
POLICE

Predicting Traffic Violation Penalty

By Lavanya Krishna and Dylan Tallis
PD: 5

01

Goal & Data

Project Goal

- Model can aid in training police officers
- Identify factors contributing to more severe penalties
- Reduce human bias using objective rather than subjective characteristics
  - Time and location
  - Age and race

Description of Data

Traffic violations in Montgomery County, Maryland

2014 - present, updated daily

43 attributes

299,777 missing values

Instances

All Data

1.96 million

2024 Data

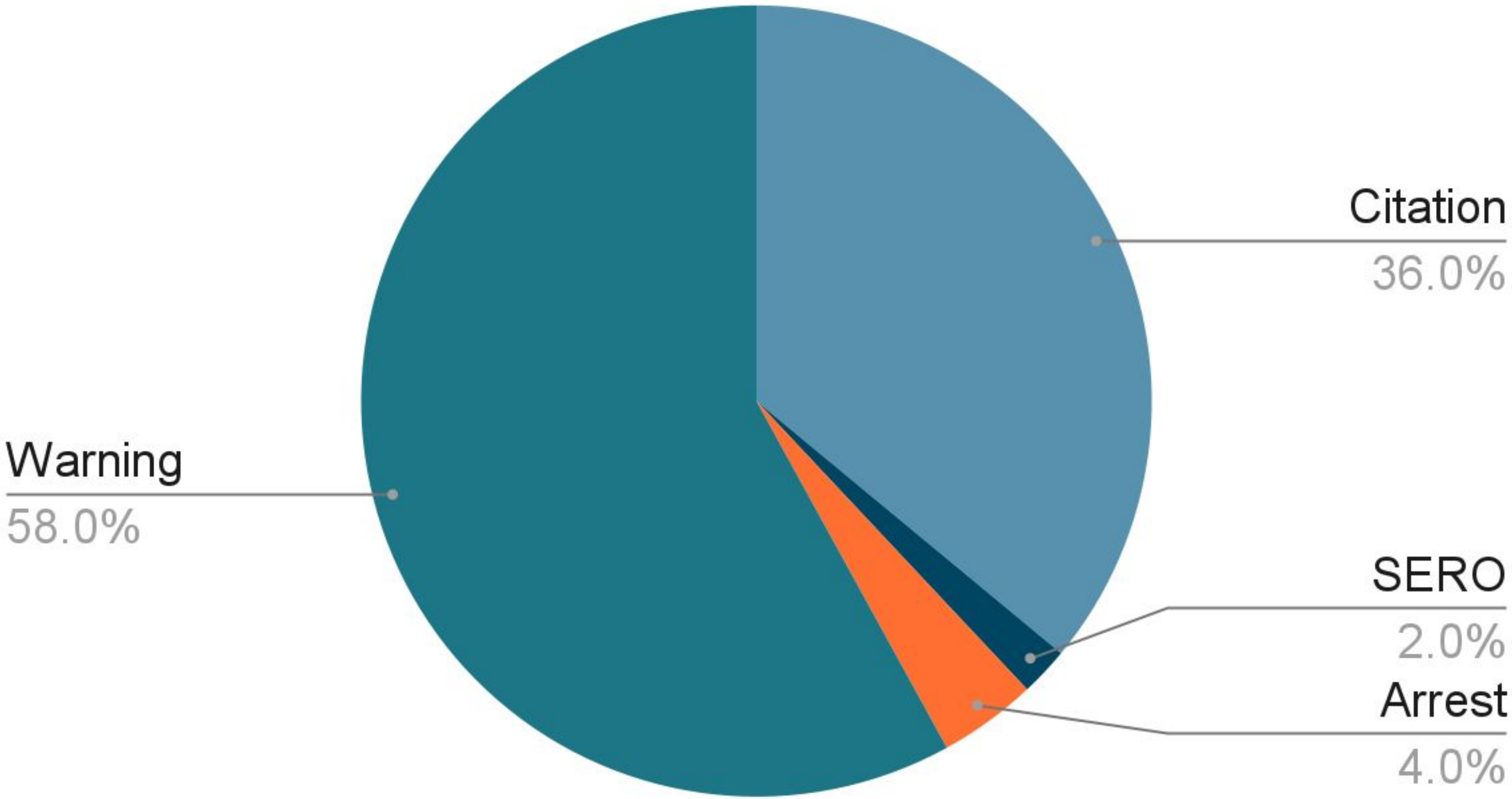
58,679

Description of Data

Class: "Search Outcome"

- Warning
- Citation
- Safety Equipment Repair Orders (SERO)
- Arrest

Distribution of Class Values



02
Tools & Preprocessing

Tools Used

Montgomery Data: 2024 data

Visual Studio Code: Preprocessing.

Google Colab: Stratified samples and splitting datasets

Notepad: Change attribute data types

Weka: Attribute selection and classifier model testing

Google Sheets: Graphing results

Initial Data Cleaning

Removed:

- Apostrophes, double quotes, and return characters
- 4 attributes with over 97% missing values
    - Search Disposition, Reason, Type, and Arrest Reason
- Instances with missing class value
    - 40% of dataset
    - 35,208 instances

Random Sampling

Stratified Random Sampling: 1000 instances

Removed:

- Commercial Vehicle, Agency, Fatal, HAZMAT, Alcohol, and Work Zone: one value
- Article: bias
- Geolocation: derivable
- Violation Type: same as class
- SeqID: index

Cleaning and Splits

Change attribute type from string to nominal

Change attribute type to date

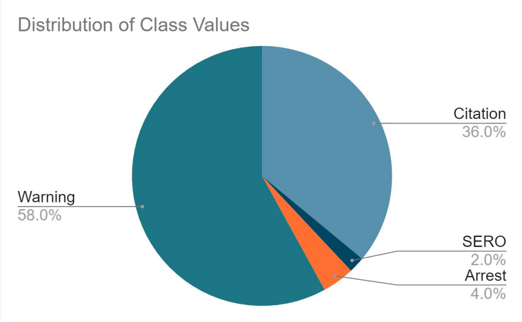
- Save as .arff

Training, Validation, and Testing

- train_test_split
- Did not use

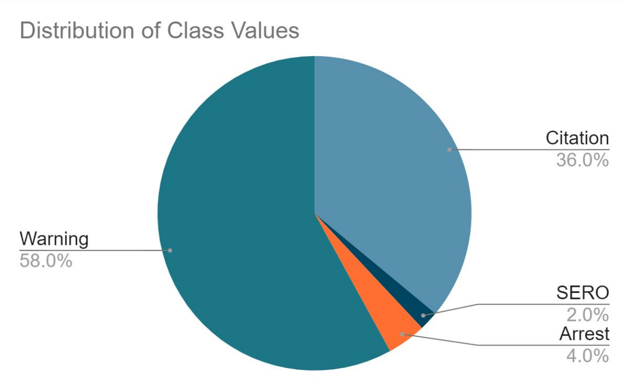
```python
import pandas as pd

df = pd.read_csv("/content/drive/MyDrive/ML_Project/traffic_data.csv")
df.head(2)

from google.colab import drive
drive.mount('/content/drive')

condition_a = df['Search Outcome'] == "Warning"
strata_a = df[condition_a]

condition_b = df['Search Outcome'] == "Citation"
strata_b = df[condition_b]

condition_c = df['Search Outcome'] == "Arrest"
strata_c = df[condition_c]

condition_d = df['Search Outcome'] == "SERO"
strata_d = df[condition_d]

strata_a_sample = strata_a.sample(n = 580, random_state = 0)

strata_b_sample = strata_b.sample(n = 359, random_state = 0)
```

```python
strata_c_sample = strata_c.sample(n = 43, random_state = 0)

strata_d_sample = strata_d.sample(n = 18, random_state = 0)

stratified_sample_df = pd.DataFrame()

stratified_sample_df = pd.concat([strata_a_sample, strata_b_sample, strata_c_sample,
strata_d_sample])

print(stratified_sample_df.head())

from google.colab import files
files.download('stratified_sample.csv')
```



```python
import pandas as pd

df =
pd.read_csv("/content/drive/MyDrive/machine_learning/d_stratified_sample.c
sv")

from sklearn.model_selection import train_test_split

cols = df.columns.tolist()
cols = cols[0:21] + cols[22:] + [cols[21]]
df = df[cols]
df.head(1)

X = df.iloc[:, 0:-1]
y = df.iloc[:, -1]
X.head(1)

y.head(1)

X_train, X_val_test, y_train, y_val_test = train_test_split(X,y,
test_size=0.3, random_state=0, stratify=y)
X_val, X_test, y_val, y_test = train_test_split(X_val_test, y_val_test,
test_size=0.5, random_state=0, stratify=y_val_test)

X_train['index'] = X_train.index
y_train = y_train.reset_index()

train = pd.merge(X_train, y_train, on = 'index')

train.to_csv('train.csv',index=False)

X_val['index'] = X_val.index
y_val = y_val.reset_index()
```

```python
validation = pd.merge(X_val, y_val, on = 'index')

validation.to_csv('validation.csv',index=False)

X_test['index'] = X_test.index
y_test = y_test.reset_index()

test = pd.merge(X_test, y_test, on = 'index')

test.to_csv('test.csv',index=False)
```



03

Attribute Selection

CorrelationAttributeEval

Pearson's correlation between attribute and class

Cutoff value: 0.1

Selected attributes:

- Search Conducted
- Accident
- Contributed To Accident
- Property Damage
- Gender

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 (y_i - \bar{y})^2}}$$

```
=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 28 Search Outcome):
        Correlation Ranking Filter
Ranked attributes:
 0.21601    12 Search Conducted
 0.15221     7 Accident
 0.15221    21 Contributed To Accident
 0.13588    10 Property Damage
 0.11972    23 Gender
 0.09767     9 Personal Injury
```

CfsSubsetEval

Individual predictive ability and redundancy

Selected attributes:

- Location
- Personal Injury
- Property Damage
- Search Conducted

```
=== Attribute Selection on all input data ===

Search Method:
        Greedy Stepwise (forwards).
        Start set: no attributes
        Merit of best subset found:    0.334

Attribute Subset Evaluator (supervised, Class (nominal): 28 Search Outcome):
        CFS Subset Evaluator
        Including locally predictive attributes

Selected attributes: 4,9,10,12 : 4
                     Location
                     Personal Injury
                     Property Damage
                     Search Conducted
```

InfoGainAttributeEval

Information gain with respect to class, entropy

Cutoff value: 0.2

Selected attributes:

- Location
- Model
- Charge
- Search Reason For Stop
- Search Conducted
- Driver City

```
=== Attribute Selection on all input data ===

Search Method:
	Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 28 Search Outcome):
	Information Gain Ranking Filter

Ranked attributes:
 1.207508    4 Location
 0.525144   18 Model
 0.509508   20 Charge
 0.427613   13 Search Reason For Stop
 0.213413   12 Search Conducted
 0.208574   24 Driver City
 0.199371   17 Make
```

GainRatioAttributeEval

Gain ratio in relation to class

Cutoff value: 0.1

Selected attributes:

- Search Conducted
- Personal Injury
- Property Damage
- Accident
- Contributed To Accident
- Location

```
=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 28 Search Outcome):
        Gain Ratio feature evaluator

Ranked attributes:
 0.68543    12 Search Conducted
 0.15778     9 Personal Injury
 0.15359    10 Property Damage
 0.12621     7 Accident
 0.12621    21 Contributed To Accident
 0.12446     4 Location
 0.0867     20 Charge
```

OneRAttributeEval

Set of rules for one attribute with lowest error rate

Cutoff value: 60

Selected attributes:

- Charge
- Search Reason For Stop
- Search Conducted
- Race

```
=== Attribute Selection on all input data ===

Search Method:
	Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 28 Search Outcome):
	OneR feature evaluator.

	Using 10 fold cross validation for evaluating attributes.
	Minimum bucket size for OneR: 6

Ranked attributes:
67                     20 Charge
64.7                   13 Search Reason For Stop
61.7                   12 Search Conducted
60                     22 Race
59.9                   21 Contributed To Accident
```

04
Classifier Models

Classifier Models

Naive Bayesian

- Based on Bayes' Theorem
- Assumes features are independent
- Predicts the class with the highest probability
- Simple yet effective for supervised learning
- Accessed via:
    - bayes → NaiveBayes

OneR Classifier

- Creates simple rules based on a single feature
- Selects the feature with the lowest error rate to form a rule
- High accuracy in some cases despite simplicity
- Accessed via:
    - rules → OneR

Classifier Models

J48 Classifier

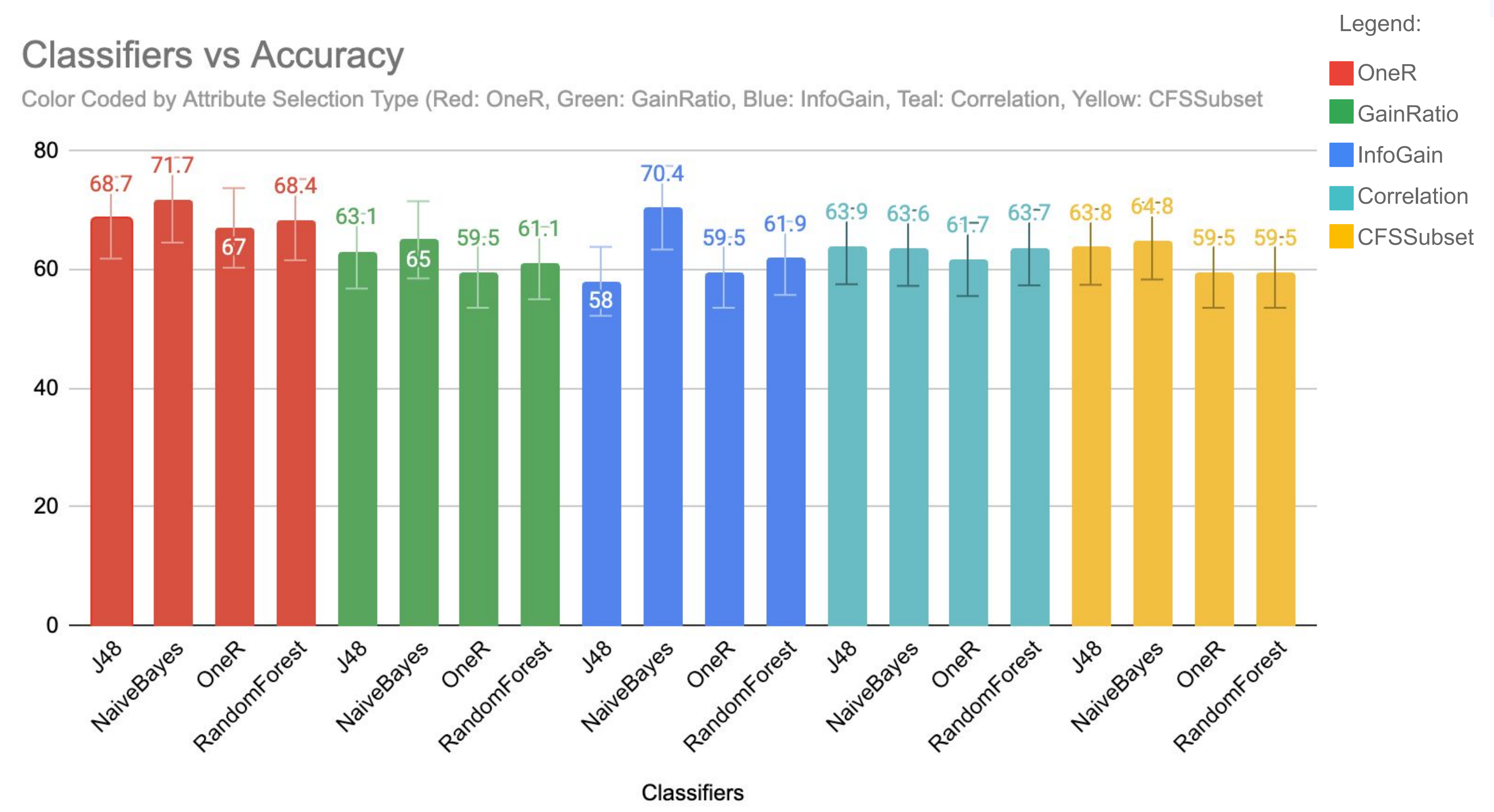
- Builds a decision tree by splitting data based on best attribute
- Minimizes classification error
- Decision tree used to predict the class of new instances
- Balances simplicity and accuracy
- Accessed via:
    - trees → J48

Random Forest Classifier

- Builds a forest of random trees, each making independent predictions
- Final prediction is based on majority voting across all trees
- Trees are equally weighted
- Accessed via:
    - tree → RandomForest

05

Results



Classifiers vs Accuracy
Color Coded by Attribute Selection Type (Red: OneR, Green: GainRatio, Blue: InfoGain, Teal: Correlation, Yellow: CFSSubset
Legend:
OneR
GainRatio
InfoGain
Correlation
CFSSubset
80
60
40
20
0
68.7
71.7
67
68.4
63.1
65
59.5
61.1
58
70.4
59.5
61.9
63.9
63.6
61.7
63.7
63.8
64.8
59.5
59.5
J48
NaiveBayes
OneR
RandomForest
J48
NaiveBayes
OneR
RandomForest
J48
NaiveBayes
OneR
RandomForest
J48
NaiveBayes
OneR
RandomForest
J48
NaiveBayes
OneR
RandomForest
Classifiers

Results

Highest Accuracy: OneR with Naive Bayes at 71.1%

Second Highest: Info Gain with Naive Bayes at 70.4%

Low error rates

- Mean absolute, root mean squared, relative absolute, root relative squared

High precision and F-measure

High accuracy and low error are indicators of consistent performance

06

Conclusions

Common Attributes

Naive Bayes OneR and Info Gain shared attributes:

- Charge
- Search Reason For Stop
- Search Conducted

Charge is the numeric code for the specific charge

Class Imbalance

Many models classified the majority as "Warning"

- Confusion matrices show zeros in all columns except for "Warning"
- Many True Positives and False Positives

Class imbalance: 58% of instances classified as "Warning"

Models perform well on majority class but poorly on minority classes

Class Imbalance - Improvements

- Oversample minority classes
- Undersample majority class
- Assign class weights to emphasize minority class misclassification

Human Bias

Each violation penalty is given in a unique situation

Variations in penalty due to officer's…

- Emotions
- Fatigue
- Level of experience
- Personal bias towards driver

Human Bias - Improvements

Create a dataset with a minimum level of experience

- Only consider traffic violation data that was reported by officers who have achieved that level

Thank you!
Questions?