Analyzing Policing Patterns and Demographic Disparities in U.S. Traffic Stops

Isabelle Krampert
igk9
December 10th, 2025

Links

Github Repository: https://github.com/izkrampert19/IK_CS439_FinalProject
Project Demo:
https://drive.google.com/file/d/1FyQxgJ6M2GWpCiQEsyn83wpltwgMyZ5V/view?usp=sharing

Stanford Open Policing Datasets:
https://openpolicing.stanford.edu/data/
States Used: CO, CT, MA, MS, MT, ND, VT (State Patrol datasets)

US Census Bureau Dataset:
https://www.census.gov/data/tables/time-series/demo/popest/2010s-state-detail.html#par_textimage_673542126
NOTE: Clicking on the link directly may lead to an error page. Please try pasting this link into the search bar instead.

Project Statement, Relevance, & Challenges

Core Question: Can the likelihood of a traffic stop be predicted on race, age, or sex?

Problem Statement:
With this project, I wanted to investigate differences in traffic stop patterns across race, age, and gender. The Stanford Open Policing project, one of my main sources of data, had already studied and found racial disparities in U.S. police stops, but I wanted to go a bit further and examine if there were any disparities between genders and age groups of stopped drivers.

Novelty and Importance:
Although a major study has already been conducted on these datasets concerning racial disparity and policing patterns, I wanted to further investigate if factors like age and sex also influence the likelihood of a stop. This project focuses solely on the chance of a stop alone based on these factors, and does not consider if an arrest or citation is made afterward, unlike the main study. The importance of this project lies in the fact that it is focused on intersectional demographic analysis – for example, contrasting the percentage stops between younger white men vs. older black women – and if this data can improve predictive performance while highlighting police bias in any state. By doing this, the project will hopefully show existing disparities (if any), and whether they are prevalent to make the likelihood of a traffic stop predictable.

Project Limitations / Difficulties:

I experienced many difficulties when cleaning and consolidating both datasets! Since the Stanford Open data counts Hispanic as a "race" and the Census counts it as an ethnicity, I needed to find a way to adapt the Census data to the Stanford Open entries. I ended up just adapting the Census data by marking any synthetic entry marked as Hispanic in origin as "Hispanic" in the final dataset regardless of race, then taking a stratified sample of the synthetic data to prevent bias. I also originally intended to have more state data that had the column info I needed, but some of these sets were so large they weren't able to download properly at all. Plus, I think some states like NY and IL have broken download links, so although these had the columns I needed I wasn't able to view them whatsoever!

There are also limitations with my project; my model is examining just race, age, and gender, when actual traffic stops depend on a multitude of other factors (time of day, location, driving behavior, etc.). This means that it cannot capture the full complexity of policing results. Also, even though I took stratified samples from each dataset, the volume of the data was reduced, which could have impacted the models' performances when it came to race categories.

Models and Algorithms

Logistic Regression:

This is my main predictive model for this project – I chose it because it fits binary classification problems quite well (stopped vs. not stopped) and I thought it would provide more accurate probability estimates than a Naive Bayes model would alone. It captured the relationships between demographic prediction rates and a "stopped" well.

Naive Bayes:

This was my secondary model for this project. It has shown itself to be less accurate than the Logistic Regression model, however it had a simple implementation and it handled categorical data effectively. It also served as a good comparison against Logistic Regression, which highlighted some of Naive Bayes' shortcomings (oversimplifying assumptions, less accurate probability estimates).

```
# Load in CSV
df = pd.read_csv(file_path, low_memory=False)

# Drop null values for key fields
df = df.dropna(subset=["subject_race", "subject_sex", "subject_age", "date"])

# Obtain year from original "date" column
df["Year"] = pd.to_datetime(df["date"], errors="coerce").dt.year
df = df[df["Year"] == YEAR]

# Standardizing race and sex
df["Driver_Race"] = (
    df["subject_race"]
    .str.lower()
    .map(RACE_MAP)
)
df = df[df["Driver_Race"].isin(VALID_RACES)]
```

Progress and Contributions

(Note: I worked solo on this project! I did not have a partner.)

CSV Conversion → csv_conv.R

A very short R script I wrote to convert RDS files into CSV files. For some reason I was unable to properly download any of my chosen state datasets as CSVs, so I downloaded the RDS files instead and just converted them for use in my project!

Data Cleaning → clean_SOP.py:

With this cleaning script, I loaded and standardized the chosen state CSVS from the Stanford Open Policing website. All entries that were missing the information I needed (driver race, age, or sex) were dropped, and the rest had their demographic categories standardized with a mapping dictionary. I also filtered all of the data in these CSVs to be from 2014, just so everything was standardized and would match up with the Census data. After getting all of the data filtered, I took stratified samples from every state (~50,000 people per state) in order to prevent states with large populations from making the cleaned stop dataset biased. Finally, the script puts all these samples into the final, cleaned dataset for stopped individuals. The output CSV is named "cleaned.csv", and is saved to the "processed" folder within the "data" folder. A snippet of the cleaning code is shown above.

Baseline Population Construction → merge_USBC_SOP.py:

Since the Stanford Open dataset only included stopped entries, I needed a general population baseline of non-stopped entries to use as a comparison. I used the merge_USBC_SOP.py script to not only clean the US Census Dataset, but to also generate a synthetic population as a baseline for the cleaned Stanford Open set. It adds a new column to the final dataset called "Stopped" which can be either 1 (stopped) or 0 (nonstopped). It also merges the cleaned synthetic Census data with the cleaned Stanford open data, making a dataset that is ready to be used for training by both my logistic regression and naive bayes models. The output CSV is named "training_data.csv", and is saved to the "processed" folder also within the "data" folder. Below is an image of how I mapped the codes to each demographic.

```python
# Create mappings for demographic codes used in CSV
# Further info about KEY found in "sc-est2019-alldata5_KEY.pdf"
CENSUS_MAPPINGS = {
    'sex': {
        # 0 will eventually be excluded
        0: 'Total',
        1: 'Male',
        2: 'Female'
    },
    'origin': {
        # 0 will eventually be excluded
        0: 'Total',
        1: 'Not Hispanic',
        2: 'Hispanic'
    },
    'race': {
        # As stated in sc-est2019-alldata_KEY.pdf
        # These do not match up with the Stanford Open Data currently, but this will be fixed later
        1: 'White Alone or in Combination',
        2: 'Black Alone or in Combination',
        3: 'American Indian and Alaska Native Alone or in Combination',
        4: 'Asian Alone or in Combination',
        5: 'Native Hawaiian and Other Pacific Islander Alone or in Combination'
    }
}
```

Model Training → model_training.py:

This script handles the logistic regression and naive bayes models' training. It encodes categorical variables, then splits the data into training and testing sets with stratification. After the models are trained on training_data.csv, they're then saved to the "models" folder along with the training data itself. Once this script finishes running, the next script can be run. If model_training.py is not run before model_analysis, the latter will not work, as it relies on the output models generated from the training script!
A snippet of code from the model training function is displayed on the right.

```
print("Training Models...")
# logistic regression
lr_model = LogisticRegression(
    random_state=42,
    max_iter=1000,
    solver='lbfgs'
)
lr_model.fit(X_train, y_train)

# naive bayes
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
```

```
# Making the actual predictions
y_pred = model.predict(X_test)
y_pred_proba = model.predict_proba(X_test)[:, 1]

# Calculating the metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred_proba)
con_mat = confusion_matrix(y_test, y_pred)

# Storing the results
results[model_name] = {
    'accuracy': accuracy,
    'precision': precision,
    'recall': recall,
    'f1': f1,
    'roc_auc': roc_auc,
    'y_pred': y_pred,
    'y_pred_proba': y_pred_proba,
    'confusion_matrix': con_mat
```

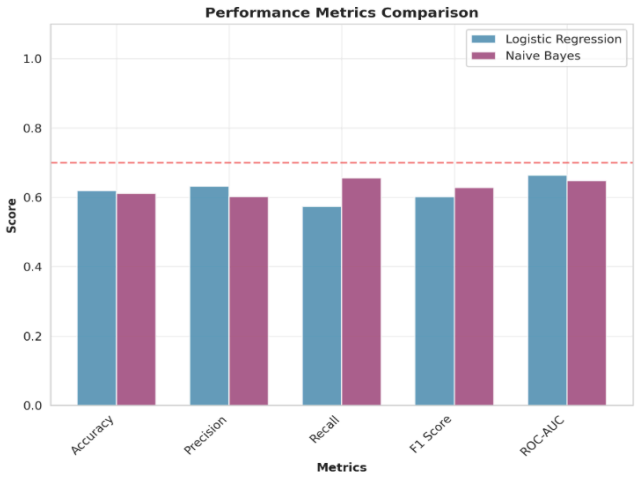
Model Analysis → model_analysis.py:

This script handles model evaluation and analysis for logistic regression and naive bayes. It generates the fairness CSV with the final model performance metrics, broken down into each demographic. Additionally, I've made it so that it creates a PNG with four plots (performance metrics comparison, ROC curves, confusion matrices for logistic regression and naive bayes models). This way, I could have a visual aid when comparing and contrasting model performance. It also prints performance metrics directly in the terminal as the program is running – I saved this printed output in a text file called "terminal_analysis.txt" To the left is a snippet of my code that makes the predictions and calculates metrics.

Experimental Design

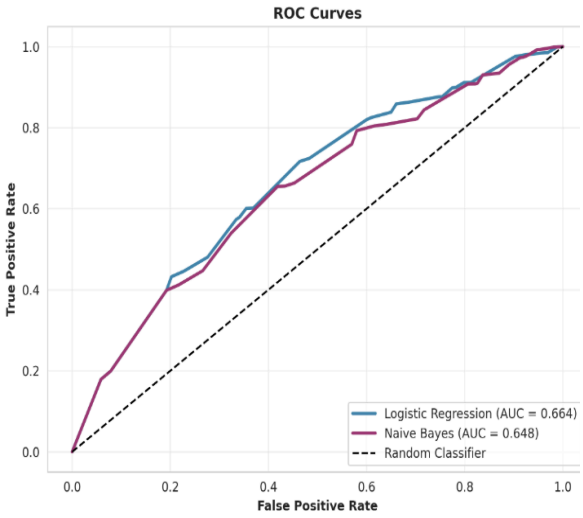
Experiment:

My own prediction before implementing the model was that it would show a higher stop likelihood for less populous racial demographics, younger age groups, and for men. I hypothesized that if race, age, and sex were to greatly influence stop likelihood in real data to begin with, then machine learning models should be able to predict



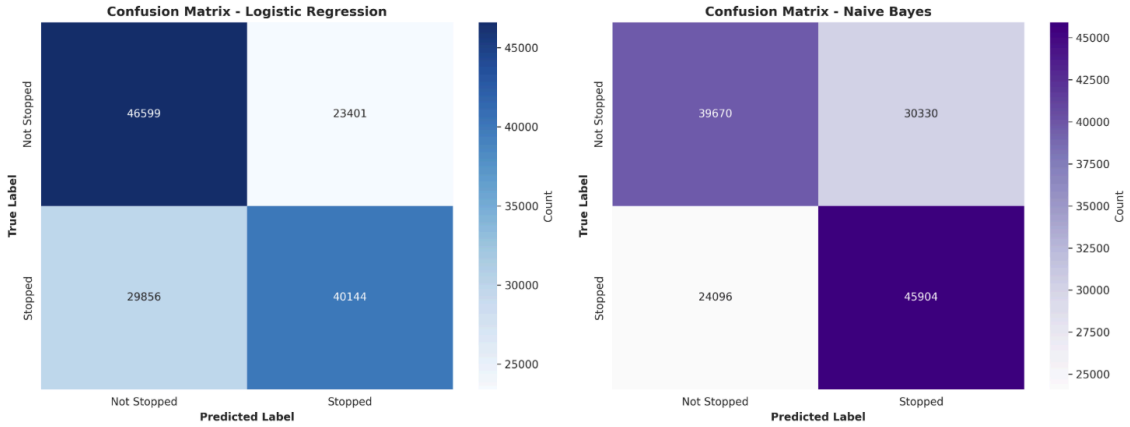
differences in stop likelihood probabilities amongst different demographics. To do this, I trained my models on real recorded data for the stopped individuals, and a synthetically generated baseline for non-stopped individuals. Afterward, I evaluated the models and their predictions, and found that the Logistic Regression model performed better overall against the Naive Bayes model.

Key Findings and Results



Overall Model Performance:

The bar chart above, the graph to the left, and the confusion matrices on the bottom are outputs of my model_analysis.py script. Overall, based on the performance metrics from my model analysis script, *Logistic Regression performed better than Naive Bayes*. The ROC curves display that both of these models do have some accuracy when it comes to demographic-related predictions, but I don't think either of them have strong enough predictive power to fully and accurately model traffic stop outcomes. I stated in my project proposal that I would consider a ROC-AUC score of above 0.7 to be considered sufficient, but neither of my models met this requirement (Logistic Regression had a score of 0.6637, Naive Bayes 0.6481).



Fairness Analysis Results

As mentioned previously – and in the context of what this project is analyzing – Logistic Regression and Naive Bayes have their limitations. They were able to capture the fact that age and sex were strong predictors of a driver being stopped, but when it came to race, the models failed. Regarding age group, the models show that the 25-39 age group had the highest predicted stop rates (Actual 65.26%, Predicted 85.49%), denoting that the Logistic Regression model had strongly associated this group with "stopped".

Again, we can see that sex is a strong predictor of being stopped, as there is an incredibly wide gap between the predicted stop rate for men vs. women (65.23% vs. 17.9%, respectively).

Shortcomings

In contrast to these, there are no large gaps shown between race categories' prediction probabilities, which means that race is not a strong enough predictor for the model to determine a predicted stop rate. When the predicted rates = 0, it means the models completely failed to obtain a probability or learn any patterns – we see this for the "Under 25" age group and the "Asian/Pacific Islander" race category. We can also see the Naive Bayes model failing for the "Asian/Pacific Islander" and "Black" race categories. This could be indicating that there were oversights in my sampled populations (perhaps the synthetic baseline had underrepresented categories?) or that the simplicity of Logistic Regression and Naive Bayes models' prevented them from learning certain things. Below is a picture of the fairness_analysis CSV generated by model_analysis.py.

Category	Predicted_Stop_Rate	Count	Actual_Stop_Rate	Avg_Probability	NB_Predicted_Rate	Demographic
Asian/Pacific Islander	0	29817	0.2082	0.3101	0	Race
Black	0.1943	79387	0.5547	0.389	0	Race
Hispanic	0.3166	87295	0.2492	0.4466	0.3771	Race
White	0.5451	503501	0.5521	0.538	0.6924	Race
25-39	0.8549	193496	0.6526	0.6111	0.7752	Age_Group
40-64	0.5143	239113	0.5183	0.5378	0.5143	Age_Group
65+	0.3924	74148	0.2429	0.4432	0.4637	Age_Group
Under 25	0	193243	0.4232	0.3636	0.3838	Age_Group
Female	0.179	293894	0.3954	0.3951	0.179	Sex
Male	0.6523	406106	0.5757	0.5759	0.8099	Sex

Conclusion & Takeaways:

To conclude, the outcome of this project shows that age, sex, and race do have an impact on the likelihood of a traffic stop, but to an extent. The models were able to pick up on larger trends (higher stop rates for men and people in the 25-39 age group). Despite this, the models struggled with certain demographic categories that were smaller – even with stratified sampling – indicating that model performance is not dependent on demographics alone, and high accuracy is simply not possible with both the Logistic Regression and Naive Bayes models in place. Ultimately, this project helped me understand both the limitations and capabilities of predictive modeling.