Predicting Traffic Stop Outcomes

Sri Santhosh Hari, Ker-Yu Ong, Masha Vasilenko, Yi Qiang Zhao
MSAN 697 Final Project

Objective

Our goal for this analysis was to predict whether or not a traffic or pedestrian stop would result in an arrest or citation.

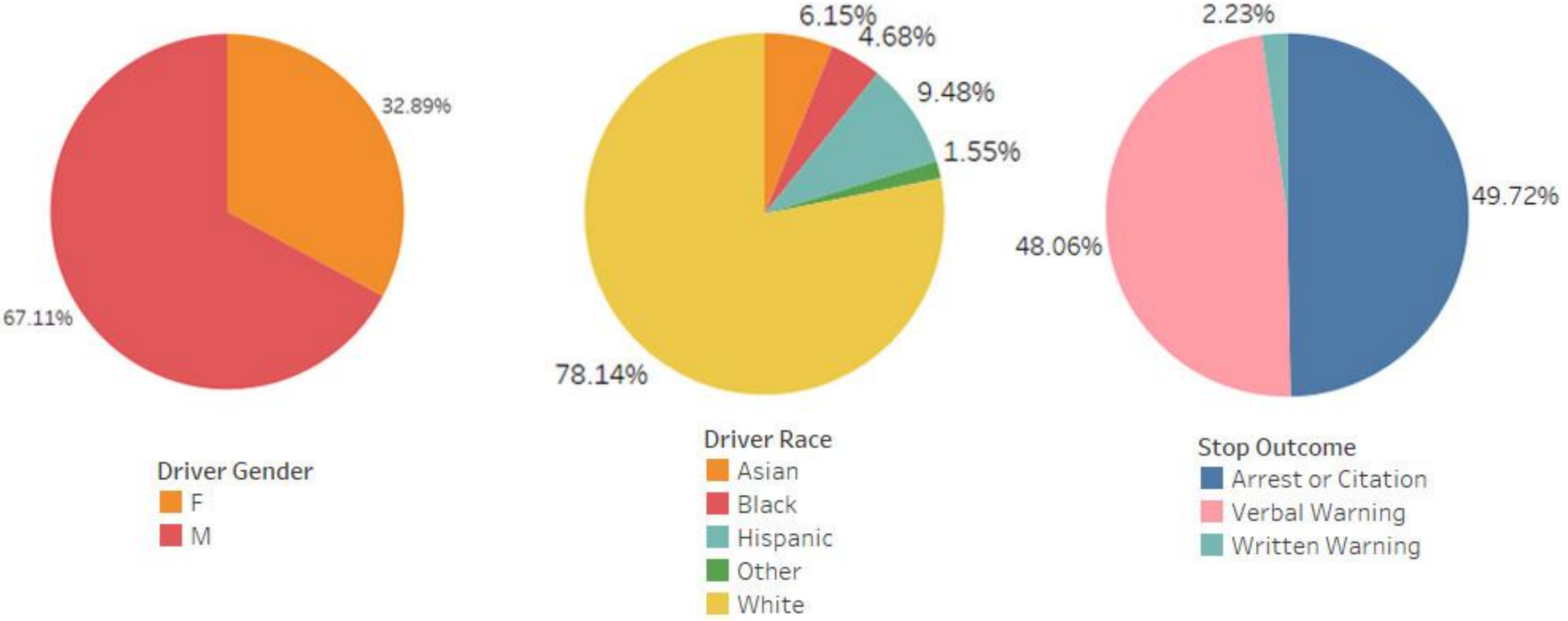
We hypothesized that attributes related to the driver and/or traffic police are predictive of stop outcome.

This is interesting from a sociological perspective and can be used as an awareness tool to address bias in policing.

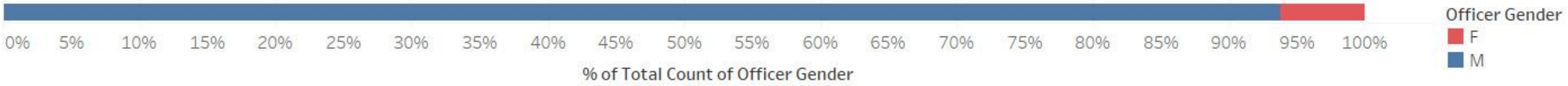
Data Description

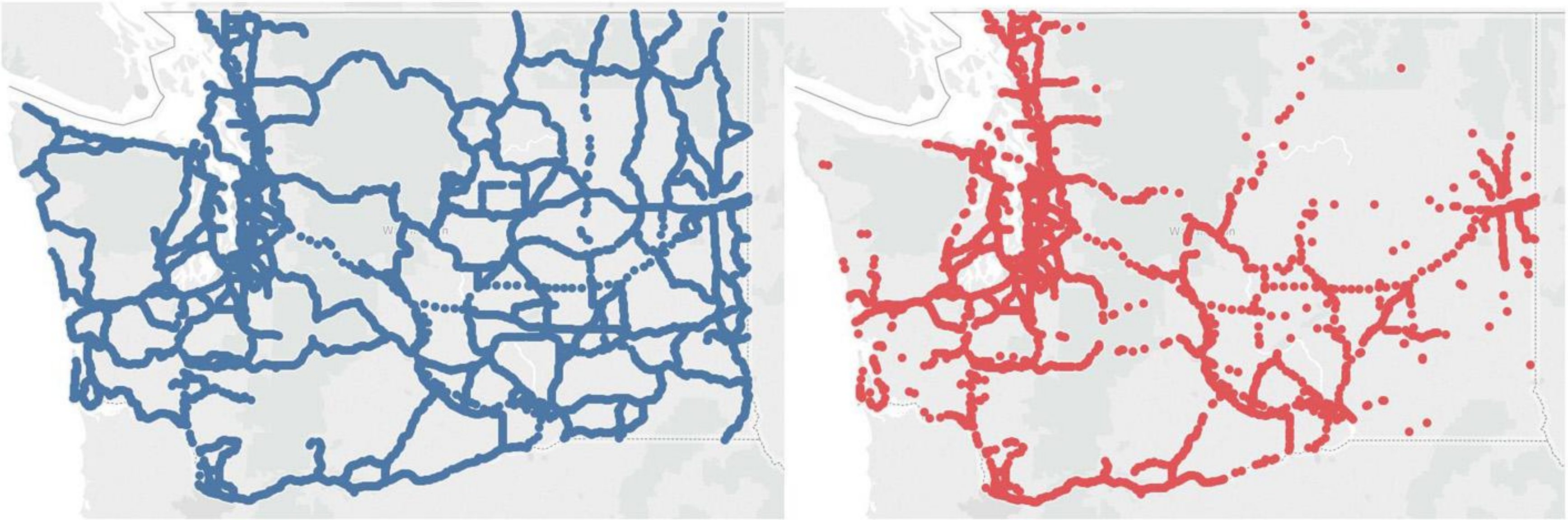
- 8.6M traffic stops from 2009-2016 in Washington State
- Source: The Stanford Open Policing Project
- Format: csv
- Sample fields: stop_date, driver_gender, driver_age, driver_race, officer_gender, officer_race, violation, contraband_found, highway_type, stop_outcome,

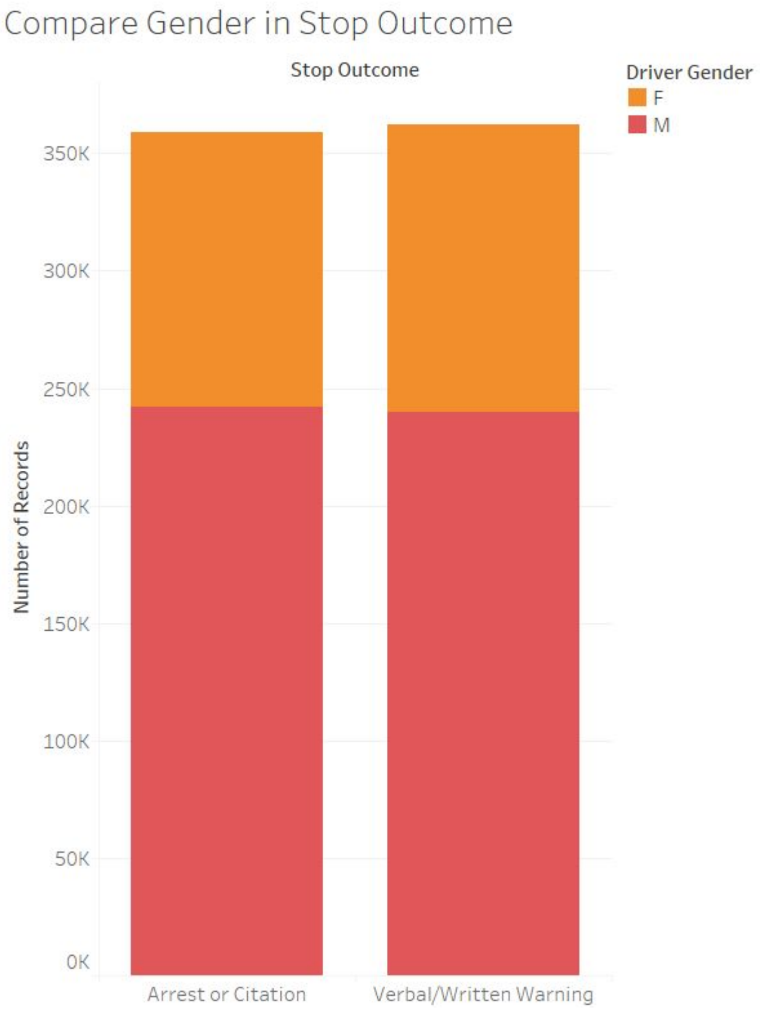
Exploratory Data Analysis

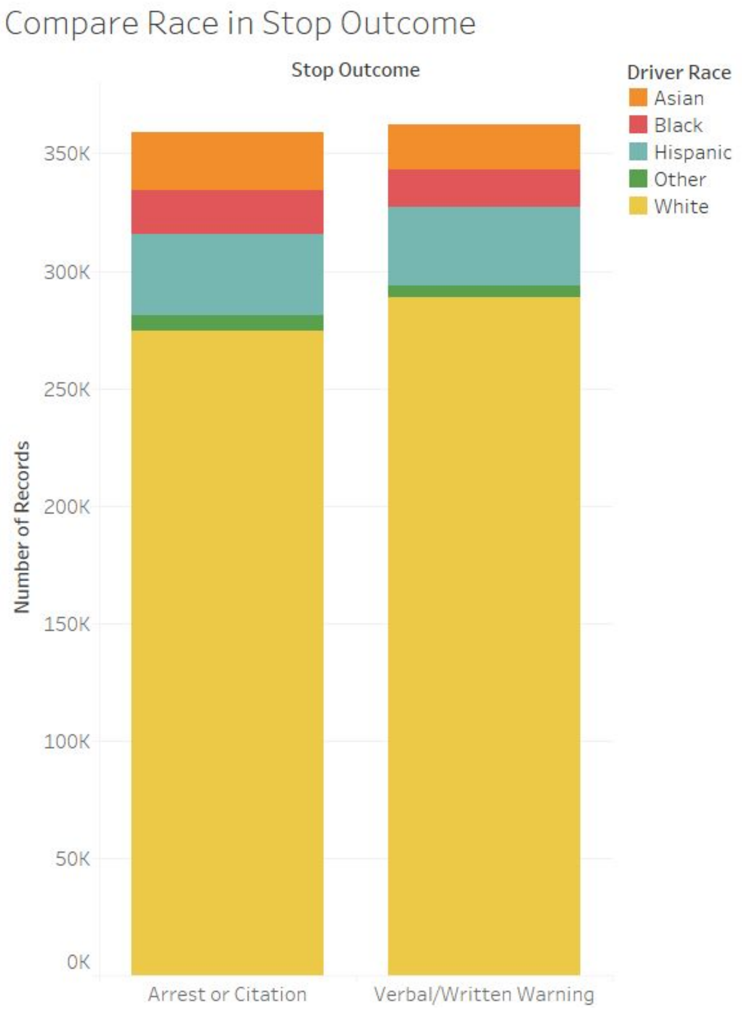


Officer Gender and Stop Locations

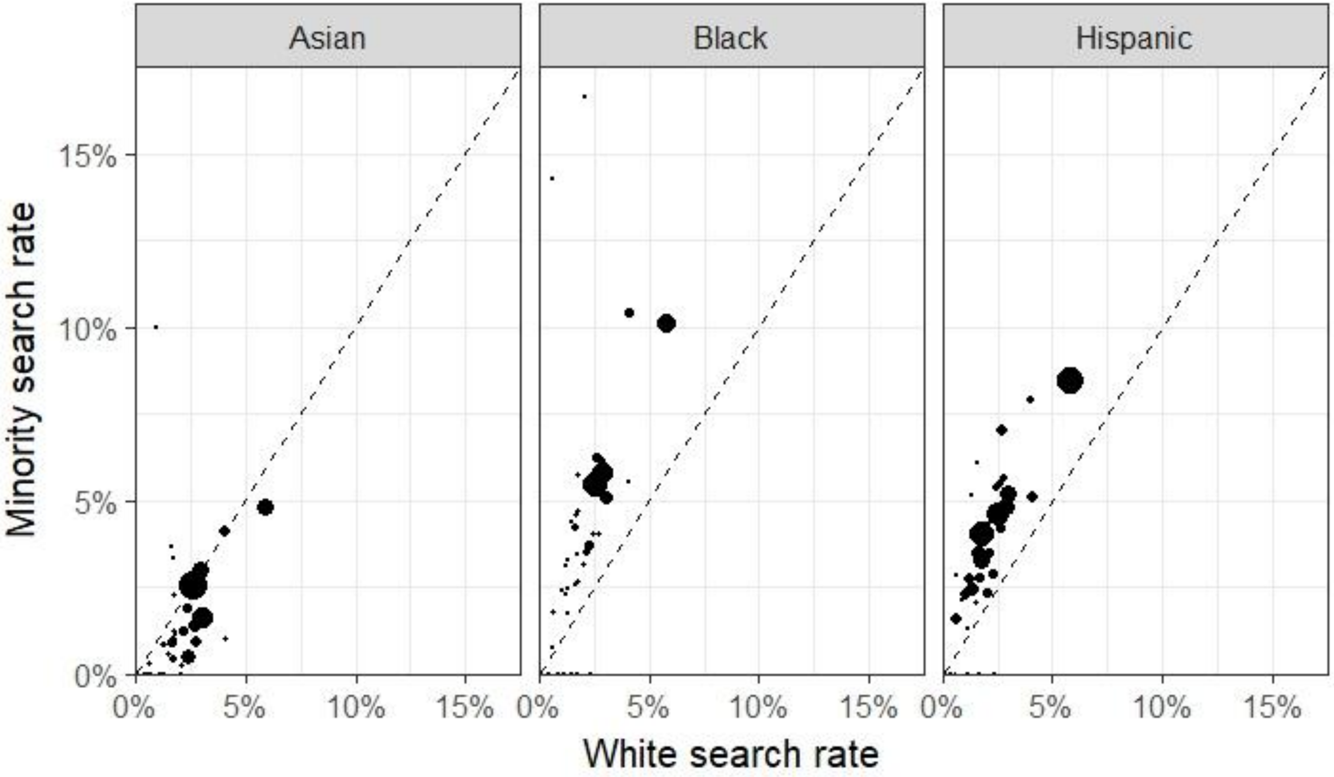




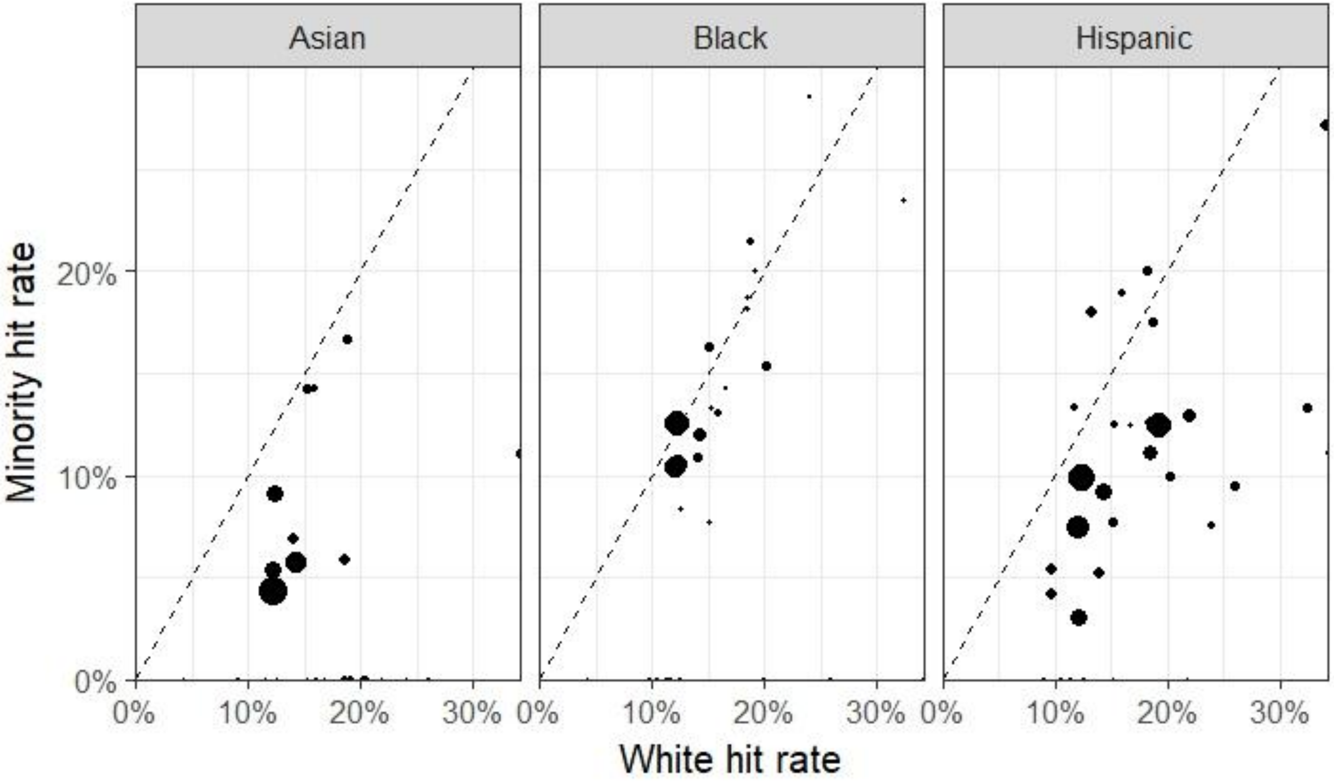

Compare Gender in Stop Outcome
Stop Outcome
Driver Gender
F
M
Number of Records
350K
300K
250K
200K
150K
100K
50K
0K
Arrest or Citation
Verbal/Written Warning


Compare Race in Stop Outcome
Stop Outcome
Driver Race
Asian
Black
Hispanic
Other
White
Number of Records
350K
300K
250K
200K
150K
100K
50K
0K
Arrest or Citation
Verbal/Written Warning

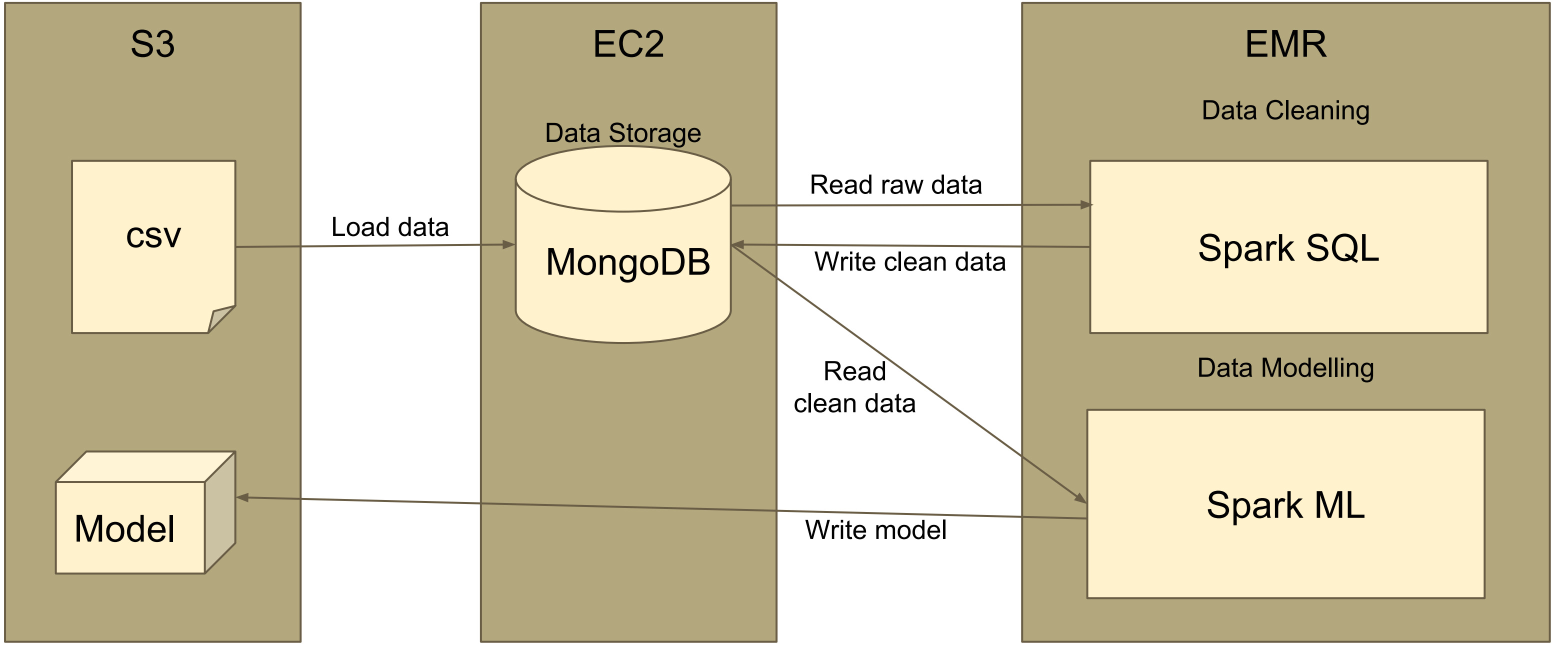
Comparison of Search Rates for Minority and White Drivers



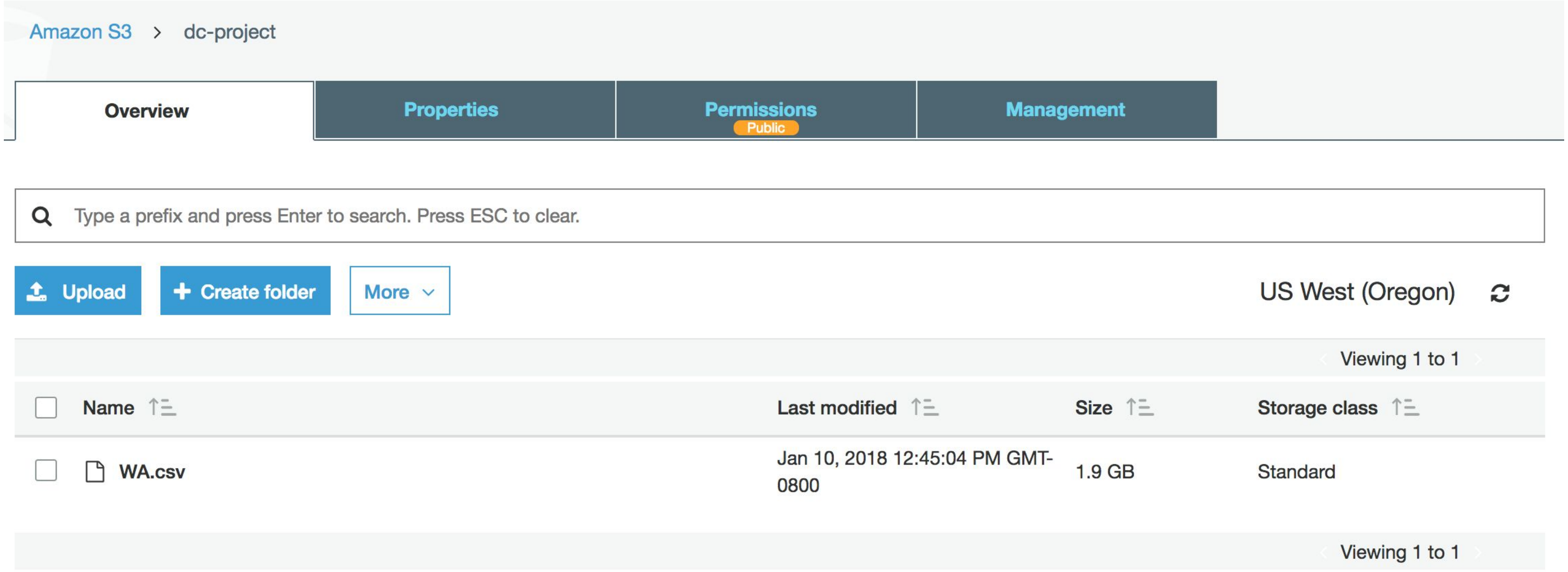
Comparison of Hit Rates for Minority and White Drivers



Data Analytics Pipeline



Data in S3



Amazon S3 › dc-project
Overview
Properties
Permissions
Public
Management
Type a prefix and press Enter to search. Press ESC to clear.
Upload
Create folder
More
US West (Oregon)
Viewing 1 to 1
Name
Last modified
Size
Storage class
WA.csv
Jan 10, 2018 12:45:04 PM GMT-0800
1.9 GB
Standard
Viewing 1 to 1

MongoDB

To set up our MongoDB database for this project, we:

- Set up an EC2 instance via the AWS console
- Installed and configured MongoDB on the server
- Copied data from S3 and loaded it into the database

MongoDB

Sample query: pull one record where driver_race = “Black”:

```
[ec2-user@ip-172-31-3-149 ~]$ mongo
MongoDB shell version: 3.2.18
connecting to: test
> use dc_project
switched to db dc_project
> db.wa_data.find({driver_race: "Black"}).limit(1)
{ "_id" : ObjectId("5a57f1afcea7149791716b43"), "id" : "WA-2009-0000009", "state" : "WA", "stop_
date" : "2009-01-01", "stop_time" : "00:00", "location_raw" : "", "county_name" : "", "county_fi
ps" : "", "fine_grained_location" : "C-017-991", "police_department" : "", "driver_gender" : "M"
, "driver_age_raw" : 33, "driver_age" : 33, "driver_race_raw" : "African American", "driver_race
" : "Black", "violation_raw" : "Lane Travel,License Susp/Rev 3rd Deg,Signal", "violation" : "Lic
ense,Safe movement", "search_conducted" : "FALSE", "search_type_raw" : "No Search", "search_type
" : "", "contraband_found" : "FALSE", "stop_outcome" : "Arrest or Citation", "is_arrested" : "",
 "violations" : "185,16,12", "officer_id" : 650, "officer_gender" : "M", "officer_race" : "White
", "highway_type" : "C", "road_number" : 17, "milepost" : 991, "lat" : "", "lon" : "", "contact_
type" : "Self-Initiated Contact", "enforcements" : "1,3,3", "drugs_related_stop" : "FALSE" }
>
```

MongoDB

Sample query: find all violation types where driver_race = “Black”

```
> db.wa_data.distinct("violation")
[
        "Equipment",
        "Speeding",
        "License,Lights,Paperwork",
        "Safe movement",
        "DUI,License,Speeding",
        "License,Safe movement",
        "Paperwork,Safe movement,Speeding",
        "Lights,Paperwork",
        "Lights,Safe movement,Speeding",
        "Paperwork",
        "Safe movement,Seat belt",
        "DUI,Safe movement",
        "DUI,Safe movement,Speeding",
        "Lights",
        "Safe movement,Speeding",
        "DUI,Paperwork,Safe movement",
        "Equipment,License,Other,Paperwork,Registration/plates,Safe movement",
```

Data Processing Goals

- Handle missing data: excluding records, mean/median imputation
- Split fields with concatenated values: e.g. multiple violations concatenated
- Create additional features: e.g. extract date/time parts from stop_date and stop_time, flags for officer-driver gender/race similarity
- Encode categorical variables appropriately for analysis: numerical and binary encoding

RDD/Data Frame Creation

To begin our analysis, we read our data from MongoDB into a Spark

```
conf = SparkConf().setMaster("local").setAppName(app_name)
sc = SparkContext(conf = conf)
sqlContext = SQLContext(sc)

df_raw = sqlContext.read.format("com.mongodb.spark.sql.DefaultSource")\
                    .option("uri", "mongodb://34.216.30.252/dc_project.wa_1m")\
                    .load()

print df_raw.show(5)
~
~
~
```

RDD/Data Frame Creation

To begin our analysis, we read our data from MongoDB into a Spark

```
+--------------------+--------------------+----------------+-----------+-------------------+----------+--------------+-------------+-----------+---------------+------------------+------------+------------
---------+------------+---------------+-----------+---------+------------+-----------+--------+--------------+----------+------------+-----------------+-----------+----------------+-----------+-----------
----+-----+----------+------------------+---------+--------------------+--------------------+---------------+
|                 _id|        contact_type|contraband_found|county_fips|        county_name|driver_age|driver_age_raw|driver_gender|driver_race|driver_race_raw|drugs_related_stop|enforcements|fine_grained
_location|highway_type|             id|is_arrested|      lat|location_raw|        lon|milepost|officer_gender|officer_id|officer_race|police_department|road_number|search_conducted|search_type|search_type
_raw|state| stop_date|      stop_outcome|stop_time|           violation|       violation_raw|     violations|
+--------------------+--------------------+----------------+-----------+-------------------+----------+--------------+-------------+-----------+---------------+------------------+------------+------------
---------+------------+---------------+-----------+---------+------------+-----------+--------+--------------+----------+------------+-----------------+-----------+----------------+-----------+-----------
----+-----+----------+------------------+---------+--------------------+--------------------+---------------+
|[5a590efedb95ca7e...|  Aggressive driving|           FALSE|           |                   |      16.0|          16.0|            M|      White|          White|             FALSE|   1,3,3,3,3|
 C-027-NA|           C|WA-2015-0516111|           |         |            |           |        |             M|       539|       White|                 |         27|           FALSE|           |      No Se
arch|   WA|2015-05-08|Arrest or Citation|    20:00|Paperwork,Safe mo...|Lane Change,Lane ...|13,12,19,37,152|
|[5a590efedb95ca7e...| Self-Initiated Co...|           FALSE|      53009|     Clallam County|      32.0|          32.0|            M|      White|          White|             FALSE|           1|
S-101-252|           S|WA-2015-0516112|           |48.105721|     Clallam|-123.357797|     252|             M|       411|       White|                 |        101|           FALSE|           |      No Se
arch|   WA|2015-05-08|Arrest or Citation|    20:00|            Speeding|       Speed - Radar|              6|
|[5a590efedb95ca7e...| Self-Initiated Co...|           FALSE|           |                   |      27.0|          27.0|            M|      White|          White|             FALSE|         3,3|
 C-032-NA|           C|WA-2015-0516113|           |         |            |           |        |             M|       524|       White|                 |         32|           FALSE|           |      No Se
arch|   WA|2015-05-08|    Verbal Warning|    20:00|  Paperwork,Speeding|Oper Licence - I,...|          6,181|
|[5a590efedb95ca7e...| Self-Initiated Co...|           FALSE|      53027|Grays Harbor County|      46.0|          46.0|            M|      Asian|    East Indian|             FALSE|           1|
  S-008-2|           S|WA-2015-0516114|           |47.018939|Grays Harbor|-123.350639|       2|             M|       732|       White|                 |          8|           FALSE|           |      No Se
arch|   WA|2015-05-08|Arrest or Citation|    20:00|            Speeding|       Speed - Radar|              6|
|[5a590efedb95ca7e...|     Emphasis patrol|           FALSE|      53037|    Kittitas County|      20.0|          20.0|            M|      White|          White|             FALSE|         1,3|
I-090-102|           I|WA-2015-0516110|           |47.054229|    Kittitas|-120.666276|     102|             M|        40|       White|                 |         90|           FALSE|           |      No Se
arch|   WA|2015-05-08|Arrest or Citation|    20:00|Safe movement,Spe...|Lane Change,Speed...|           6,37|
+--------------------+--------------------+----------------+-----------+-------------------+----------+--------------+-------------+-----------+---------------+------------------+------------+------------
---------+------------+---------------+-----------+---------+------------+-----------+--------+--------------+----------+------------+-----------------+-----------+----------------+-----------+-----------
----+-----+----------+------------------+---------+--------------------+--------------------+---------------+
only showing top 5 rows
```

SparkSQL

Next, we explore features, handle missing values...

```
# Explore features
print df_raw.groupBy(df_raw["driver_gender"]).count().orderBy("count",ascending=False).show()
print df_raw.select("driver_gender").distinct().count()
```

```
# Replace empty values with null
def blank_as_null(x):
    return when(col(x) != "", col(x)).otherwise(None)

df_raw2 = df_raw1.cache()

for c in df_raw2.columns:
    df_raw2 = df_raw2.withColumn(c, blank_as_null(c))
```

```
# Count missing values
df_raw2.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in df_raw2.columns]).show()
```

SparkSQL

... Perform feature engineering

```
# fill driver gender with the most frequent values
df_impute = df_raw3.withColumn("driver_gender_imp",when(col('driver_gender').isNull(),'M').otherwise(col('driver_gender
df_impute = df_impute.drop('driver_gender')
df_impute = df_impute.withColumnRenamed('driver_gender_imp', 'driver_gender')
df_impute.cache()
```

```
# Feature engineering:
# Flag if a driver's gender is different from the officer's
df_feat1 = df_impute2.withColumn("gender_diff", when(col("driver_gender")==col("officer_gender"), 0).otherwise(1
# Flag if a driver's age is different from the officer's
df_feat1 = df_feat1.withColumn("race_diff", when(col("driver_race")==col("officer_race"), 0).otherwise(1))
# Time of the day when the stop occured
df_feat1 = df_feat1.withColumn("time_of_day", when(col("stop_hour")<'07', 1)\
                              .when(col("stop_hour")<'13',2)\
                              .when(col("stop_hour")<'19', 3)\
                              #.when(col("stop_hour")<'21', 3)
```

SparkSQL

… And make corresponding changes to the data frame

```
# numerically encode categorical variables
%time df_clean1 = indexStringColumns(df_feat1,\
                              ['contact_type',"driver_gender", "driver_race",\
                               "highway_type", "officer_gender", "officer_race","stop_outcome", "search_conducted"])
```

```
df_clean2 = df_clean1\
                    .withColumn('stop_date_year', year(df_clean1.stop_date.cast(DateType())))\
                    .withColumn('stop_date_month', month(df_clean1.stop_date.cast(DateType())))\
                    .withColumn('stop_date_dayofmonth', dayofmonth(df_clean1.stop_date.cast(DateType())))\
                    .withColumn('stop_date_weekofyear', weekofyear(df_clean1.stop_date.cast(DateType())))\
```

```
va = VectorAssembler(outputCol="features", inputCols=input_cols)
#lpoints - labeled data.
df_assembled = va.transform(df_hot).select("features", "stop_outcome").withColumnRenamed("stop_outcome", "label")
```

Machine Learning

- Logistic regression model to classify traffic and pedestrian stops
- Analysis of regression coefficients suggest, holding all other variables constant
  - Collision/aggressive driving are more likely to lead to an arrest (duh....)
  - Female officers are 15% less likely to make an arrest
  - Stops on Interstate highway are 30% more likely to lead to an arrest
  - Asian american officers are 20% less likely to make an arrest
  - Driver-Officer gender/race difference has little impact

Processing Time Comparison

Property / Task	Local Machine	EMR Cluster (3 - m3.xlarge)	
Data Volume	1 Mil rows	1 Mil rows	~8.6 Mil rows
Clean data and write to MongoDB	6m07.68s	3m39.22s	26m18.10s
Encoding, Logistic Regression and Model Saving	3m48.24s	3m12.19s	26m10.19s
Encoding, Random Forest Classification and Model Saving	19m05.75s	27m42.42s	-

Lessons Learned - Distributed Computing

- Data wrangling on the cluster is much faster than on a local machine and scales linearly without any additional tuning
- Tree based models on small datasets may not see a performance boost due to expensive optimizations/approximation techniques used

Lessons Learned - Machine Learning

- *Data leakage*: look closely for the sources of data leakage! We initially included ‘*violation_type*’, ‘*contraband_found*’ and ‘*enforcement_type*’ as predictive features and got unrealistically high accuracy rates. We had to drop these variables as they directly pointed to the outcome.
- *Areas for further research:* dependencies between the variables *‘search_conducted’, ‘drugs_related_stop’* , *‘contraband_found’* and driver/officer/location characteristics.

References

1. Data source: https://www.kaggle.com/stanford-open-policing/stanford-open-policing-project-washington-state/data
2. E. Pierson, C. Simou, J. Overgoor et al. (October 2017). *A large-scale analysis of racial disparities in police stops across the United States.* Retrieved from *https://openpolicing.stanford.edu/publications/*