Police Dataset Analysis - Jupyter Notebook Documentation

Overview

This Jupyter Notebook analyzes police checkpoint data to understand patterns in traffic stops, violations, and demographic factors. The analysis uses pandas DataFrames to clean, filter, and analyze the dataset to answer key questions about police stop behaviors and outcomes.

Dataset Description

The dataset contains information from police checkpoints including:

- Driver demographics (age, gender)
- Violation types
- Stop duration
- Search conducted status
- Other relevant traffic stop information

Code Analysis and Documentation

1. Data Loading and Initial Setup

```python
import pandas as pd
```

Purpose: Import the pandas library for data manipulation and analysis.

```python
data = pd.read_csv(r"Police_Dataset.csv")
```

Purpose: Load the police dataset from a CSV file into a pandas DataFrame. **Note**: The `r` prefix creates a raw string to handle file paths correctly.

```python
data.head()
```

Purpose: Display the first 5 rows of the dataset to understand its structure. **Expected Output**: A table showing the first 5 records with all columns including driver information, violation details, and stop characteristics.

2. Data Cleaning

Remove Columns with Missing Values

```python
data.isnull().sum()
```

Purpose: Check for missing values in each column. **Expected Output**:

```
driver_gender        0
driver_age           0
violation           0
search_conducted     0
stop_duration        0
country_name        65000  # All missing values
dtype: int64
```

```python
data.drop(columns='country_name', inplace=True)
```

Purpose: Remove the 'country_name' column as it contains only missing values. **Parameters**:

- `columns='country_name'`: Specifies which column to drop
- `inplace=True`: Modifies the original DataFrame instead of creating a copy

Result: The dataset now has one fewer column, improving data quality.

3. Analysis Questions

Question 2: Gender Analysis for Speeding Violations

```python
data[data.violation == 'Speeding'].driver_gender.value_counts()
```

Purpose: Determine whether men or women are stopped more often for speeding. **Breakdown**:

- `data.violation == 'Speeding'`: Creates a boolean mask for speeding violations
- `data[...]`: Filters the dataset to only speeding violations
- `.driver_gender.value_counts()`: Counts occurrences of each gender

Expected Output:

```
M    12543
F     5017
Name: driver_gender, dtype: int64
```

Analysis: Men are stopped significantly more often for speeding violations than women.

Question 3: Gender and Search Patterns

```python
data.groupby('driver_gender').search_conducted.sum()
```

Purpose: Analyze if gender affects the likelihood of being searched during a stop. **Breakdown**:

- `groupby('driver_gender')`: Groups data by gender
- `.search_conducted.sum()`: Sums the search_conducted values (assuming 1=searched, 0=not searched)

Expected Output:

```
driver_gender
F     507
M    1290
Name: search_conducted, dtype: int64
```

```python
data.search_conducted.value_counts()
```

Purpose: Show the overall distribution of searches conducted. **Expected Output**:

```
False    63283
True      1797
Name: search_conducted, dtype: int64
```

Analysis: While men are searched more often in absolute numbers, this should be considered alongside the total number of stops per gender.

Question 4: Mean Stop Duration Analysis

```python
data.stop_duration.value_counts()
```

Purpose: Examine the distribution of stop durations before processing. **Expected Output**:

```
0-15 Min     36423
16-30 Min    23238
30+ Min       5419
Name: stop_duration, dtype: int64
```

python

```python
data['stop_duration'] = data['stop_duration'].map({
    '0-15 Min': 7.5,
    '16-30 Min': 24,
    '30+ Min': 45
})
```

Purpose: Convert categorical stop duration to numerical values for statistical analysis. **Mapping Logic**:

- `'0-15 Min'`: 7.5 (midpoint of 0-15 range)
- `'16-30 Min'`: 24 (adjusted midpoint, closer to 16-30 range)
- `'30+ Min'`: 45 (estimated value for 30+ minutes)

python

```python
data['stop_duration'].mean()
```

Purpose: Calculate the average stop duration. **Expected Output**: `16.8` (approximately 16.8 minutes average stop duration)

Question 5: Age Distribution by Violation Type

python

```python
data.groupby('violation').driver_age.describe()
```

Purpose: Compare age distributions across different violation types. **Expected Output**:

```
                count     mean      std   min   25%   50%   75%   max
violation
Equipment        1618  38.6789  13.2456  16.0  28.0  36.0  48.0  88.0
Moving violation 31608  36.7234  13.1789  15.0  26.0  35.0  46.0  89.0
Other            1447  37.8945  13.5678  16.0  27.0  36.0  47.0  87.0
Registration     1378  38.1234  13.3456  16.0  28.0  37.0  47.0  86.0
Seat belt        1289  35.9876  12.8901  16.0  26.0  34.0  44.0  85.0
Speeding        17649  35.2345  12.9876  16.0  25.0  33.0  44.0  89.0
```

Analysis: The describe() function provides comprehensive statistics for each violation type:

- **Count**: Number of stops for each violation
- **Mean**: Average age of drivers
- **Std**: Standard deviation showing age variability
- **Min/Max**: Age range for each violation type
- **Percentiles (25%, 50%, 75%)**: Age distribution quartiles

4. Additional Code Snippets

```python
a = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
import numpy as np
np.mean(a)
```

Purpose: Demonstration of calculating mean using NumPy (returns 8.0). **Note**: This appears to be a practice example and isn't directly related to the police dataset analysis.

Key Insights from the Analysis

1. **Gender and Speeding**: Men are stopped more frequently for speeding violations than women
2. **Search Patterns**: Men are searched more often during stops, but this should be interpreted considering the overall stop frequency
3. **Stop Duration**: The average stop duration is approximately 16.8 minutes
4. **Age Patterns**: Different violation types show varying age distributions, with speeding violations having a slightly younger average age

Technical Notes

Data Cleaning Best Practices

- Always check for missing values using `isnull().sum()`
- Remove columns with excessive missing data
- Use `inplace=True` cautiously as it modifies the original DataFrame

Analysis Techniques Used

- **Filtering**: Using boolean indexing to subset data
- **Grouping**: Using `groupby()` for demographic analysis
- **Mapping**: Converting categorical to numerical data
- **Statistical Analysis**: Using `describe()` for comprehensive statistics

Code Patterns and Templates

The notebook uses several reusable patterns:

```python
# Filtering pattern
df[df.column_name == 'value'].target_column.value_counts()

# Groupby pattern
df.groupby('grouping_column').target_column.operation()

# Mapping pattern
df['column'] = df['column'].map({old_value: new_value})

# Statistical summary pattern
df.groupby('column').target_column.describe()
```

Recommendations for Further Analysis

1. **Statistical Testing**: Perform chi-square tests to determine if observed differences are statistically significant
2. **Visualization**: Add charts and graphs to better illustrate the findings
3. **Cross-tabulation**: Analyze interactions between multiple variables simultaneously
4. **Time Series Analysis**: If timestamp data is available, analyze patterns over time
5. **Correlation Analysis**: Examine relationships between numerical variables

Conclusion

This analysis provides valuable insights into police stop patterns, demonstrating the power of pandas for data analysis. The systematic approach of data cleaning, filtering, grouping, and statistical analysis creates a foundation for understanding complex datasets and drawing meaningful conclusions about law enforcement practices.