.0,4.0})
      Predict: 0.0
     Else (feature 0 not in {3.0,4.0})
      If (feature 0 in {2.0})
       Predict: 0.0
      Else (feature 0 not in {2.0})
       Predict: 0.0
    Else (feature 2 not in {1.0})
     If (feature 0 in {3.0,4.0})
      If (feature 0 in {4.0})
       Predict: 0.0
      Else (feature 0 not in {4.0})
       Predict: 0.0
     Else (feature 0 not in {3.0,4.0})
      If (feature 0 in {2.0})
       Predict: 0.0
      Else (feature 0 not in {2.0})
       Predict: 0.0
   Else (feature 0 not in {1.0,2.0,3.0,4.0})
    If (feature 2 in {1.0})
     Predict: 0.0
    Else (feature 2 not in {1.0})
     Predict: 0.0
  Else (feature 1 not in {2.0})
   If (feature 0 in {1.0,2.0,3.0,4.0})
    If (feature 2 in {1.0})
     If (feature 1 in {0.0})
      If (feature 0 in {4.0})
       Predict: 0.0
      Else (feature 0 not in {4.0})
       Predict: 0.0
     Else (feature 1 not in {0.0})
      If (feature 0 in {3.0,4.0})
       Predict: 0.0
      Else (feature 0 not in {3.0,4.0})
       Predict: 0.0
    Else (feature 2 not in {1.0})
     If (feature 1 in {0.0})
      If (feature 0 in {3.0,4.0})
       Predict: 0.0
      Else (feature 0 not in {3.0,4.0})
       Predict: 0.0
   Else (feature 1 not in {0.0})
    If (feature 0 in {3.0,4.0})
     Predict: 0.0
    Else (feature 0 not in {3.0,4.0})
     Predict: 0.0
 Else (feature 0 not in {1.0,2.0,3.0,4.0})
  If (feature 2 in {1.0})
   If (feature 1 in {0.0})
    Predict: 0.0
   Else (feature 1 not in {0.0})
    Predict: 0.0
  Else (feature 2 not in {1.0})
   If (feature 1 in {0.0})
    Predict: 0.0
   Else (feature 1 not in {0.0})
    Predict: 0.0
```

Fig. 7. Decision Tree

```
STATE: OH_cleaned
Search Conducted
Coefficients: 0.0  0.0   Intercept: -Infinity
CB Found
Coefficients: [0.0,0.0] Intercept: -Infinity
Arrested
Coefficients: [0.0,0.0] Intercept: -5.9293358008598105
```

Fig. 8. Coefficients and intercepts for Ohio State

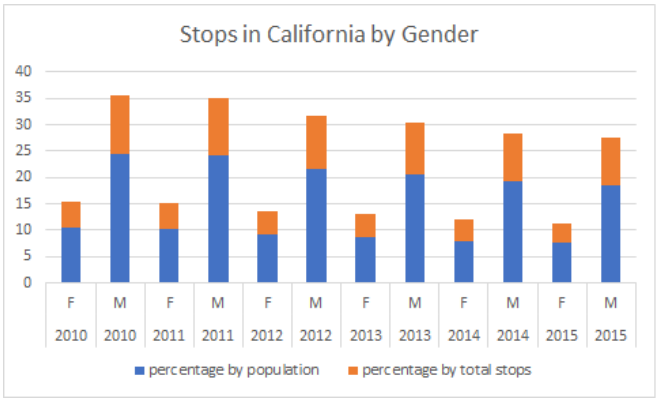

Fig. 9. Stops in California by Gender

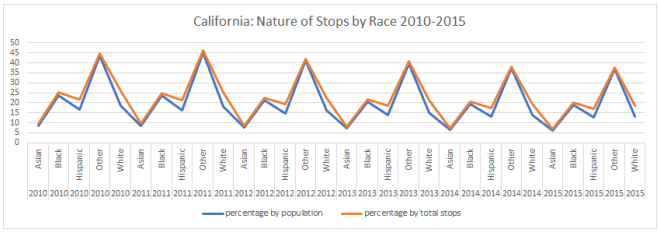

Fig. 10. Nature of Stops By Race in California.

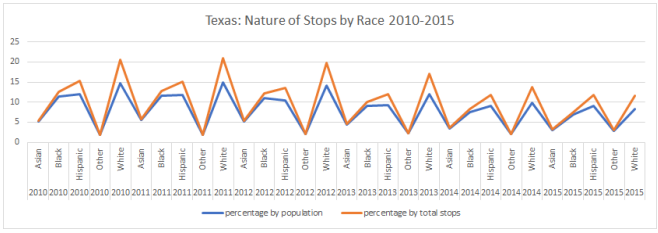

Fig. 11. Nature of Stops By Race in Texas.

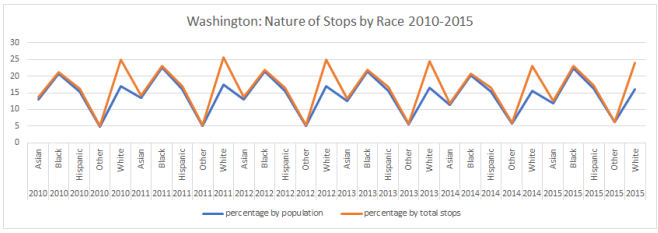

Fig. 12. Nature of Stops By Race in Washington.

that although the number of stops conducted over the years has slightly declined, the driver's who are Black, Hispanic or of Other races are stops more than driver's of Asian or White race. This disparity is preserved over the 5 year span that we have analyzed for three states.

VI. Related Work

The Stanford group has invested significant efforts into collecting, cleaning and standardizing the data in an automated manner. They have devised sophisticated statistical techniques[3][4] to analyze phenomenon such as racial bias[1][5]and discrimination.[2]

VII. Conclusion

Various analysis tasks have been performed on Stanford open policing data.To understand the trends in traffic stops, aggregation tasks have been performed. Number of people

stopped in each state their race, gender and po These resulted in some interesting patterns in the data. In few states some sort of racial bias have been found. Logistic Regression model is built to predict the binary outcomes and to know can age, gender race be weak features to predict. More analysis and some more interesting results can be found here- Results .

REFERENCES

[1] E. Pierson, C. Simoiu, J. Overgoor, S. Corbett-Davies, V. Ramachandran, C. Phillips, S. Goel. (2017) A Large-Scale Analysis Of Racial Disparities In Police Stops Across The United States. Stanford University.

[2] Sharad Goel, Maya Perelman, Ravi Shroff and David Alan Sklansky(2017) Combatting Police Discrimination In The Age Of Big Data. Stanford University.

[3] Emma Pierson Sam Corbett-Davies Sharad Goel Fast Threshold Tests for Detecting Discrimination

[4] The Problem Of Infra-Marginality In Outcome Tests For Discrimination by Camelia Simoiu, Sam Corbett-Davies and Sharad Goel, Stanford University.

[5] Sharad Goel, Justin M. Rao And Ravi Shroff Precinct Or Prejudice? Understanding Racial Disparities In New York Citys Stop-And-Frisk Policy, Stanford University.

[6] Jongbin Jung, Connor Concannon, Ravi Shro, Daniel G. Goldstein and Sharad Goel Simple Rules for Complex Decisions, Stanford University.

[7] Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel and Aziz Huq Algorithmic Decision Making and the Cost of Fairness

[8] Sharad Goel, Justin M. Rao, and Ravi Shroff Personalized Risk Assessments in the Criminal Justice System

[9] Wenjie Xing and Ming Guan, "The Research Proposal of Racial Inequalities against Asian American in College Admission based on Regression Model", ACMDL

[10] K. Shvachko, H. Kuang, S. Radia and R. Chansler, "The Hadoop Distributed File System," 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), Incline Village, NV, 2010, pp. 1-10. doi: 10.1109/MSST.2010.5496972

[11] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauly, Michael J. Franklin, Scott Shenker, Ion Stoica: "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing."NSDI 2012: 15-28

[12] Xiangrui Meng, Joseph Bradley, Burak Yavuz , Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, Doris Xin, Reynold Xin, Michael J. Franklin, Reza Zadeh, Matei Zaharia and Ameet Talwalkar, MLlib: Machine Learning in Apache Spark

[13] Shivaram Venkataraman, Zongheng Yang1, Davies Liu2, Eric Liang2, Hossein Falaki, Xiangrui Meng, Reynold Xin, Ali Ghodsi, Michael Franklin, Ion Stoica1and Matei Zaharia,"SparkR: Scaling R Programs with Spark" AMPLab UC Berkeley, Databricks Inc., MIT CSAIL

[14] Bar Akgn, ule Gndz dc, Streaming Linear Regression on Spark MLlib and MOA

[15] Mohan, Pradeep, et al. "Should SDBMS support a join index?: a case study from CrimeStat." Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems. ACM, 2008.

[16] Shekhar, Shashi, et al. Crime pattern analysis: A spatial frequent pattern mining approach. No. TR-12-015. MINNESOTA UNIV MINNEAPOLIS DEPT OF COMPUTER SCIENCE AND ENGINEERING, 2012.