# 

# 

# 

# 

# 

**Machine Learning - Dr. Yilmaz Period 5**

**Predicting Crash Type from Crash Report Incident**

**Team Members: Gabriel Xu, Andrew Chen**

**10/07/2024**

### **Table of Contents**

[Table of Contents 2](#_vz530d6rc7oi)

[Part 1: Dataset Overview 3](#_8j1emcdlaa16)

[Part 2: Preprocessing 7](#_qqr3jy873kw4)

[Part 3: Attribute Analysis 10](#_mqjsar88sn0t)

[Part 4: Classifier Models 13](#_ot9ve3l7dbpj)

[Part 5: Discussion and Conclusion 14](#_ru4r5ndwyxn)

### 

### **Part 1: Dataset Overview**

**Section 1.1: Introduction**

The dataset we used is the “Crash Data” dataset, linked here: <https://catalog.data.gov/dataset/crash-data>. There are 24241 instances of 42 attributes of crashes occurring in the Town of Cary, North Carolina. We plan to classify **how severe** the results of the crash were. This means predicting whether there were no injuries or fatalities, injuries but no fatalities, or fatalities. This dataset will be useful for finding how certain **conditions** such as location, weather, and the road combine to **impact** how **dangerous** a crash could be.

**Section 1.2: Meaning of Attributes**

**1. tamainid**

* **Meaning**: Unique ID for each traffic accident entry.
* **Example**: 48247, 48253 – these are unique IDs for individual accidents

**2. location\_description**

* **Meaning**: Describes the location of the accident (e.g., street names or distances from landmarks).
* **Example**: “30 FEET FROM SR3977 (SW CARY PKWY)” – this accident occurred 30 feet from a specific street,

**3. rdfeature**

* **Meaning**: Describes special road features near the accident location.
* **Example**: “RAILROAD CROSSING” – The accident took place near a railroad crossing.

**4. rdcharacter**

* **Meaning**: The road’s physical characteristics, such as whether it’s straight or curved.
* **Example**: “STRAIGHT, LEVEL” – the road was straight and level at the time of the accident.

**5. rdclass**

* **Meaning**: Classification of the road, such as whether it’s a local street or highway.
* **Example**: “LOCAL STREET” – the accident occurred on a smaller, local street

**6. rdconfigur**

* **Meaning**: Describes the configuration of the road, such as whether it’s divided or undivided.
* **Example**: “TWO-WAY, DIVIDED, POSITIVE MEDIAN BARRIER” – the road has two-way traffic with a median barrier.

**7. rdsurface**

* **Meaning**: The type of surface the road has.
* **Example**: “SMOOTH ASPHALT” – the road surface was smooth asphalt

**8. rdcondition**

* **Meaning**: The condition of the road at the time of the accident.
* **Example**: “DRY” – the road was dry when the accident occurred.

**9. lightcond**

* **Meaning**: Describes the lighting conditions at the time of the accident.
* **Example**: “DAYLIGHT” – the accident occurred during daylight hours

**10. weather**

* **Meaning**: Weather conditions during the time of the accident.
* **Example**: “CLEAR” – the weather was clear during the accident

**11. trafcontrl**

* **Meaning**: Describes the traffic control present at the location (e.g., traffic lights, stop signs).
* **Example**: “RR GATE AND FLASHER” – A railroad crossing gate and flasher were present.

**12. lat**

* **Meaning**: Latitude coordinate of the accident location.
* **Example**: -78.821706 – the latitude of the accident site

**13. lon**

* **Meaning**: Longitude coordinate of the accident location.
* **Example**: 35.761999 – the longitude of the accident site

**14. lon2**

* **Meaning**: A second longitude value, possibly for marking a different point in the accident area.
* **Example**: -78.787907 – Another longitude point related to the accident.

**15. lat2**

* **Meaning**: A second latitude value, similar to lon2 for another geographic point in the accident.
* **Example**: 35.716448 – Another latitude point.

**16. tract**

* **Meaning**: A geographic subdivision, probably a census tract
* **Example**: “P054” – A specific census tract for geographic reference.

**17. zone**

* **Meaning**: Refers to a traffic or urban zone (e.g., residential, school zone).
* **Example**: “116” – Numeric representation of the zone where the accident occurred.

**18. fatality**

* **Meaning**: Indicates if there was a fatality (Yes/No).
* **Example**: “No” – No fatalities were recorded for this accident.

**19. possblinj**

* **Meaning**: Indicates if there were possible injuries (Yes/No).
* **Example**: “No” – No injuries were recorded for this accident.

**20. numpassengers**

* **Meaning**: The number of passengers in the vehicles involved in the accident
* **Example**: “1” – There was one passenger involved.

**21. numpedestrians**

* **Meaning**: The number of pedestrians involved in the accident.
* **Example**: “0” – No pedestrians were involved in the accident.

**22. contrcir1\_desc**

* **Meaning**: Describes a contributing factor to the accident (e.g., distracted driving, speeding).
* **Example**: “NONE” – No specific contributing factor was recorded.

**23. contrcir2\_desc**

* **Meaning**: Describes an additional contributing factor, if applicable.
* **Example**: “NONE” – No secondary contributing factor was recorded.

**24. contrcir3\_desc**

* **Meaning**: Describes a third contributing factor, if applicable.
* **Example**: “NONE” – No third contributing factor was recorded.

**25. contrcir4\_desc**

* **Meaning**: Describes a fourth contributing factor, if applicable.
* **Example**: “NONE” – No fourth contributing factor was recorded.

**26. vehicle1**

* **Meaning**: Describes the first vehicle involved in the accident.
* **Example**: “PICKUP” – The first vehicle was a pickup truck.

**27. vehicle2**

* **Meaning**: Describes the second vehicle involved, if applicable.
* **Example**: “SPORT UTILITY” – The second vehicle was a sport utility vehicle.

**28. vehicle3**

* **Meaning**: Describes the third vehicle involved, if applicable.
* **Example**: “PASSENGER CAR” – The third vehicle was a passenger car.

**29. vehicle4**

* **Meaning**: Describes the fourth vehicle involved, if applicable.
* **Example**: “None” – No fourth vehicle was involved.

**30. vehicle5**

* **Meaning**: Describes the fifth vehicle involved, if applicable.
* **Example**: “None” – No fifth vehicle was involved.

**31. workarea**

* **Meaning**: Indicates whether the accident occurred in a work zone (Yes/No).
* **Example**: “NO” – The accident did not occur in a work zone.

**32. records**

* **Meaning**: Could refer to the record number or entry in the database.
* **Example**: “10003” – This is the database record number.

**33. ta\_date**

* **Meaning**: The date when the traffic accident occurred.
* **Example**: “2021-07-07” – The accident occurred on July 7, 2021.

**34. ta\_time**

* **Meaning**: The time when the traffic accident occurred.
* **Example**: “2:18:32 PM” – The accident occurred at this specific time.

**35. crash\_date**

* **Meaning**: The timestamp for when the crash was officially recorded.
* **Example**: “2021-07-07T18:18:32+00:00” – The crash was recorded at this time in UTC format.

**36. geo\_location**

* **Meaning**: The latitude and longitude of the crash combined for easy geographic reference.
* **Example**: “35.716440073, -78.78796424” – The location of the accident.

**37. year**

* **Meaning**: The year in which the accident occurred.
* **Example**: “2021” – The accident occurred in 2021.

**38. fatalities**

* **Meaning**: If there were fatalities in the accident.
* **Example**: “No” – No fatalities resulted from this accident.

**39. injuries**

* **Meaning**: If there were injuries in the accident.
* **Example**: “No” – No injuries were reported.

**40. month**

* **Meaning**: The month when the accident occurred.
* **Example**: “7” – The accident occurred in July.

**41. contributing\_factor**

* **Meaning**: Main contributing factor that caused the accident.
* **Example**: “NONE, NONE” – No contributing factors were recorded.

**42. vehicle\_type**

* **Meaning**: Types of vehicles involved in the accident.
* **Example**: “PICKUP” – A pickup truck was involved in the accident.

**Section 1.3: Preprocessing Plans**

Most of the columns with categorical data types are all skewed towards certain values. The class variables, the number of fatalities and injuries are also heavily skewed towards lower values. For preprocessing we will need to fix areas such as missing values and deleting columns that won’t be helpful. The columns and rows with a high amount of missing values (greater than ~70%) will be deleted and the remaining ones will likely have missing values replaced. Another thing to fix would be how certain features contain a list of individual characteristics that we should further separate. Another obvious step is to normalize the data since the attributes are on different scales.

### 

### **Part 2: Preprocessing**

**Section 2.1: Transform Class Column**

Using the columns of “fatalities” and “injuries”, construct a new “**class**” column with possible values of “crash”, “injury”, or “fatalities”. If “fatalities” is Yes, the value will be “fatalities”. Then if “injuries” is Yes, the value will be “injury”. Otherwise, the value will be “crash”. We then obviously removed the “injuries” and “fatalities” features. This dataset is titled as “cpd-crash-incidents.csv” in the Google Drive folder.

**Section 2.2: Delete Useless Columns and Rows**

We dropped the columns “tamainid” and “records” because they likely don't have any correlation with the class and are just IDs. The “fatality” and “posiblinj” columns were removed because they are redundancies of the “fatalities” and “injuries” columns. “lat” and “lon” were likewise removed because they are repeats of “lat2” and “lon2”, and they have much more missing values than their counterparts. “location\_description”, “tract”, “contributing\_factor”, and “vehicle\_type” were removed because there aren't a lot of repeats among instances as there are a lot of different possible values, so basic classifier models wouldn’t be able to properly process this text. “ta\_date”, “crash\_date”, and “year” for similar reasons as above. “geo\_location” is removed since it is derived from longitude and latitude. Although “zone” is technically derived from the location to some extent, we thought it was worth keeping for now since the relationship is not a clear, direct one.

We also checked for repeat instances and removed them.

The code to complete Sections 2.1 and 2.2 is here:

data=[row.split(';') for row in open("cpd-crash-incidents.csv").read().strip().split("\n")]

new\_data=[]

to\_delete=["tamainid", "records", "fatality", "year", "possblinj", "lat", "lon", "location\_description", "tract", "contributing\_factor", "vehicle\_type", "ta\_date", "crash\_date", "geo\_location"]

for row in data:

a=[]

for i, x in enumerate(row):

if data[0][i] not in to\_delete:

a.append(x)

if a[-3]=="Yes": a.append("fatalities")

elif a[-2]=="Yes": a.append("injury")

else: a.append("crash")

a=a[:-4]+a[-2:]

new\_data.append(a)

new\_data[0][-1]="class"

**Section 2.3: Clean up N/A/Missing Values**

All the cells with a value of N/A, “None”, “Other”, and “Unknown”, were converted to an empty cell. We decided to substitute missing values in the “numpedestrians” column with the value of 0.

The code to do this is here:

for i in range(len(new\_data)):

for j in range(len(new\_data[0])):

if new\_data[i][j] in ["NONE", "OTHER \*", "UNKNOWN"]:

new\_data[i][j]=""

if new\_data[i][j]=="" and j==13:

new\_data[i][j]="0"

**Section 2.4: Transform Attributes into Usable Form**

We added a new column “vehicles” based on the number of values for “vehicle1” through “vehicle5” are not empty.

for i in range(1, len(new\_data)):

count=0

for j in range(len(new\_data[0])):

if new\_data[0][j][:-1]=="vehicle" and new\_data[i][j]!="":

count+=1

new\_data[i]=new\_data[i][:-1]+[count]+new\_data[i][-1:]

new\_data[0]=new\_data[0][:-1]+["vehicles"]+new\_data[0][-1:]

The “ta\_time” column was transformed into groups for each hour of the day, as minutes are too specific for models to use.

for i in range(1, len(new\_data)):

for j in range(len(new\_data[0])):

if new\_data[0][j]=="ta\_time":

new\_time=int(new\_data[i][j][:new\_data[i][j].index(':')])

if new\_data[i][j][-2:]=="PM":

if new\_time!=12:

new\_time+=12

elif new\_time==12:

new\_time=0

new\_data[i][j]=new\_time

with open("final.csv", "w") as f:

for row in new\_data:

f.write(";".join(map(str, row))+"\n")

**Section 2.5: Handle Empty Cells**

The columns and rows with more than >=70% missing values were dropped. First, we removed the attributes of “contrcir1\_desc”, “contrcir2\_desc”, “contrcir3\_desc”, “contrcir4\_desc”, “vehicle3”, “vehicle4”, and “vehicle5”.

There were no instances we needed to delete.

Then we ran WEKA’s method filter of replacing missing values with the mode/mean of that attribute.

**Section 2.6: Normalize Data**

We then decided to use z-score normalization on all the discrete attributes through the standardize filter (except for the class).

**Section 2.8: Summary**

After all of these preprocessing steps, the dataset (the one titled “final.csv”) now has 24241 instances of 20 columns (not including class). For the class labels, 0.18% are “fatalities”, 13.34% are “injury”, and 86.47% are “crash”.

### **Part 3: Attribute Analysis**

1. **non-Weka Selection Method**

After careful consideration, we chose to keep the following attributes, as they would likely have a considerable impact on crash severity:

1) **weather** – weather conditions (e.g. rainy or snowy) can impact visibility and road conditions, leading to more severe accidents

2) **trafcontrl** -- the presence of traffic controls (e.g. stop signs) impacts crash severity

3) **lightcond** -- poor lighting conditions (night or dawn/dusk) could increase crash severity

4) **rdcondition** (Road Condition) – slippery or damaged roads can lead to increased crash severity

5) **rdfeature** – road features like curves, intersections, and narrow lanes may lead to more severe crashes

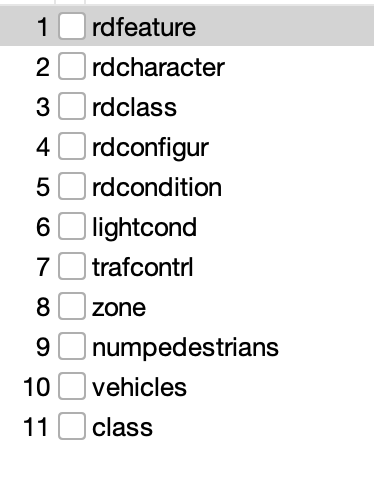
6) **vehicle1** and **vehicle2** – types of vehicles involved (e.g., cars, trucks, motorcycles) could impact the severity, with larger vehicles likely being more severe

7) **numpassengers** – a higher number of passengers could influence the overall impact as well as severity

8) **numpedestrians** – the presence of pedestrians likely increases the likelihood of severe injury.

1. **CorrelationAttributeEval**

We used Weka for this approach. This approach evaluates attributes and ranks them based on their individual correlations with the class label (crash, injury, fatalities). We only kept attributes with a **correlation score of 0.02 or higher**. This threshold ensures that the most relevant features are selected while also minimizing noise from less meaningful attributes.

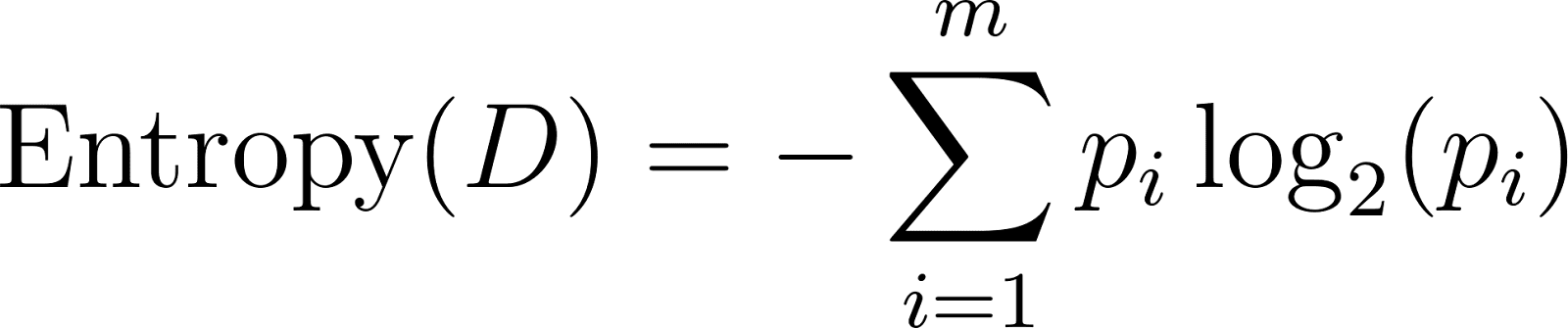
The attributes that remain after applying this selection algorithm are:  


1. **InfoGainAttributeEval**

We used Weka for this approach. This approach ranks attributes based on how much information they provide about the class label (crash, injury, fatalities). In other words, this method measures the **Information Gain** for each attribute.

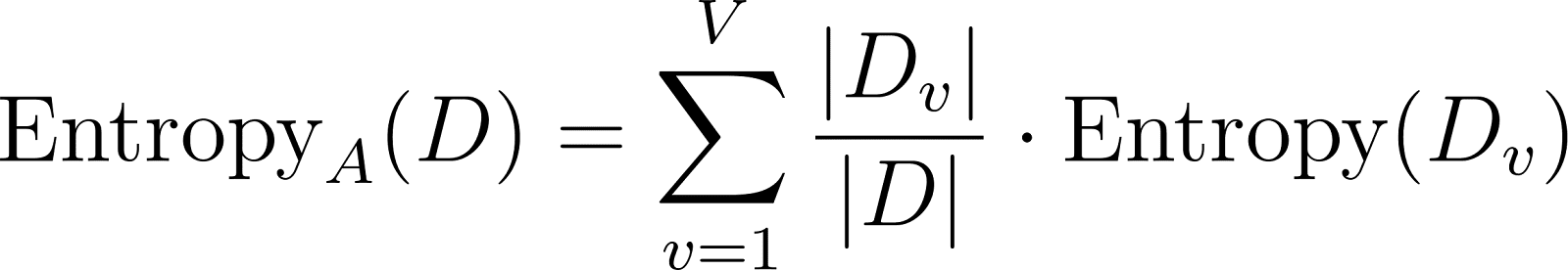
Information Gain can be calculated by these three steps.

1. **Entropy of dataset :**



Where  is the proportion of instances in class  in the dataset , and  is the number of classes.

1. **Entropy after splitting on attribute A**:

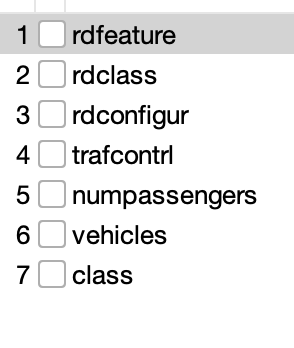


Where [](https://www.codecogs.com/eqnedit.php?latex=D_v#0) is the subset of [](https://www.codecogs.com/eqnedit.php?latex=D#0) where attribute [](https://www.codecogs.com/eqnedit.php?latex=A#0) has value [](https://www.codecogs.com/eqnedit.php?latex=v#0), and [](https://www.codecogs.com/eqnedit.php?latex=V#0) is the number of distinct values of [](https://www.codecogs.com/eqnedit.php?latex=A#0).

1. **Information Gain**:



A higher IG for an attribute indicates that it provides more information about the class labels. The attributes that remain after applying this selection algorithm are below. Attributes were only selected if they had **InfoGain of 0.01 or higher**.



**D. ReliefF**

We used Weka for this approach. This technique evaluates the importance of each attribute by comparing how similar instances are within the same class (nearest hits) versus how different they are from different classes (nearest misses). The ReliefF Algorithm then ranks attributes based on how well they separate different classes.

For an attribute [](https://www.codecogs.com/eqnedit.php?latex=A#0), the weight of the attribute is calculated by:

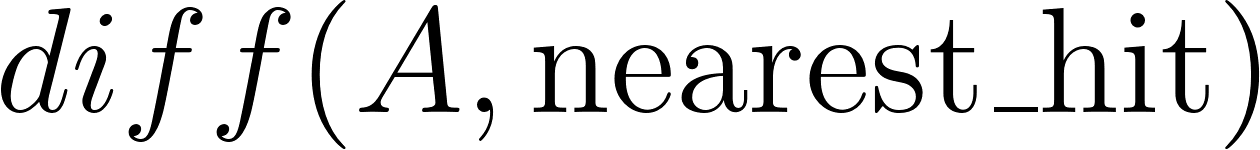


Where:

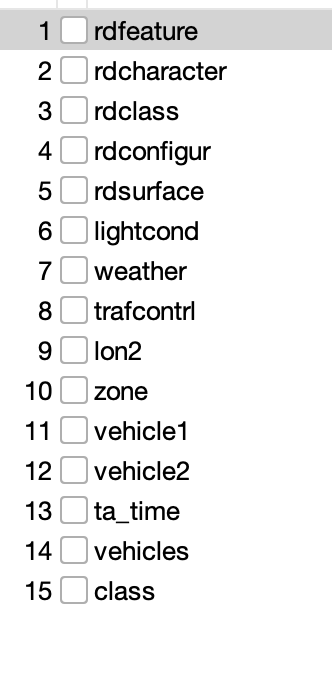
[](https://www.codecogs.com/eqnedit.php?latex=m#0) is the number of sampled instances.

[](https://www.codecogs.com/eqnedit.php?latex=%5Ctext%7Bnearest%5C_hit%7D#0) refers to the closest instance of the same class.

[](https://www.codecogs.com/eqnedit.php?latex=%5Ctext%7Bnearest%5C_miss%7D#0) refers to the closest instance from a different class.

[](https://www.codecogs.com/eqnedit.php?latex=diff(A%2C%20%5Ctext%7Bnearest%5C_hit%7D)#0) is the difference in attribute A between the instance and its nearest hit, and similarly for the nearest miss.

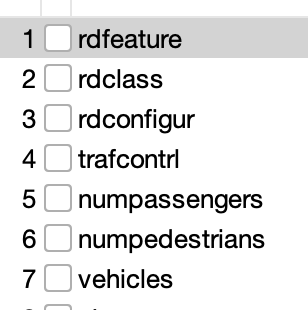
We selected attributes with a weight of **0.02 or higher**. The attributes that remain after applying the algorithm are below:



**E. CfsSubsetEval**

We used Weka for this approach, which selects a subset of features based on how well they predict and how little they overlap each other (both relevant and not redundant).

The attributes that remain after applying this model are:

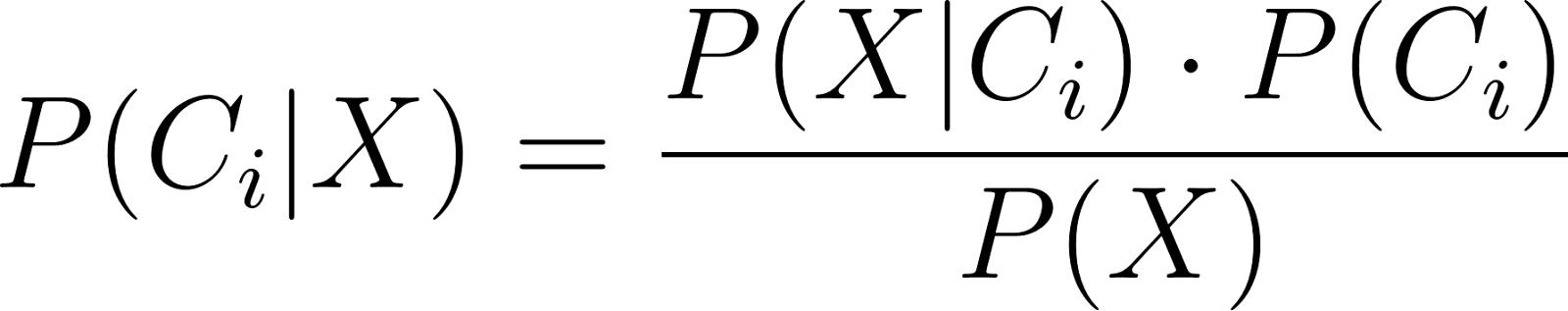


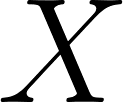
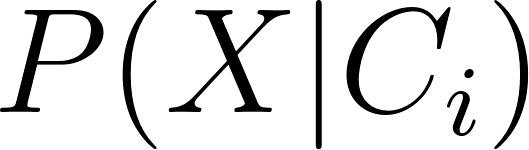
### **Part 4: Classifier Models**

**1. bayes.NaiveBayes**

Naive Bayes is a probabilistic classifier based on Baye’s theorem. Its called “naive” because it assumes that attributes of a dataset are independent and don’t affect each other. The algorithm chooses the class with the highest conditional probability.

Naive Bayes uses Bayes’ theorem:



Where  is the probability of class  given the data . Because it assumes all features as independent, we simplify [](https://www.codecogs.com/eqnedit.php?latex=P(X%7CC_i)#0) as:



Then, the class with the highest  is predicted

**2. tree.J48**

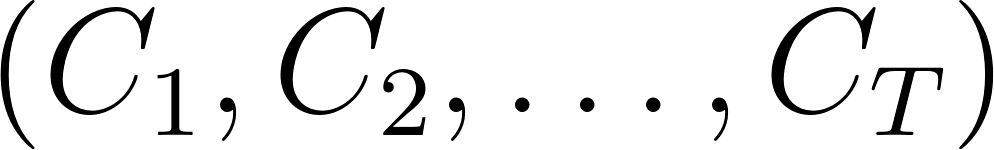
J48 is an implementation of the C4.5 decision tree algorithm. It recursively splits the dataset and selects the attribute with the highest information gain. Each internal node represents a decision on an attribute, and the leaf nodes represent the final class prediction.

**3. tree.RandomForest**

RandomForest is a learning method that builds a collection of decision trees, known as a “forest”. Each tree is trained on a random subset of the dataset.

Random Forest selects the class based on majority voting:



Where  are the predictions from individual trees.

The final class is the one with the highest number of votes from the trees in the forest.

**4. rules.DecisionTable**

DecisionTable creates a table of rules based on the attributes. Each rule corresponds to a unique set of conditions on the attributes. If multiple rules apply, the classifier selects the class based on majority rule.

The Decision Table then evaluates each condition:

[](https://www.codecogs.com/eqnedit.php?latex=R_i%3A%20%5Ctext%7Bif%7D%20%5C%20A_1%20%3D%20x_1%20%5C%20%5Ctext%7Band%7D%20%5C%20A_2%20%3D%20x_2%20%5C%20%5Cdots%20%5C%20%5Ctext%7Bthen%7D%20%5C%20C_i#0)

**5. rules.OneR**

Unlike DecisionTable which creates multiple rules, the OneR classifier creates one rule for each attribute in the data. The algorithm then selects the rule with the smallest error rate, following the below pseudocode.

For each attribute

For each value of the attribute

count the frequency of each class

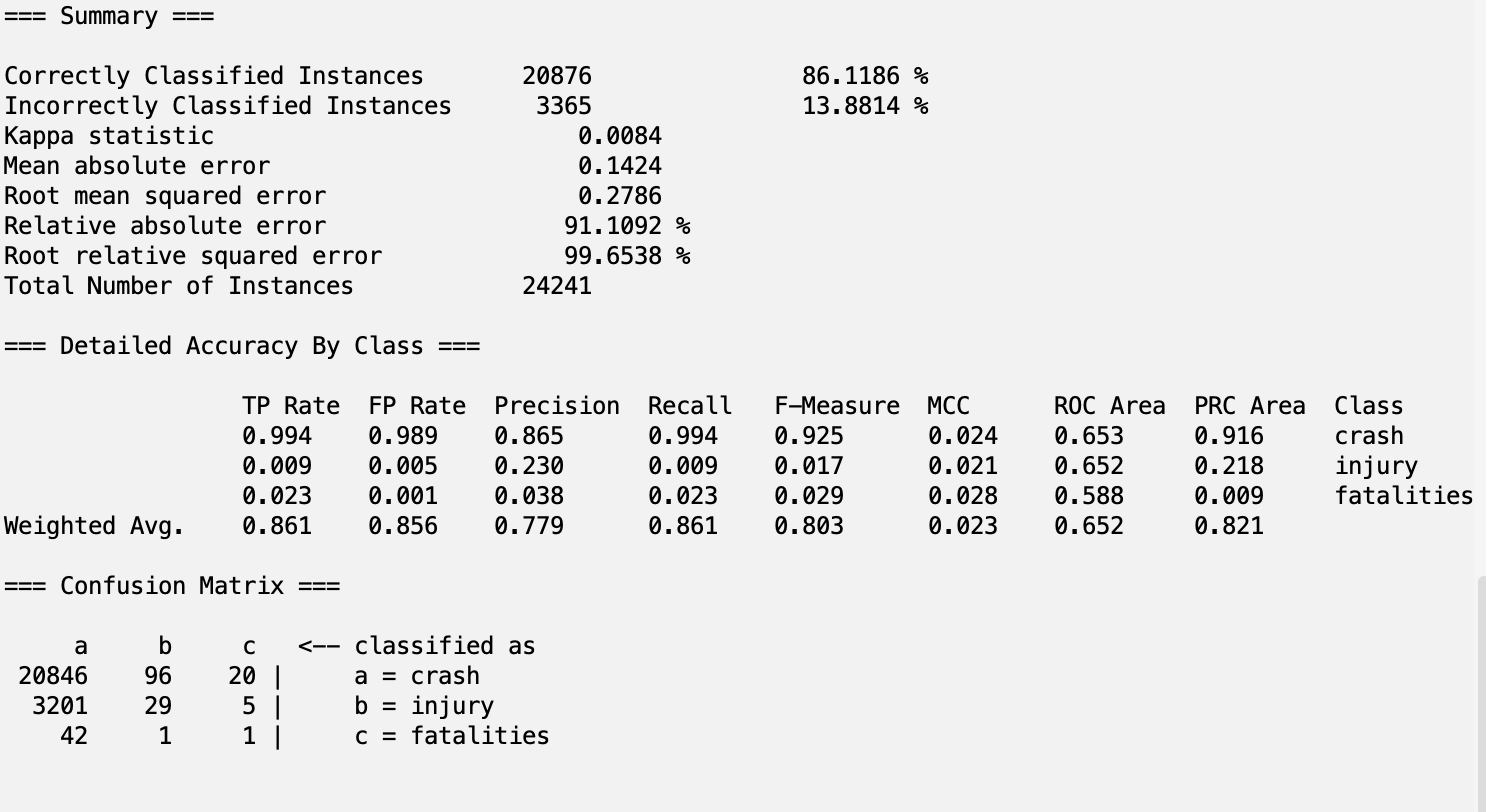
find the most frequent class

make rule: assign that class to this attribute-value

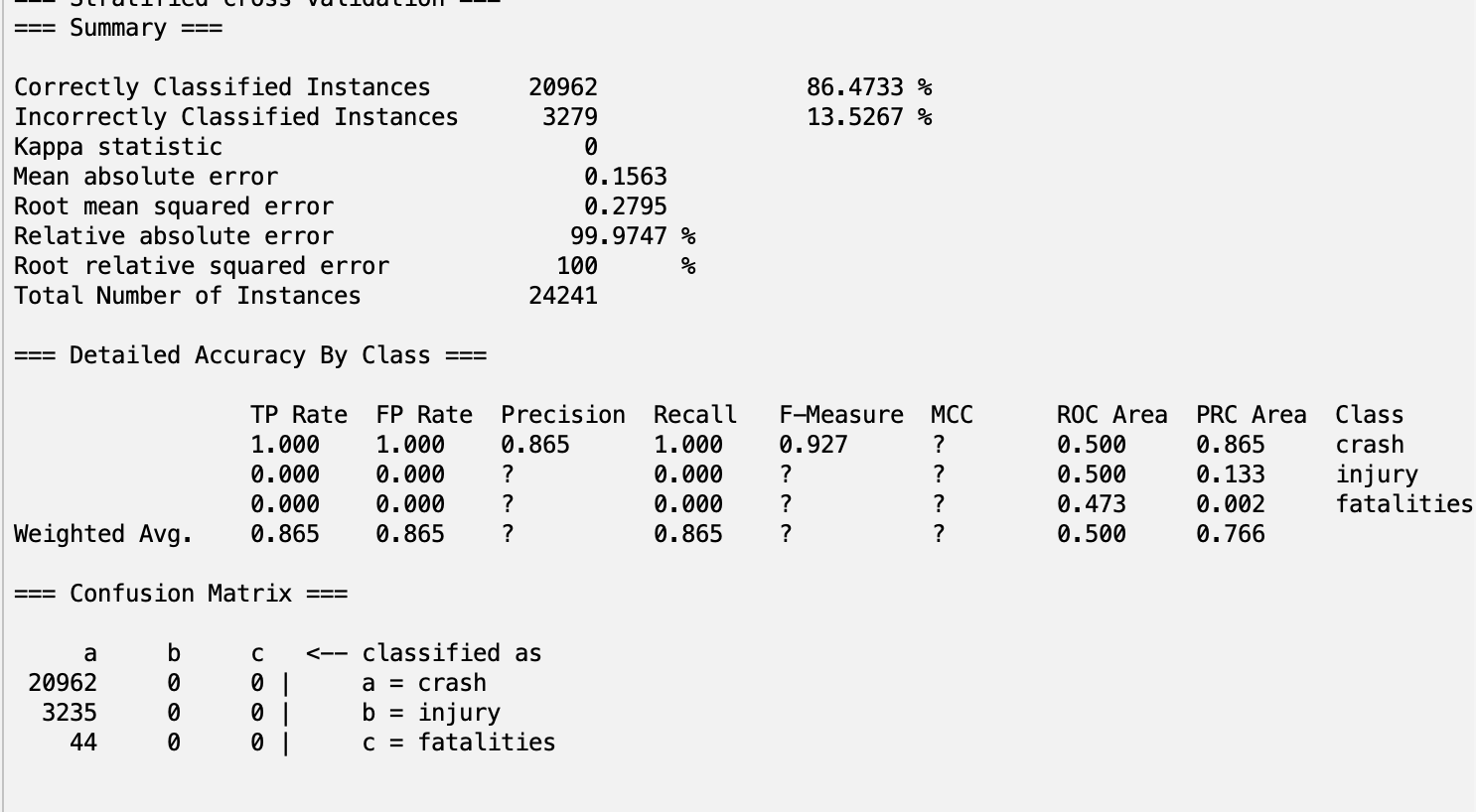
Compute error rate of the rules (of this attribute)

Choose the rules with the smallest error rate

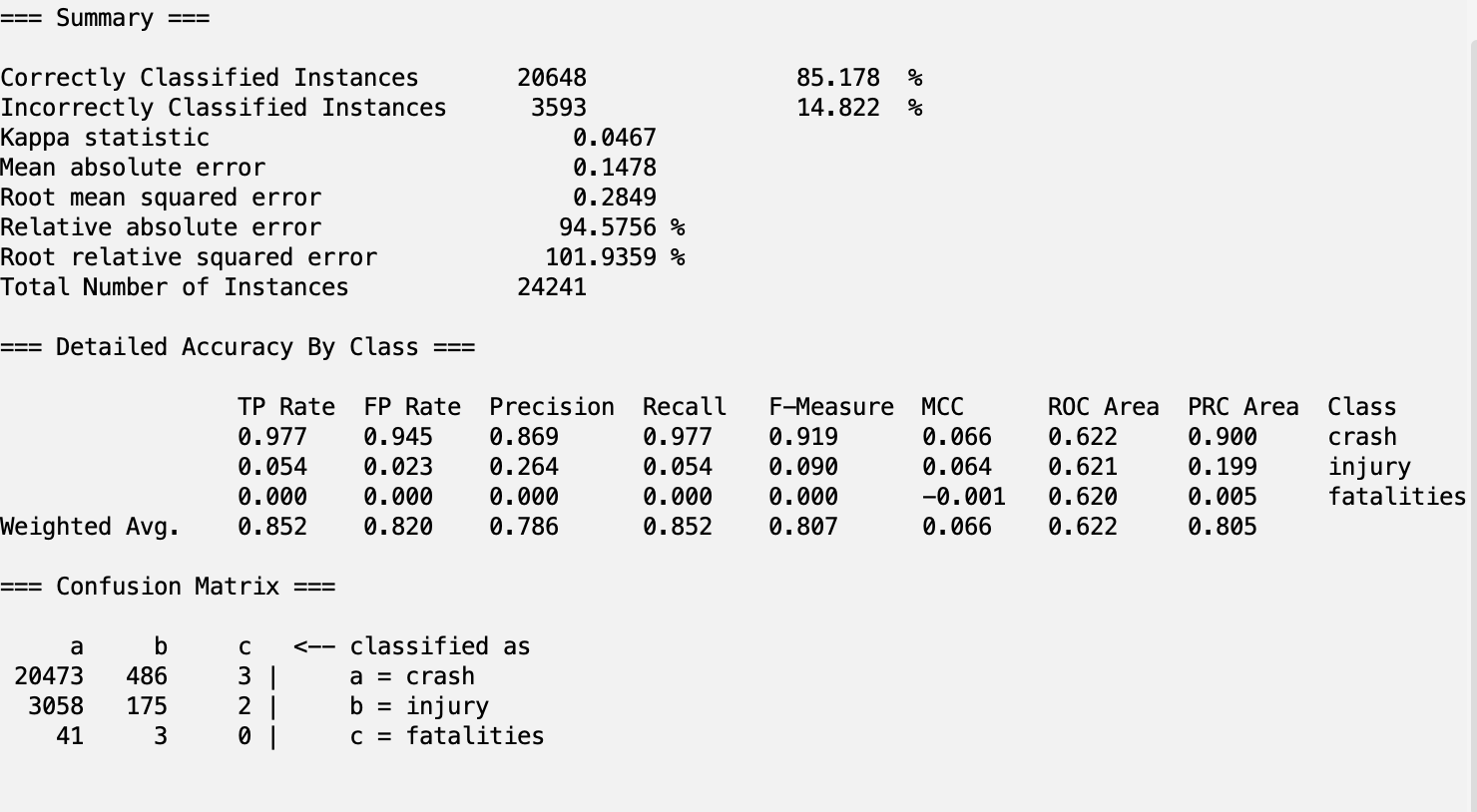
**Non-Weka with Naive Bayes**



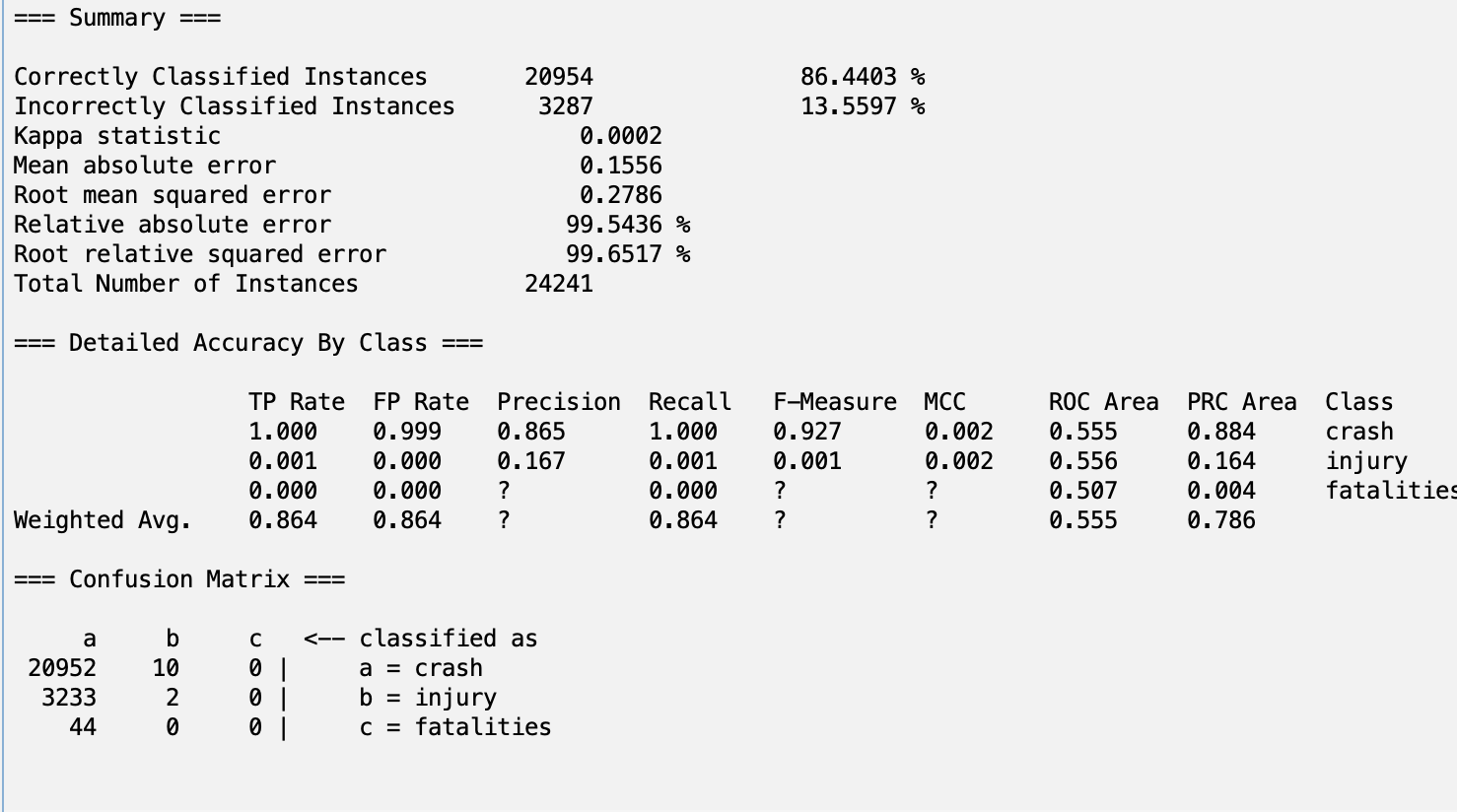
**Non-Weka with J48**

****

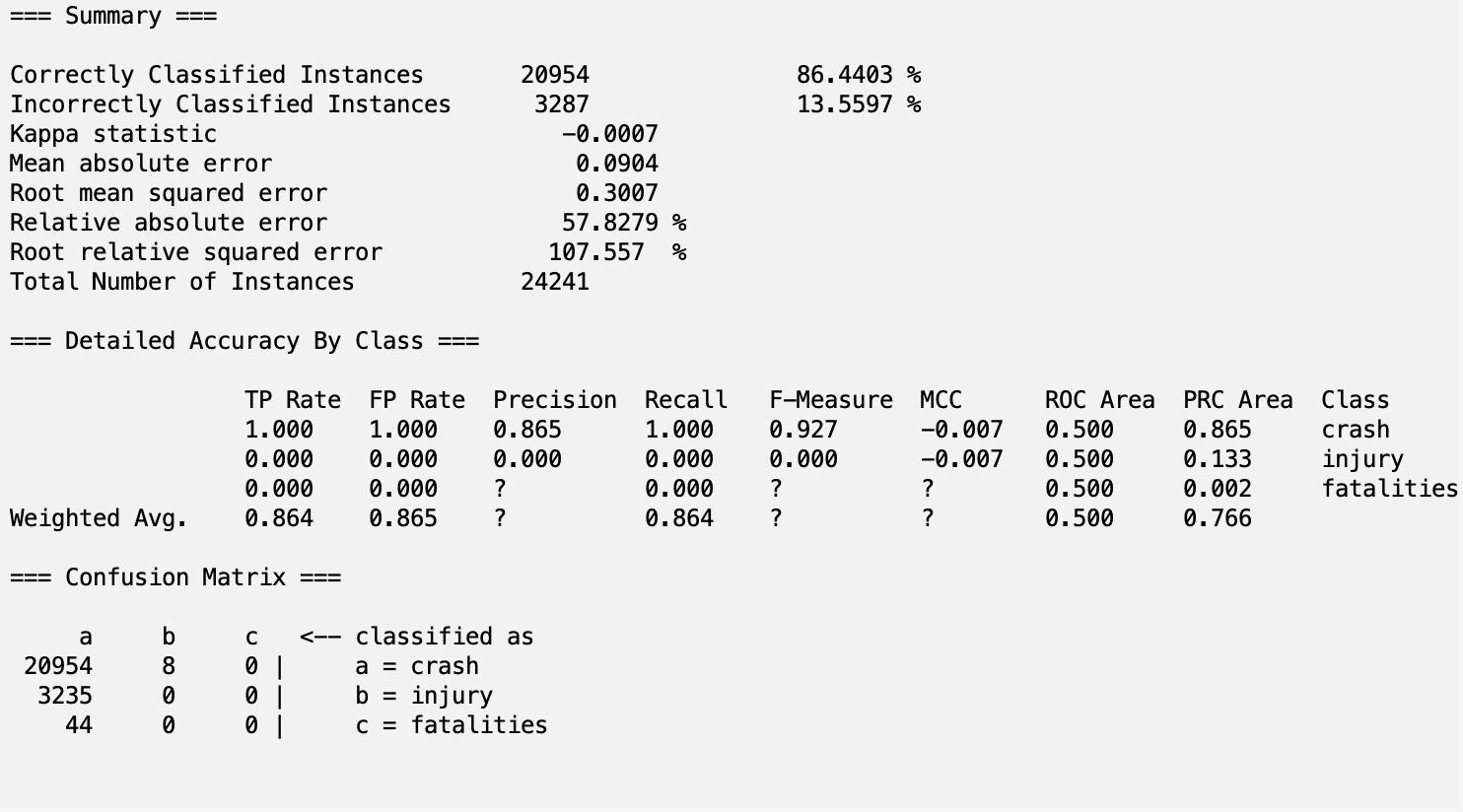
**Non-Weka with RandomForest**



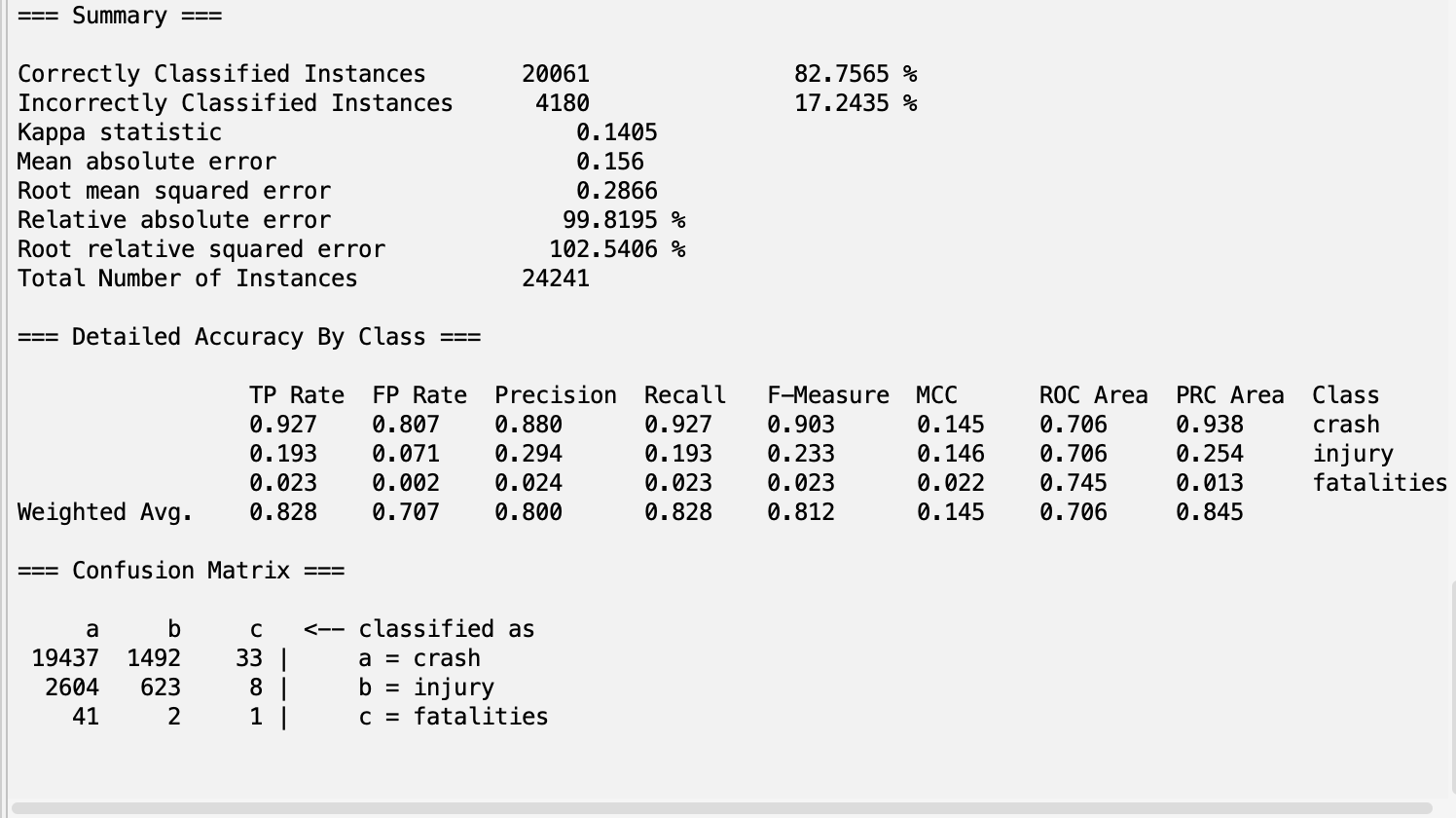
**Non-Weka with DecisionTable:**

****

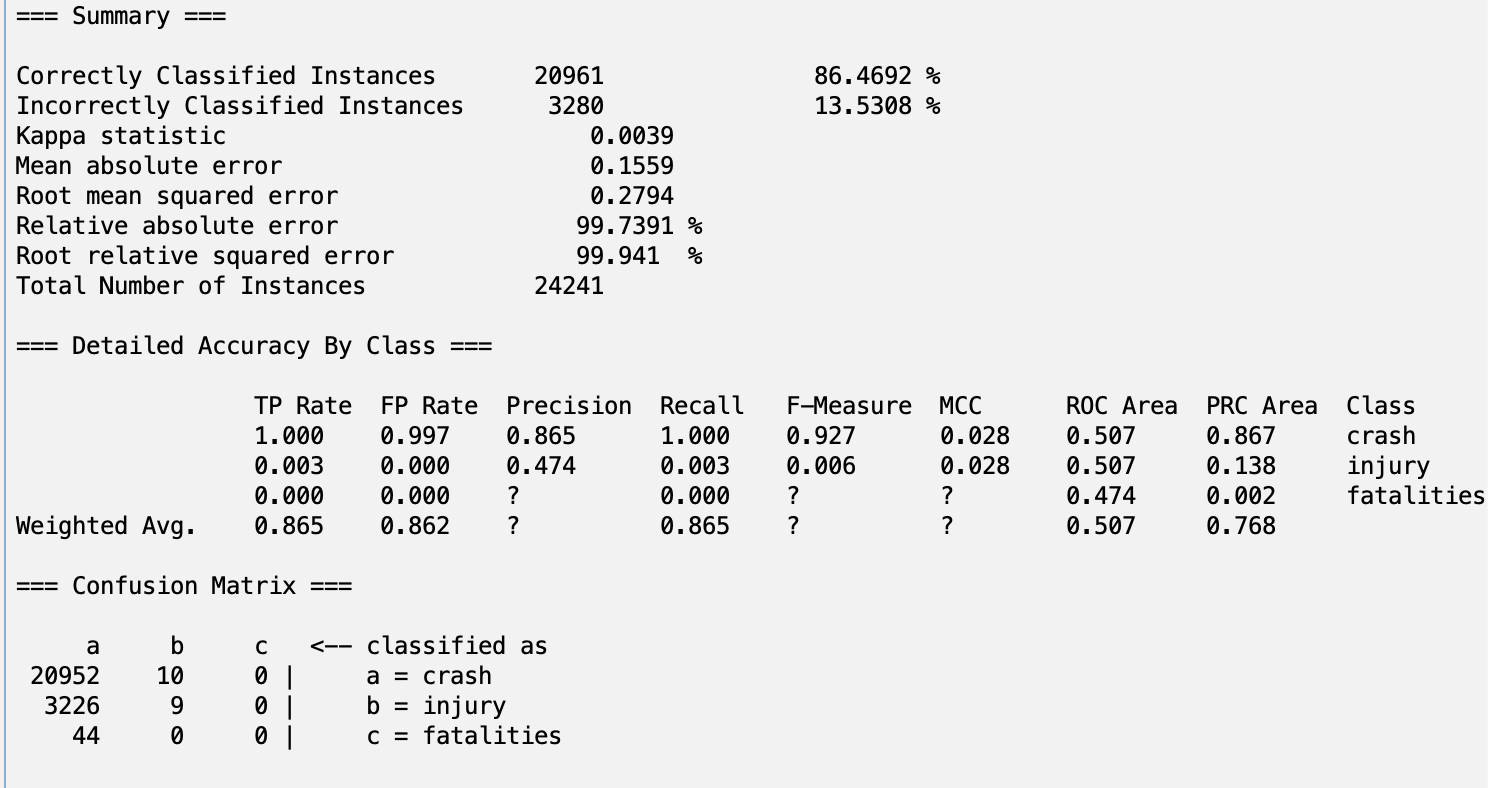
**Non-Weka with OneR:**

****

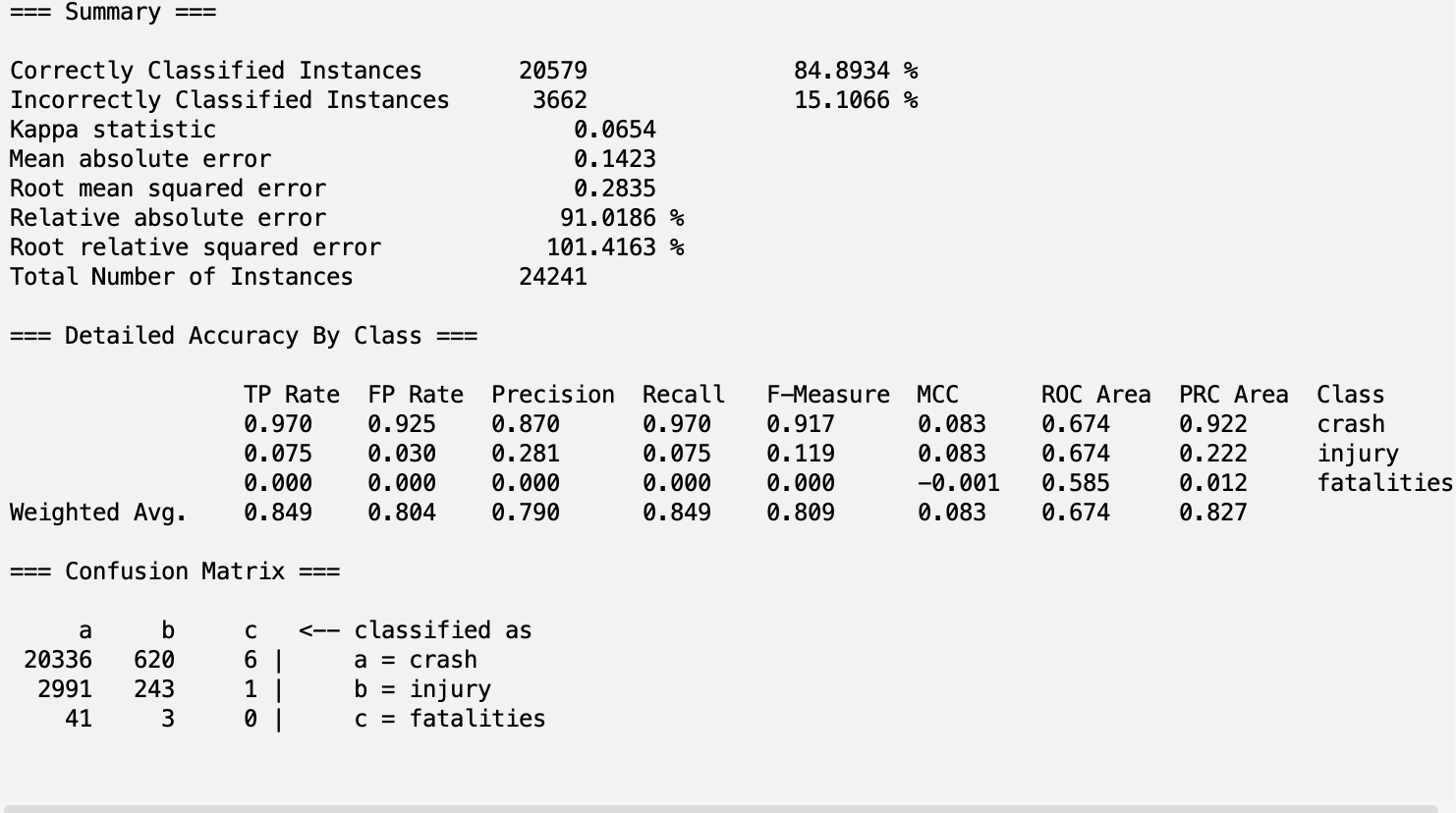
**CorrelationAttributeEval with Naive Bayes:**



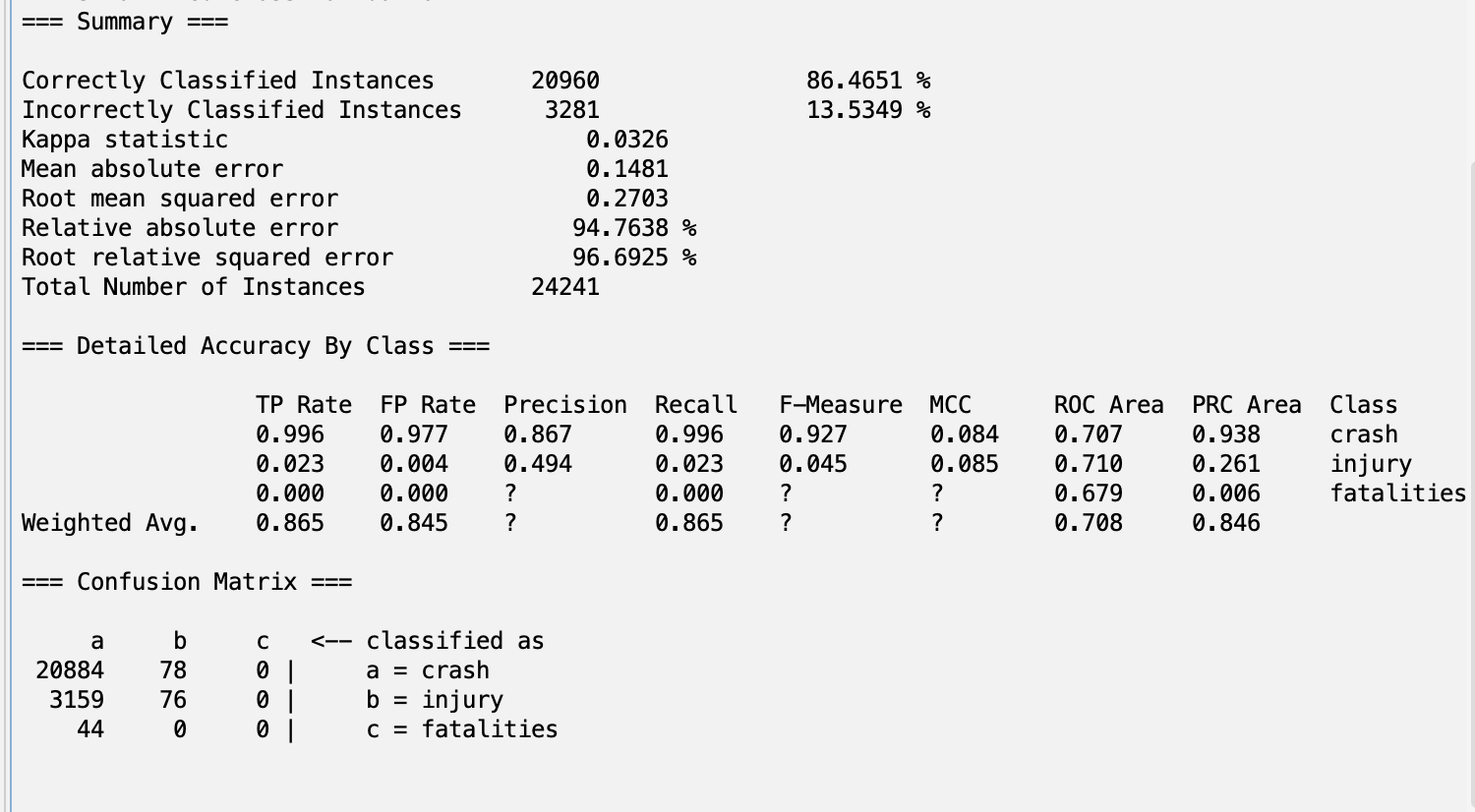
**CorrelationAttributeEval with J48**



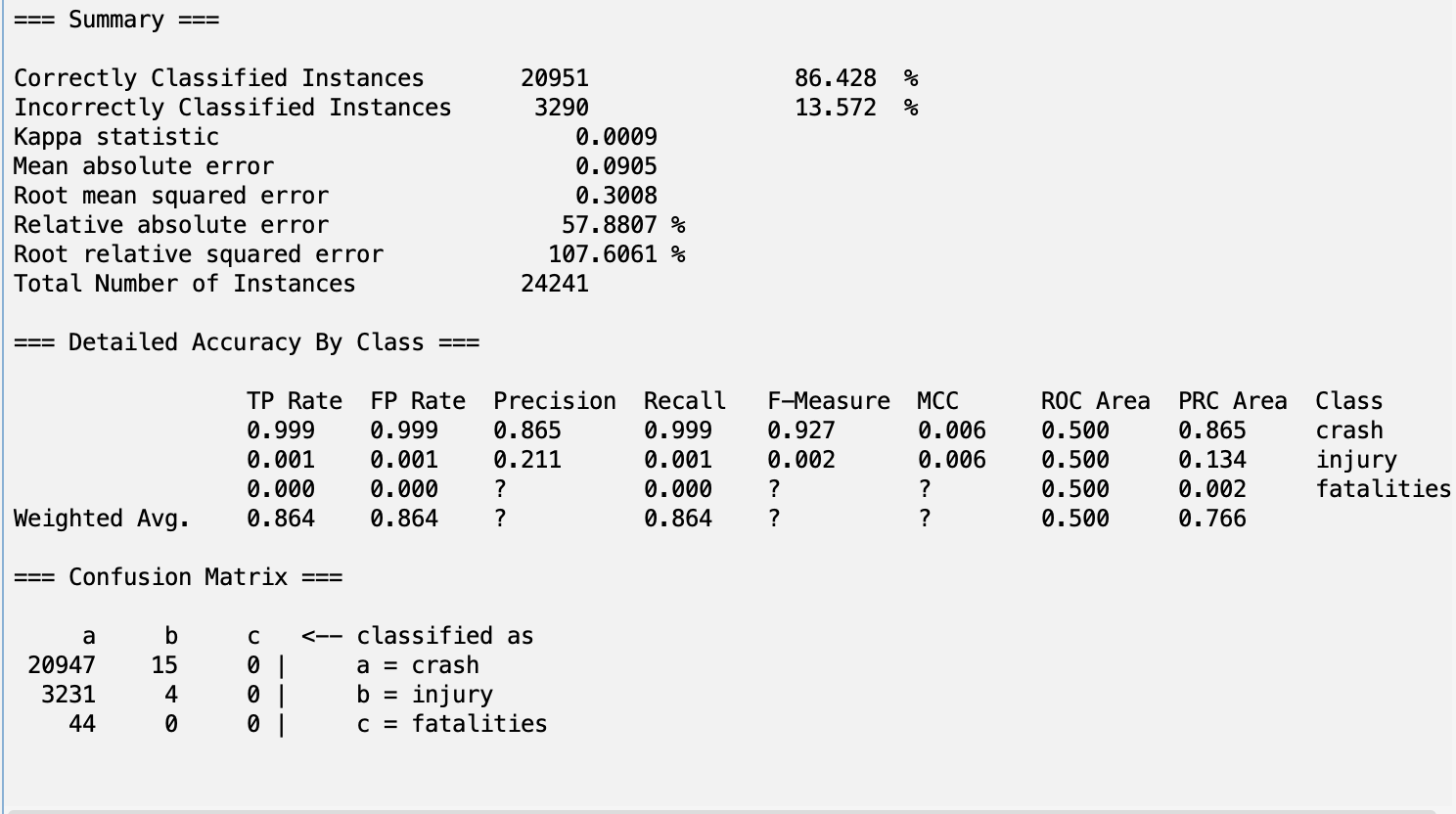
**CorrelationAttributeEval with RandomForest**



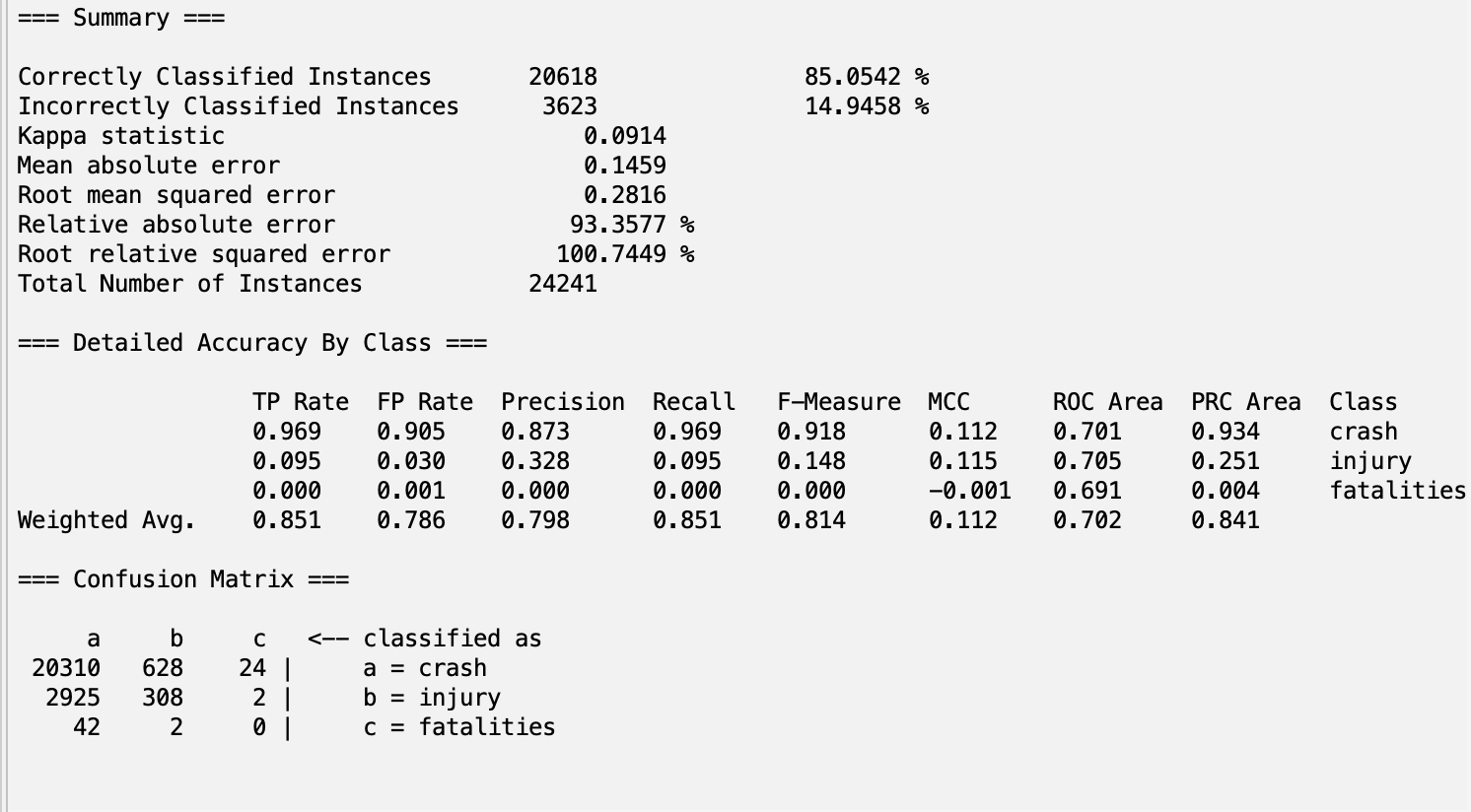
**CorrelationAttributeEval with DecisionTable**



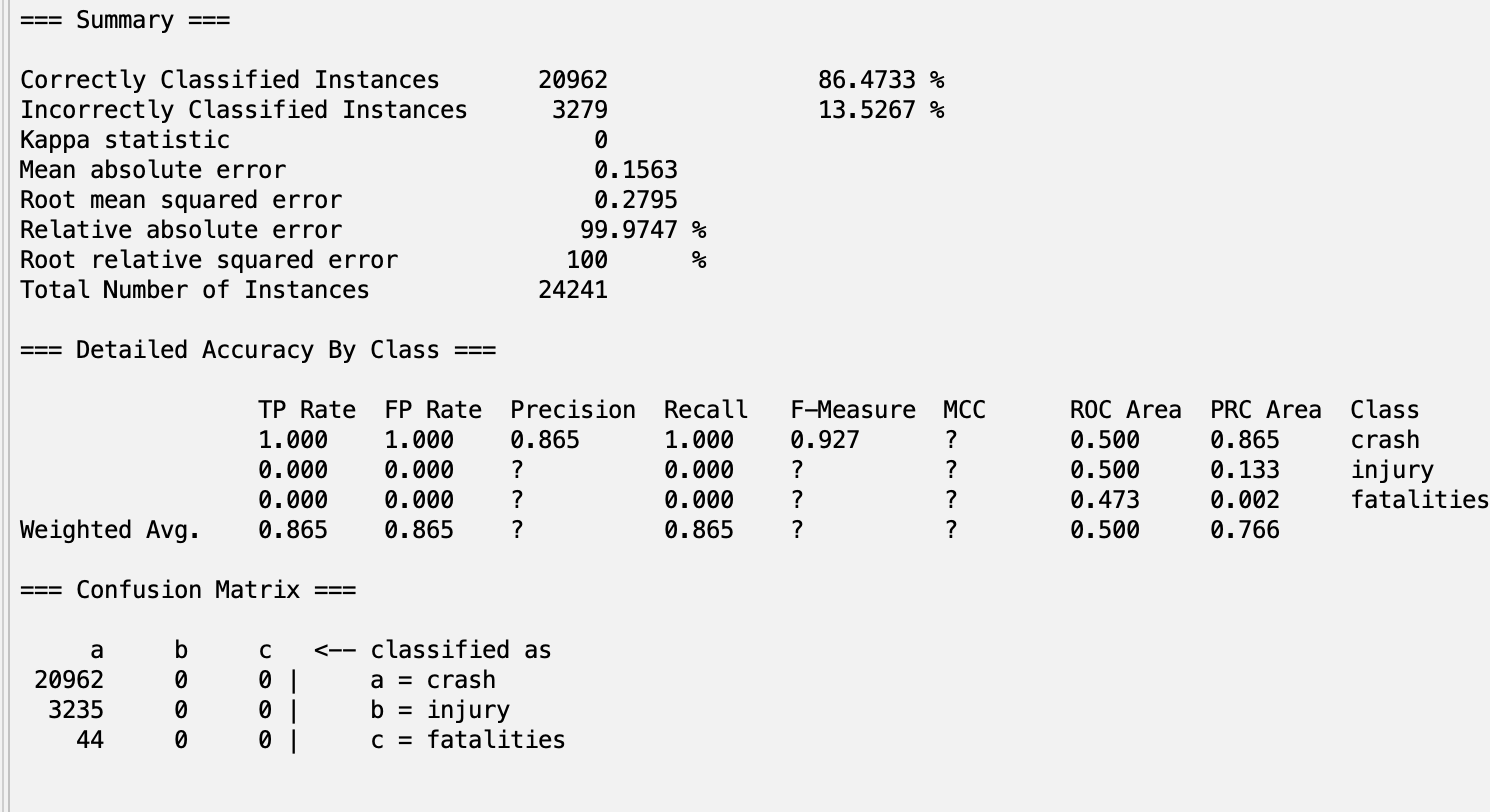
**CorrelationAttributeEval with OneR**



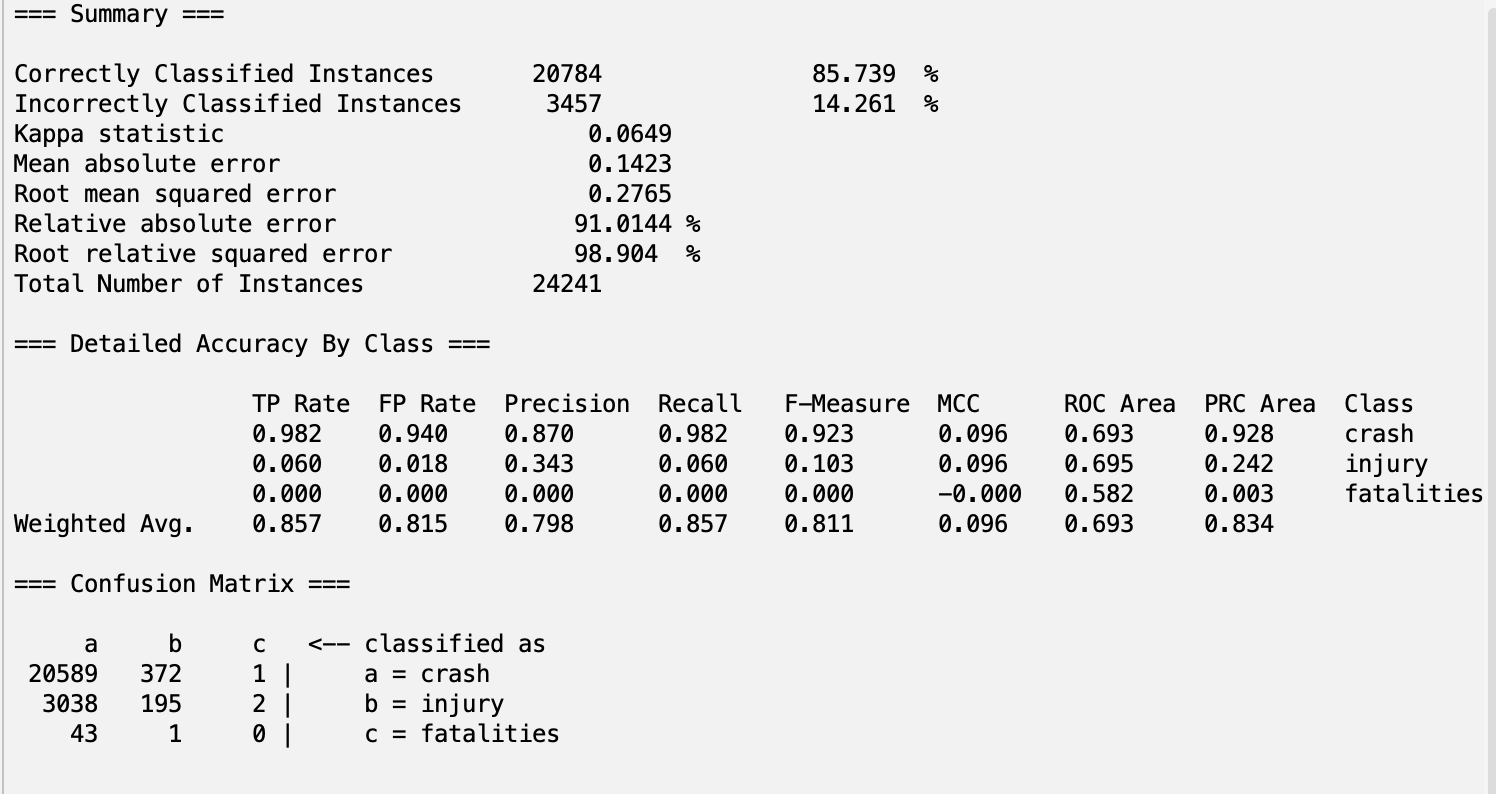
**InfoGainAttributeEval with Naive Bayes:**



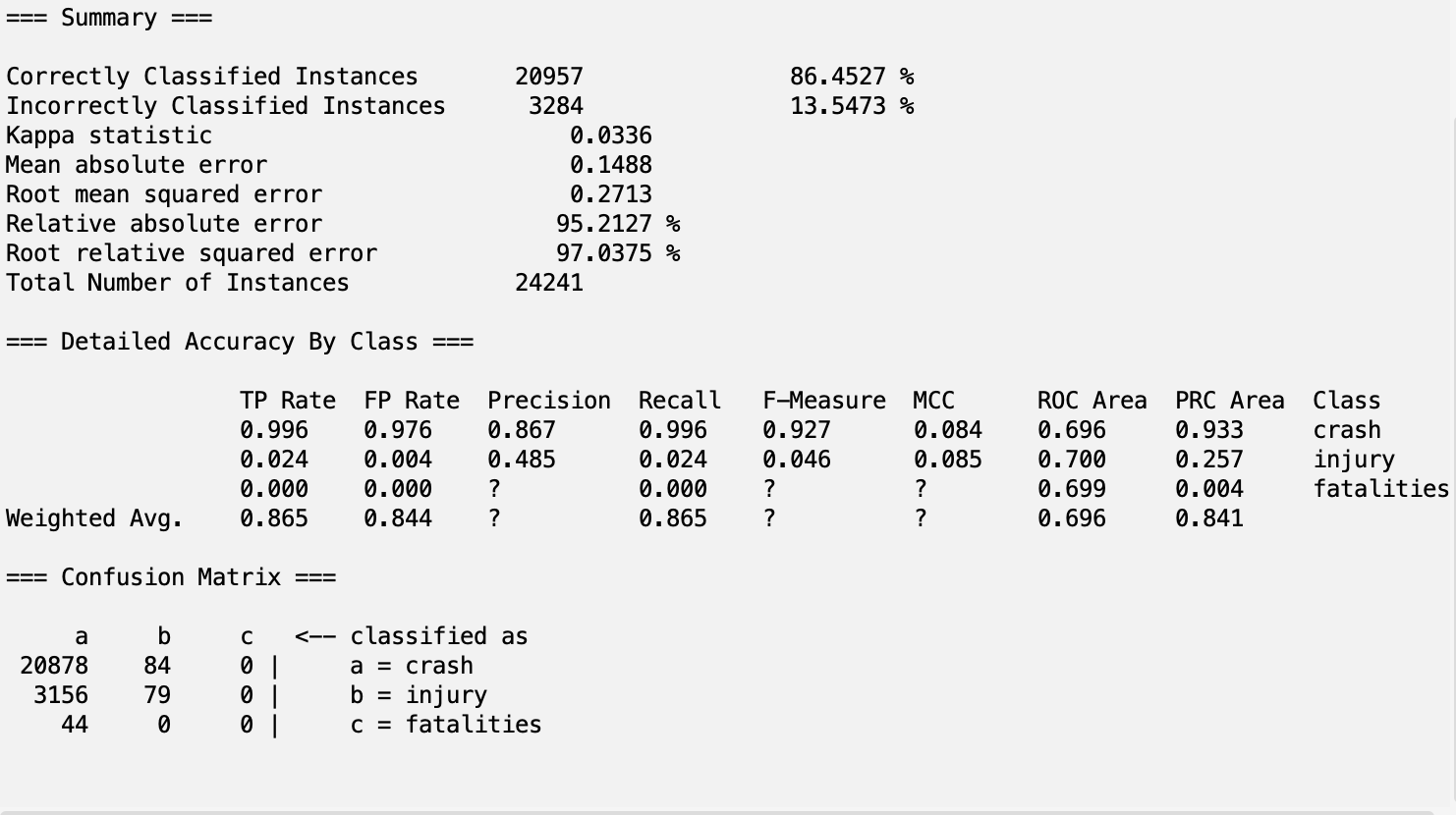
**InfoGainAttributeEval with J48:**

****

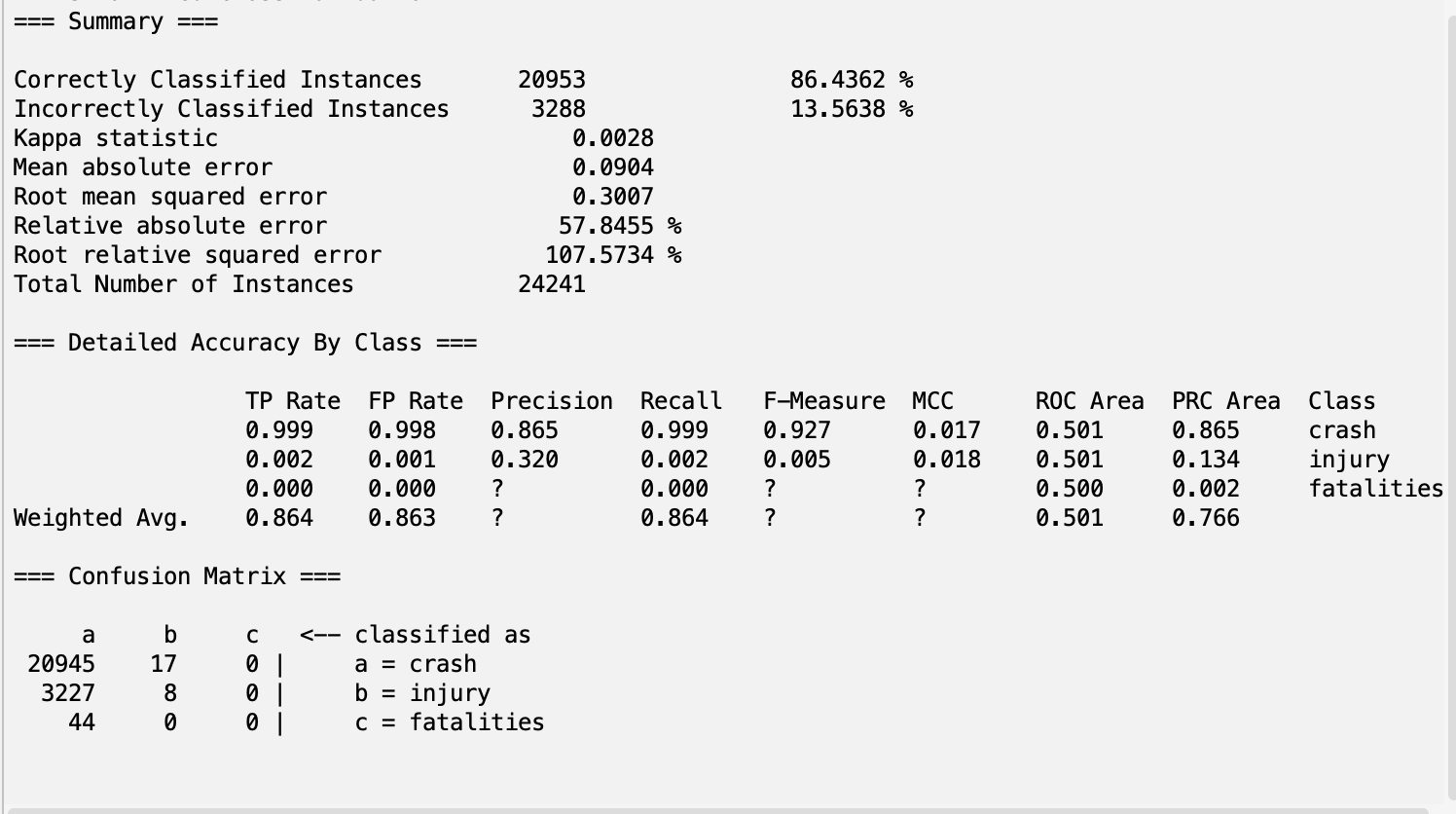
**InfoGainAttributeEval with RandomForest:**

****

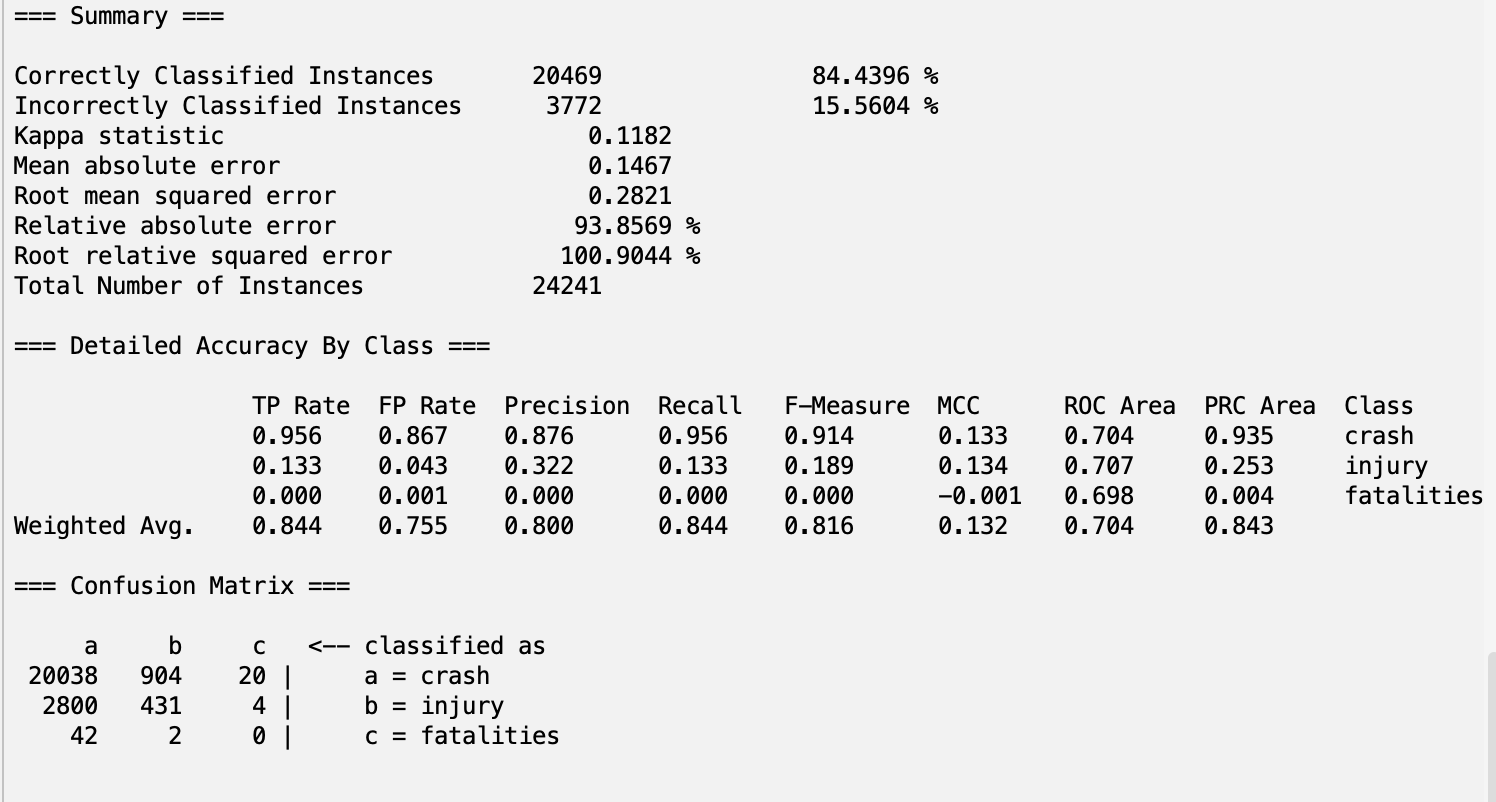
**InfoGainAttributeEval with DecisionTable:**

****

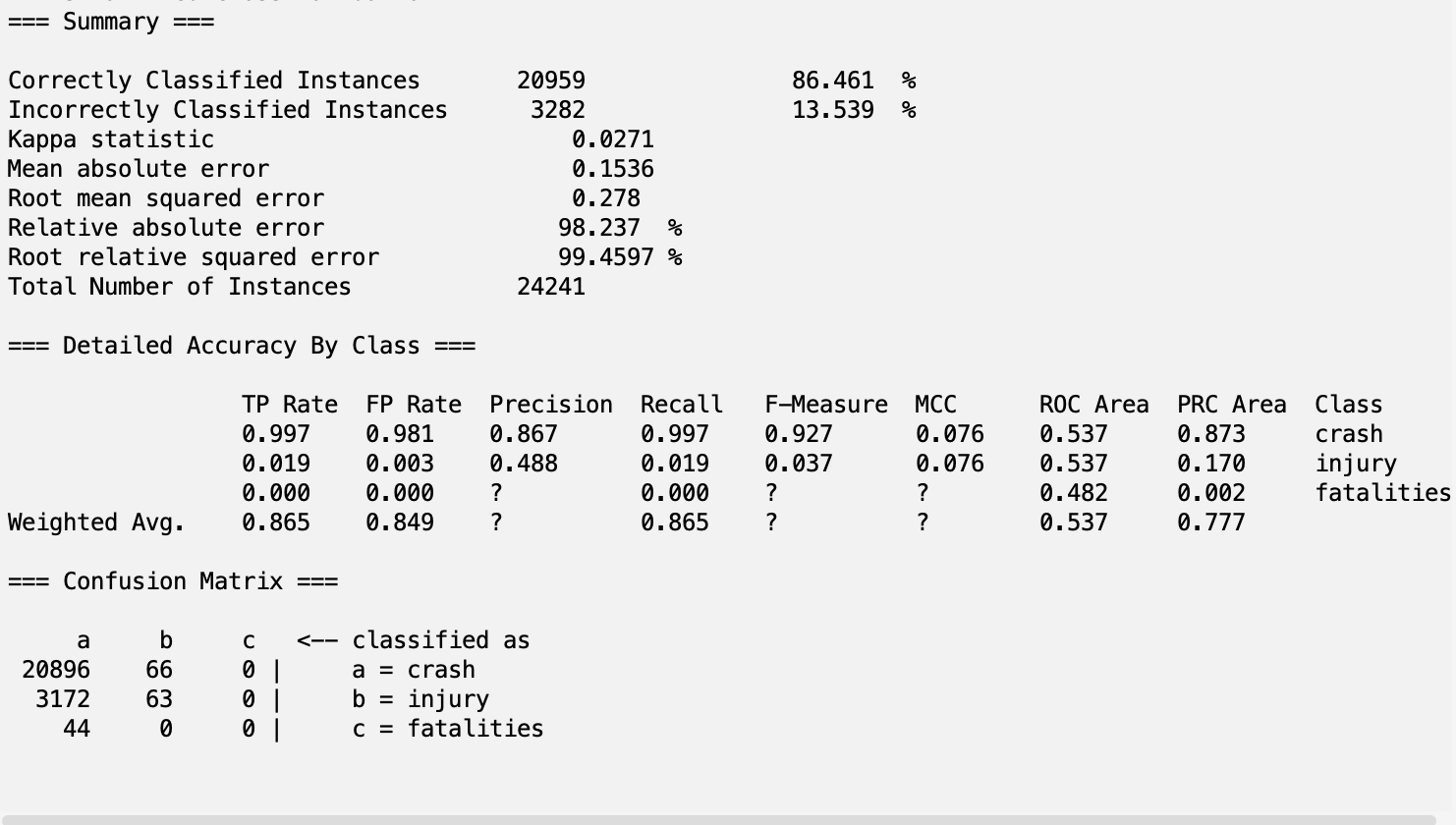
**InfoGainAttributeEval with OneR:**



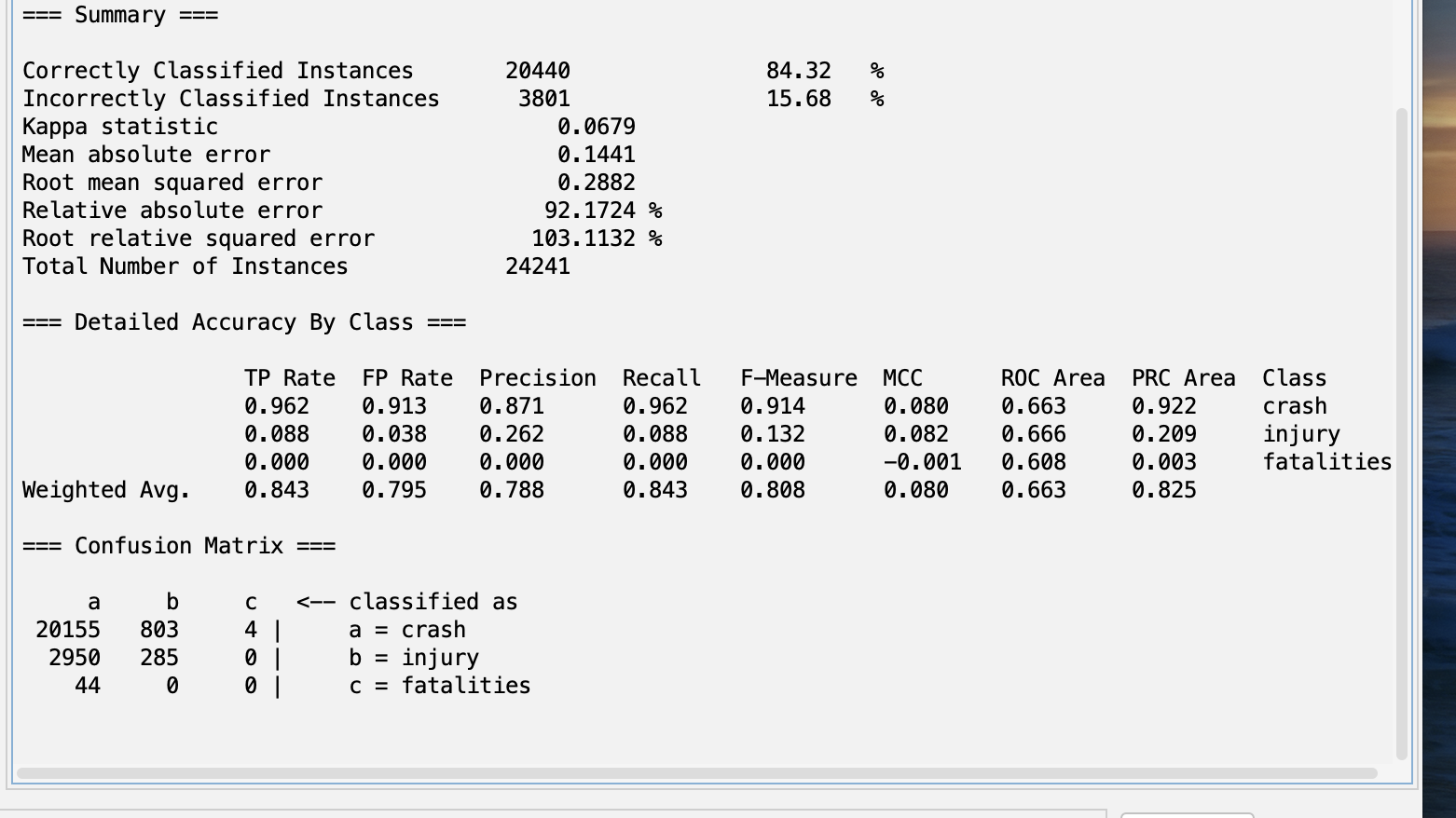
**ReliefF with Naive Bayes:**

**~~~~**

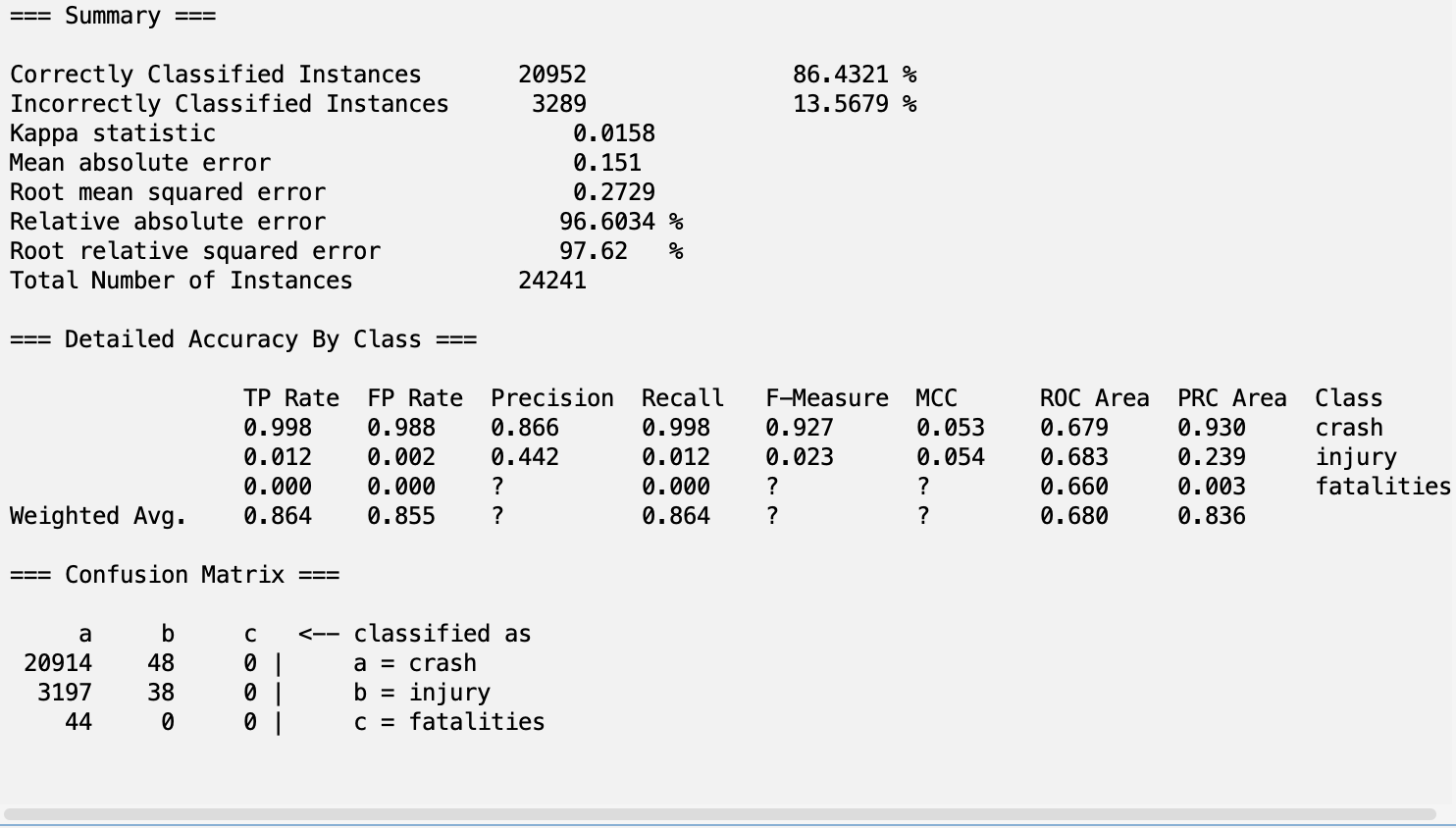
**ReliefF with J48:**

****

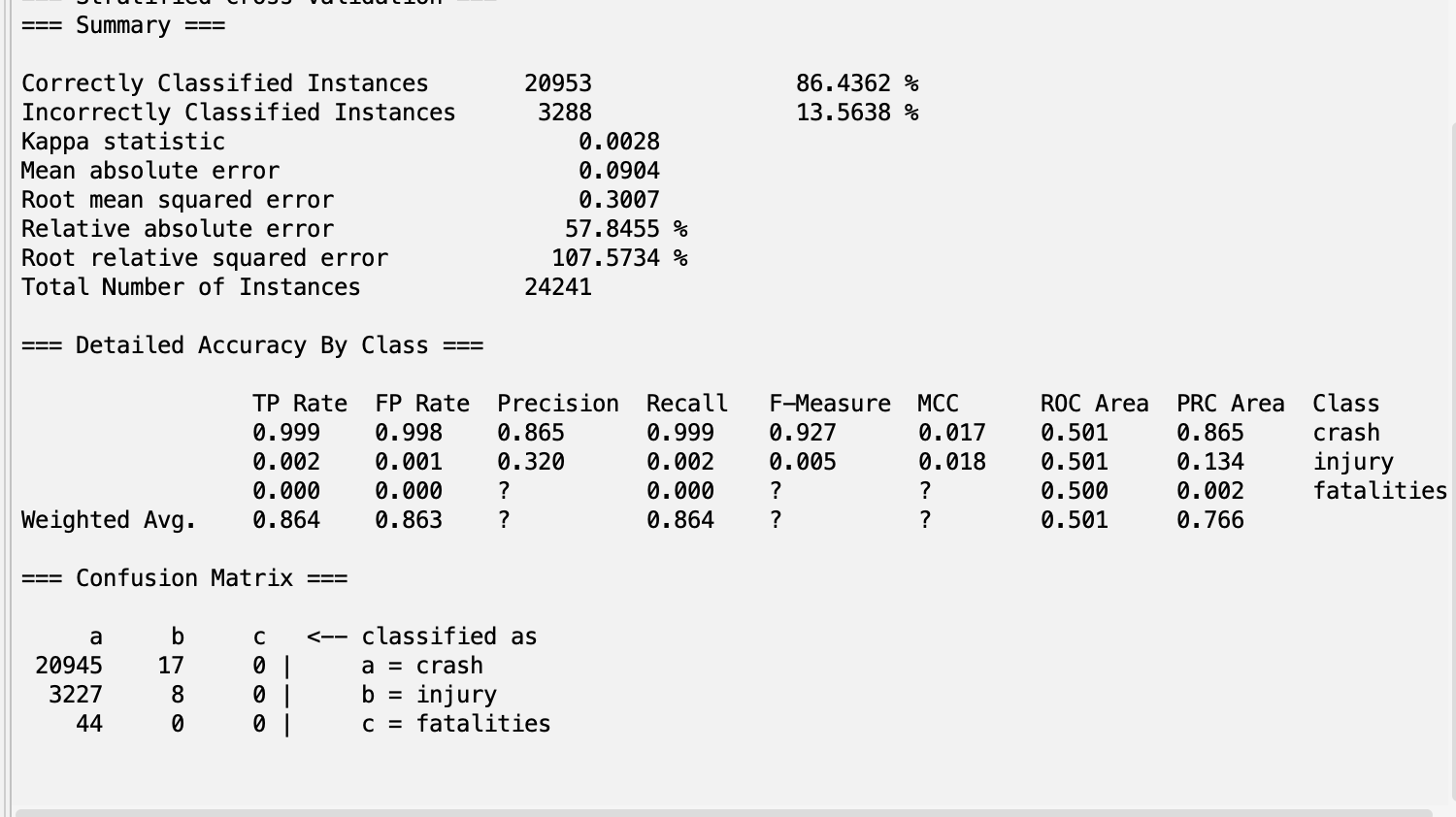
**ReliefF with RandomForest:**

****

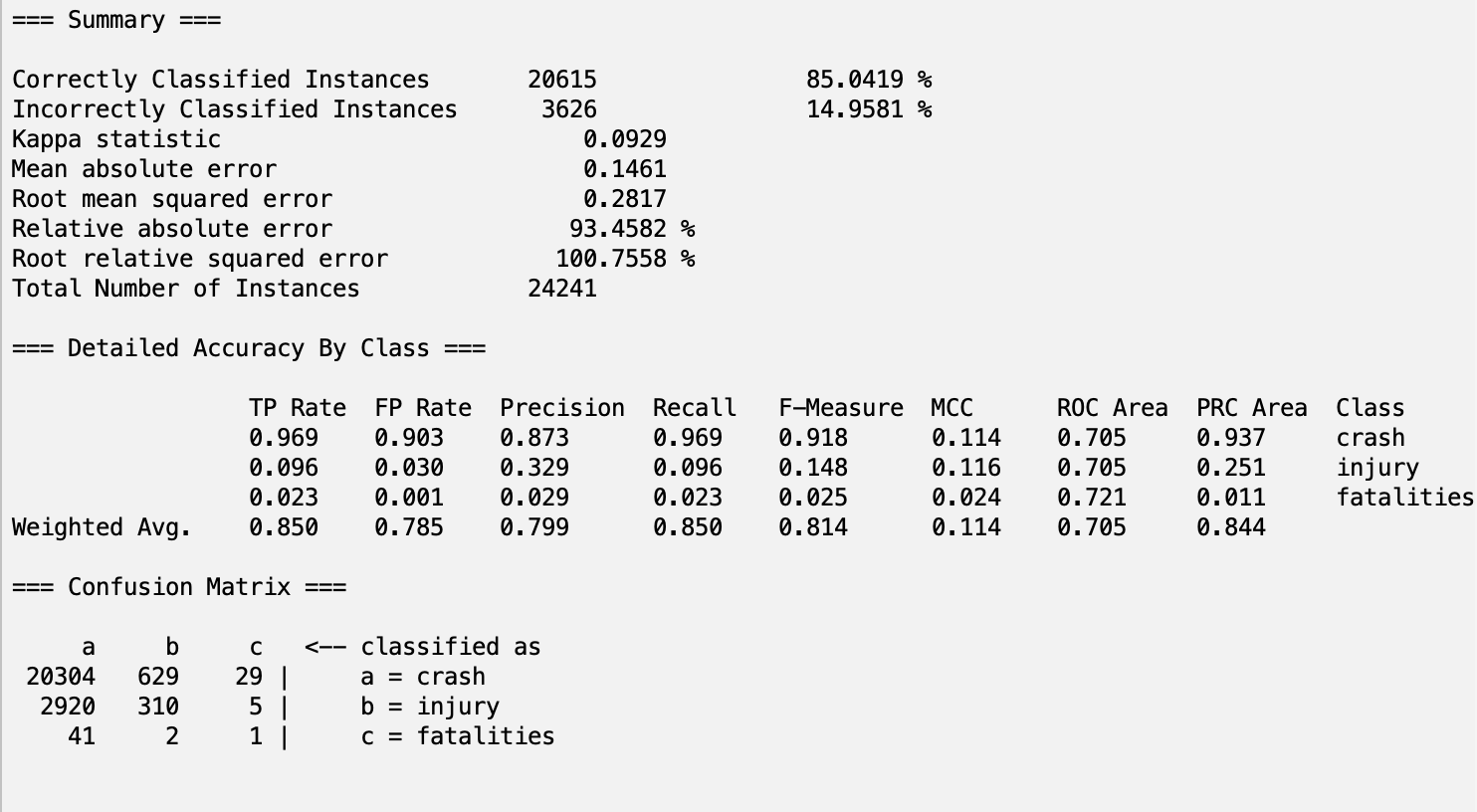
**ReliefF with DecisionTable:**

****

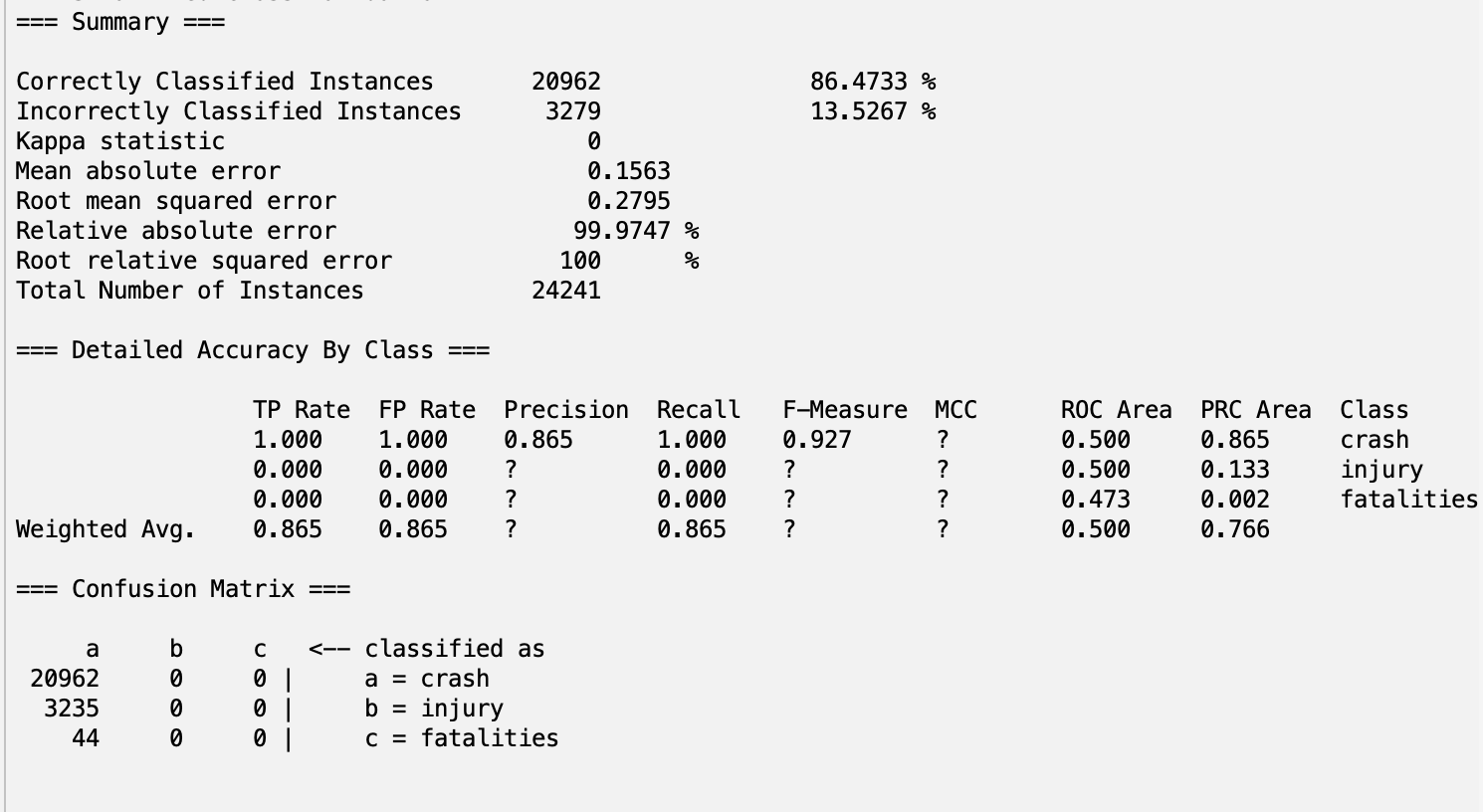
**ReliefF with OneR:**

****

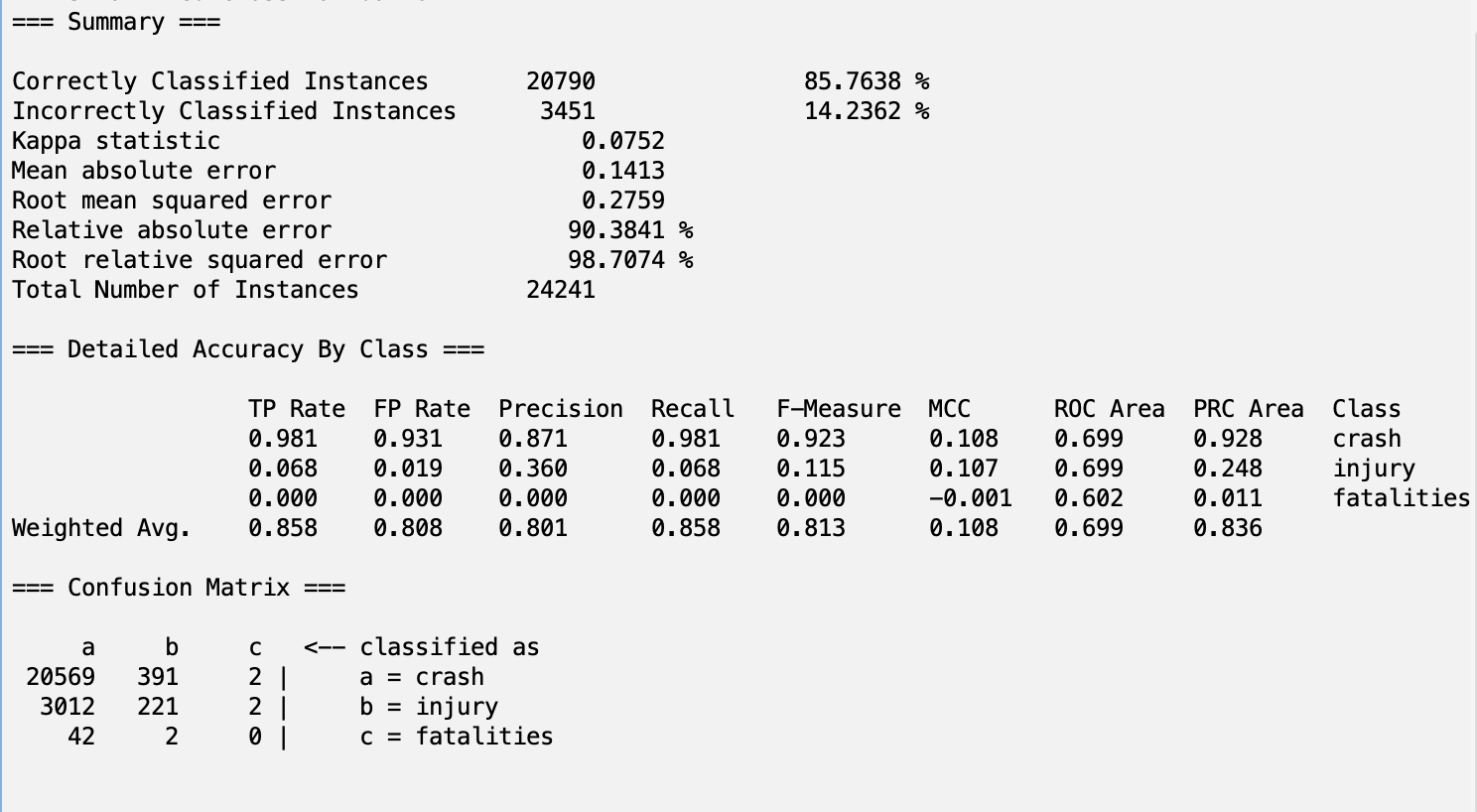
**CfsSubsetEval with Naive Bayes:**

****

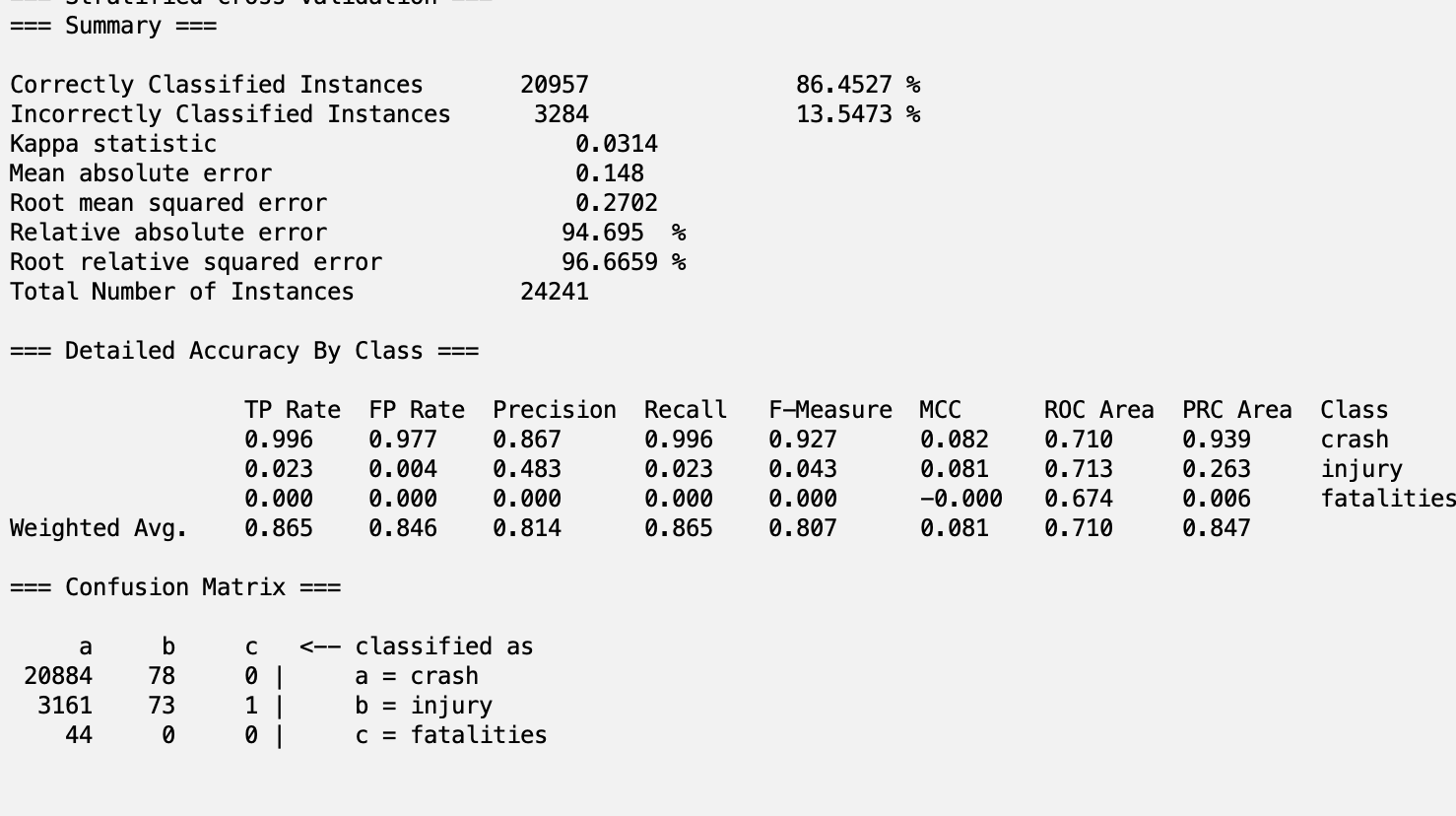
**CfsSubsetEval with J48:**

****

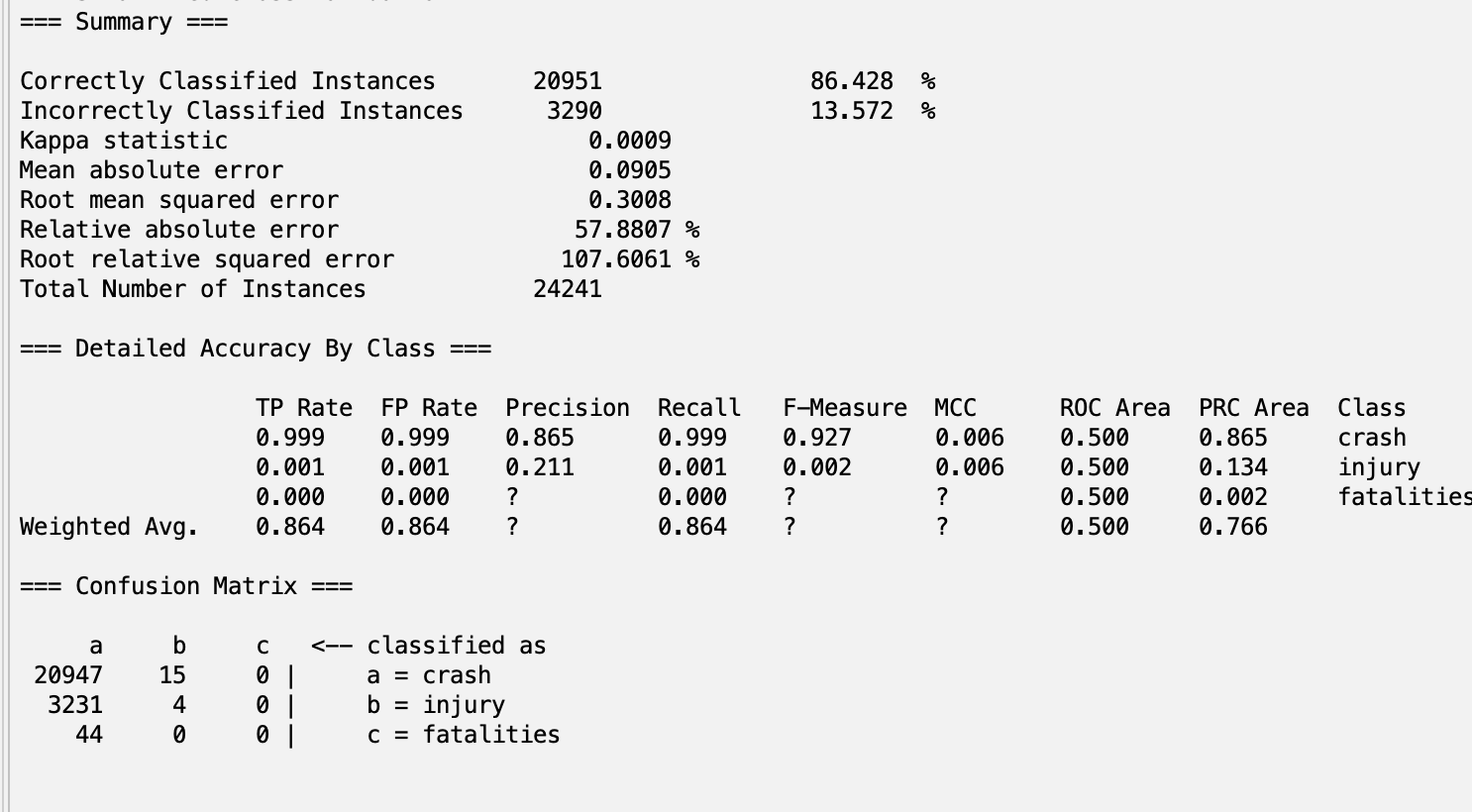
**CfsSubsetEval with RandomForest:**

****

**CfsSubsetEval with DecisionTable:**

****

**CfsSubsetEval with OneR:**



### **Part 5: Discussion and Conclusion**

**Section 5.1: Looking at Results**

Our best combination of attribute analysis and a classifier model was CfsSubsetEval with Naive Bayes, achieving accuracy of 85.0519%, TP rate of 0.850, FP Rate of 0.785, precision of 0.799, recall of 0.850, f-measure of 0.814, and MCC of 0.114. We chose this as our best model because it had the highest f-measure and MCC, which considers the parts of the confusion matrix and precision and recall. However, there were other combinations of attribute selection algorithms and a classifier that achieved better performance in the metric of accuracy. The tradeoff between accuracy and precision and recall in this scenario is largely due to how imbalanced the class labels are in the dataset. This leads to the model preferring to classify instances as “crash” since there are few “injury” and “fatalities” labels. High accuracy will come from correctly classifying the crashes as crashes, but this often results in misclassifying the instances with “injury” and “fatalities” class labels. We thought rather than being extremely biased towards the majority class and always predicting “crash”, it would be better to measure the model’s usefulness in predicting the other classes by considering non-accuracy metrics.

**Section 5.2: Future Work**

In the future, we can work on addressing the challenges posed by the imbalanced class distribution within the dataset. One promising avenue is to explore advanced sampling techniques, such as synthetic minority over-sampling or under-sampling methods, to create a more balanced dataset for training. We attempted to use WEKA’s method of balancing classes by adjusting the weights, but this resulted in the majority of the feature selection algorithms being unusable. We can try utilizing other methods online that directly artificially add or remove instances instead of changing WEKA weights.

**Section 5.3: Team Member Roles**

Finding the Data & Dataset Overview: Gabriel

Preprocessing: Gabriel

Attribute Analysis: Andrew

Classifier Models: Andrew

Discussion and Conclusion: Gabriel