Stop Violations

Independent Project

Technical Report

By:  
Manuel Andrade

Contents

[Introduction 3](#_Toc152366788)

[Data Exploration & Cleaning 3](#_Toc152366789)

[Driver Age Raw and Driver Age 4](#_Toc152366790)

[Categorical Columns: Driver Race, Violation Raw, Violation, Stop Outcome, Is Arrested, and Stop Duration 5](#_Toc152366791)

[Driver Race 5](#_Toc152366792)

[Violation Raw 6](#_Toc152366793)

[Violation 7](#_Toc152366794)

[Is Arrested 7](#_Toc152366795)

[Stop Outcome 8](#_Toc152366796)

[Stop Duration 8](#_Toc152366797)

[Driver Age 9](#_Toc152366798)

[Date Column: Stop Date 10](#_Toc152366799)

[Heatmap Analysis And Cleaning Continued 12](#_Toc152366800)

[Stop Outcome vs Is Arrested 13](#_Toc152366801)

[Stop Duration 14](#_Toc152366802)

[Heatmap Retake 16](#_Toc152366803)

[Superficial Data Analysis 16](#_Toc152366804)

[Decomposition Tree 16](#_Toc152366805)

[Naiive Forecast and Yearly Rank 18](#_Toc152366806)

[Moving Average 20](#_Toc152366807)

[Gender Timeline 22](#_Toc152366808)

[Driver Race Outcome Proportions 23](#_Toc152366809)

[Retrieving Data From Power BI 24](#_Toc152366810)

[Python Logistic Regression 25](#_Toc152366811)

[Untrained Multinomial Logistic Regression 25](#_Toc152366812)

[Confusion Matrix Change 28](#_Toc152366813)

[Multinomial Logistic Regression Training and Testing 29](#_Toc152366814)

[Changing Cutoff for Training and Testing 31](#_Toc152366815)

[Logistic Regression With Low Correlation 31](#_Toc152366816)

[Binary Logistic Regression Model 34](#_Toc152366817)

[Binary Confusion Matrix 35](#_Toc152366818)

[Power BI Logistic Regression Chart 36](#_Toc152366819)

[Conclusion & Recommendation 37](#_Toc152366820)

# Introduction

Thousands of people get pulled over each year in Maryland for various reasons. However, it is interesting to know the result of these stops after the stop. Thus, I have developed a small project to try and address this problem.

The following project is related to traffic stops and their outcomes, where various predictors are used to predict the outcome of a given traffic stop. This involves using Power BI, DAX, Excel, DAX studio, Power Query, and Python to clean and analyze data and then predict outcomes.

Aside from this, the data set is derived from another dataset and is based on an arrest dataset from a county in Maryland. It is important to note that the data set does not represent traffic stops across the USA and applies only to the Maryland County from which this dataset originates.

Therefore, the dataset will be cleaned and analyzed, and predictions will be attempted. Also, some of the steps used to get to the outcome will be briefly discussed.

# Data Exploration & Cleaning

First, the dataset originally contained 11 variables, which were:

* Stop date: date of violation
* Stop time: time of the violation
* Driver gender: gender of violators
* Driver age: age of violators
* Driver age raw: the year the driver was born.
* Driver race: race of violators
* Violation: category of violations which have the following categories:
  + Speeding
  + Moving Violation (Reckless Driving, Hit and run, Assaulting another driver, pedestrian, improper turns and lane changes, etc)
  + Equipment (Window tint violations, Headlight/taillights out, Loud exhaust, Cracked windshield, etc.)
  + Registration/Plates
  + Seat Belt
  + other (Call for Service, Violation of City/Town Ordinance, Suspicious Person, Motorist Assist/Courtesy, etc.)
* Search Conducted: whether a search is conducted in True and False form
* Stop outcome: the result of a violation and our outcome variable
* Is arrested: whether a person was arrested in True or False form
* Stop duration: detained time for violators approx. in minutes
* Drugs related stop: whether a person was involved in a drug crime in true or false format
* Country Name: name of country
* Search type: whether a search was conducted and the type of search.

However, many anomalies are noted when looking at the data through Power Query in Power BI. Specifically, some empty columns and columns had a lot of missing data. The solution was to delete columns with 90% or more missing data points. This involved deleting “country name” and “search type” columns.

Likewise, some columns had missing data points that were much less than 90%. These columns were “driver gender”, “driver race”, “violation raw”, “violation”, “stop outcome”, “is arrested”, and “stop duration”, among others. One possible solution was deleting empty rows in Power Query. This is because there was no way to find the missing data since this dataset is engineered from another dataset. However, note that when a column had missing data in a row, other columns had missing data in that same row.

## Driver Age Raw and Driver Age

Specifically, driver age and driver age raw had missing data in the same rows, as seen below.

A screenshot of a computer

Description automatically generated

Figure - Missing rows

Therefore, the solution became to delete rows that exhibited those characteristics that deleted related rows. For instance, “driver age raw” and “driver age” rows with empty rows were deleted because if there was a row with missing data in “driver age raw”, then driver age had missing data. Even worse, empty cells could not replaced with the average for “driver age raw” because if it were done, the same rule would have to be applied for “driver age”, in which case the averages were different.

Unfortunately, those rows could not be deleted. This is because those were empty and were not considered “null” by Power Query. The solution was to replace those values with 0 and then with “null” and exclude them. Then, delete all values in driver age with a 0.

Aside from this, driver age raw exhibited some extreme values. For instance, the max value for “driver age raw” is 8801. This is the year they were born, which makes no sense, so they were deleted. Also, there are no “driver age” values after 1997, so values that proceeded on or after 1997.

More disappointingly, after researching more into the dataset, it was noted that some of the “driver age raw” years were estimates. Also, the “driver age raw” would not help derive more information and likely cause correlation issues when we apply a model to the dataset. Therefore, it was deleted for the time being. However, this column can be created from “driver age” and “stop date” if necessary. Fortunately, the steps implemented for driver age raw did not affect “driver age.”

## Categorical Columns: Driver Race, Violation Raw, Violation, Stop Outcome, Is Arrested, and Stop Duration

“Driver Race”, “Violation Raw”, “Violation”, “Stop Outcome”, “Is Arrested”, and “Stop Duration” had the same issue where they had empty values in the same rows.

A screenshot of a computer

Description automatically generated

Figure - Power Query Missing Rows

Fortunately, these columns were unrelated, allowing us to explore 2 solutions. These 2 solutions are to either combine missing rows into a new column called “Other” or an existing column on a case-by-case basis.

### Driver Race

Driver Race had 7 instances where the row was empty. However, when looking at a bar chart (see below) it can be noted that there are few observations for Asian, and Other drivers. This will present a problem when implementing a learning model. The solution here is to combine them into the category “Other”. This is because these observations cannot be incorporated into Asian since knowing the driver’s race is impossible.

A graph with numbers and a bar

Description automatically generated

Figure - Driver Race

### Violation Raw

“Violation Raw” had many categories that a driver can be labeled as. However, many categories have few observations compared to others. This can be seen below in the bar chart.

A screen shot of a graph

Description automatically generated

Figure - Violation Raw Distribution

In this case, the observations with 700 observations or less will be combined into “Other.” The “Violation” column has a different approach.

### Violation

Unlike “Violation Raw,” the column “Violation” frequency is not too dispersed, as seen below. A similar approach to “Violation Raw” was taken to put categories with few observations into “Other.”

A graph with blue squares

Description automatically generated

Figure - Violation Distribution

### Is Arrested

Unfortunately, categories with few observations cannot be put in the blank into an “Other” category as the column is binary. Likewise, whether those stops ended in an arrest cannot be determined. The solution is to get rid of those 7 observations.

A graph with a bar

Description automatically generated with medium confidence

Figure - Is Arrested Frequency

### Stop Outcome

In the “Stop Outcome” column, there are various outcomes. However, they are very dispersed with many categories, as seen below. In this case, categories with less than 348 observations will be put into the “Other” category.

A white background with blue lines

Description automatically generated

Figure - Stop Outcome Distribution

### Stop Duration

Unlike the other categorical columns, the “stop duration” is not too extreme and will be left as is. This is seen below.

A graph with blue rectangles

Description automatically generated

Figure - Stop Duration Distribution

## Driver Age

When it comes to age, the distribution must be checked. After this, outliers above or below the lower and upper limits defined using DAX in Power BI will be checked. The 25th and 75th percentiles must be calculated inclusively (example below), then the lower and upper limit values (example below).

DAX Code:

25th Per =

PERCENTILE.INC(

    traffic\_violaions[driver\_age],

    .25

)

Lower Limit =

VAR IQR = [75th Per] - [25th Per]

VAR LLim = [25th Per] - (1.5\*IQR)

RETURN

LLim

Aside from this, looking at the We note that the distribution is skewed to the right, which means that most of the observations are to the right. This is based on the average being greater than the median.

A graph of a number of people

Description automatically generated with medium confidence

Figure - Age Histogram With U/L Limits

Although the lower limit is mathematically correct, it must be adjusted to 0. This is because there are no negative ages. The upper limit is correct and makes sense, considering the context. This brings us to 2 Possible Solutions:

* 1) Ages above 73 can be changed to 34 to keep data.
* 2)Delete ages above 73. In this case, we only have 153 observations with ages over 73 and 49,556 observations.

After careful consideration, only 0.31% of the observations are above 73, meaning they will be deleted.

## Date Column: Stop Date

For the column date, it is essential to note that there are no missing values. However, there would be an issue if a star schema is to be implemented.

This is because the dates themselves are not complete. For example, looking at the monthly distributions, the months do not have equal distributions (see below). In a star schema, we want the date column to have all the dates for the given range. This means every day in that range is needed from January 1st 2021 to December 31st 2021.

A graph with numbers and a number of days

Description automatically generated with medium confidence

Figure - Monthly Distribution

Thankfully, this issue may be fixed by creating a date column for the dimension table. Please note that creating a dimension table is not necessary for this project but serves as a demonstration.

First, a new table is created with all the dates from the range in the fact table. A new table is created using DAX and Power BI to create a date dimension table. Then, the following DAX code is implemented:

Date\_Dim = CALENDAR(

    DATE(2005,1,1),

    DATE(2011,10,31)

)

This creates the first column needed to create a date key and other columns like year, month, day, and similar. Then, the “Year” column is built based on the date column created using the following code:

Year =

YEAR(

Date\_Dim[Date]

)

A similar approach is taken to create month and day columns.

Aside from this, the “Date Key” column must be created in the Date Dimension first and then referenced in the fact table. This is because the filter direction of the relationship goes from the dimension table to the fact table, which allows us to insert attributes from the dimension and fact table into a figure and display them. After completing this, further analysis will be performed using a heatmap.

# Heatmap Analysis And Cleaning Continued

Before implementing the model, checking for high correlation among the variables is crucial. If not, the risk of high p-values and poor model performance is taken. Expressly, variables greater than 0.8 in absolute will be excluded, and further investigation will be done before dropping them.

Moreover, two approaches exist to develop a heatmap: either utilizing Power BI or employing Python or R within Power BI. In this case, Python within Power BI is chosen. This decision is driven by the recognition that using Power BI alone to create a heatmap involves extensive work and is often not recommended. It must be ensured that Python is installed within Power BI and that the pandas, matplotlib, and seaborn libraries are installed, too.

After these initial considerations, the Python script logo and the necessary columns are selected on a blank page. It is essential to transform category columns into dummy columns to implement them successfully into the heatmap. Notably, the exclusion of the date column is emphasized to prevent the creation of an excessive number of categories. The code used for the heatmap is as follows:

* + - # The following code to create a dataframe and remove duplicated rows is always executed and acts as a preamble for your script:
    - # dataset = pandas.DataFrame(Date\_Key, driver\_age, driver\_gender, driver\_race, drugs\_related\_stop, is\_arrested, search\_conducted, stop\_duration, stop\_outcome, violation, violation\_raw)
    - # dataset = dataset.drop\_duplicates()
    - # Paste or type your script code here:
    - import pandas as pd
    - import matplotlib.pyplot as plt
    - import seaborn as sns
    - dataset = pd.get\_dummies(dataset, columns=['driver\_gender', 'driver\_race', 'drugs\_related\_stop', 'is\_arrested', 'search\_conducted', 'stop\_duration', 'stop\_outcome', 'violation', 'violation\_raw'])
    - dataset = dataset.drop(columns=['driver\_gender\_F', 'drugs\_related\_stop\_False', 'is\_arrested\_False', 'search\_conducted\_False'])
    - f, ax = plt.subplots(figsize=(16,12))
    - sns.heatmap(dataset.corr(), annot=True, fmt=".1f", ax=ax)
    - plt.show()

The result is shown below:

A red and pink squares with black and white text

Description automatically generated

Figure - Initial Heatmap

Unfortunately, there were variables with an absolute value greater than 0.8, which must be investigated.

Of note are the following columns: “Stop Outcome” vs “Is Arrested” and “Stop Duration” vs “Stop Duration”. This high correlation is likely due to few observations or because they are closely related.

### Stop Outcome vs Is Arrested

When examining a histogram, one observes only 1665 arrested drivers in the "Stop Outcome" and 1865 drivers in the arrested column. This inconsistency arises because the encounter ends when someone is arrested. Consequently, dropping the "Is Arrested" column and concentrating on the “Stop Outcome” column becomes logical. Additionally, the focus is on potential outcomes, and the dependent variable is "Stop Outcome."

A screenshot of a graph

Description automatically generated

Figure - Is Arrested vs Stop Outcome

### Stop Duration

The high correlation within “Stop Duration” may be due to a few observations. Specifically, a notable correlation exists between the "16-30 Min" and "30+ Min" categories only. This suggests combining these two categories into a new "16+ Min" category to indicate that some stops take longer than 15 minutes. This consolidation will be executed using Power Query, as it can revert a step in case of an error. This process allows a reevaluation of whether there is a high correlation between the predictors themselves and the outcome variable.

A graph with blue rectangular bars

Description automatically generated with medium confidence

Figure - Stop Duration Distribution

### Heatmap Retake

The technique worked, and now the heatmap has no absolute correlations greater than 0.80. The following image depicts the result.  
A screenshot of a computer

Description automatically generated

Figure - Heatmap Reattempt

Now that the data has been cleaned up as best as possible, it is time to do some quick analysis.

# Superficial Data Analysis

Various superficial analyses were conducted, encompassing a decomposition tree, naive forecast, yearly rank, moving average, gender time analysis, and driver outcome proportions. The objective was to extract insights from the data.

## Decomposition Tree

Initially, the aim is to discern which violations and driver races predominantly lead to specific outcomes. Unfortunately, in each instance, the selected violation resulted in a higher population of White drivers, indicating a lack of proportionality in the data. However, it's noteworthy that Black drivers were proportionally arrested more than white drivers in all cases. An example of this is provided below.

A screenshot of a computer

Description automatically generated

Figure - White Drivers Tree Map

A screenshot of a computer

Description automatically generated

Figure - Black Drivers Tree Map

This is intriguing and calls for more exploration to see what the future looks like and what years had the most arrests and citations.

## Naiive Forecast and Yearly Rank

To create a yearly rank, DAX will be used to make an annual rank chart to rank the number of violations by year. First, the number of violations is counted for the table matrix itself by using the following code:

* + countrows =
  + COUNTROWS(
  + traffic\_violaions
  + )

Next, the ranking process ensures that the value does not sum in the totals section. This necessitates the utilization of the HASONEVALUE() and IF() functions to return only one value when selected; otherwise, nothing will be selected. Additionally, the RANKX() function is employed to rank the counts. It's crucial to note that the year row must align with the Date\_Dim[Year] in the functions. The code for this process is presented below:

* + Rank of Years =
  + IF(
  + HASONEVALUE(
  + Date\_Dim[Year]
  + ),
  + RANKX(
  + ALL(Date\_Dim[Year])
  + ,
  + [countrows]
  + )
  + )

Apart from this, to create a Naive Forecast, a line chart is chosen with Year on the x-axis and Count Rows on the y-axis. Following this, the "Forecast" option is selected, and the forecast length is set to 4 years, indicating a projection 4 years into the future.

Following these steps, a slicer for the stop outcome is added, resulting in the generation of the following outcomes:

A graph on a white background

Description automatically generated

Figure - Naiive Forecast & Yearly Rank

Interestingly, note that 2006 had the most stops, and the forecast was flat. However, this changes when we slice by arrest driver. This is because the forecast gains a minor uptrend but then falls. The rank is still the same.

A graph on a screen

Description automatically generated

Figure - Naiive Forecast & Yearly Rank For Arrest Drivers

As for citations, note that the forecast and the rank do not differ when all the categories are included. Also, this brings the moving average into consideration.

A graph on a screen

Description automatically generated

Figure - Naiive Forecast & Yearly Rank For Citations

### Moving Average

To develop a moving average, the process involves creating an Average Years dimension containing a series to define the moving average. Also, it is necessary to initiate a new table and then input the following code to generate a series of 4 ranging from 1 to 4 to create this table:

* Average Years =
* GENERATESERIES(
  + 1,
  + 4,
  + 1
* )
* Then create the following DAX measure to select 1 by default in case more than one value is selected:  
  Average Years Value =
* SELECTEDVALUE(
  + 'Average Years'[Value],
  + 1
* )

Moreover, a matrix needs to be created with Year in the rows. For the values, incorporate the countrows measure (the previously defined measure) and introduce a new measure that computes the moving average years. To formulate this measure, implement the following code:

* Moving Avg Yr =
* VAR LastTransactionYr = MAX(Date\_Dim[Year])
* VAR AvgYr = [Average Years Value] -- here we put the average years series into a variable to later implment it
* VAR PeriodInVisual =
* FILTER(
* ALL(
  + Date\_Dim[Year] -- do not want any filters applied
* ),
* AND(
  + Date\_Dim[Year] > LastTransactionYr - AvgYr, -- want the year you are at to be less than the difference between the max year and the selected average year
  + Date\_Dim[Year] <= LastTransactionYr -- want the year you are at to be less than or equal to the max date
* )
* )
* VAR Output =
* CALCULATE(
* AVERAGEX(
  + 'Date\_Dim',
  + [countrows] -- count the average stops per date
* ),
* PeriodInVisual -- enter in our variable filter
* )
* Return
* Output

Proceeding with the process, incorporate a slicer for the moving average series, driver gender, and driver race.

Subsequently, observe that the moving average values are consistently low regardless of the chosen moving average, gender, or race. This low-value occurrence stems from the absence of records for certain months and dates within the date range. In simpler terms, it is advisable not to rely too much on this information.

A screenshot of a computer

Description automatically generated

Figure - Moving Average Matrix

## Gender Timeline

Interestingly, the timeline pattern for both driver genders was the same, as seen below. There was a peak in 2006. After 2006, there was a gradual decline in stops.

A graph of a graph

Description automatically generated with medium confidence

Figure - Gender Timeline

Moreover, both genders peaked in arrests and citations in 2006 and gradually declined. However, the number of warnings increased after 2008, which may help explain the decrease in arrests and citations.

A graph of a person and person

Description automatically generated

Figure - Gender Timeline Arrests

A graph of a person and person

Description automatically generated

Figure - Gender Timeline Citations

A graph of a person and person

Description automatically generated

Figure - Gender Timeline Warning

Nonetheless, diving deeper into stop outcomes proportioned by the driver races would be interesting.

## Driver Race Outcome Proportions

Various interesting insights emerge when creating a matrix and examining the proportions for the “Stop Outcomes” broken down by driver race. Initially, black drivers exhibit the highest proportion of arrests compared to other drivers, coupled with the second-highest warning rate proportionally. Meanwhile, Hispanic drivers register the second-highest arrest rate proportionally among drivers but have the second-lowest rate proportionally, indicating a lower frequency of warnings.

A screenshot of a graph

Description automatically generated

Figure - Driver Stop Outcomes Proportions

Further investigation into the distribution of stop outcomes for Black and Hispanic drivers intrigued me, especially due to the high proportion of arrests. Surprisingly, we observed that the distribution is roughly the same for both types of drivers. This similarity in distribution may pose challenges when implementing our model.

A screen shot of a graph

Description automatically generated

Figure - Black & Hispanic Driver Outcomes

# Retrieving Data From Power BI

Before implementing the model, it's necessary to retrieve the current data from Power BI. Power BI is not intended for use as a cleaning or retrieval tool. However, there are two possible options to accomplish this task:

1. Navigate to the table view, select the desired table, right-click a column, and copy and paste the results into an Excel CSV file.
2. Open DAX Studio while Power BI is running, then open an empty Excel file. Proceed to the data tab, select "Get Data," and choose "Launch Power Query Editor." From there, go to the home tab and select a new source, choosing "Database" and then "Analysis Service." Finally, copy the DAX Studio Local Hostname into the hostname and select "Close and Apply."

The easiest option is Option 1, but it may not be suitable for large datasets. On the other hand, Option 2 requires more work after importing the data, such as changing column names. In this case, it's suggested that if one wants to follow along, choose Option 1.

# Python Logistic Regression

Now that the data is imported into a CSV file, the modeling stage will take initiation. Specifically, the model that will be used on the data is the logistic regression model. Also, not going over most of the steps except the crucial ones. The steps can be found inside the Python file. Likewise, the following columns were dropped because they would generate too many categorical variables:

* "Date Key," "Stop Date," "Stop Time," "Violation Raw"

Moreover, dropping "Violation Raw" was crucial because it had the same categories as "Violation." As for "Date Key," it was used as a key only.

## Untrained Multinomial Logistic Regression

After reading the CSV file and converting categorical columns into categories, a multinomial logistic regression model was implemented using the entire dataset. This was done to identify any underlying issues and gauge the model's performance during training and testing.

Fortunately, no issues were encountered, and the multinomial logistic regression model was successfully optimized. A summary of the results and various interesting details are provided.A screenshot of a computer

Description automatically generated

Figure - Multinomial Logistic Regression Model Summary with Entire Data

First, the poor-performing variables in the outcomes were driver gender, and drugs related stop because they are greater than 0.05, which means they failed the hypothesis test. Also, a higher pseudo R-squared suggests that the model explains a more significant proportion of the variance in the dependent variable. In this case, it is somewhat moderate. Likewise, it is worth noting that the interpretation of goodness of a Pseudo R-Squared value depends on many factors, like the context of the data analysis. The Pseudo R-squared does not have a universal benchmark.

Nonetheless, I implemented a confusion matrix to understand the predictions better. Specifically, some observations were observed, but the model indicated no observations or predictions. This is a major issue that the current cutoff rate may cause.

A screenshot of a computer

Description automatically generated

Figure - Initial Confusion Matrix With Default Cutoff

In fact, there are recalls of 0 for 2 and 3. This means that the model failed to identify any positive instances. Thus, the model did not capture any of the true positives, resulting in a recall failure for the positive instances in the dataset.

Likewise, classes 2 and 3 have an f1 score of 0, resulting from the recall being 0. This means the model cannot identify any positive instances for these classes. Also, note that the precision is 1 for both these classes, which means there is an inability to balance precision and recall effectively.

A screenshot of a computer screen

Description automatically generated

Figure - Initial Default Confusion Matrix

### Confusion Matrix Change

Again, the cutoff rate can be decreased to attempt to gain some positive insights. Additionally, the poor outcome may be linked to there being very few observations for the "Other" (2) class and Warning (3) class, resulting in a high correlation. The histogram and heatmap from the previous stop outcome histogram can be examined to verify this suspicion.

Next, the cutoff is changed to 0.5 and 0.2 for all the categories. However, no improvement is observed, so an experiment is conducted by changing the cutoff individually for each category.

A screenshot of a computer screen

Description automatically generated

Figure - 0.5 Cutoff Initial Matrix Report

A screenshot of a computer screen

Description automatically generated

Figure - 0.2 Cutoff Initial Matrix Report

There was an improvement when we held the cutoff for the first 2 classes at 0.5, and changed the cutoff for “other” and “warning” to 0.2. Specifically, there was an improvement in “other”, but that improvement was minimal.

A screenshot of a computer screen

Description automatically generated

Figure - Initial Cutoff Changed Individually

## Multinomial Logistic Regression Training and Testing

Finally, the data is split into training and testing to assess how the model will perform using an 80/20 split. This approach provides better insight into how the model performs with unseen data.

A screenshot of a data sheet

Description automatically generated

Figure - Training & Splitting Multinomial Logistic Regression Results

Unsurprisingly, the results were not significantly different from the previous results. This can be further seen with the confusion matrix and the classification report.

A white square in a black and white square

Description automatically generated

Figure - Confusion Matrix for Training and Testing

A screenshot of a computer screen

Description automatically generated

Figure - Confusion Matrix Report for Training and Testing

### Changing Cutoff for Training and Testing

Similarly, the results were still poor even after changing the cutoff for the confusion for all the classes and individually for each class.

A screenshot of a computer screen

Description automatically generated

Figure - Cutoff Change to 0.2 For All Classes For Training and Testing

A screenshot of a computer screen

Description automatically generated

Figure - Individual Cutoff Change: Arrest Driver = 0.5, Citation = 0.5, Other = 0.2, Warning = 0.5

Further configurations will be done, and categories will be combined or dropped.

## Logistic Regression With Low Correlation

When revisiting the Power BI heatmap, a few noteworthy details become apparent. First, in examining the outcome categories, a high correlation is observed between "Driver Race Black" and "Driver Race Hispanic." This is likely attributed to a few observations, with their correlation being 0.5 in absolute terms.

As previously mentioned, the distribution between "Driver Race Black" and "Driver Race Hispanic" is similar. To address this, the solution involved converting the Hispanic driver’s class to "Other" to preserve those observations.

A screenshot of a computer screen

Description automatically generated

Figure - Distribution of Black and Hispanic Drivers from Power BI

Similarly, a high correlation is observed between the "Warning" and "Arrest Driver" outcome categories, likely due to few observations for these outcome categories. Consequently, these were moved into the "Other" category, transforming the model into a binary model.

Additionally, the violation predictor exhibited two highly correlated categories with "Citation": "Speeding" and "Moving Violation," with absolute correlations of 0.7 and 0.5, respectively. The reason may be that "Speeding" and "Moving Violation" could sometimes be interpreted differently. Hence, "Speeding" was reclassified as a "Moving Violation" since it is a moving violation.

Before proceeding, examining a correlation heatmap to identify any existing correlations is essential. Unfortunately, there remains a high correlation between Black and White drivers at 0.69 in absolute correlation. Nevertheless, there is some improvement in correlation. The chosen solution involves converting the category for Black drivers into "Other" due to the higher number of White drivers than Black drivers, resolving the correlation issue and signaling the readiness to move on to the final binomial logistic regression model.

A screenshot of a computer

Description automatically generated

Figure - Heatmap Correlation Between Black and White Drivers

A screenshot of a computer

Description automatically generated

Figure - Heatmap With No Significant Correlation

## Binary Logistic Regression Model

After splitting the model, the summary results for the model were as follows:  
A screenshot of a computer

Description automatically generated

Figure - Initial Summary Results Binary Logistic Model

There are high p-values that are greater than 0.05. This leaves the option of dropping a one of those predictors to see if a difference. Therefore, “Driver Age” will be dropped since it has the highest age.

Again, the model is run, but there are still p-values greater than 0.05. This time, all p-values greater than 0.05 will be dropped, and results will show “Driver Gender” and “Driver Race”.

A screenshot of a computer

Description automatically generated

Figure - 2nd Attempt After Initial Drop

Finally, the model was implemented, and there was a major improvement. Specifically, there were no p-values greater than 0.05.

A screenshot of a computer

Description automatically generated

Figure - No Significant P-Values

### Binary Confusion Matrix

Likewise, the confusion matrix was an improvement at a cutoff of 0.50. The accuracy is 92%. However, it must be noted that the model is flawed as it is extremely good at predicting 0s but awful at predicting 1s. This is because there are very few observations for 1 (Other). Specifically, the model demonstrates excellent precision (93%) and recall (99%) for class 0, indicating accurate identification of this class. However, for class 1, precision is lower (46%), suggesting a higher rate of false positives, and recall is only 10%, indicating a limited ability to capture instances of class 1.

A screenshot of a graph

Description automatically generated

Figure - Binary Classification Report

A screenshot of a computer screen

Description automatically generated

Figure - Binary Confusion Matrix Final Model

Now that the model has been developed, it is time to deploy it in Power BI to create an image.

# Power BI Logistic Regression Chart

Finally, to implement the data frame that contains the predictions and actual outcomes, the model needed to be implemented in Power Query, and a data frame had to be created in Python with these 2 attributes. This was intended to display the results in a pie chart to visualize the results. The results are further exacerbated when looking at them visually. The category “Citation” is predicted well with a 5.8% error. However, the category "Other” is not predicted well. This is because, despite the 5.8% error, there are only 729 observations in the testing data. Not to mention that the model was trained with very few observations of the category “Other” compared to “Actual.”

A screenshot of a computer

Description automatically generated

Figure - Power BI Actual vs. Predict Outcome

# Conclusion & Recommendation

Overall, the data for traffic violation stops for Maryland drivers was explored visually and analyzed thoroughly using DAX and Power Query. Also, this involved cleaning the data with Power Query and Python. This led to implementing a logistic regression that was not accurate at an error rate of 8% for a dataset with few observations for the “Other” category. Nonetheless, there was an improvement in sensitivity for class 1 (“Other”) of 10%, which means that the model captured 10% of the instances belonging to class 1. Aside from this, there must be more data to make better predictions in the future, hopefully 99% accurate predictions, for Maryland drivers who get stopped in a traffic violation.