# Project report: BDAS iteration of analysis of gender bias in policing in Rhode Island

By Ahmed Suhail
For INFOSYS722

Link to the code repository: https://github.com/iahmedsuhail/722-bdas

## Table of Contents

# Business Understanding

## 1.1. Business Objectives

Analysis of the gender bias in policing, to address the problem of gender equality.

How often does gender come into play when traffic cops stop motorists on the road? For several years now, especially since the Black Lives Matter movement started, people have advocated for better policing. There is an immediate need for analysis of how policing is done, spot discrimination (if any!). There is also an immediate need to empower women and address the problem of Gender Equality, as is stated in the UN's Sustainable Development Goals (https://bit.ly/1Qk5cqI).

## 1.2. Situation assessment

### Resources

An interdisciplinary team of statisticians, computer scientists, and journalists at Stanford University came together, and "filed public record requests with all 50 states to obtain details of each stop carried out by state patrol officers over the last 10 years". Data was then made to the available to the public (Link: openpolicing.stanford.edu) to facilitate further discussion of policing practices. Over 200 million records from dozens of state and local police departments across the country are available on the website for analysis.

### Requirements

The requirements for successful conduction of analysis include the availability of feature rich data, that possibly consists of all the features known to a police officer before they make a stop. A

successful analysis of such data would possibly be more suitable to suggest correlation between stops and discrimination.

### Assumptions

- All features of data are consistently recorded.
- Data from all the stops was provided by the police department, i.e. stops that might have been police officers harassing citizens were also recorded.

### Risks

- Police data is often private, and not all features are safe for release to the public. Rigorous analysis is difficult.
- Because of the concerns for privacy, licence plate numbers of stopped vehicles was not made available to public. The licence plate number has information known to the police officer that they might have considered whilst making a stop, hence making a known feature not available.
- Police officers might have been biased in their stops and might have falsely added some attributes, such as listing suspicious behaviour as reason for search when, in actuality, the person was just fine.

### Contingencies

- Even though an analysis with the base population of Rhode Island is possible, it is out of the scope of this assignment. A first point look at discrimination would be to look at the population of males and females in a given area and then compare the number of stops they face. Such statistics might not be a clear indicator of discrimination, but they would broaden the scope of insight into the situation.
- A comparison with the drivers of Rhode Island would be even more helpful. To my knowledge, no such database exists in the public. Even if a dataset of the drivers ***of*** Rhode Island were obtained, it would be difficult to obtain a dataset of drivers driving ***through*** Rhode Island.

## 1.3. Data Mining Goals

- Analysis of gender discrimination in police stops in parts of USA (Rhode Island for this report)
- Evaluate the effects of date, race, and other features available on traffic stops by the police.
- Analyse for discrimination in stops by gender using a test that effectively categorizes discriminatory stops from non-discriminatory ones, building on the OSAS Iteration.
- Successful examination and study of the rate at which male and female drivers are stopped and arrested, of the hypotheses that arrests are discriminatory and being able to state with confidence if they are/are not.

## 1.4. Project Plan

The project spanned over a total of four weeks, with the individual activities during each week described in the below Gantt Chart.

| Phase | Week 1 | Week 2 | Week 3 | Week 4 |
|---|---|---|---|---|
| Business understanding | | | | |
| Data understanding | | | | |
| Data preparation | | | | |
| Modeling (Model Selection) | | | | |
| Modeling (Implementing model) | | | | |
| Evaluation | | | | |
| Deployment | | | | |

# Data Understanding

## 2.1. Data Collection

The data was made public by the Stanford open policing project, which is available on the linked website. Hence, I had no issues collecting the data. However, the people at Stanford Open Policing project had filed *public record requests* with 50 states of the United States of America to obtain details of each stop carried out by state patrol officers over the last 10 year. Over a 100 million records from 31 states were collected. The remaining 19 states had a variety of responses, whose data was not suitable for analysis.

A 100 million stops are again, out of scope of this project. So, for this Iteration, I chose to focus on the stops in Rhode Island. The data was a fairly clean dataset with about half a million stops conducted. However, the risks related to all crime analysis still apply, so the possibility of data not being accurate, as indicated with other risks in section 1.2 still apply.

## 2.2. Data Description

i. The data was in files of the format .csv (Comma Separated Values) and .rds for RStudio. The data was loaded into a dataframe, which makes data manipulation easy when we're using Spark. An interesting feature of dataframes is that they can store different types of data, making modelling and manipulation easier.

ii. Data was publicly available for download around June 2017, for analysis and help in getting better insights. By studies of location-specific data and evaluation of effects of different features on stops, better analyses were be made.

iii. For the BDAS iteration, focus was on data from Rhode Island, expanding on the findings from OSAS Iteration. The State Patrol from Rhode Island had provided data of all vehicular stops from January 2005 to December 2015. This assignment conducts a specific statistical test for discrimination, inspired by the tests conducted by the team at Stanford Open Policing Project.

iv. The available data consisted of the following features:

- raw_row_number: A record ID for reference to the raw data
- Date: Stop date in the format of YYYY-MM-DD
- Stop Time: Time of the Stop, in a 24 hour, HHMM format.
- Stop Location: Textual, non-standardised location.
- Driver Race: Race of the stopped subject.
- Driver Age: Age of the stopped subject.
- Driver Sex: Gender of the stopped subject.
- Search Conducted: Boolean value that indicates whether a search was conducted.
- Contraband Found: Boolean value that indicates if contraband was found
- Citation Issued: Boolean value indicating outcome, if a citation was issued.
- Warning Issued: If a stop resulted in the issue of warning, this Boolean was selected.
- Frisk Performed: If a quick Frisk was performed, this Boolean value was set True

- Arrest Made: If the person were arrested, this Boolean value was set True.
- reason_for_stop: The visible reason for stopping the driver.

## 2.3. Data Exploration

Initial exploration of data reveals the need to prepare data and fix errors, and missing values before plots for better understanding can be made. The effects of reasons for searching and zone on male and female arrests were also evaluated, which suggest males are more likely to be arrested than females, whether they be in a zone or they be committing the same crimes as women!

```
df.groupBy("subject_sex").count().show()
```

```
+-----------+------+
|subject_sex| count|
+-----------+------+
|         NA| 29097|
|     female|131138|
|       male|349446|
+-----------+------+
```

```
df.groupBy(["zone", "subject_sex"]).count().show()
```

```
+----+-----------+-----+
|zone|subject_sex|count|
+----+-----------+-----+
|  X3|     female|26653|
|  X3|         NA| 4627|
|  X3|       male|62778|
|  X4|         NA| 9679|
|  X4|       male|94953|
|  K3|     female|29097|
|  X1|         NA| 3491|
|  X1|     female| 2702|
|  K3|       male|79771|
|  X1|       male|10522|
|  K1|         NA| 2252|
|  K2|     female|28114|
|  K1|       male|32255|
|  NA|         NA|   10|
|  K3|         NA| 4916|
|  K1|     female|13855|
|  K2|         NA| 4122|
|  X4|     female|30717|
|  K2|       male|69167|
+----+-----------+-----+
```

```
df.groupBy(["subject_sex", "reason_for_stop"]).count().show()
```

```
+-----------+--------------------+------+
|subject_sex|     reason_for_stop| count|
+-----------+--------------------+------+
|     female|Other Traffic Vio...| 17911|
|       male|Special Detail/Di...| 12977|
|     female|Equipment/Inspect...| 14039|
|     female|                 APB|   109|
|     female|Violation of City...|   216|
|         NA|    Call for Service|     4|
|         NA|Equipment/Inspect...|     2|
|       male|    Call for Service|  5237|
|       male|Registration Viol...| 14181|
|       male|            Speeding|182538|
|         NA|            Speeding|     8|
|       male|Motorist Assist/C...|   657|
|       male|   Suspicious Person|   268|
|     female|  Seatbelt Violation|  3550|
|     female|Registration Viol...|  5649|
|       male|Other Traffic Vio...| 72317|
|         NA|                  NA| 29073|
|     female|   Suspicious Person|    74|
|         NA|  Seatbelt Violation|     3|
|       male|                 APB|   376|
+-----------+--------------------+------+
```

## 2.4. Data Quality

### Missing Values

Since we're testing for discrimination against a particular gender, data that misses subject_sex isn't useful at all. Since a fairly large amount of data is available for modelling, I chose to drop the rows with missing values of subject_sex.

```
df = df.filter(df.subject_sex.endswith('ale'))
```

```
df.groupBy("subject_sex").count().show()
```

```
+-----------+------+
|subject_sex| count|
+-----------+------+
|     female|131138|
|       male|349446|
+-----------+------+
```

The regular query for spark .na.drop wasn't effective since data was not actually null but a string named "NA", data had to be filtered like shown in the code snippet.

### Data Quality Patterns

- Initially, I thought NA values in other rows also need to be tackled. But the thought of so many NA values in contraband_found intrigued me. This is actually fine, because what NA means here is that since search_conducted is false, that is a search was not conducted in this stop, there is no way that contraband could have been recovered. These NA values are

hence fine for analysis, but will be evaluated upon further in section 3.2.

```
df.groupBy("contraband_found").count().show()
```

```
+----------------+------+
|contraband_found| count|
+----------------+------+
|           FALSE| 11183|
|              NA|462822|
|            TRUE|  6579|
+----------------+------+
```

- No other irregularities, such as duplicates were found in the data. But for safety, the dropduplicates() method was run with no argument for subset, so rows where all values were identical would be dropped.
- After the dropping of null values from subject_sex, the feature subject_race was checked to verify quality of data, and revealed good quality data, as no missing values were present.

```
df.groupBy("subject_race").count().show()
```

```
+--------------------+------+
|        subject_race| count|
+--------------------+------+
|               white|344716|
|               black| 68577|
|            hispanic| 53123|
|       other/unknown|  1344|
|asian/pacific isl...| 12824|
+--------------------+------+
```

```
In [12]: #total number of values in data BEFORE dropping duplicates
         df.count()
```

```
Out[12]: 480584
```

```
In [13]: #dropping duplicates from data
         df = df.dropDuplicates()
```

```
In [14]: #total number of values in data AFTER dropping duplicates
         df.count()
```

```
Out[14]: 480584
```

# Data Preparation

## 3.1. Data Selection

For the data selection in this section, only features that are not so relevant to the analysis are dropped. Data selection is throughout, though & multiple iterations of modelling involve selection of different types of data for each iteration.

Primary Identifiers for data that are not relevant for modelling (raw_row_number and department id) are dropped.

```
In [18]: df = df.drop("raw_row_number", "department_id")
```

```
In [19]: df.printSchema()
```

```
root
 |-- date: string (nullable = true)
 |-- time: string (nullable = true)
 |-- zone: string (nullable = true)
 |-- subject_race: string (nullable = true)
 |-- subject_sex: string (nullable = true)
 |-- type: string (nullable = true)
 |-- arrest_made: string (nullable = true)
 |-- citation_issued: string (nullable = true)
 |-- warning_issued: string (nullable = true)
 |-- outcome: string (nullable = true)
 |-- contraband_found: string (nullable = true)
 |-- contraband_drugs: string (nullable = true)
 |-- contraband_weapons: string (nullable = true)
 |-- contraband_alcohol: string (nullable = true)
 |-- contraband_other: boolean (nullable = true)
 |-- frisk_performed: string (nullable = true)
 |-- search_conducted: string (nullable = true)
```

An analysis of the type feature revealed all the types of stops were vehicular. Since all the values in this feature are the same, it is a redundant feature and hence, dropped. All other analysis is done keeping in mind that we're only dealing with vehicular stops.

```
#Check feature type
df.groupBy("type").count().show()
```

```
+---------+------+
|     type| count|
+---------+------+
|vehicular|480584|
+---------+------+
```

```
#Drop type feature from data, since it has only one value
df = df.drop('type')
```

## 3.2. Data Cleaning

- A look into the types of values of the fields of outcome indicated the following different values to be present in it.

```
#Check outcome column
df.groupBy("outcome").count().show()
```

```
+--------+------+
| outcome| count|
+--------+------+
|      NA|  6763|
|citation|428378|
|  arrest| 16603|
| warning| 28840|
+--------+------+
```

This is exactly similar to the other 3 features present in the schema, as can be seen here:

```
|-- arrest_made: string (nullable = true)
|-- citation_issued: string (nullable = true)
|-- warning_issued: string (nullable = true)
```

The feature *outcome* is hence redundant and dropped. A question might arise here as to why outcome was dropped, rather than the other 3 features. This is because the other 3 features are somewhat linearly independent and might train the model better.

- The other type of dependent features present in the data are the following ones, related to the type of contraband being found when the search was conducted.

```
|-- contraband_found: string (nullable = true)
|-- contraband_drugs: string (nullable = true)
|-- contraband_weapons: string (nullable = true)
|-- contraband_alcohol: string (nullable = true)
|-- contraband_other: boolean (nullable = true)
```

As wil be discussed later in this report, I'm going to test for discrimination as was conducted by the team at Stanford Open Policing. As stated by the team in their research, "The Nobel Prize–winning economist Gary Becker proposed an elegant solution to this conundrum in the 1950s: Becker suggested one should look not at search rates but at success rates. Absent discrimination, he argued, officers should find contraband on searched minorities at the same rate as on searched whites" (Pierson et al., 2017).

A better test would hence be possible if we test for contraband being found using the *contraband_found* (it takes into consideration all types of contraband) and removing the other features.

```
In [30]: #Drop features linearly dependent
         df = df.drop('contraband_drugs', 'contraband_weapons', 'contraband_alcohol', 'contraband_other')
```

## 3.3. Data Construction

Coming back to the issue discussed in section 2.4 under Data Quality Patterns, with NA values being present in some of the features, such as contraband_found. To check if this was an actual issue, the

feature search conducted was checked side by side contraband_found.

```
#NA Values in some fields
df.groupBy("search_conducted").count().show()
```

```
+----------------+------+
|search_conducted| count|
+----------------+------+
|           FALSE|462822|
|            TRUE| 17762|
+----------------+------+
```

```
#NA Values in some fields
df.groupBy("contraband_found").count().show()
```

```
+----------------+------+
|contraband_found| count|
+----------------+------+
|           FALSE| 11183|
|              NA|462822|
|            TRUE|  6579|
+----------------+------+
```

What the above grouping suggests is that the NA values in contraband found are actually Not Applicable, i.e. you wouldn't expect contraband to be found if the search was never conducted in the first place. Hence, the NA values are not a problem and don't require to be dealt with. Further reading regarding the effects of this on StackOverFlow revealed a solution of how this could be tackled with for better modelling. A new field was formed with 1 for the values being NA in contraband_found and 0 if the value in contraband_found was either False or True. Both the fields were fed together into the model every time.

```
from pyspark.sql.functions import when
#Create a new column for contraband_found
df = df.withColumn("contraband_found_resolved", when(df.contraband_found == "NA", 0).otherwise(1))
```

```
#Check newly created column
df.groupBy("contraband_found_resolved").count().show()
```

```
+------------------------+------+
|contraband_found_resolved| count|
+------------------------+------+
|                       1| 17762|
|                       0|462822|
+------------------------+------+
```

## 3.4. Data Integration

It can be seen in the Jupyter Notebook that the dataset was manually split into two and then combined using spark.read.csv method.

```
# Importing data.
df = spark.read.csv('dataset/ri_statewide_2019_02_25.csv', header=True, inferSchema=True)
```

## 3.5. Data Reformatting

The date and time columns in the dataset were deemed important for modeling, as they were an important attribute that contribute to vehicles being stopped. It might be the case that officers

might be stopping more females at night, which is an important factor to consider while analysing for discrimination. The date and time columns were strings and were converted to integers for modeling, by creating new columns using the withColumn and split method. The results from split function were cast to IntegerType for modeling.

In [29]:
```
#splitting date into different columns for LR
from pyspark.sql.types import IntegerType
split_date = split(df['Date'], '-')
df = df.withColumn('Year', split_date.getItem(0).cast(IntegerType()))
df= df.withColumn('Month', split_date.getItem(1).cast(IntegerType()))
df= df.withColumn('Day', split_date.getItem(2).cast(IntegerType()))
```

In [30]:
```
#splitting time into different columns for modeling

split_time = split(df['time'], '-')
df = df.withColumn('Hour', split_date.getItem(0).cast(IntegerType()))
df= df.withColumn('Minute', split_date.getItem(1).cast(IntegerType()))
df= df.withColumn('Second', split_date.getItem(2).cast(IntegerType()))
```

# Data Transformation

## 4.1. Data Reduction

Since the dataset is not one with high dimensionality, a feature selection algorithm is not relevant. Logical selection of the features, is however, critical for a test for discrimination. For an effective test for discrimination, it is important to only use features for modelling that are known to the officer prior to making the stop. The dataframe schema was hence analysed for features, and I concluded that the following features, which were recorded after making the stop, needed to be dropped from the dataset.

In [32]:
```
#Drop features not known prior to making the stop
df = df.drop('arrest_made', 'frisk_performed', 'search_basis', 'reason_for_search', 'reason_for_stop')
```

In [33]:
```
df.printSchema()
```

```
root
 |-- date: string (nullable = true)
 |-- time: string (nullable = true)
 |-- zone: string (nullable = true)
 |-- subject_race: string (nullable = true)
 |-- subject_sex: string (nullable = true)
 |-- citation_issued: string (nullable = true)
 |-- warning_issued: string (nullable = true)
 |-- outcome: string (nullable = true)
 |-- contraband_found: string (nullable = true)
 |-- search_conducted: string (nullable = true)
 |-- vehicle_make: string (nullable = true)
 |-- vehicle_model: string (nullable = true)
 |-- contraband_found_resolved: integer (nullable = false)
```

## 4.2. Data Projection by Transformations

An analysis of the values of the feature contraband_found revealed that a lot of the values are NA. The values are NA for not applicable, since not all of the traffic stops were searched.

```
df.groupBy("contraband_found").count().show()
```

```
+----------------+------+
|contraband_found| count|
+----------------+------+
|           FALSE| 11183|
|              NA|462822|
|            TRUE|  6579|
+----------------+------+
```

An analysis of the values of type search_conducted further revealed that the contraband_found is not applicable when search was never conducted in the stop. The values were hence deemed important for modelling and another column was projected, which was 0 when contraband_found was not applicable and 1 otherwise.

```
#NA Values in some fields
df.groupBy("search_conducted").count().show()
```

```
+----------------+------+
|search_conducted| count|
+----------------+------+
|           FALSE|462822|
|            TRUE| 17762|
+----------------+------+
```

```
from pyspark.sql.functions import when
#Create a new column for contraband_found
df = df.withColumn("contraband_found_resolved", when(df.contraband_found == "NA", 0).otherwise(1))
```

# Data Mining Method Selection

## 5.1. Match DM Methods with the context of DM Objectives

The primary goal, as was stated in section 1.3 is the assessment of discrimination on the basis of gender in traffic stops, for this iteration, in Rhode Island. There are many ways to do this, and in previous iterations many ways were tried. The different ways this could have been achieved are:

- Association, which is an unsupervised learning method, could be used to form an association between the different features in the dataset. Association rule mining is the data mining process of finding the rules that may govern associations and causal objects between sets of items. Apriori algorithm could formulate a relationship between the different features in a dataset and relationships between the features could have been evaluated. The metrics of the model would have allowed to see how influential the gender of a person is when a police officer is making a stop. This, however, with other methods was tried in the ISAS, using SPSS modeler, and wasn't very helpful in formulation of any rules.
- Discrimination could also have been tested for by using Clustering, which again is unsupervised learning. KMeans Clustering was used to find patterns in the data that miss the eye, but initial runs of the model didn't yield any helpful results. I didn't want to waste much time there, and instead build on the OSAS Modeler Iteration, so I focused mainly on Logistic Regression.

### 5.2. Selection of appropriate DM Method

**Logistic Regression** is conventionally used for Predictive Analytics, to predict whether and with what likelihood an event will occur in the future. Unlike linear regression, it is a classification algorithm. Many people, such as the authors of the book (Kleinbaum & Klein, 2002) introduce it as a predictive analytic technique for when the dependent variable is binary.

Another typical use of Logistic Regression, however, is the use of **Odds ratio to predict discrimination**, as talked about by the authors in the same book cited above (Kleinbaum & Klein, 2002)and for better understanding in the article by (Ed, n.d.). Logistic regression will be used to test for discrimination, as was done by the team at Stanford Open Policing Project, where the odds ratio is used to describe the type of relationship between the dependent variable, which in our case will be male or female. We're hence going to use Logistic Regression in a not so conventional way, to test for discrimination. The idea will be further explained in section 6.2

## Data Mining Algorithm Selection

### 6.1. Exploratory Analysis

An exploratory analysis is not relevant to my project, as the other methods to calculate odds ratio to test for discrimination are very statistical in nature and not present in PySpark. The idea follows from the strategy adopted by the researchers of the Stanford Open Policing Project, where they argue that **a stop is not discriminatory if it results in contraband being found in males at the same rate as it results in contraband being found in females.** The team followed the principles of Gary Becker, a Nobel Prize winner, who suggested one should look not at search rates but at success rates. Absent discrimination, he argued, officers should find contraband on searched males at the same rate as on searched females.

If searches of males turn up contraband at lower rates than searches of females, for example, it indicates officers are applying a double standard and searching males on the basis of less evidence. If searches of females turn up contraband at lower rates, it suggests officers are searching females on the basis of less evidence.

### 6.2. Selection of Algorithm

The use of Logistic Regression to test for discrimination is explained here. Logistic regression if run on a binary predictor such as Gender, can be interpret safely with the p value being low enough for the result to be significant. When the odds ratio is greater than 1, it describes a positive relationship. The positive relationship means as one shifts from one gender to another, the odds of being arrested might increase. Therefore, the odds ratio will be used to predict searches in contraband of which gender are more likely to turn up contraband. The best-case scenario would be when the searches turn up contraband with the same results.

### 6.3. Building the model with appropriate parameters

For modeling the data, the data that were in strings needed to be converted to numerical values. This was done with the use of the String Indexer class, which encodes a column of string labels/categories to a column of indices.

```
# Create a string indexer (convert every string into a number, such as male = 0 and female = 1).
# A number will be assigned to every category in the column.
gender_indexer = StringIndexer(inputCol='subject_sex',outputCol='SexIndex')
```

For effective results, so that the model doesn't confuse the categories of race and sex we have in our data as having an order with them, one hot encoding was used. One hot encoder maps the label

indices to a binary vector representation with at the most a single one-value. The spark one hot encoder takes the indexed label/category from the string indexer and then encodes it into a sparse vector. Since neither the race, sex, zone or vehicles have an associated order, all were one hot encoded.

```
# Now we can one hot encode these numbers. This converts the various outputs into a single vector.
gender_encoder = OneHotEncoder(inputCol='SexIndex',outputCol='SexVec')

#Similar to the above.
race_indexer = StringIndexer(inputCol='subject_race',outputCol='raceIndex')
race_encoder = OneHotEncoder(inputCol='raceIndex',outputCol='raceVec')

#Similar to the above.
zone_indexer = StringIndexer(inputCol='zone',outputCol='zoneIndex')
zone_encoder = OneHotEncoder(inputCol='zoneIndex',outputCol='zoneVec')

#Similar to the above.
vehicle_indexer = StringIndexer(inputCol='vehicle_make',outputCol='vehicleIndex')
vehicle_encoder = OneHotEncoder(inputCol='vehicleIndex',outputCol='vehicleVec')
```

Another step is to consolidate the various columns into a single column. This is necessary since the Machine Learning algorithms in spark operate on a single vector of predictors, although each element in that vector may consist of multiple values. An instance of VectorAssembler is used to transform the data and then consolidated into a vector. All of the predictors are ultimately consolidated. In simpler words, it's job is to combine the raw features and features generated from various transforms into a single feature vector.

```
# Now we can assemble all of this as one vector in the features column.
assembler = VectorAssembler(inputCols=['Year', 'Month', 'Day', 'Hour', 'Minute', 'Second',
 'SexVec',
 'zoneVec',
 'search_conducted',
 'vehicleVec',
 'raceVec'],outputCol='features')
```

A pipeline was created to streamline the process of modeling. It made streamlining the feature transformations easier. A workflow was hence created, and complex transformations made easier.

```
# Pipeline for complex workflow
pipeline = Pipeline(stages=[gender_indexer,race_indexer, zone_indexer, vehicle_indexer,
                            gender_encoder,race_encoder, zone_encoder, vehicle_encoder,
                            assembler,log_reg])
```

The model was then fit on the training data.

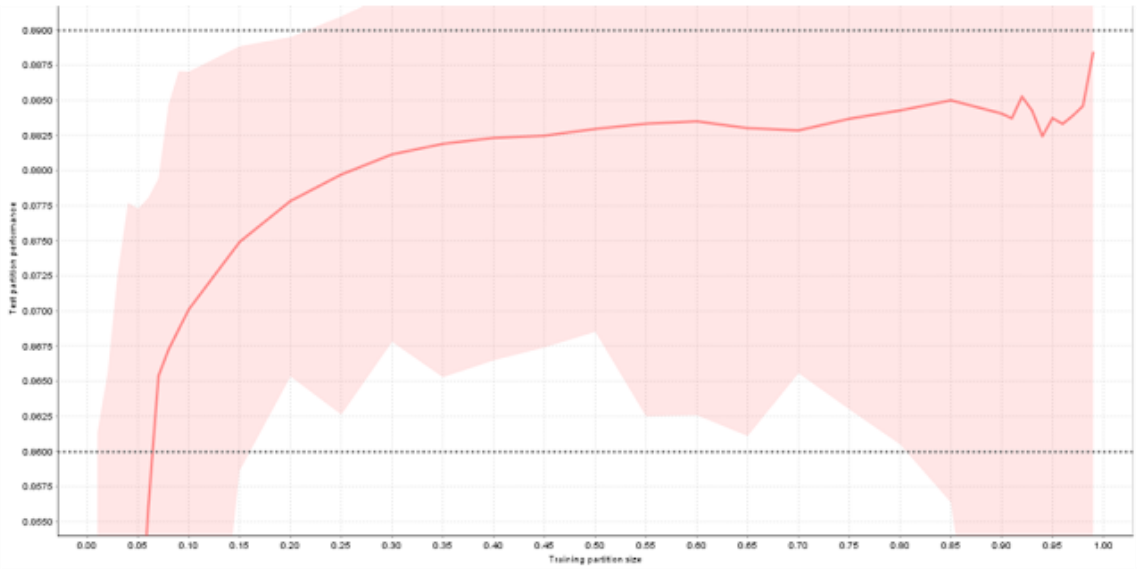
```
#modeling
fit_model = pipeline.fit(train_data)
```

# Data Mining

## 7.1. Train - Test Designs

For effective modeling, the data was split into training and testing sets, and made sure that the model while training never sees the training data. A universally accepted rule of conducting data analysis is using the 70:30 split. This was also adapted for my study because there is huge evidence that when the data is trained over more than 80% of the dataset, even though there is benefit in the accuracy, but the benefit is not as much as compromising the validity of your accuracy scores. The

saturation point achieved in the graph below is proof of a train test pattern that validates the 70:30 split, and since the dataset I had was huge vertically as compared to horizontally, the 70:30 split seemed to be perfect for modeling.



## 7.2. Model must run

Model was fit on the data.

```
#modeling
fit_model = pipeline.fit(train_data)
```

## 7.3. Model's output patterns documented

The model was run effectively and then tested on the test data, formed by the random 70:30 split. A Binary Class evaluator was used to for evaluation metrics because the label column, contraband_found was binary. As can be seen from the select query, all the predictions that showed up in the top 20 rows were correct.

```
# Evaluate the model using the binary classifer.
from pyspark.ml.evaluation import BinaryClassificationEvaluator, MulticlassClassificationEvaluator

my_eval = BinaryClassificationEvaluator(rawPredictionCol='prediction',
                                        labelCol='contraband_found')
```

```
# If we select the actual and predicted results, we can see that some predictions were correct while others were wrong.
results.select('contraband_found','prediction').show()
```

```
+----------------+----------+
|contraband_found|prediction|
+----------------+----------+
|               1|       1.0|
|               1|       1.0|
|               1|       1.0|
|               1|       1.0|
|               1|       1.0|
|               1|       1.0|
|               1|       1.0|
|               1|       1.0|
|               1|       1.0|
|               1|       1.0|
|               1|       1.0|
|               1|       1.0|
|               1|       1.0|
|               1|       1.0|
|               1|       1.0|
|               1|       1.0|
|               1|       1.0|
|               1|       1.0|
|               1|       1.0|
|               1|       1.0|
+----------------+----------+
only showing top 20 rows
```

The Area under ROC was plotted using a stage -1 of the pipeline, and came out to be fairly high. ROC Curves are a way to see how any predictive model can distinguish between the true positives and negatives. In order to be successful, a model needs to not only correctly predict a positive as a positive, but also a negative as a negative. The ROC curve plots out the sensitivity and specificity for every possible decision rule cutoff between 0 and 1 for a model. In simpler terms, higher the AUC, better the model in predicting if contraband was found or not. And as we can see, we got a fairly high value of it. The AUC might seem alarmingly high to the reader, but an explanation is offered in section 8.1 in the in depth discussion of results.

```
lrm = fit_model.stages[-1]

summary = lrm.summary

print("areaUnderROC: " + str(summary.areaUnderROC))
```

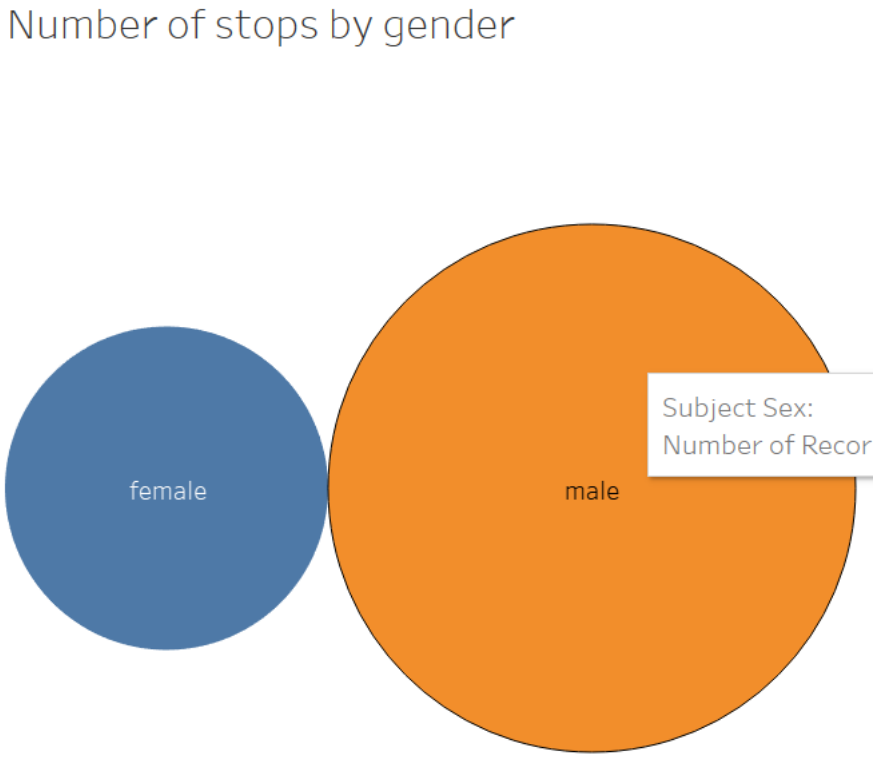
```
areaUnderROC: 0.9930572623607522
```

# Interpretation

## 8.1. In depth discussion

- Data: The data fed to the model was rich vertically, but not so rich horizontally. The features had to be reduced to the ones that an officer knew prior to making the stop to be able to test for discrimination, and there is a limitation to how many of such features are reported by law agencies and are mandated for recording by the law after making the stop. Nevertheless, we still had enough data to be able to suggest/discard correlation strength between gender and traffic stops.
- Patterns: Peeps into the data and primary visualisations revealed that men are more likely to be stopped and even arrested in Rhode Island. This was achieved by visualizations in Tableau as PySpark doesn't allow you to plot data directly. Initial looks at data suggest that males are more likely to be stopped than females. This however, is not all of the story as no

comparison with the base population of Rhode Island was taken into consideration.




- Results: To test for discrimination, a test using Logistic Regression was to be conducted. The idea is that if searches in males and females turn up contraband with the same frequency, then officers are not discriminatory in their stops. To judge the frequency of contraband being found the Odds ratio attribute, which can be calculated from the coefficients was used. The coefficients of the logistic regression were as following:

```
print("Coefficients: " + str(lrm.coefficients))
```

```
Coefficients: [8.845392567027978e-06,0.0003652767562964292,5.954016231847854e-05,8.845392567027978e-06,0.0003652767562964292,5.
954016231847854e-05,-0.01736236927311298,-0.014091565261873319,-0.0027275705117718475,0.00763638960409699,0.006903921732383211,
0.004288404902808234,-0.7329858454320288,0.004350162465412992,0.0019404553297236225,-0.0025593886316416744,0.002856185167700001
3,-0.009043816054850733,-0.004820648513783765,-0.013888199751280814,0.013262481356300043,0.0015191200019898372,0.00935897489587
661,5.190261417940348e-05,-0.005285883947159624,0.004637543344813744,0.021451231736913958,-0.007855977087362336,-0.035439161978
68823,0.007655243936754905,0.007593912447261756,-0.010661320447735366,-0.006641048774579483,-0.002262007057727082,-0.0072701752
885169476,0.001356282379124956,0.011807336639168341,-0.016775880305950057,0.0016147072331473777,0.0013913171239973029,-0.036695
21564180566,-0.0002038565426866327,0.010360390593962276,-0.01605200511809535,0.02116250278188753,0.023359954597500118,0.0232522
97167813053,-0.010018961733266848,-0.039294635113528 28,0.02254757595855481,0.01585476577524864,0.019805689383353565,-0.01548349
0192102503,0.026923128449661375,-0.010561373258520433,-0.020153842320270374,0.01942248641768323,-0.026676303982526243,-0.034762
194964432384,-0.010193468703854064,-0.01639391333932485,0.0203151345648897,-0.046912162199374 02,0.008673276301370778,-0.0221965
84276569432,-0.024426493125014572,0.026905579379417507,0.02675985081285263,0.027159890584728782,0.02748632194789479,-0.05009380
628811495,0.026776337678013636,-0.15054885722156 02,0.026979992472139842,0.02676061313539343,0.026792800148469294,0.026539246442
49097,0.026697345362944495,0.027063498687531513,0.026745922026825 2,0.026875798313215283,0.026538983688504 2,0.026715633371674 52
4,0.026622223625757252,0.026616289763800566,0.026791214248652 06,0.026787400284337546,0.026669500189844 65,0.026856968638048036,
0.026718595939092 13,0.026890709006825 53,0.026572960378397195,0.026629922648766673,0.027417555549443362,0.026597967738385 73,0.02
667345266703586 8,0.026630098709538006,0.026589349711872393,0.026779703051468106,0.026535607223340 2,0.026498652163774 61,0.026684
54171005228,0.026648700041931 9,0.026711373531507 94,0.021772780213536 7,-0.021707093822926 15,-0.020981659066670044,0.009972431553
863707]
```

As can be seen, for a model with not more than 10 features, different coefficients were released. An explanation of this is was caused by the usage of One Hot Encoding. Even though encoding the data using the one hot encoder had the added benefit of LR Model not misinterpreting data that is categorical to have an order (which led to a model of higher AUC, i.e. better model), the one hot encoder formed a vector which made interpreting the coefficients impossible. A second iteration was hence conducted, removing the one hot encoder for interpretation of coefficients.

## Iteration 2 Results

Another iteration was hence done, removing the one hot encoder for possible interpretation of the coefficients and hence calculation of the odds-ratio, which is important if we want to see which gender is more likely to turn up contraband if a search is performed.

```
In [86]: lrm = fit_model.stages[-1]

         summary = lrm.summary

         print("areaUnderROC: " + str(summary.areaUnderROC))

         areaUnderROC: 0.9928861584724064

In [87]: print("Coefficients: " + str(lrm.coefficients))

         Coefficients: [6.145157087201099e-06,0.0003545451430219321,6.452545578054053e-05,6.145157087201099e-06,0.0003545451430219321,6.
         452545578054053e-05,0.017604446811487682,0.004640212253076101,-0.7364572525059239,-9.28515411498677e-05,-0.009179725177856826]

In [88]: import numpy as np
         np.exp(lrm.coefficients)

Out[88]: array([1.00000615, 1.00035461, 1.00006453, 1.00000615, 1.00035461,
                1.00006453, 1.01776032, 1.00465099, 0.47880721, 0.99990715,
                0.99086228])
```

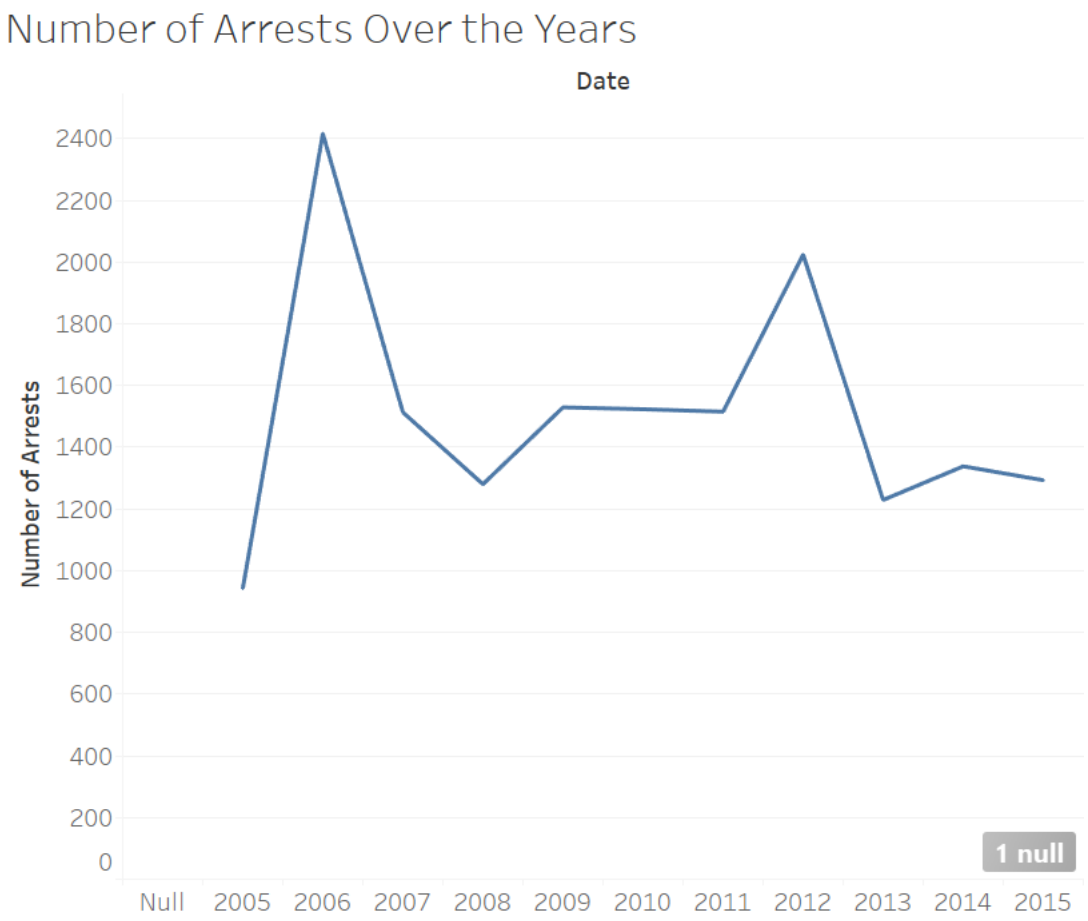
The iteration 2 resulted in a model with slightly lesser accuracy but as can be seen from the screenshots, the coefficients were used to calculate odds ratio using NumPy. The odds ratio for subject_sex is 1.000000615, whose interpretation is in section 8.3.

### Q: Why is the AUC alarmingly high?

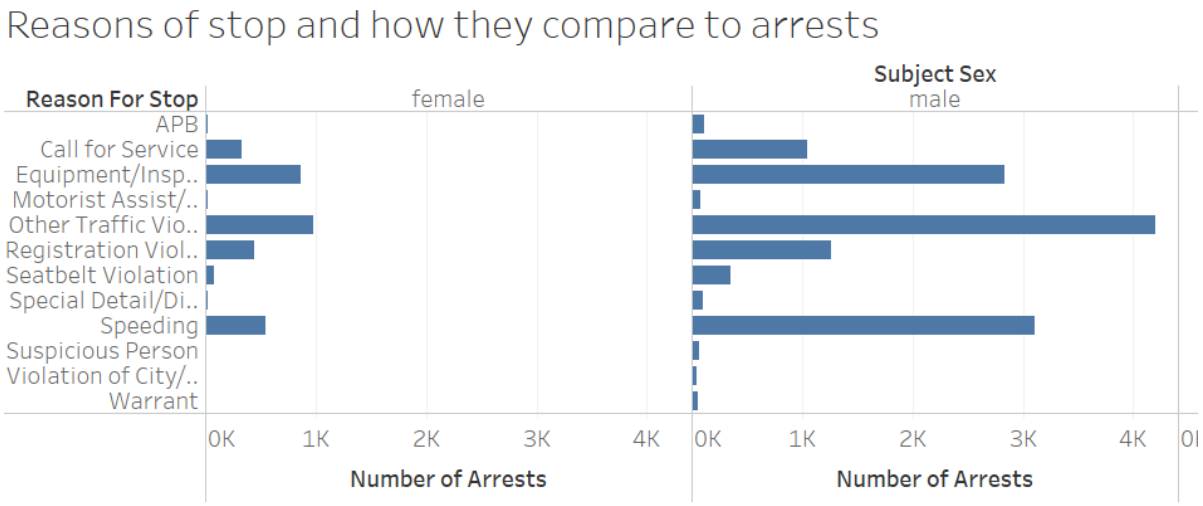
Another discussion that is eminent here is the unusually high accuracy of the logistic regression model. A reason to why the model might have unusually high accuracy is that Logistic Regression performs best when the features fed to the model are linearly dependent, as was done in the third, OSAS iteration, where accuracy was logical. Here, the AUC is unusually high because we've fed a linearly dependent feature, search_conducted, to the model. Since almost always, whenever the Boolean search_conducted is true, contraband_found is also true. But for the test, search_conducted is vital since it is a feature that lists the reason the search was conducted, which the officer knew prior to making the stop and should have been fed to the model. The odds ratio is influenced in a way that search_conducted becomes much more important and subject_sex loses it's value. The stark opposition in iterations 3 and 4 are because of feeding different features to the model, and which one is more likely the case is difficult to interpret. Discrimination to some extent is seen, but it is not exactly clear to what extent there is discrimination in stops.

## 8.2. Visualization of the results

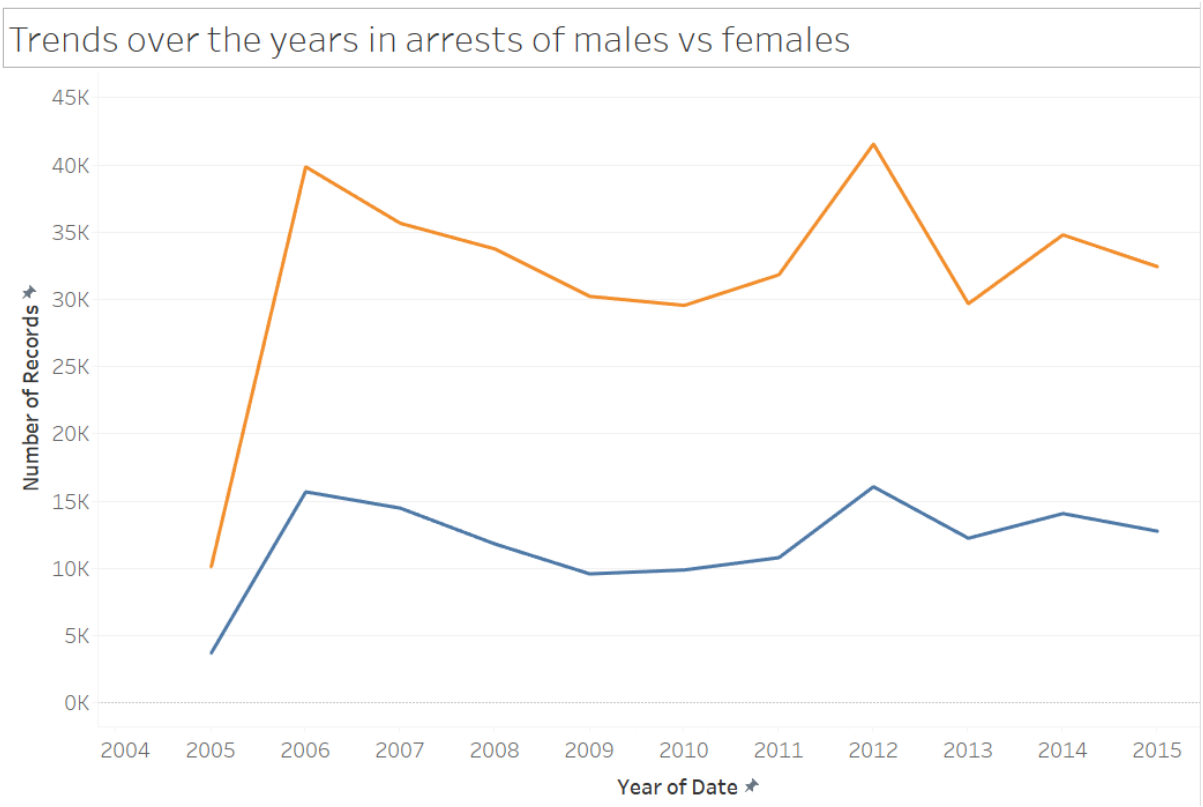
PySpark doesn't allow us to plot data directly. Instead, the easiest way to visualize the data was through Tableau. The following visualisations give an insight into the situation.




The number of arrests increased from the year 2005 to 2006, drastically, but have averaged around 1600 since then, with having its falls and rises.




Males are more likely to be arrested for all different types of violations, expect for APBP where female arrests are higher.




The trends in female and male arrests are not discriminatory, whenever female arrests have spiked, male arrests have also spiked.

## 8.3. Interpretation of the results

From the visualization of processed data, it can be seen that males are more likely to be arrested for all sorts of crimes and are stopped almost twice as often as females. This, in no way is a complete insight into discrepancy, as the baseline comparison with drivers of Rhode Island is not available. To test for discrimination, a test was proposed, but the coefficients of individual features in Logistic Regression could not be obtained in the first iteration, where the model built had higher accuracy.

An analysis of Odds ratio of subject_sex = male, in the second iteration, reveals it to be 1.000000615. A value almost equal to one reveal that police officers are not discriminatory when it comes to CONDUCTING SEARCHES IN DIFFERENT GENDERS, since the searches in different genders are likely to turn up contraband at the same amount. **Whilst males are more likely to be arrested for the same features, searches are not discriminatory with searches turning up contraband in females at the same amount as females.**

## 8.4. Assessment of the results

The second iteration, with the one hot encoder removed, resulted in a model with fairly high accuracy which was attributed to a linearly dependent feature in section 8.1. The high AUC is hence justified.

An analysis of the odds ratio, calculated using NumPy from the coefficient matrix revealed the odds ratio for subject_sex to be nearly equal to 1, which leads us to conclude that there is almost no discrimination in the officers deciding to search vehicles of males and females, since they turn up

contraband at the same rates. A fairly good model is hence built, but it is arguable if the search_conducted should have been fed to the model.

search_conducted being fed to the model led to results which were alarmingly high, but it is an important feature for the model to know to be rightly weigh the coefficients for different features. Hence, it was still fed in the second iteration, and it was concluded that there is no discrimination in searches being conducted on the basis of gender in Rhode Island, but there might have been some discrimination when making arrests, which was concluded in the ISAS Iteration (and can also be seen from the Tableau visualizations in Section 8.2).

### 8.5. Multiple Iterations must take place

As can be seen in the code base and in section 8.1, multiple iterations were done with feature selection and removal of one hot encoding for better interpretation of the coefficients matrix.

## Acknowledgement

I acknowledge that the submitted work is my own original work in accordance with the University of Auckland guidelines and policies on academic integrity and copyright.
(See: https://www.auckland.ac.nz/en/students/forms-policies-and-guidelines/student-policies-and-guidelines/academic-integrity-copyright.html).

I also acknowledge that I have appropriate permission to use the data that I have utilised in this project. (For example, if the data belongs to an organisation and the data has not been published in the public domain then the data must be approved by the rights holder.) This includes permission to upload the data file to Canvas. The University of Auckland bears no responsibility for the student's misuse of data.

## References

Ed. (n.d.). There's Nothing Odd about the Odds Ratio: Interpreting Binary Logistic Regression. Retrieved from https://www.statisticssolutions.com/theres-nothing-odd-about-the-odds-ratio-interpreting-binary-logistic-regression/

Kleinbaum, D. G., & Klein, M. (2002). Logistic Regression A Self-Learning Text. In *Survival*.

Pierson, E., Simoiu, C., Overgoor, J., Corbett-Davies, S., Ramachandran, V., Phillips, C., & Goel, S. (2017). *A large-scale analysis of racial disparities in police stops across the United States*. 1–10. Retrieved from http://arxiv.org/abs/1706.05678