**Project Report: Predicting Traffic Violation Penalty**

Lavanya Krishna and Dylan Tallis

Machine Learning 1

Dr. Selma Yilmaz

October 23, 2024

**Table of Contents**

1. [Project Goal](#tw2zclrhvlao)
2. [Description of Dataset](#v4wxfblgmkyi)
3. [Tools](#zh40ynr9fd52)
4. [Preprocessing](#vm9j97jdhffu) 
   1. Initial Data Cleaning
   2. Missing Values
   3. Distributions and Sampling
   4. Additional Data Cleaning
   5. Train - Validation - Test Split
5. [Description of Attribute Selection](#1n2f734pctu7) Process
6. [Description of Classifier Models](#myhk5ovmhnf2)
7. [Result and Evaluation](#999nyg46jii5)
   1. Confusion Matrices
   2. Justification of Selection
8. [Discussions and Conclusions](#gyzh9cmcobif)
9. [Team Members and Tasks Performed](#a8aiqwmnd23r)
10. [Appendix](#3srnkrvag8mt)

**Project** **Goal**

The county of Montgomery, Maryland keeps a daily updated record of all issued electronic traffic violations within the county. The dataset includes information about the nature of the violation, the description of the car and driver, the time and location, and the classification of the penalty. The penalties include: warnings, citations, safety equipment repair orders (SERO), and arrests.

Building a model to predict traffic violation penalties can identify the factors that contribute to violations that end in more significant penalties. Additionally, predicting violations based on objective factors, as opposed to personal factors like race or gender, can help reduce human bias in penalty assignment, making law enforcement practices more fair and transparent. Lastly, the model could be used to train new police officers by giving them the information about a traffic violation and comparing the violation type they would assign with what their coworkers typically did.

**Description of Dataset**

This dataset contains traffic violation information from all of the electronic traffic violations issued in Montgomery County, Maryland. The dataset was created in June of 2014 and since then has been updated daily. At the time of our download, the dataset had 1.96 million instances and 43 attributes. For the sake of relevancy, we decided to only consider the data from 2024, which reduced the number of instances. Our dataset now had 58,679 instances and a dimension of 42 with one class variable. That dataset contained 299,777 missing values. We chose Search Outcome as our class, with values that align with the traffic violation penalties.

**Tools**

The tools we used to assist us in this project included Visual Studio Code, Google Colaboratory, Notepad, and Weka. Visual Studio Code, or VS Code, is an integrated development environment that we used during preprocessing. We used the browser-based Python coding environment Google Colaboratory’s (Colab) Jupyter Notebooks to create a stratified sample and split the sample into training and testing datasets. We used the Notepad text editor to change attribute data types for easier handling. Lastly, we used the Weka data platform to perform attribute selection algorithms and create and test our classification models.

**Preprocessing**

Initial Data Cleaning

We used VS Code’s replace all tool to remove all apostrophes and double quotes from our dataset that prevented us from opening the data in Weka. We also removed all return characters placed in the middle of some instances. Instead of scrolling through and checking each of the 58,679 instances, we found it more efficient to open the file in Weka, as it would throw an error and specify the line, which was the line containing the return.

Missing Values

We found that four attributes, Search Disposition, Search Reason, Search Type, and Search Arrest Reason, had 97% of their values missing. This percentage exceeded our cutoff of 70% so we removed the four attributes. The attribute Commercial Vehicle’s only value was “No” so we removed it as well. We found that the majority of the rest of the missing values corresponded to instances that did not have a class value, and the attributes that contained the missing values were all related to a search. Our class describes the penalty outcome of the violation after conducting a search, so if a search was not conducted, there was no data for any of the search attributes. By choosing our class attribute as Search Outcome, we removed the 40% of the dataset that had missing class values. In doing this, we reduced the number of instances in our dataset to 35,208.

Distributions and Sampling

After dealing with missing values, our dataset had the following class distributions.

Class Distributions of Dataset

| **Class Label** | **Number of Instances** | **Proportion of Dataset** (rounded) |
| --- | --- | --- |
| Warning | 20,434 | 58% |
| Citation | 12,620 | 36% |
| Arrest | 1,518 | 4% |
| SERO | 636 | 2% |

For easier handling, we used stratified random sampling in Colab to reduce the number of instances in our dataset to 1,000 with the following class distributions. The code can be found in the [Appendix](#3srnkrvag8mt).

Class Distributions of Stratified Sample

| **Class Label** | **Number of Instances** | **Proportion of Dataset** (rounded) |
| --- | --- | --- |
| Warning | 580 | 58% |
| Citation | 359 | 36% |
| Arrest | 43 | 4% |
| SERO | 18 | 2% |

Additional Data Cleaning

After creating our new dataset using stratified random sampling, we still had some missing values. The majority of the remaining missing values were in the Article attribute. We decided to delete the attribute because all of its missing values corresponded with SERO penalties and we could therefore not properly fill them without adding bias to the dataset and model. We discovered two disguised missing values, one in Make and one in Year, which we filled with the modes.

We removed the attributes Agency, Fatal, HAZMAT, Alcohol, and Work Zone because they all only had one value. We deleted the Geolocation attribute, as it was derivable data from the Longitude and Latitude attributes, and the Violation Type attribute as it was the same data as the Search Outcome. We also removed the SeqID attribute which contained a unique ID for each traffic violation.

We found that the attribute Model was saved as a string type so we used the filter “StringToNominal” to ensure that we could use the attribute selection algorithms. While doing so we found that the change would only save if you saved the dataset as an .arff and not as a .csv. We also took the advice from the Weka FAQs and changed the data type of the Date Of Stop and Time Of Stop attributes from nominal to date using Notepad.

Train - Validation - Test Split

To split the dataset into training, validation, and testing datasets, we used Scikit-learn’s train\_test\_split. We used a stratified 70-15-15 split, with 700 instances in the training dataset and 150 instances in the validation and testing datasets each. To create the datasets we added a new attribute, index, which we removed after. Each dataset is representative of the original population. The code can be found in the [Appendix](#3srnkrvag8mt).

Class Distributions of **Training** Dataset

| **Class Label** | **Number of Instances** | **Proportion of Dataset** (rounded) |
| --- | --- | --- |
| Warning | 406 | 58% |
| Citation | 251 | 36% |
| Arrest | 30 | 4% |
| SERO | 13 | 2% |

Class Distributions of **Validation** Dataset

| **Class Label** | **Number of Instances** | **Proportion of Dataset** |
| --- | --- | --- |
| Warning | 87 | 58% |
| Citation | 54 | 36% |
| Arrest | 6 | 4% |
| SERO | 3 | 2% |

Class Distributions of **Testing** Dataset

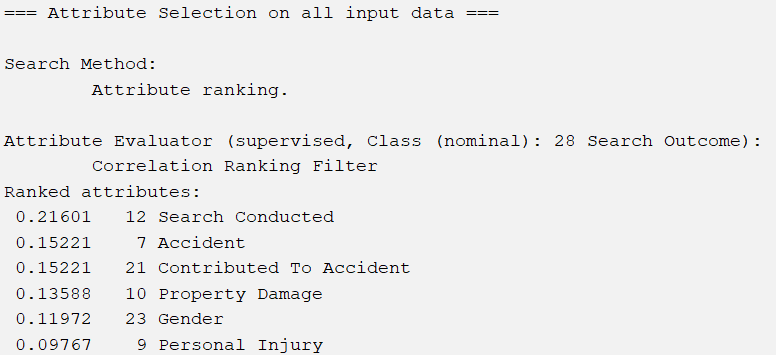
| **Class Label** | **Number of Instances** | **Proportion of Dataset** (rounded) |
| --- | --- | --- |
| Warning | 87 | 58% |
| Citation | 54 | 36% |
| Arrest | 7 | 5% |
| SERO | 2 | 1% |

**Description of Attribute Selection Process**

We used Weka for all approaches.

Pearson’s Correlation Evaluation

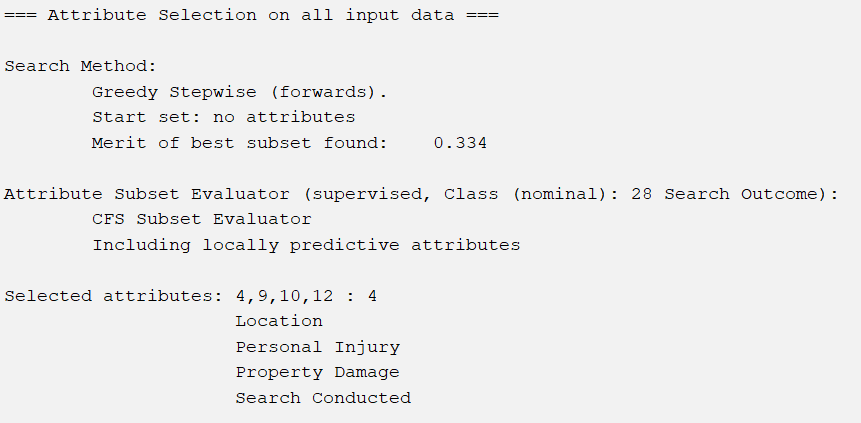
This approach evaluates the worth of each attribute by calculating the Pearson’s correlation between the attribute and the class. We set the cutoff value at 0.1.



The attributes chosen are Search Conducted, Accident, Contributed To Accident, Property Damage, and Gender.

CFS Subset Evaluation

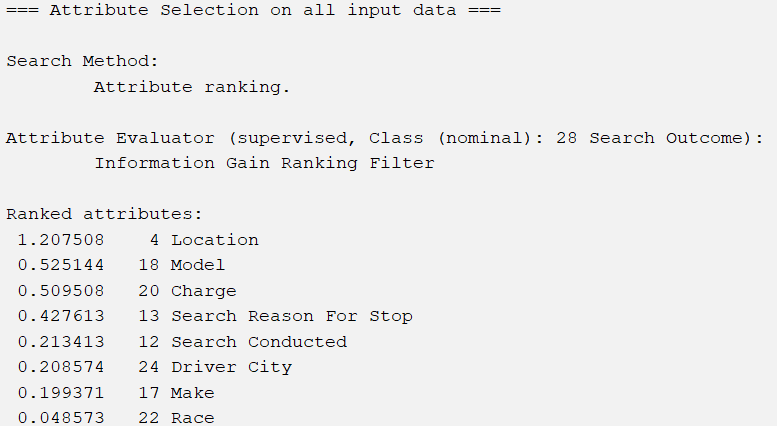
This approach evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the level of redundancy between them.



The attributes chosen are Location, Personal Injury, Property Damage, and Search Conducted

Information Gain Evaluation

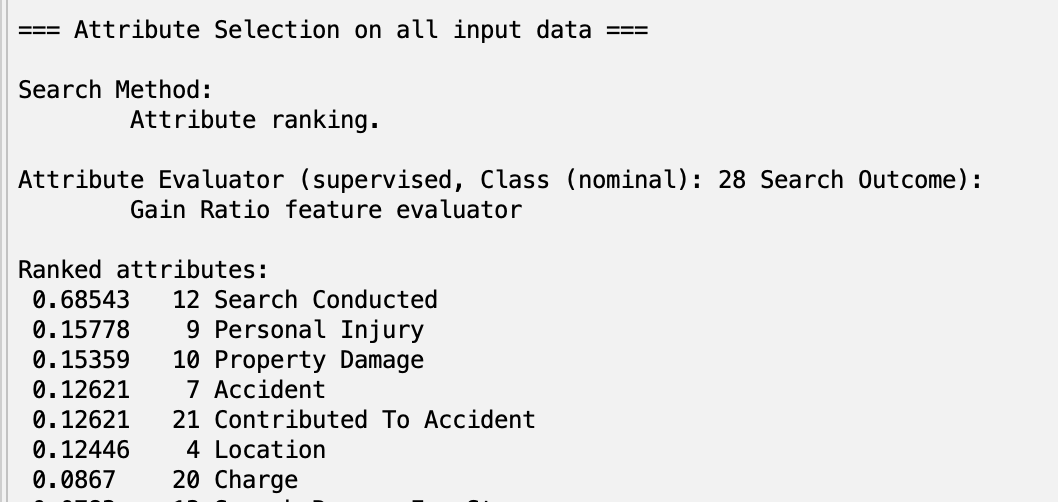
This approach evaluated the worth of an attribute by calculating the information gain with respect to the class. We set the cutoff value at 0.2.



The attributes chosen are Location, Model, Charge, Search Reason For Stop, Search Conducted, and Driver City.

Gain Ratio Evaluation:

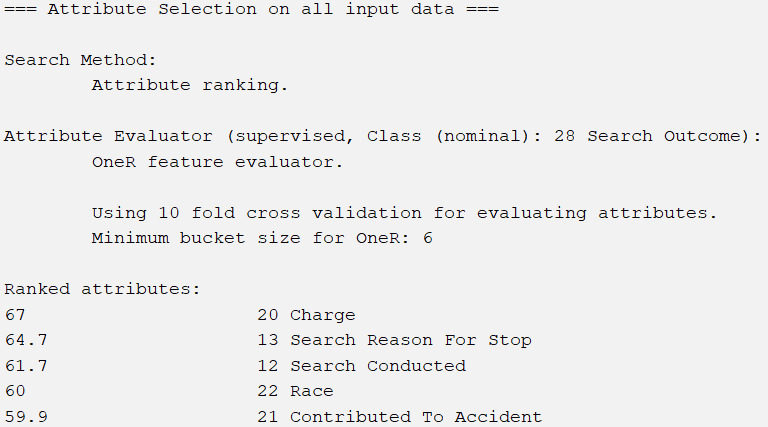
This approach assesses the value of an attribute by calculating the gain ratio in relation to the class. For this we set the cut off value to 0.1



The attributes kept from Gain Ratio Evaluation are Search Conducted, Personal Injury, Property Damage, Accident, Contributed To Accident, Location.

OneR Evaluation:

This approach evaluates the worth of each attribute by using the OneR algorithm. We set the cutoff value at 60.



The chosen attributes are Charge, Search Reason For Stop, Search Conducted, and Race.

**Description of Classifier Models**

For each dataset made from the attribute selection algorithms, we used Naive Bayesian, OneR, J48, and Random Forest classifiers.

Naive Bayesian

The Naive Bayesian Classifier uses Bayes’ Theorem to calculate the probability of each class based on input features, assuming the features are independent. It predicts the class with the highest probability, making it simple yet effective for supervised learning cases.

Accessed through bayes → NaiveBayes

OneR

The OneR classifier creates simple rules based on a single feature to make predictions. It evaluates each feature individually and selects the one with the lowest error rate to form a rule from. Despite its simplicity, it has high accuracy with certain cases.

Accessed through rules → OneR

J48

The J48 classifier builds a decision tree by recursively splitting the data based on the attribute that best separates the classes. It evaluates each feature to create branches, with the goal of minimizing classification error. The final decision tree is used to predict the class of new instances, offering a balance between simplicity and accuracy.

Accessed through trees → J48

Random Forest

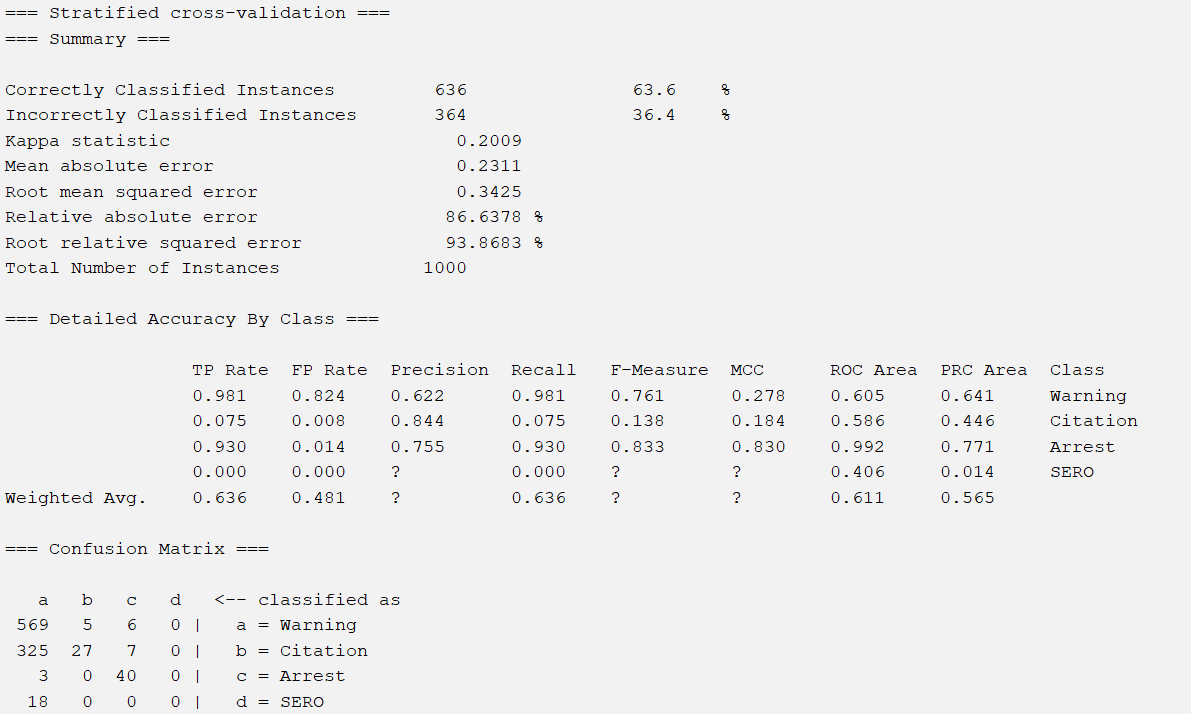
The Random Forest Classification method uses the Weka Classifier Rules to build a forest of random trees, where each tree independently makes a class prediction. The final prediction is based on the class that receives the most votes across all trees, following a simple majority rule since the trees are equally weighted.

Accessed through tree → RandomForest

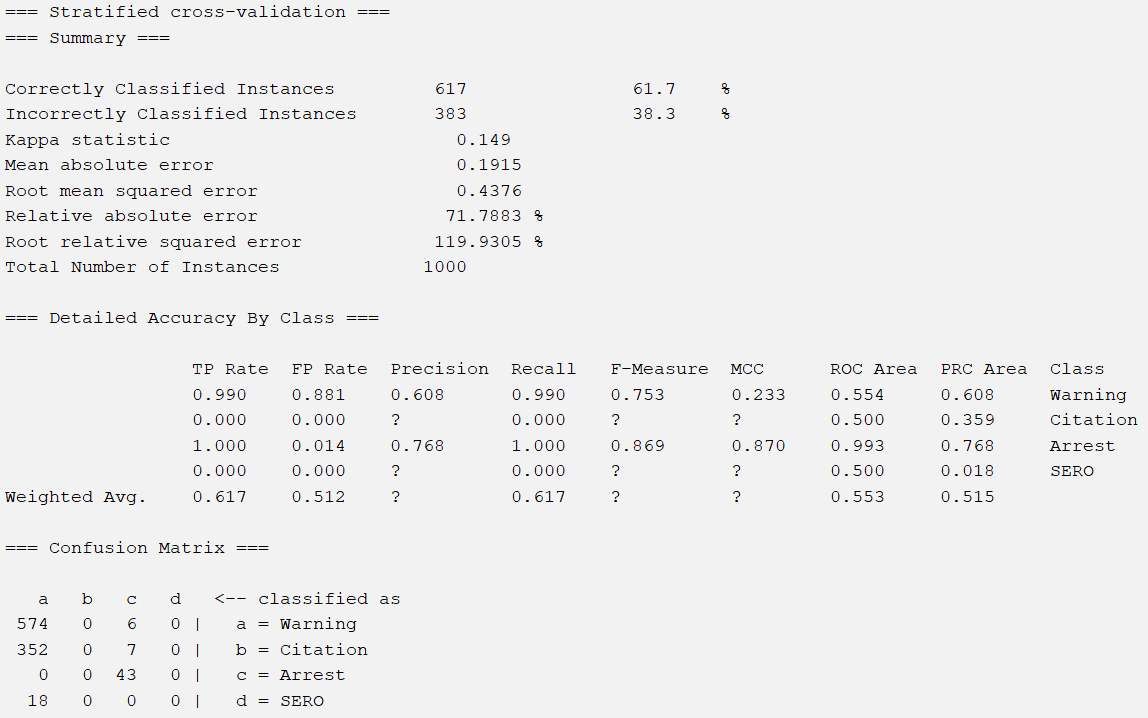
**Result and Evaluation**

Although we created training, validation, and testing datasets, we realized that Weka does not use validation datasets so we decided to use the cross-validation test option with 10 folds.

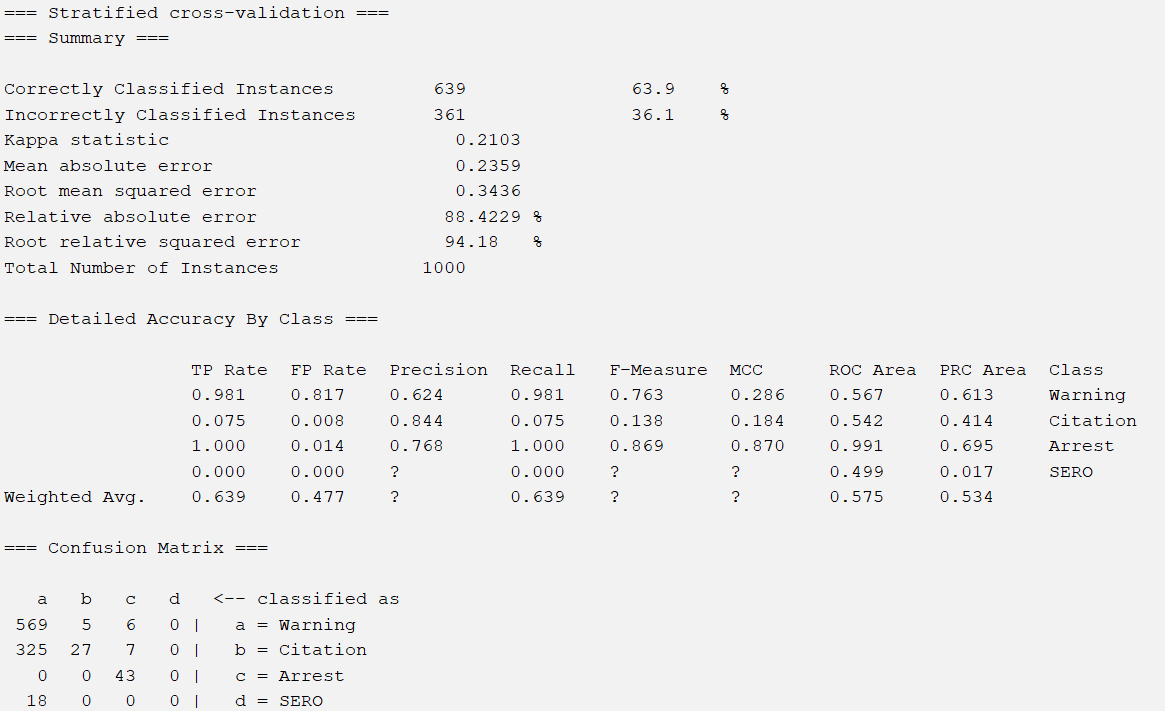
Correlation with Naive Bayes



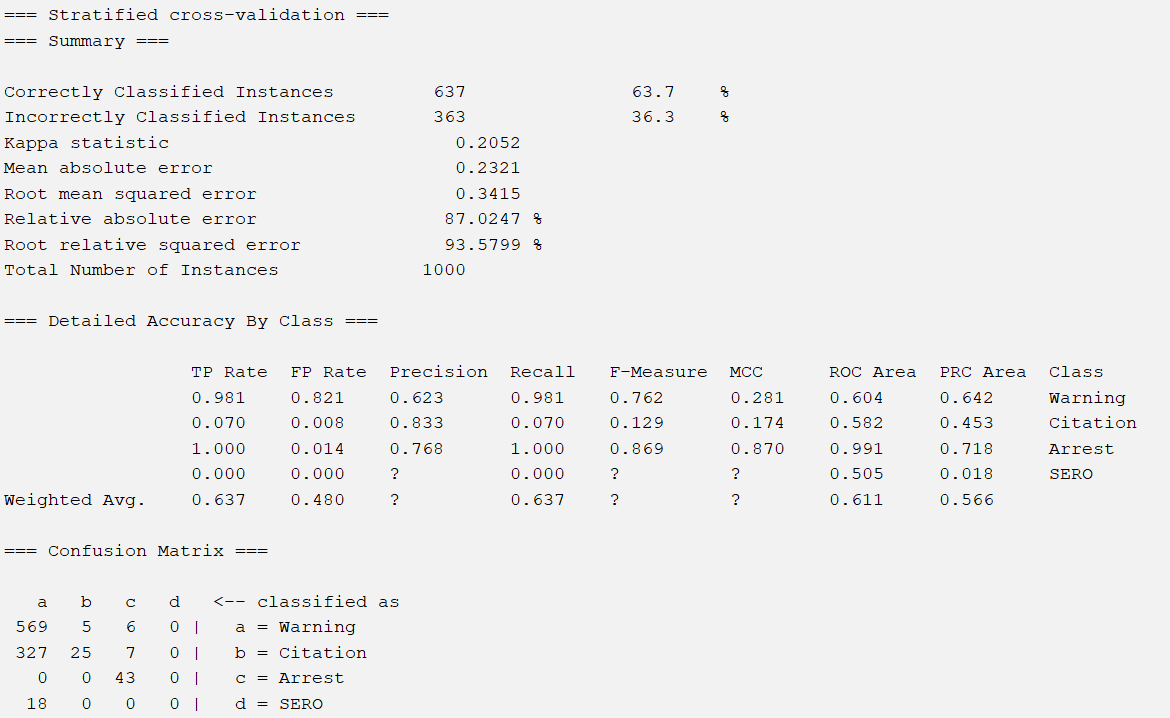
Correlation with OneR



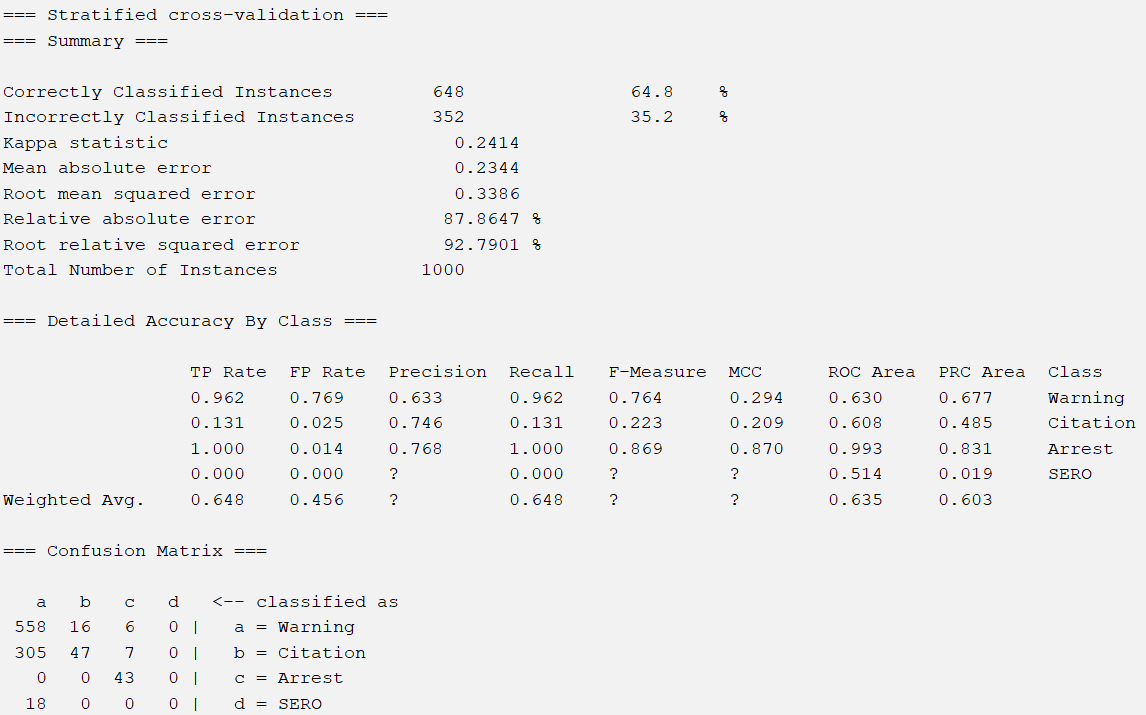
Correlation with J48



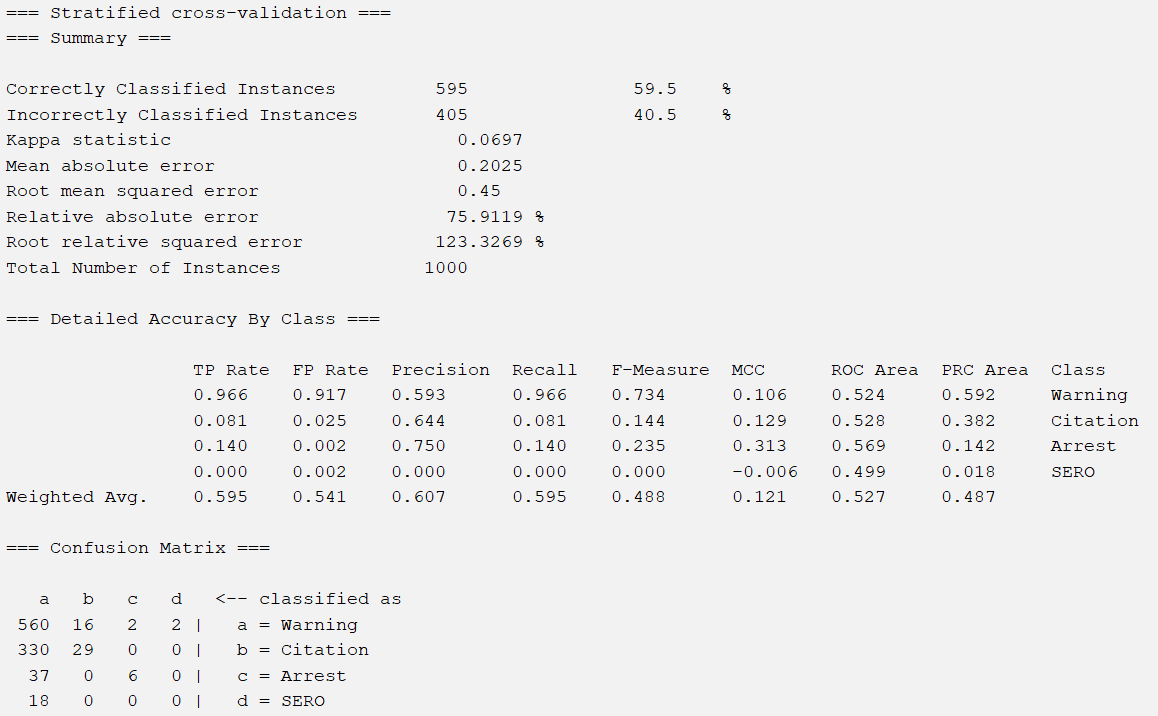
Correlation with Random Forest



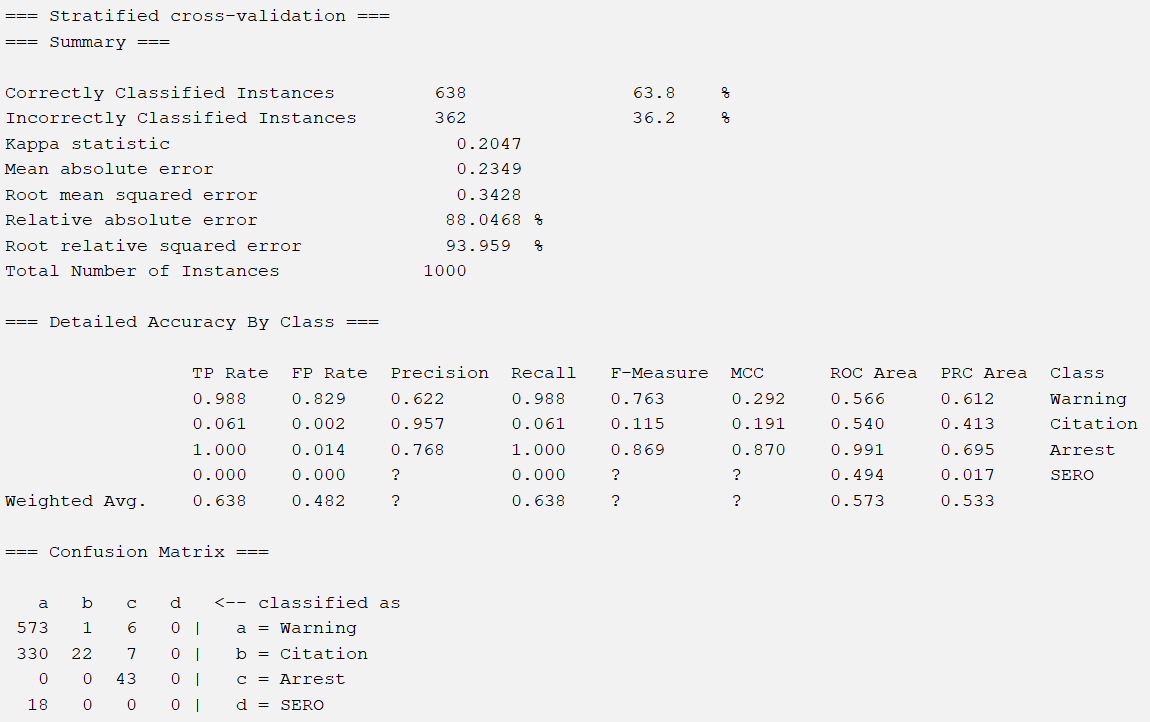
CFS Subset with Naive Bayes



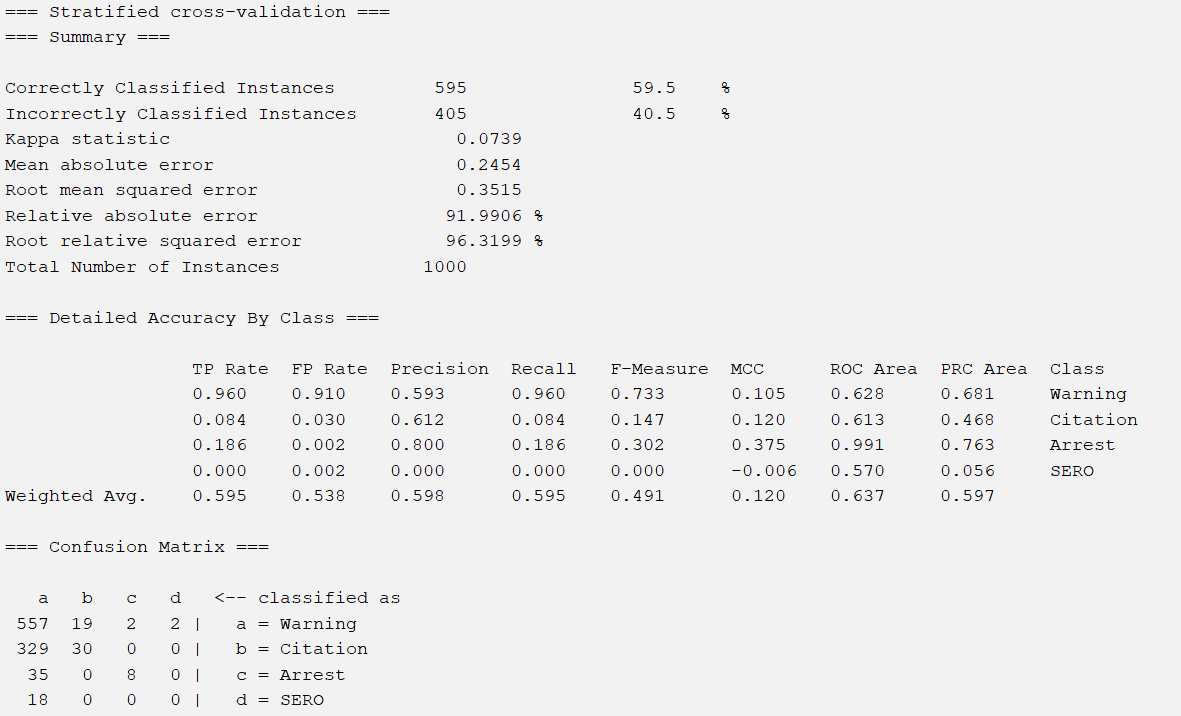
CFS Subset with OneR



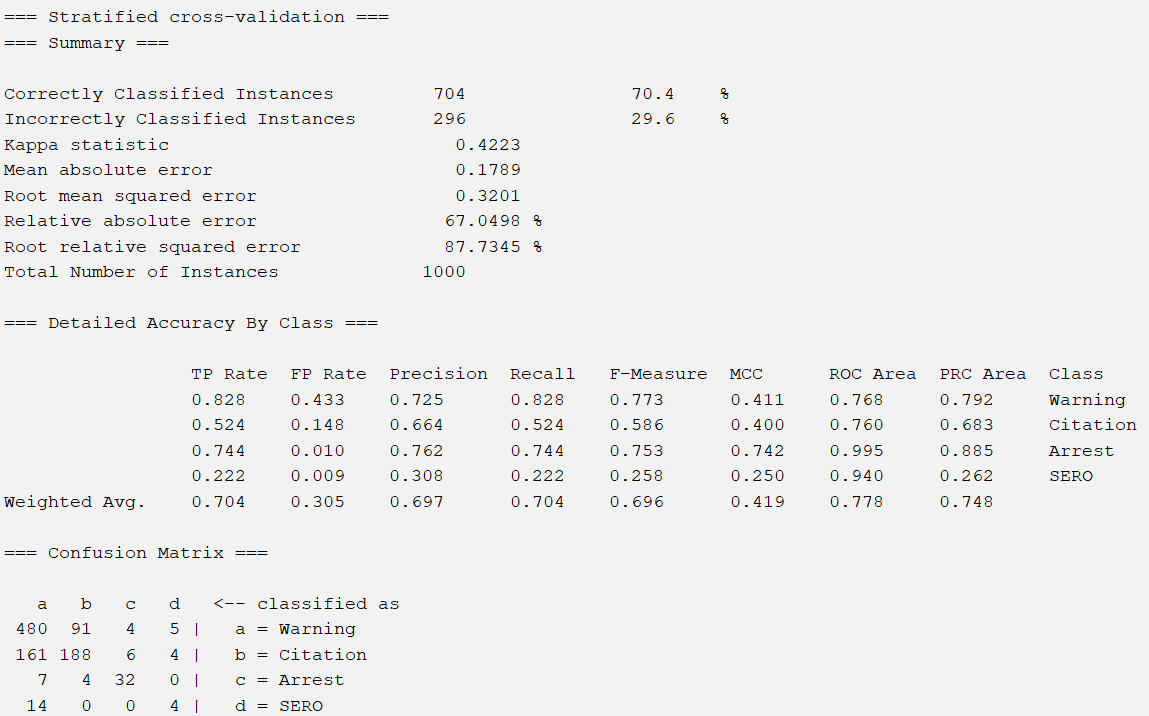
CFS Subset with J48



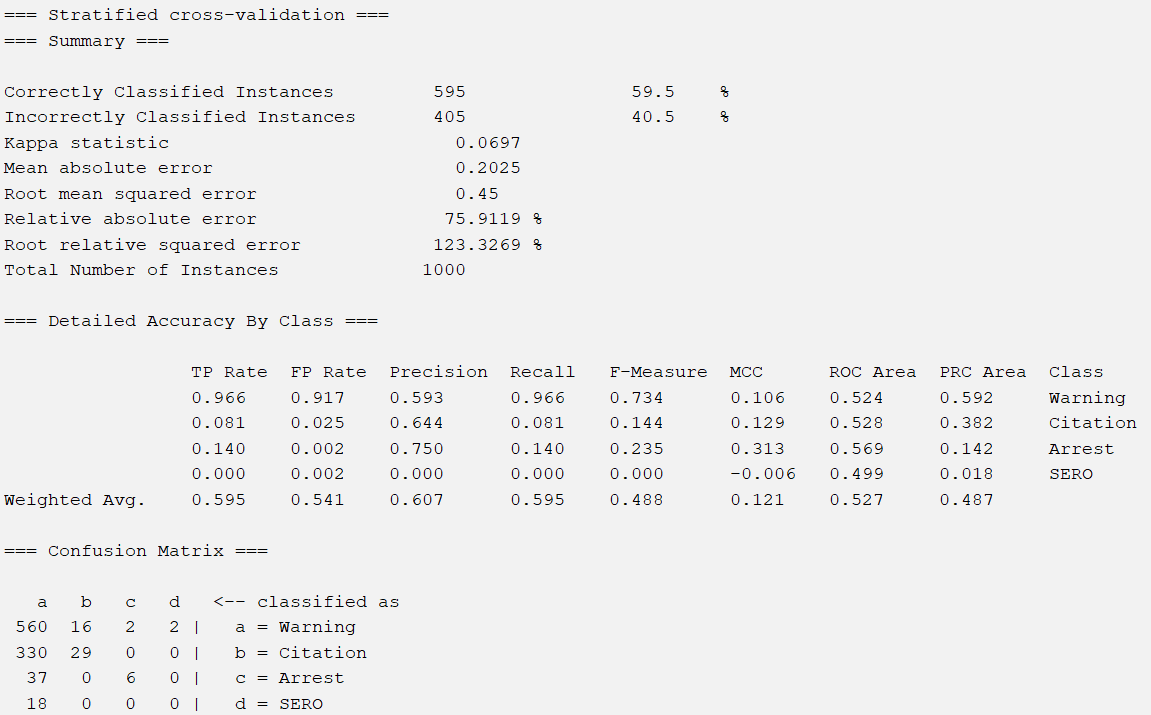
CFS Subset with Random Forest



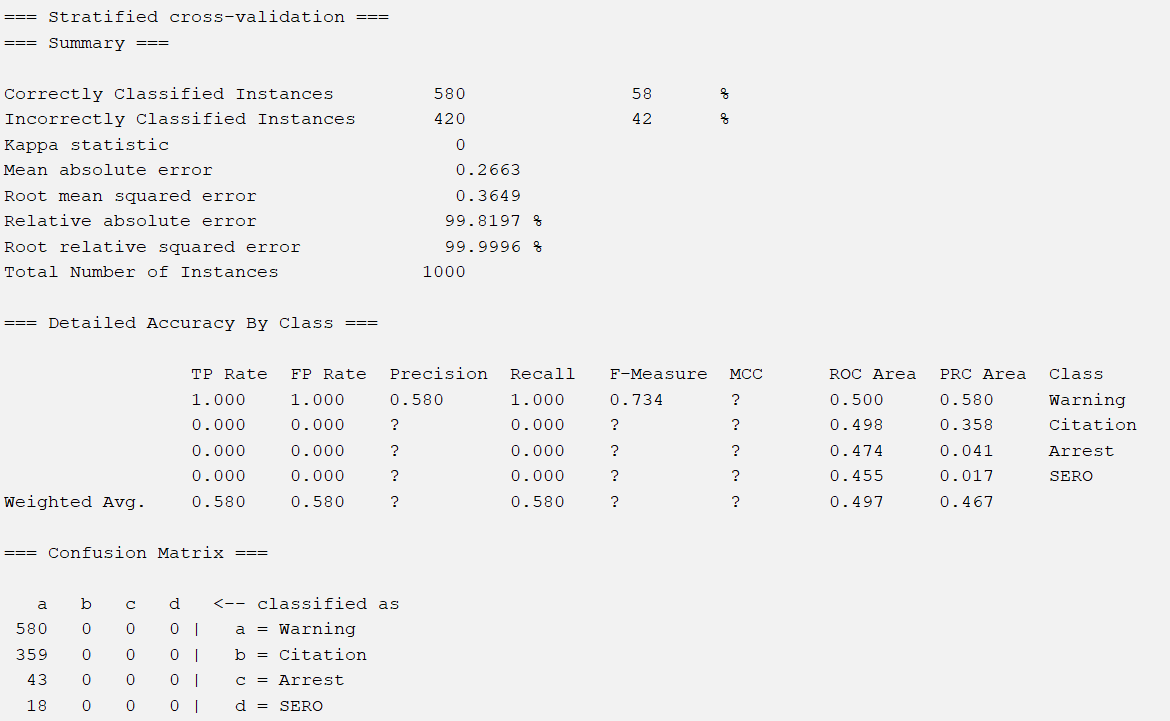
Info Gain with Naive Bayes



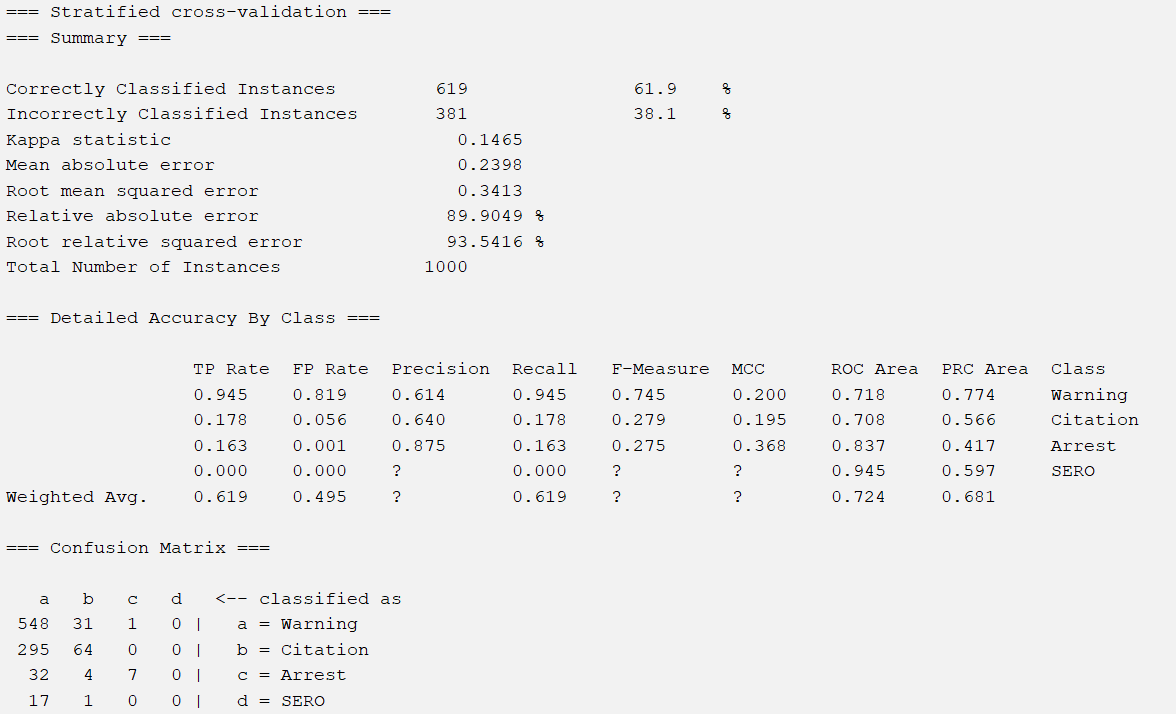
Info Gain with OneR



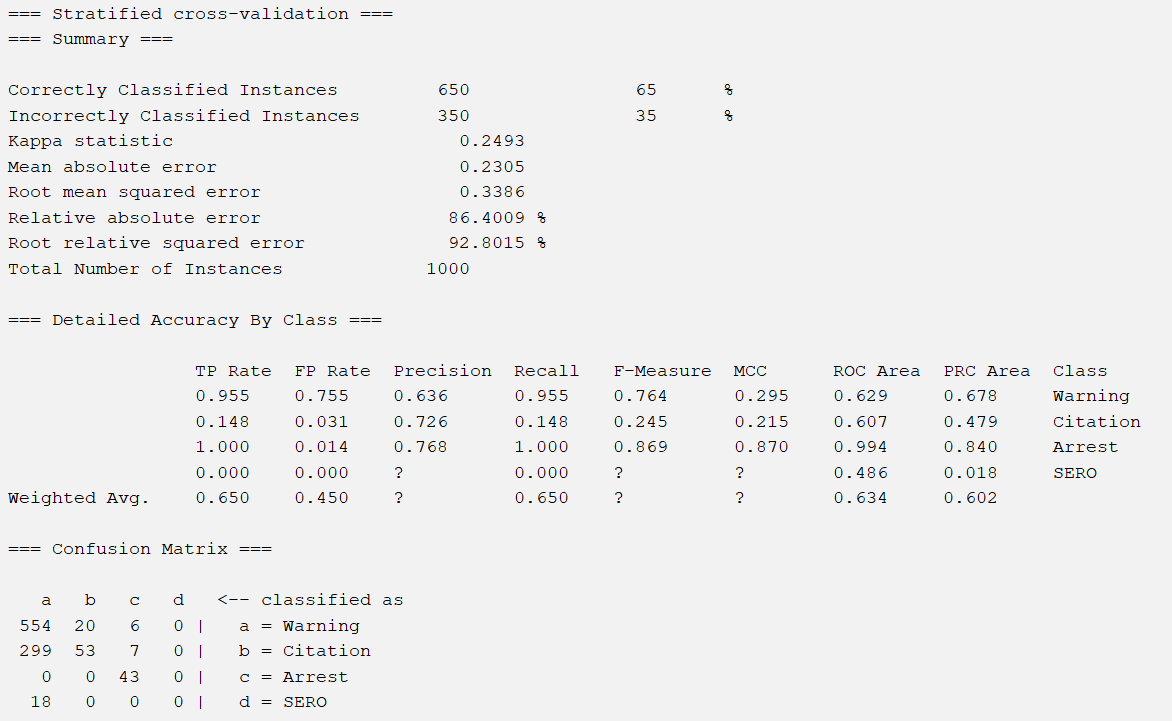
Info Gain with J48



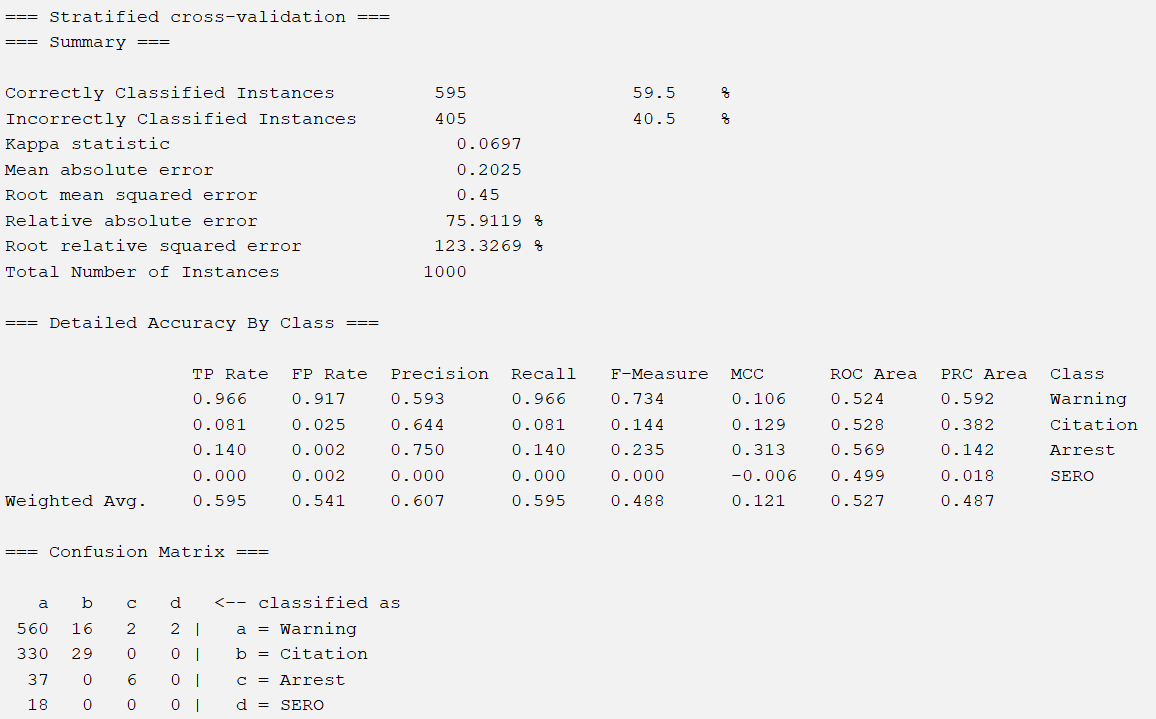
Info Gain with Random Forest



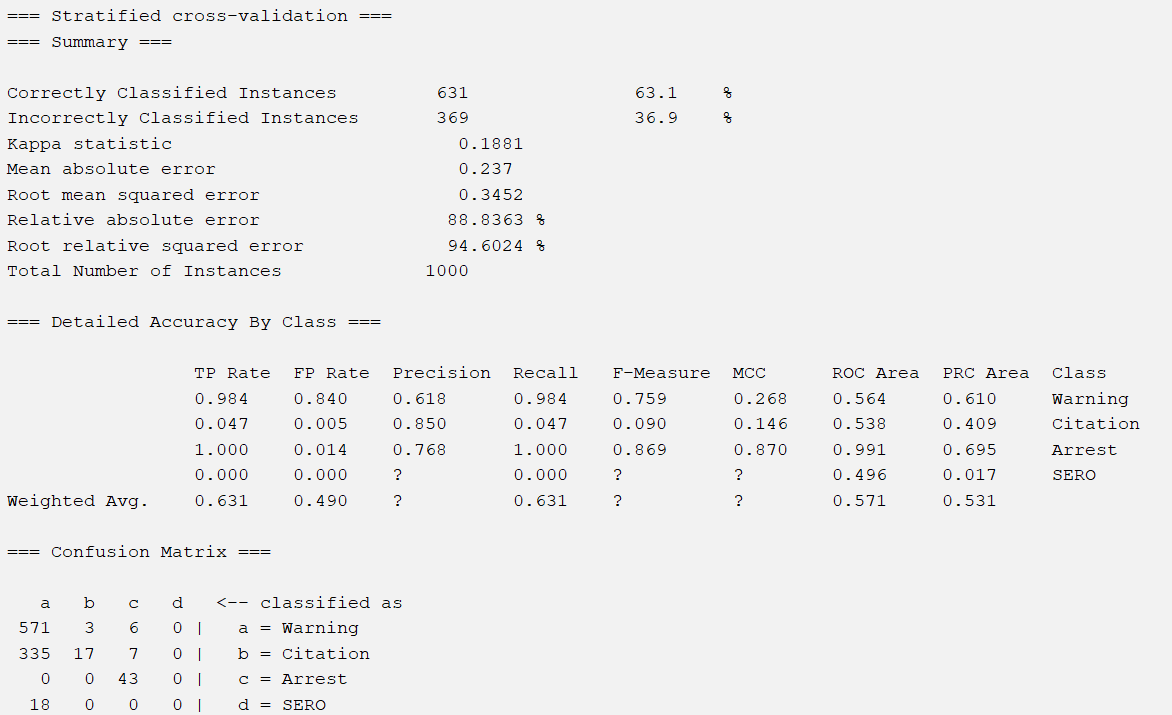
Gain Ratio with Naive Bayes



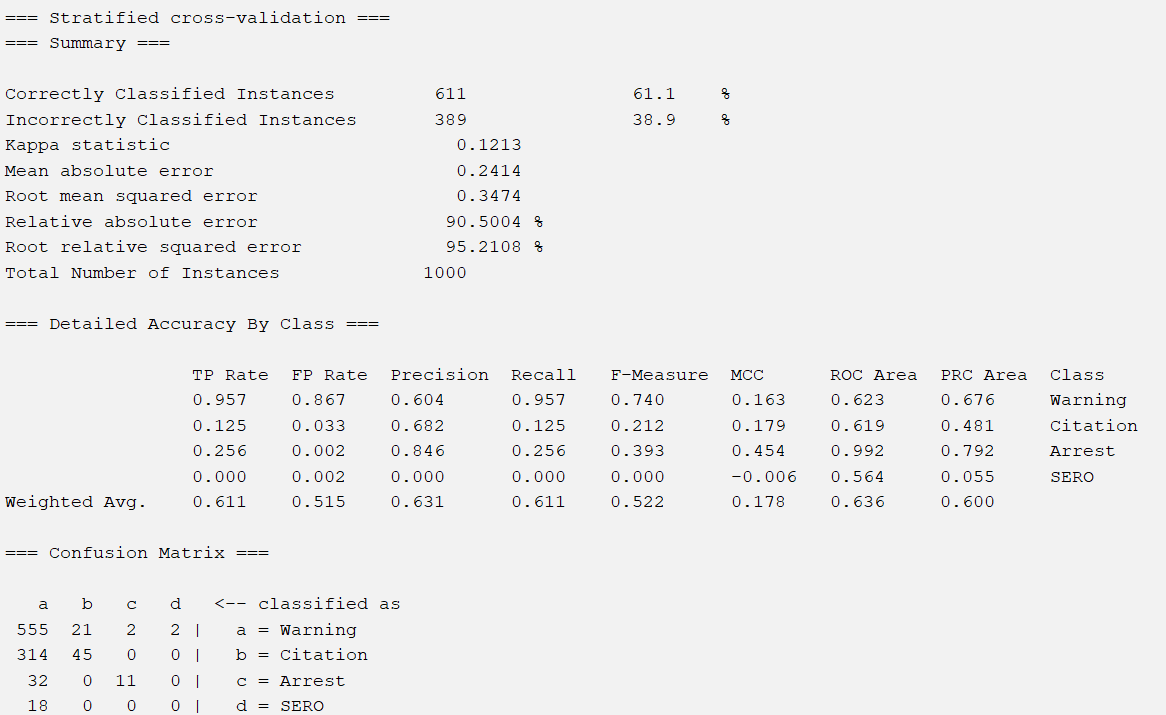
Gain Ratio with OneR



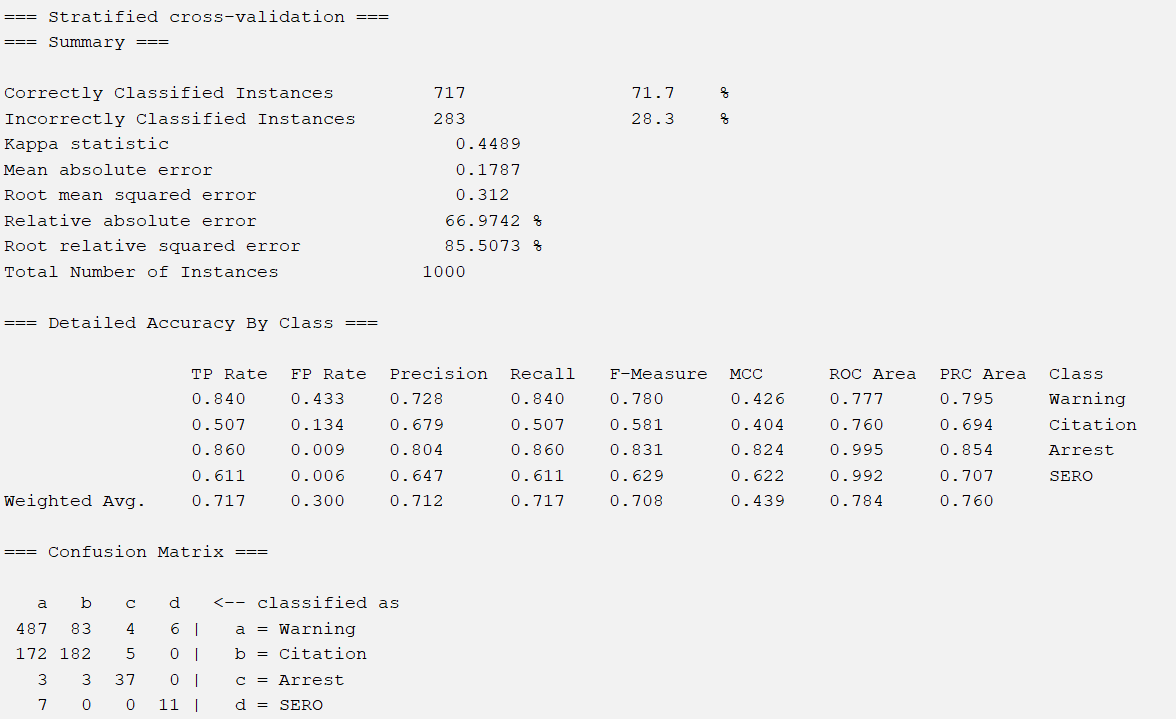
Gain Ratio with J48



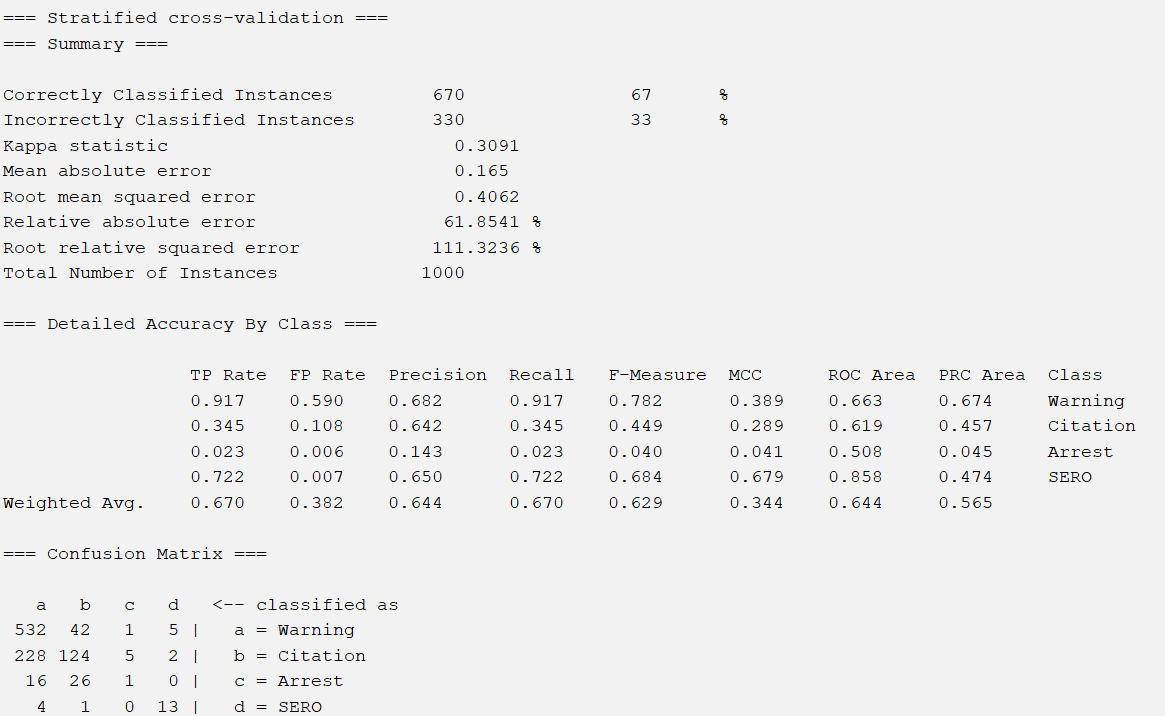
Gain Ratio with Random Forest



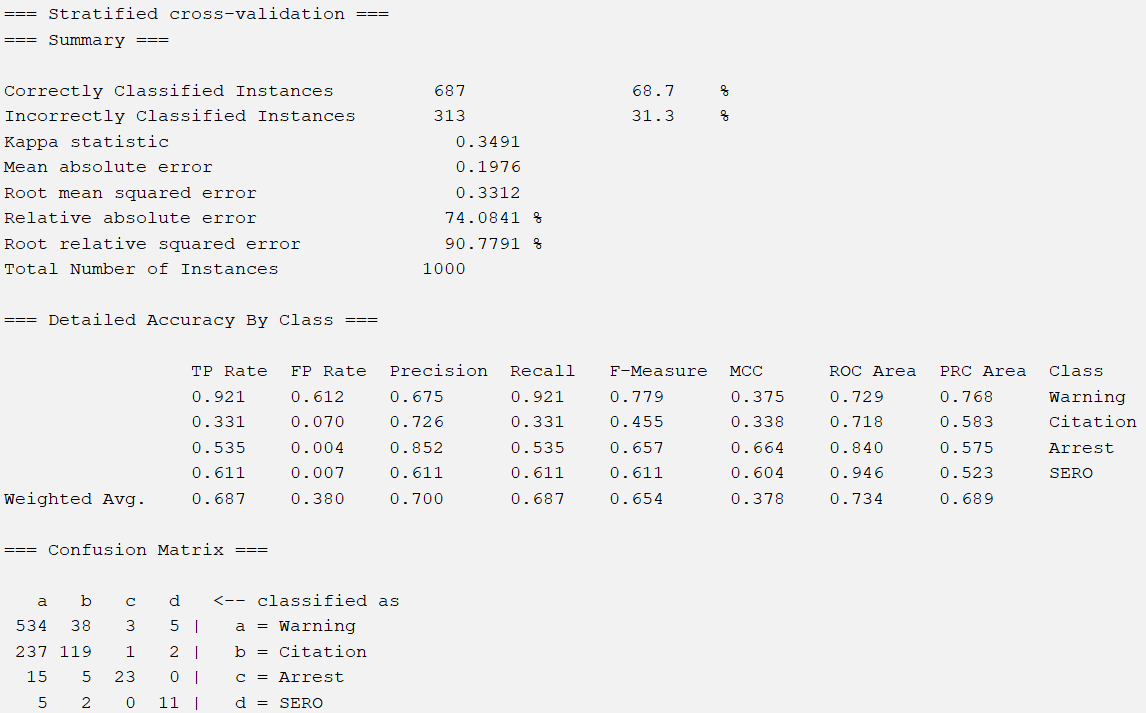
OneR with Naive Bayes



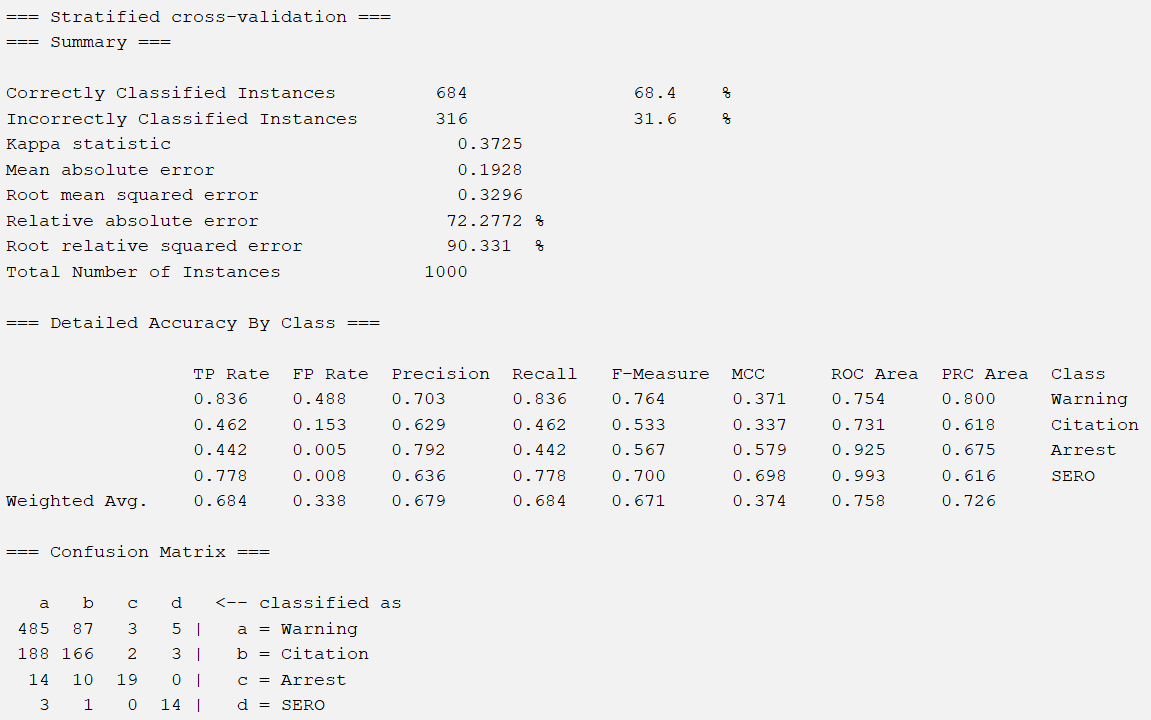
OneR with OneR



OneR with J48



OneR with Random Forest



Justification of Selection

The combination of OneR attribute selection and Naive Bayes produced the highest percent of correctly classified instances, 71.1%, closely followed by Info Gain with Naive Bayes with 70.4%. The OneR and Naive Bayes models also had among the lowest error rates, including mean absolute error, root mean squared error, relative absolute error, and root relative squared error, and the highest precision and F-measure. High accuracy and low error are good indicators of consistently satisfactory performance.

**Discussions and Conclusions**

A significant number of the models classified the majority or even all of the instances as Warning. This can be observed in the confusion matrices. Some models have zeros in all columns except the column classified as Warning. This can be explained by the skewed distribution of class values, with 58% of the instances classified as Warning. The class imbalance causes the models to perform well on the majority class value, Warning, and poorly on the minority class values.

Of the two best models, the shared attributes include Charge, Search Reason For Stop, and Search Conducted, with Charge being the highest rated by the attribute selection evaluations of OneR and Info Gain.

Although 71.1% accuracy was our best result, compared to an ideal model accuracy, our results were pretty low. There are multiple factors that could have contributed to this outcome. Because of the class imbalance, the models often had majority True Positives and False Positives. A way to mitigate the effects of the imbalance would be to balance the dataset by oversampling the minority class values or undersampling the majority class value instead of using stratified random sampling. Class weights could also be assigned to emphasize the misclassification of the minority class values. A factor that is out of our control is the inherent human bias within the data. Each traffic violation evaluation is conducted by a different police officer, at a different time, in a different place, and with different circumstances. The penalty assigned to each violation could vary depending on more trivial matters such as the officer’s emotional state or fatigue, or more significant matters such as differences in seniority or experience. The data also reflects the bias of officer’s towards the driver, if they tend to stop people and/or give harsher penalties to people of specific genders or ethnicities. A way to reduce some of the human bias is to take traffic violation data that was reported only by officers who have achieved a specific level of experience.

The model with the best results was the Naive Bayes model with OneR selection. We were able to find and preprocess a dataset with a real world application, and train and test twenty classification models to predict the penalty assigned to traffic violations in Montgomery County, Maryland. Considering the low accuracy, there is room for improvement in sampling to account for class value imbalances and filtering source data to include as little human bias as possible.

Steps to Reproduce our Chosen Model:

1. Download the Stratified Sample Dataset (found in the Appendix)
2. Open Weka and load updated\_stratified\_sample.arff
3. Click on the “Select Attributes” tab and choose “OneRAttributeEval”
   1. When prompted to select “Ranker” search, select yes
4. Select “Search Outcome” as class in drop down menu
5. Click Start
6. Use the cutoff value of 60 (inclusive) to select attributes
7. In the “Preprocess” tab select All in the Attributes window, unselect the chosen attributes and the class attribute, and press Remove
   1. This dataset can also be found in the Appendix, named OneR Evaluation Dataset
   2. Open Weka and load oner\_eval.arff
8. Navigate to the “Classify” tab
9. Choose weka → classifiers → bayes → NaiveBayes
10. Ensure “Cross-validation” with 10 folds in chosen in Test options pane
11. Ensure class in correct
12. Click Start

**Team Members and Tasks Performed**

Finding the Data: Lavanya and Dylan

Proposal: Lavanya and Dylan

Goal, Dataset, Tools: Lavanya and Dylan

Preprocessing: Lavanya

Attribute Selection Algorithms and Classifiers: Lavanya

Results Output: Dylan

Discussions and Conclusions: Dylan

Appendix: Lavanya and Dylan

Final Report: Lavanya and Dylan

Slideshow: Lavanya

**Appendix**

Data Website

* [Data Source Website](https://catalog.data.gov/dataset/traffic-violations)
* [Alternative Link](https://data.montgomerycountymd.gov/Public-Safety/Traffic-Violations/4mse-ku6q/about_data)

Datasets

* [Cleaned Dataset With Violations From 2024](https://drive.google.com/file/d/18ru4Ns_f5gZZzqCJZpAByzYnU88ZhoHB/view?usp=drive_link)
* [Stratified Sample](https://drive.google.com/file/d/1-jXxWEylqs8z-T4gHzUOx8iGVwVoLboH/view?usp=drive_link)
* [Splits (unused)](https://drive.google.com/drive/folders/1uaFPqIIV9_EVjDKA_PrYzg6udkHg-jXu?usp=drive_link)

Attribute Selection Datasets

* [Correlation Selection Dataset](https://drive.google.com/file/d/1-qDaFTURJnKpcXbqvx6DQKzJby3TeTQV/view?usp=drive_link)
* [CFS Subset Selection Dataset](https://drive.google.com/file/d/1-tut9a9OOF3hbVvSy1gY4uJKtMik_oDP/view?usp=drive_link)
* [Information Gain Selection Dataset](https://drive.google.com/file/d/1-yaX2gX2IzRMSuJSag4CwYsAm_8TqU2o/view?usp=drive_link)
* [Gain Ratio Selection Dataset](https://drive.google.com/file/d/102Sc4l2rmFUyvsiM4XKganP01GF9YBpE/view?usp=drive_link)
* [OneR Selection Dataset](https://drive.google.com/file/d/105uOQ0YlIcY2ttTvDu8pKmd0V6iyh1oj/view?usp=drive_link)

Code

* Code for Stratified Random Sampling:

import pandas as pd

df = pd.read\_csv("/content/drive/MyDrive/ML\_Project/traffic\_data.csv")

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)

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')

* Code for Training - Validation - Test Splits:

import pandas as pd

df = pd.read\_csv("/content/drive/MyDrive/machine\_learning/d\_stratified\_sample.csv")

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

cols = df.columns.tolist()

cols = cols[0:21] + cols[22:] + [cols[21]]

df = df[cols]

X = df.iloc[:, 0:-1]

y = df.iloc[:, -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()

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)

Descriptions of Attributes

1. SeqID: unique traffic stop ID
2. Date Of Stop: date of traffic violation
3. Time Of Stop: time of traffic violation
4. Agency: agency issuing the traffic violation
5. SubAgency: court code that represents the officer’s district assignment
6. Description: text description of charge
7. Location: address or intersection
8. Latitude: latitude of violation location
9. Longitude: longitude of violation location
10. Accident: binary, yes if involved in an accident
11. Belts: in accident cases, binary, yes in seat belts were used
12. Personal Injury: binary, yes if involved personal injury
13. Property Damage: binary, yes if involved property damage
14. Fatal: binary, yes if involved a fatality
15. Commercial License: binary, yes if driver has a CDL
16. HAZMAT: binary, yes if involved hazardous materials
17. Commercial Vehicle: binary, yes if the vehicle is a commercial vehicle
18. Alcohol: binary, yes if included an alcohol related suspension
19. Work Zone: binary, yes if in a work zone
20. Search Conducted: binary, yes if a search was conducted
21. Search Disposition: disposition of search
22. Search Reason: reason for search
23. Search Reason For Stop: reason for stop that lead to the search
24. Search Type: person, property, both, etc.
25. Search Arrest Reason: arrest reason from search
26. State: state of vehicle registration
27. Year: year of vehicle
28. Make: manufacturer of vehicle
29. Model: vehicle model
30. Color: vehicle color
31. Charge: numeric code for charge
32. Article: article of state law
33. Contributed To Accident: if violation contributed to accident
34. Race: race of driver
35. Gender: gender of driver
36. Driver City: city of driver’s home address
37. Driver State: state of driver’s home address
38. DL State: state issuing driver’s license
39. Arrest Type: marked, unmarked, etc.
40. Geolocation: geo-coded location information
41. Violation Type: warning, citation, ESERO
42. Search Outcome: warning, citation, arrest, SERO

Sources

GeeksForGeeks. (2024, July 10). *Naive Bayes classifiers*. GeeksForGeeks. Retrieved October 22, 2024, from https://www.geeksforgeeks.org/naive-bayes-classifiers/

Khanna, N. (2021, August 18). *J48 classification (C4.5 algorithm) in a nutshell*. Medium. Retrieved October 22, 2024, from https://medium.com/@nilimakhanna1/j48-classification-c4-5-algorithm-in-a-nutshell-24c50d20658e

*OneR*. (n.d.). Saed Sayad. Retrieved October 22, 2024, from https://www.saedsayad.com/oner.htm

*Random forest*. (2024, October 2). Wikipedia. Retrieved October 22, 2024, from https://en.wikipedia.org/wiki/Random\_forest