**Project Framework: Terry Stop Arrest Prediction**

**In**[**Terry v. OhioLinks to an external site.**](https://www.oyez.org/cases/1967/67)**, a landmark Supreme Court case in 1967-8, the court found that a police officer was not in violation of the "unreasonable search and seizure" clause of the Fourth Amendment, even though he stopped and frisked a couple of suspects only because their behavior was suspicious. Thus was born the notion of "reasonable suspicion", according to which an agent of the police may e.g. temporarily detain a person, even in the absence of clearer evidence that would be required for full-blown arrests etc. Terry Stops are stops made of suspicious drivers.**

**Build a classifier to predict whether an arrest was made after a Terry Stop, given information about the presence of weapons, the time of day of the call, etc. This is a binary classification problem.**

**Note that this dataset also includes information about gender and race. You may use this data as well. You could conceivably pitch your project as an inquiry into whether race (of officer or of subject) plays a role in whether or not an arrest is made.**

**If you do elect to make use of race or gender data, be aware that this can make your project a highly sensitive one; your discretion will be important, as well as your transparency about how you use the data and the ethical issues surrounding it.**

**1. Problem Statement**

Predict whether an arrest was made following a Terry Stop, based on contextual features such as presence of weapons, time of stop, gender, and race. The project also explores the potential influence of demographic factors on arrest outcomes.

**2. Data Preprocessing**

* **Handling Missing Data**: Clean or impute null values appropriately.
* **Encoding Categorical Variables**:
  + One-hot encoding for time of day, officer rank, etc.
  + For sensitive attributes like race and gender, ensure you encode in a way that maintains interpretability.
* **Feature Engineering**:
  + Create derived features (e.g., time of day → night/day; weapon\_present → binary).

**3. Exploratory Data Analysis (EDA)**

* Arrest rate by:
  + Time of day
  + Presence of weapons
  + Gender
  + Race
* Check for correlations and imbalance (especially across racial categories).

**4. Ethical Considerations**

* Be transparent: Clearly document how race/gender data is used.
* Consider using **fairness metrics**:
  + Demographic parity
  + Equal opportunity
  + Disparate impact
* Acknowledge limitations: Historical bias in data may influence outcomes.

**5. Modeling**

* Try multiple classifiers:
  + Logistic Regression (interpretable)
  + Random Forest (handles non-linear relationships well)
  + XGBoost (performance-optimized)
* Address **class imbalance** using techniques like:
  + SMOTE
  + Class weighting

**6. Model Evaluation**

* Metrics:
  + Accuracy, Precision, Recall, F1-Score
  + ROC-AUC
* Fairness evaluation:
  + Stratified performance across racial/gender groups

**7. Interpretation**

* Feature importance: Which features most influence arrest predictions?
* SHAP or LIME for individual predictions.

**8. Optional: Fairness Mitigation**

* Pre-processing techniques (e.g., reweighing)
* In-processing (e.g., adversarial debiasing)
* Post-processing (e.g., reject option classification)

**9. Conclusion**

* Summary of results and performance
* Ethical implications of using sensitive features
* Recommendations for law enforcement transparency

**10. Bonus Ideas (If You Want To Go Deeper)**

* Cluster analysis of stops to identify behavioral patterns.
* Time series analysis to explore patterns over the years.
* Sentiment analysis (if officer notes or narratives are present).

**✅ Terry Stop Arrest Prediction – Classification Project Outline (Using Your Dataset)**

**🧾 1. Objective**

Build a binary classifier to predict whether an **arrest was made after a Terry Stop**, based on factors like:

* **Presence of weapons**
* **Time of stop**
* **Race and gender** (with ethical care)

**📥 2. Data Loading (Python Code)**

python

CopyEdit

import pandas as pd

# Load the Terry Stops dataset

df = pd.read\_csv("Terry\_Stops\_20250718.csv")

# Preview the data

print(df.head())

print(df.info())

**🔍 3. Exploratory Data Analysis (EDA)**

Check class distribution and explore sensitive attributes:

python

CopyEdit

# Class distribution

print(df['arrest\_made'].value\_counts(normalize=True))

# Check unique values

print(df['subject\_race'].unique())

print(df['subject\_gender'].unique())

print(df['weapon\_found'].value\_counts())

Optional: Visualize arrest rates by race or gender:

python

CopyEdit

import seaborn as sns

import matplotlib.pyplot as plt

sns.barplot(x='subject\_race', y='arrest\_made', data=df)

plt.title("Arrest Rate by Race")

plt.show()

**⚙️ 4. Feature Engineering**

You may need to clean and create some variables:

python

CopyEdit

# Convert time column to hour

df['hour'] = pd.to\_datetime(df['stop\_time'], errors='coerce').dt.hour

df['is\_night'] = df['hour'].apply(lambda x: 1 if x >= 18 or x < 6 else 0)

# Create binary variable for weapon found (if not already)

df['weapon\_found'] = df['weapon\_description'].notnull().astype(int)

# Drop rows with missing target or key features

df = df.dropna(subset=['arrest\_made', 'subject\_race', 'subject\_gender', 'weapon\_found'])

**🧪 5. Train-Test Split and Preprocessing**

python

CopyEdit

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

# Define features and target

features = ['subject\_race', 'subject\_gender', 'weapon\_found', 'is\_night']

target = 'arrest\_made'

X = df[features]

y = df[target].astype(int)

# Split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, stratify=y, random\_state=42)

# Preprocessing

preprocessor = ColumnTransformer([

('num', StandardScaler(), ['weapon\_found', 'is\_night']),

('cat', OneHotEncoder(drop='first'), ['subject\_race', 'subject\_gender'])

])

**🤖 6. Model Training**

Example with logistic regression:

python

CopyEdit

from sklearn.linear\_model import LogisticRegression

from sklearn.pipeline import Pipeline

pipeline = Pipeline([

('preprocessor', preprocessor),

('classifier', LogisticRegression())

])

pipeline.fit(X\_train, y\_train)

You can also try RandomForestClassifier or XGBClassifier.

**📈 7. Evaluation**

python

CopyEdit

from sklearn.metrics import classification\_report, confusion\_matrix, roc\_auc\_score

y\_pred = pipeline.predict(X\_test)

y\_proba = pipeline.predict\_proba(X\_test)[:, 1]

print(classification\_report(y\_test, y\_pred))

print("ROC AUC:", roc\_auc\_score(y\_test, y\_proba))

**📊 8. Fairness Check (Optional but Recommended)**

python

CopyEdit

# Check arrest rate by race

print(df.groupby('subject\_race')['arrest\_made'].mean())

# Stratified evaluation

for race in df['subject\_race'].unique():

race\_df = df[df['subject\_race'] == race]

print(f"{race}: {race\_df['arrest\_made'].mean():.2f}")

**🔐 9. Ethical Reflection**

Including race and gender introduces fairness and discrimination risks. Explain how the model's predictions will be interpreted, avoid over-reliance on biased historical patterns, and consider using fairness constraints or post-processing techniques to balance outcomes.

**🧾 10. Final Report Contents**

I’ll generate a downloadable Word report for this once the tools are re-enabled. It will include:

* Executive summary
* Modeling steps
* Evaluation metrics
* Visuals (confusion matrix, ROC)
* Ethical considerations