**Quarter 1 Project**

*Predicting NBA All-Stars*

Gus Simanson and Lalit Boyapati

Thomas Jefferson High School for Science and Technology

Dr. Yilmaz

September 23, 2024

**Table of Contents**

Part 1 – Statement of Project Goal

Part 2 – Description of Dataset

Part 3 – Pre-Processing

Part 4 - Attribute Selection

Part 5: Train, Test, Validation Split

Part 6 - Classification Models

Part 7 - Performance Comparison/Results

Part 8 - Interesting Trends

Part 9 - Contribution of Team Members

Part 10 - Data/Sources

### 

### 

### 

### 

### 

### 

### 

### 

### **Part 1 - Statement of Project Goal**

Every February, the NBA selects 24 of their star players to play in an all-star game: Eastern Conference vs. Western Conference. These players are chosen through a vote: 50% is decided by fan vote, 25% by a current player vote, and 25% by a select group of NBA representatives. This game showcases the talent that the league has to offer and is always a spectacle to watch.

This leads us to an interesting question: can we predict which players will be named all-stars prior to the vote? Our goal is to retrieve data from the 2010-2023 seasons of all NBA players containing their individual statistics and train a model to predict whether or not a player will be designated as an all-star. We will also determine which metrics are most influential for this designation. Through this research, we will allow fans to make a more educated vote when the time comes and for players to have a better understanding on what stats they need to maximize to be in the running for one of those coveted spots.

### **Part 2 - Description of Dataset**

We gathered our data from [stats.nba](https://www.nba.com/stats/players/traditional) and [basketball-reference](https://www.basketball-reference.com/allstar/). Stats.nba provided us with the historical statistics of all NBA players, and basketball-reference.com gave us the list of named all-Stars. After scraping stats.nba for “traditional” statistics, we were able to construct a blanket dataset containing the players’ statistics, and then we manually put in whether or not they were named an all-star.

*Code to Web Scrape*:

import requests

import pandas as pd

import time

from google.colab import files

url = 'https://stats.nba.com/stats/leaguedashplayerstats'

header = {

'Host': 'stats.nba.com',

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36',

'Accept': 'application/json, text/plain, \*/\*',

'Accept-Language': 'en-US,en;q=0.9',

'Referer': 'https://www.nba.com/',

'Origin': 'https://www.nba.com',

'Connection': 'keep-alive'

}

all\_data = []

def scrape\_nba\_stats\_for\_season(season):

params = {

'College': '',

'Conference': '',

'Country': '',

'DateFrom': '',

'DateTo': '',

'Division': '',

'DraftPick': '',

'DraftYear': '',

'GameScope': '',

'GameSegment': '',

'Height': '',

'LastNGames': '0',

'LeagueID': '00',

'Location': '',

'MeasureType': 'Base',

'Month': '0',

'OpponentTeamID': '0',

'Outcome': '',

'PORound': '0',

'PaceAdjust': 'N',

'PerMode': 'PerGame',

'Period': '0',

'PlayerExperience': '',

'PlayerPosition': '',

'PlusMinus': 'N',

'Rank': 'N',

'Season': season,

'SeasonSegment': '',

'SeasonType': 'Regular Season',

'ShotClockRange': '',

'StarterBench': '',

'TeamID': '0',

'TwoWay': '0',

'VsConference': '',

'VsDivision': '',

'Weight': ''

}

response = requests.get(url, headers=header, params=params)

if response.status\_code == 200:

data = response.json()

headers = data['resultSets'][0]['headers']

rows = data['resultSets'][0]['rowSet']

df = pd.DataFrame(rows, columns=headers)

df['Season'] = season

all\_data.append(df)

print(f"Data for season {season} successfully retrieved")

else:

print(f"Failed to retrieve data for season {season}. Status code: {response.status\_code}")

seasons = [f'{year}-{str(year+1)[2:]}' for year in range(2010,2024)]

for season in seasons:

scrape\_nba\_stats\_for\_season(season)

combined\_df = pd.concat(all\_data)

combined\_csv\_name = 'nba\_player\_stats\_all\_seasons.csv'

combined\_df.to\_csv(combined\_csv\_name, index=False)

files.download(combined\_csv\_name)

Our dataset included 68 attributes, but at first glance the 27 most notable seem to be:

Age: Age of player during that season

GP: Games Played

W: Wins

L: Losses

MIN: Average minutes played a game

PTS: Average points a game

FGM: Field Goals Made per game – the number of field goals made by the player on average per game. This is any shot or tap in besides a free throw.

FGA: Field Goals Attempted - the number of field goals attempted by the player on average per game.

FG\_PCT: Field Goal Percentage - the percentage of field goals made per game by the player on average.

FG3M: Three Pointers Made - the number of three pointers made by the player on average per game.

FG3\_PCT: Three Pointer Percentage - the percentage of three pointers made by the player on average per game

FT\_PCT: Free Throw Percentage - the percentage of free throws made by the player on average per game.

OREB: Offensive Rebounds per game - the number of offensive rebounds grabbed by the player on average per game.

DREB: Defensive Rebounds per game - the number of defensive rebounds grabbed by the player on average per game.

AST: Assists - the number of assists passed by the player on average per game.

TOV: Turnovers - the number of turnovers caused by the player on average per game.

STL: Steals - the number of steals forced by the player on average per game.

BLK: Blocks - the number of shots blocked by the player on average per game.

PF: Personal Fouls - the number of personal fouls committed by the player on average per game.

NBA\_FANTASY\_POINTS: The number of fantasy points generated by the player on average per game. Fantasy points are generated using a variety of stats.

DD2: Double-Doubles - the number of games in which a player achieves double digits in two statistical categories

TD3: Triple-Doubles - the number of games in which a player achieves double digits in three statistical categories

PLUS\_MINUS: The point differential when a player is on the court.

And our class is:

All\_Star\_Selection: Whether or not player was named All-Star

These are not necessarily the attributes we will end up using, most likely the list will become much shorter. We will preprocess and undergo attribute selection to craft the strongest possible dataset to build the model from.

The dimension of our dataset is 68 with a total of 7190 instances. Since we scraped our data directly from stats.nba.com there are little to no missing values in the dataset. Our data is very heavily skewed as there are only 24 all-star players per year, while everyone else is a non-all-star. Therefore, the total 7190 instances only contain 538 all-star selections.

### **Part 3 - Pre-Processing**

*3.1 Missing Attributes:*

As mentioned previously, we will not have to fill in missing values as the data was gathered from a comprehensive source.

*3.2 Unrelated Attributes*

We reviewed the attributes in our dataset and identified those that were irrelevant to the model. Attributes like Player ID, Player Name, Nickname, Team ID, WNBA Fantasy Rank, Season (year), and Team Abbreviation were determined to be unnecessary for predicting whether a player becomes an All-Star, even without performing attribute selection.

*3.3 Derived Attributes*

We discovered several derived attributes—such as games played, which can be calculated from wins and losses, REB (rebounds), which can be derived from adding DREB and OREB, Win PCT, which can be derived from wins divided by the total number of games, FT PCT, which can be derived from made and attempted free throws, FG PCT, which can be derived from made and attempted non-free throws, and FG3 PCT, which can be derived from made and attempted three point shots. We removed all of these from the dataset.

*3.4 Normalization*

Many of the attributes contain vastly different ranges of values, as the rank attributes can be in the 100’s while blocks and steals typically stayed smaller. There were not many outliers within the data, so we decided to do min-max normalization on all of the remaining attributes because all of them had unique and wide ranges.

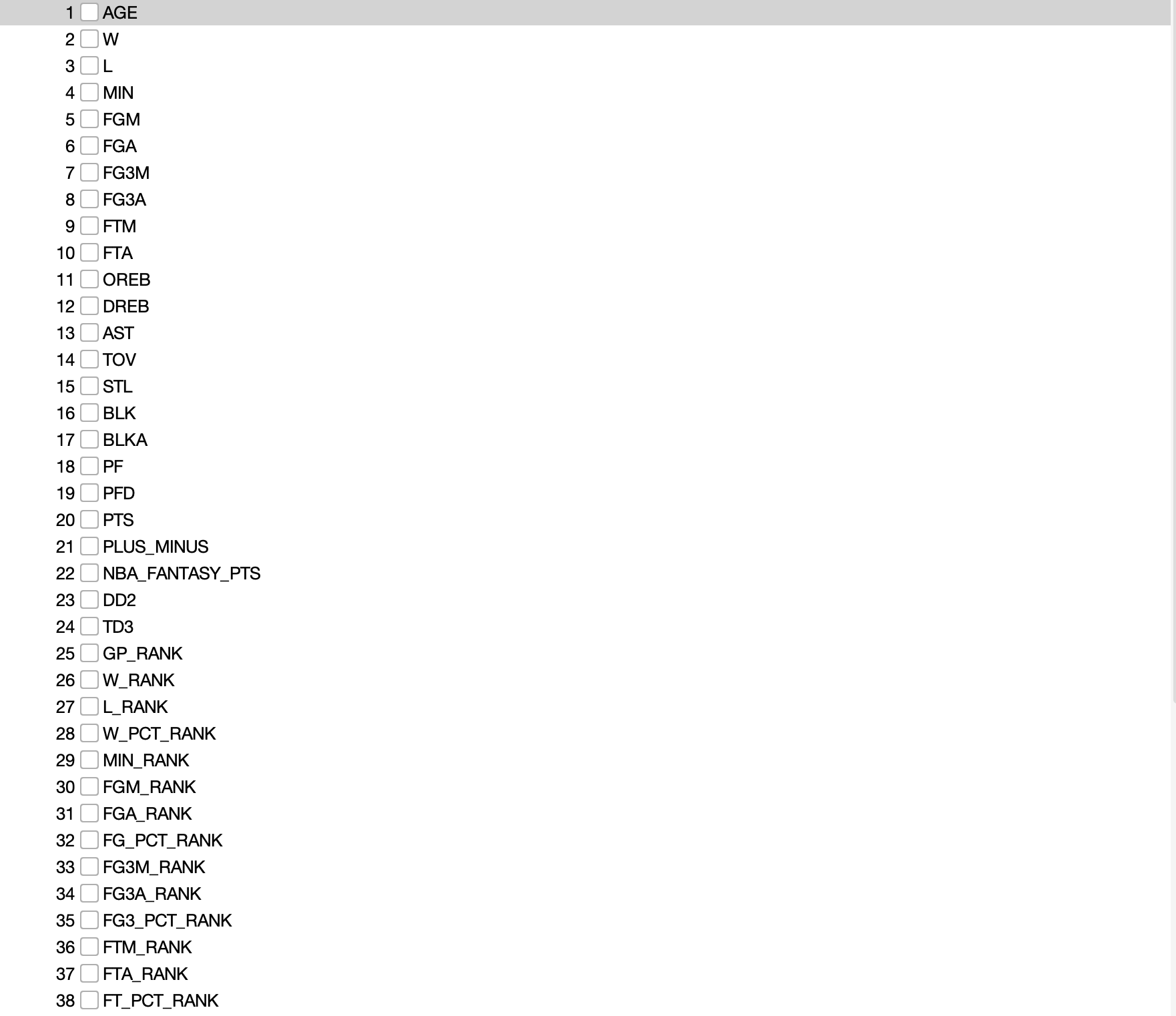
*3.5 Numeric to Nominal*

We had to switch the class attribute, All Star Selection, from numeric to nominal in order to apply attribute selection and SMOTE later on in this process.

*3.6 SMOTE*

There is a very large class imbalance in the dataset , with only 538 of the 7190 instances being all-stars (7%). To rectify this, we applied the Synthetic Minority Oversampling Technique (SMOTE, which resamples a dataset to add more instances of the minority class. SMOTE works through picking a random instance from the minority class, identifying a user specified number of nearest neighbors (typically five), and then for one randomly selected neighbor, it takes the difference between the instance and that neighbor, multiplies the difference by a random number between zero and one, and adds that difference to the instance. This provides you with a synthetic instance in your minority class. We ran SMOTE twice on the data, and achieved a more balanced ratio of 2152 all-stars to 8804 instances (24%).

3.7 Remaining Attributes



### **Part 4 - Attribute Selection**

After importing the preprocessed dataset into Weka, we decided to run the CorrelationAttributeEval, CfsSubsetEval, InfoGainAttributeEval, and OneRAttributeEval. Weka’s output for each is displayed below. We also created our own dataset, with attributes we felt to be the most appropriate.

*4.1 CorrelationAttributeEval*

CoorelationAttributeEval is one of the attribute selection algorithms we chose to run on our dataset. This calculates the Pearson correlation coefficient between each feature and the class. This then ranks the features in a listed manner as shown below in the screenshot: 

By setting out correlation value cutoff at 0.02 we can see the relevant attributes based on CorrelationAttributeEval are [AGE, PLUS\_MINUS, PLUS\_MINUS\_RANK, W\_RANK, BLK\_RANK, FTA\_RANK, W, W\_PCT\_RANK, FT\_PCT\_RANK, OREB\_RANK, STL\_RANK, DREB\_RNK]

*4.2 CfsSubsetEval*

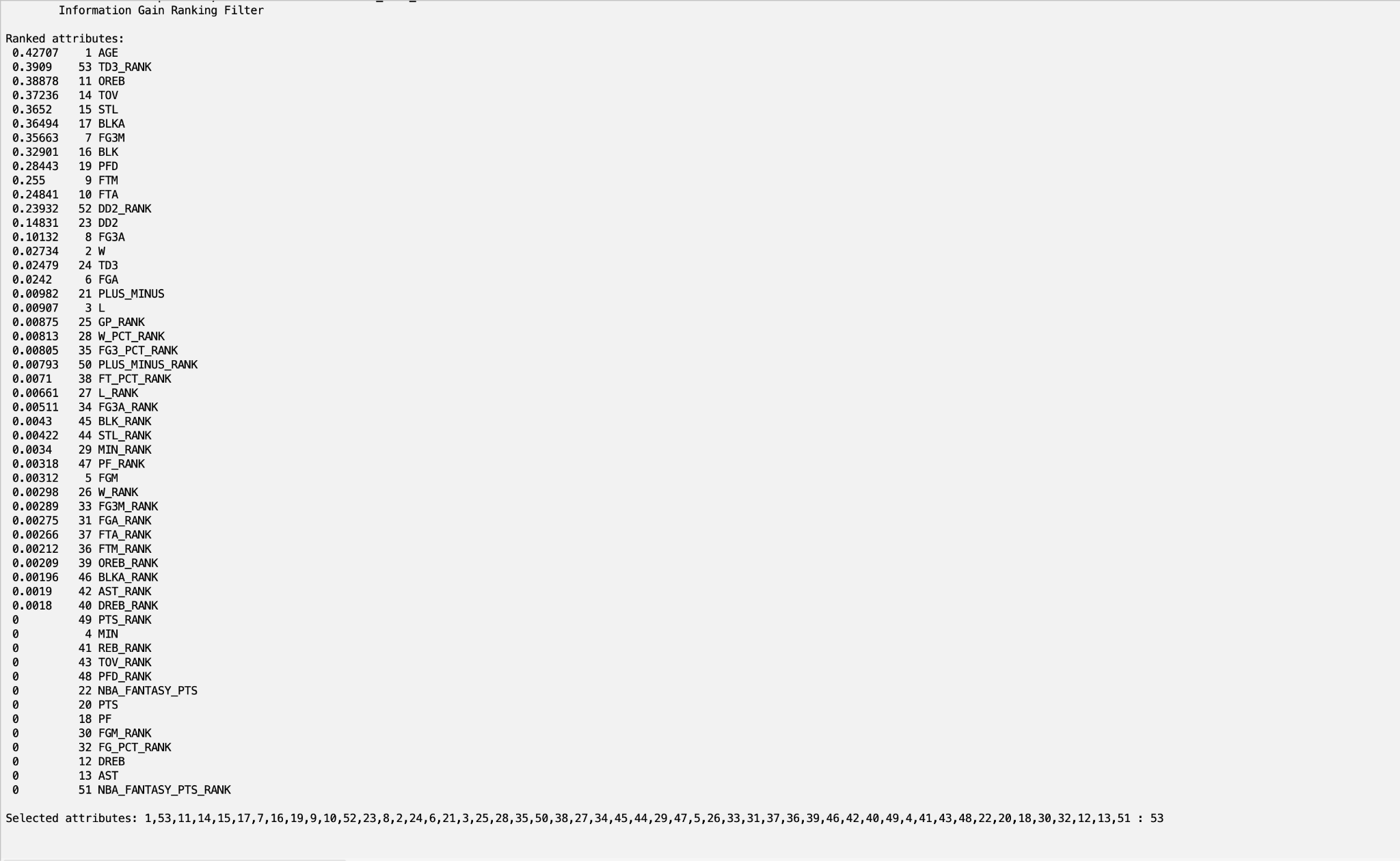
CfsSubsetEval is one of the attribute selection algorithms we chose to run on our dataset. CfsSubsetEval works by evaluating the degree of redundancy among features associated with the class. It selects features that are highly correlated with the target variable but not correlated with each other.



This created a dataset with [AGE, FG3M, OREB, TOV, STL, BLK, BLKA, PLUS\_MINUS, TD3\_RANK] as the attributes.

*4.3 InfoGainAttributeEval*

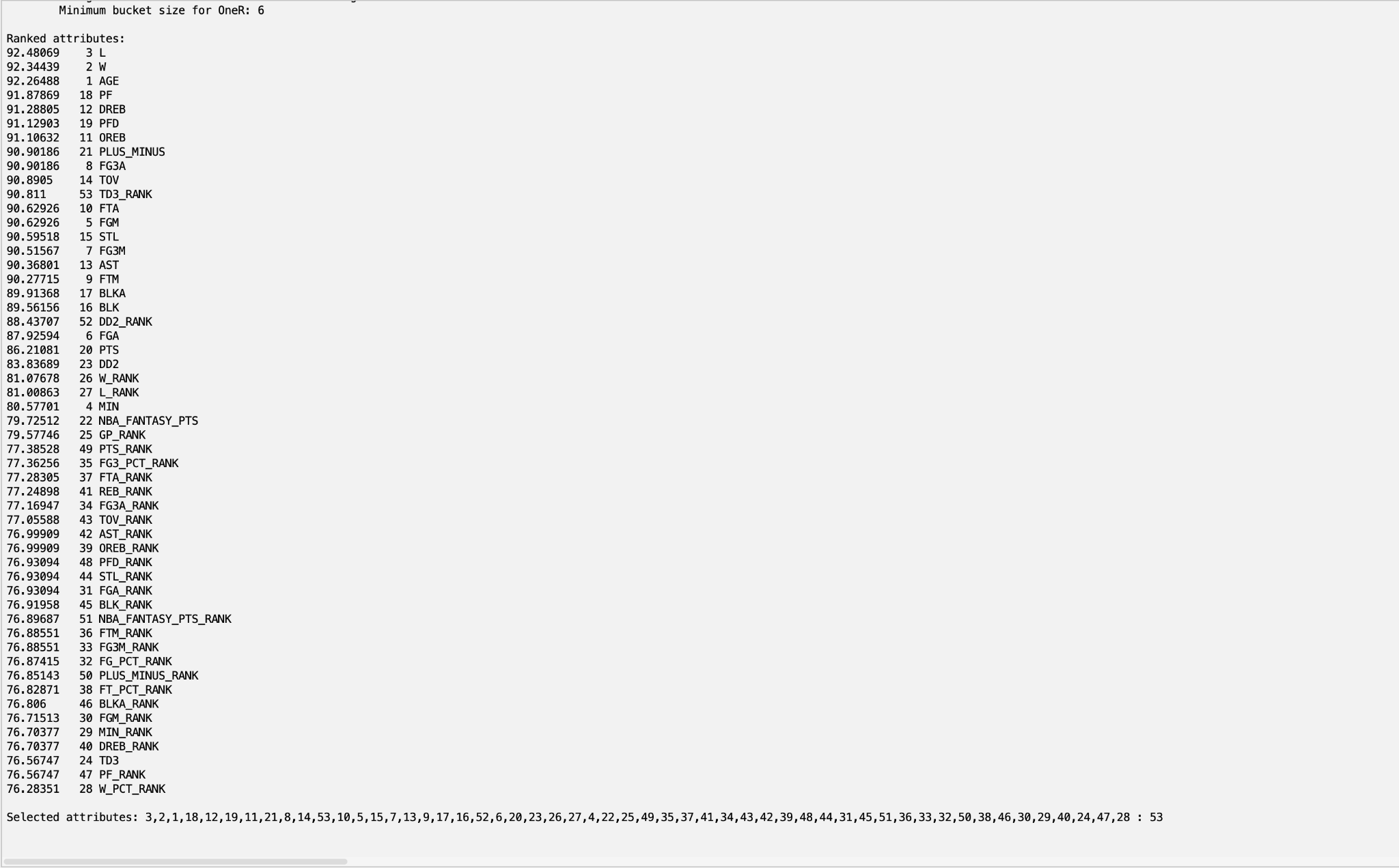
InfoGainAttributeEval determines how well a given attribute separates the training examples according to their class labels. The higher the ranking means the attribute is more informative for classification.



By setting the cutoff value to 0.2, we were left with a dataset containing the attribute set of [AGE, TD3\_RANK, OREB, TOV, STL, BLKA, FG3M, BLK, PFD, FTM, FTA, DD2\_RANK]

*4.4 OneRAttributeEval*

OneRAttributeEval in WEKA evaluates attributes by creating one-rule classifiers based on each attribute and measuring their classification error rates. It selects the attribute that produces the lowest error rate.



We decided to use a cutoff value of 90.8, leaving us with an attribute set of [L, W, AGE, PF, DREB, PFD, OREB, PLUS\_MINUS, FG3A, TOV, TD3\_RANK].

*4.5 Set chosen by us*

Based on our background knowledge about the NBA and player production, we chose a dataset with **[**W, FGM, PLUS\_MINUS, PLUS\_MINUS\_RANK, BLK, STL, AST, FG3M, and MIN] as the attributes because in our experience as avid basketball fans we believe stellar performance in these attributes to be key in order to produce an elite basketball player.

### **Part 5 - Train, Test, Validation Split**

After attribute selection, we split the data into a train, test, validation split using the code below. This split was 80% train, 10% test, and 10% validation.

*Code to split:*

from google.colab import files

import pandas as pd

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

uploaded = files.upload()

file\_name = next(iter(uploaded))

data = pd.read\_csv(file\_name)

train\_data, temp\_data = train\_test\_split(data, test\_size=0.20, random\_state=42)

val\_data, test\_data = train\_test\_split(temp\_data, test\_size=0.50, random\_state=42)

print(f"Training set: {train\_data.shape}")

print(f"Validation set: {val\_data.shape}")

print(f"Test set: {test\_data.shape}")

train\_data.to\_csv('train\_data.csv', index=False)

val\_data.to\_csv('val\_data.csv', index=False)

test\_data.to\_csv('test\_data.csv', index=False)

files.download('train\_data.csv')

files.download('val\_data.csv')

files.download('test\_data.csv')

### **Part 6 - Classification Models**

For each of the datasets created through our attribute selection we created four classification models: NaiveBayes, J48, OneR, and Logistic.

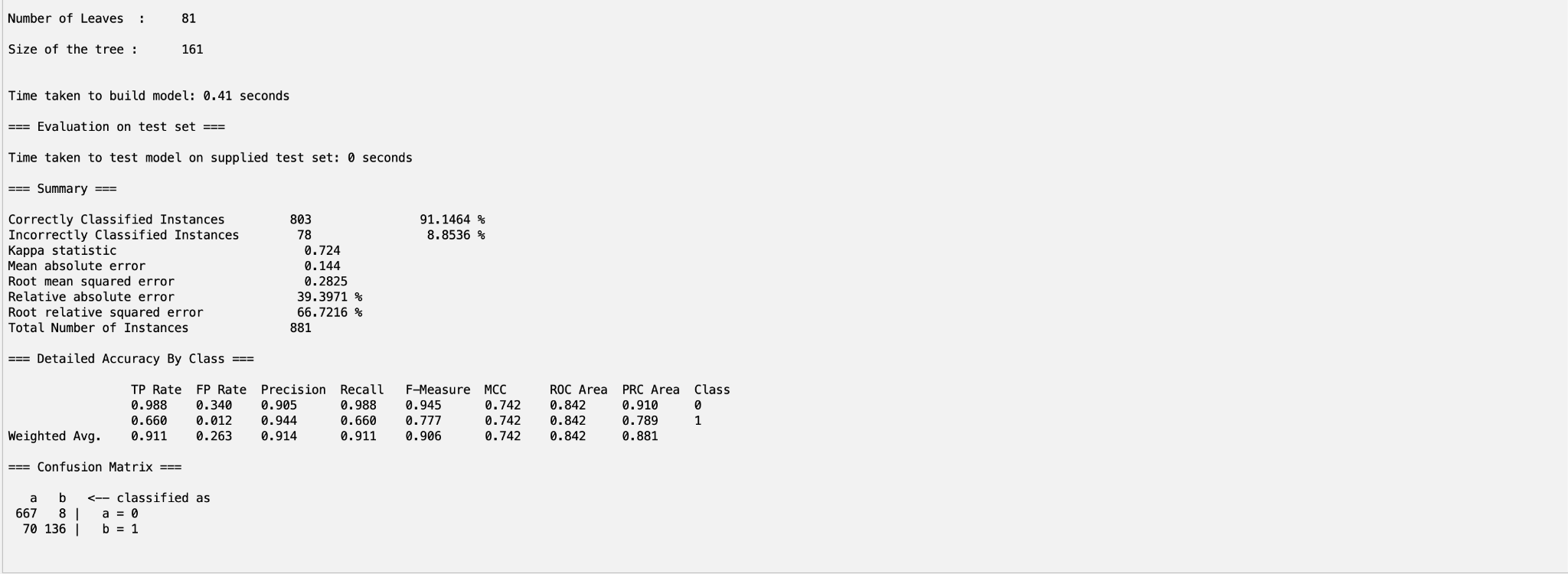
NaiveBayes is a probabilistic classifier that assumes the presence of each feature is independent of the other features.

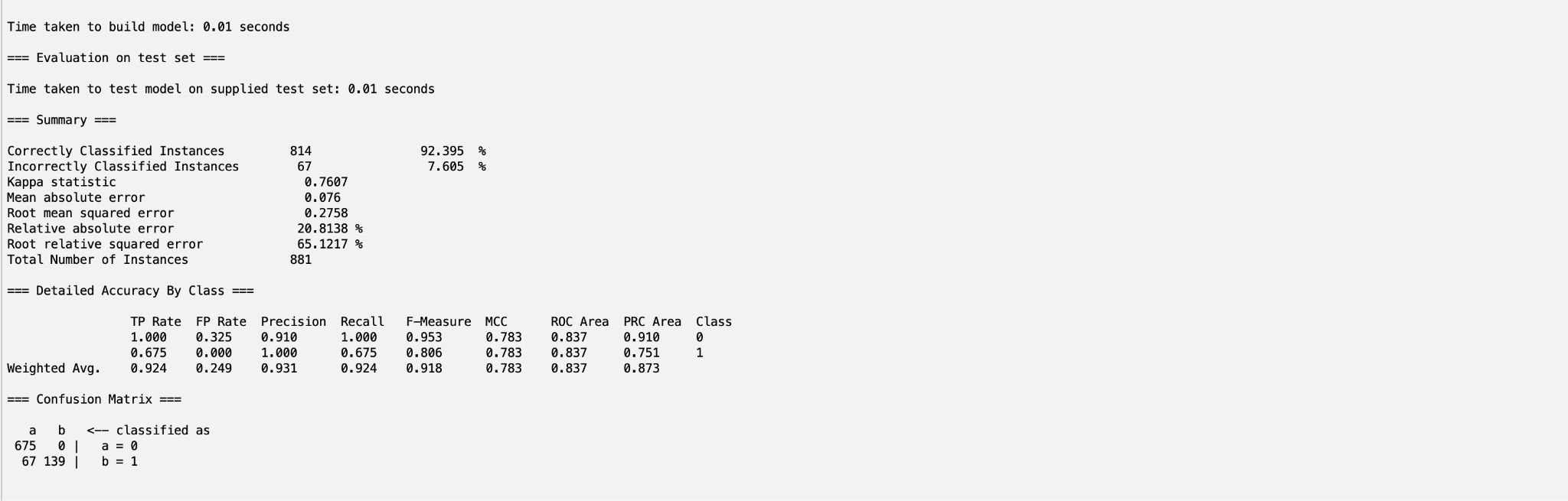
J48 is a decision tree algorithm that splits the data based on attribute values. It creates tree-structures that can handle both categorical and continuous data.

OneR is a simple, rule-based classifier that generates one rule for each predictor and selects the one that performs the best.

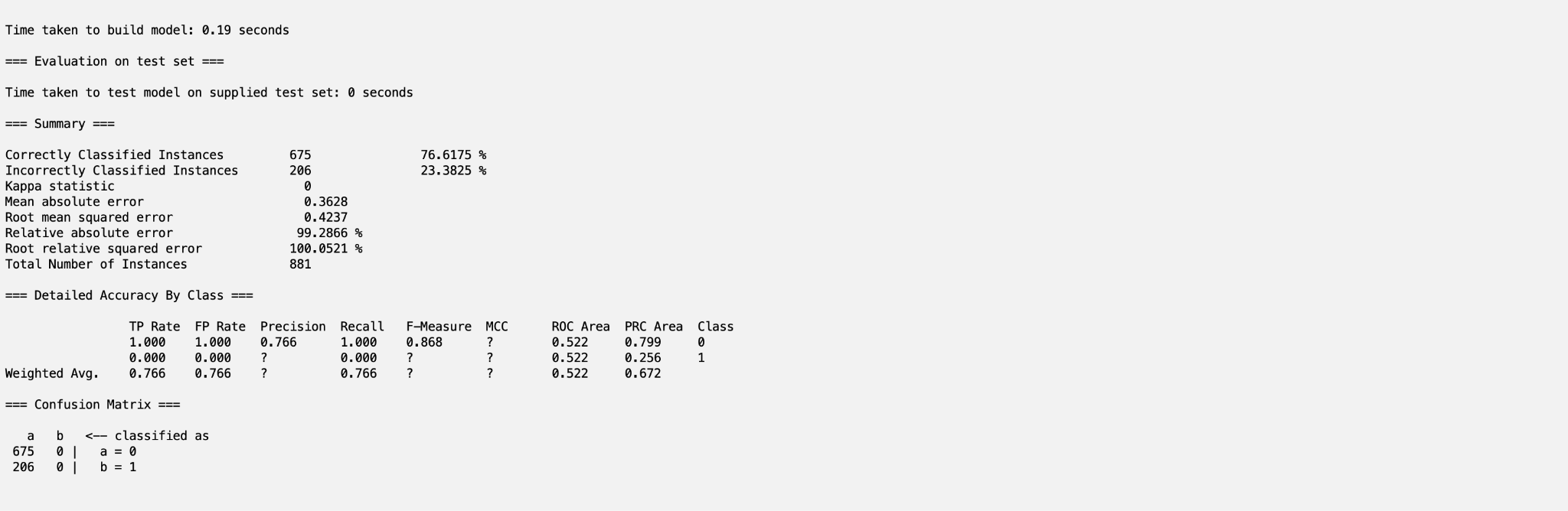
Logistic is a statistical model used for binary classification. It assumes a linear relationship between the independent variable and the log odds of the dependent variable.

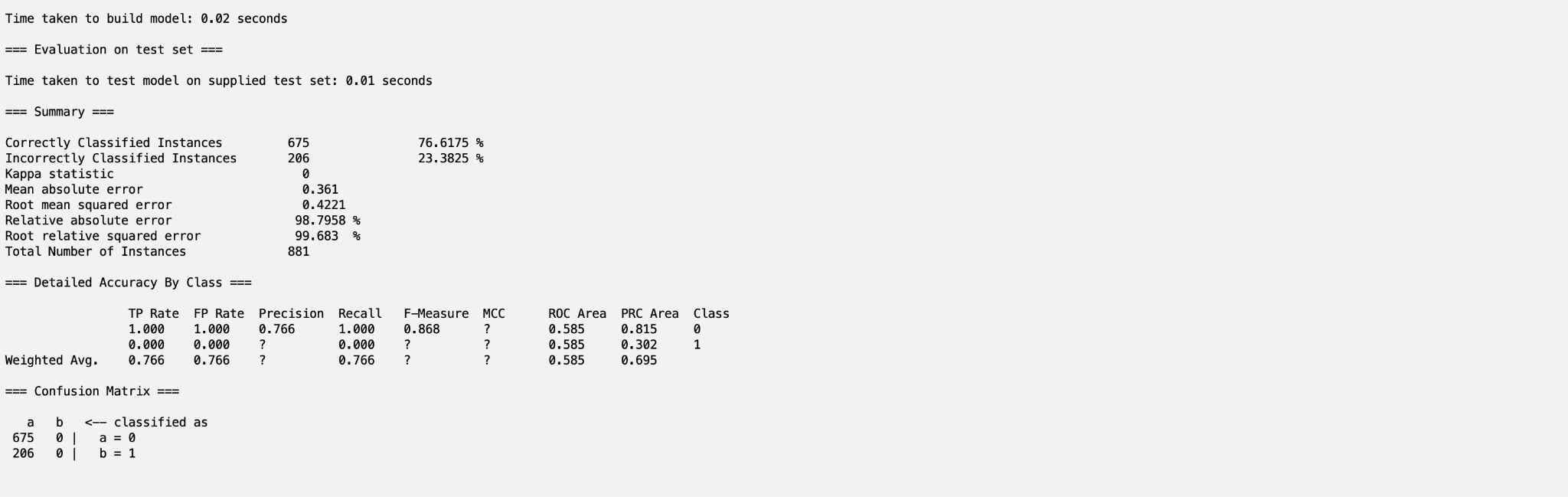
*6.1 CorrelationAttributeEval with NaiveBayes Classification*

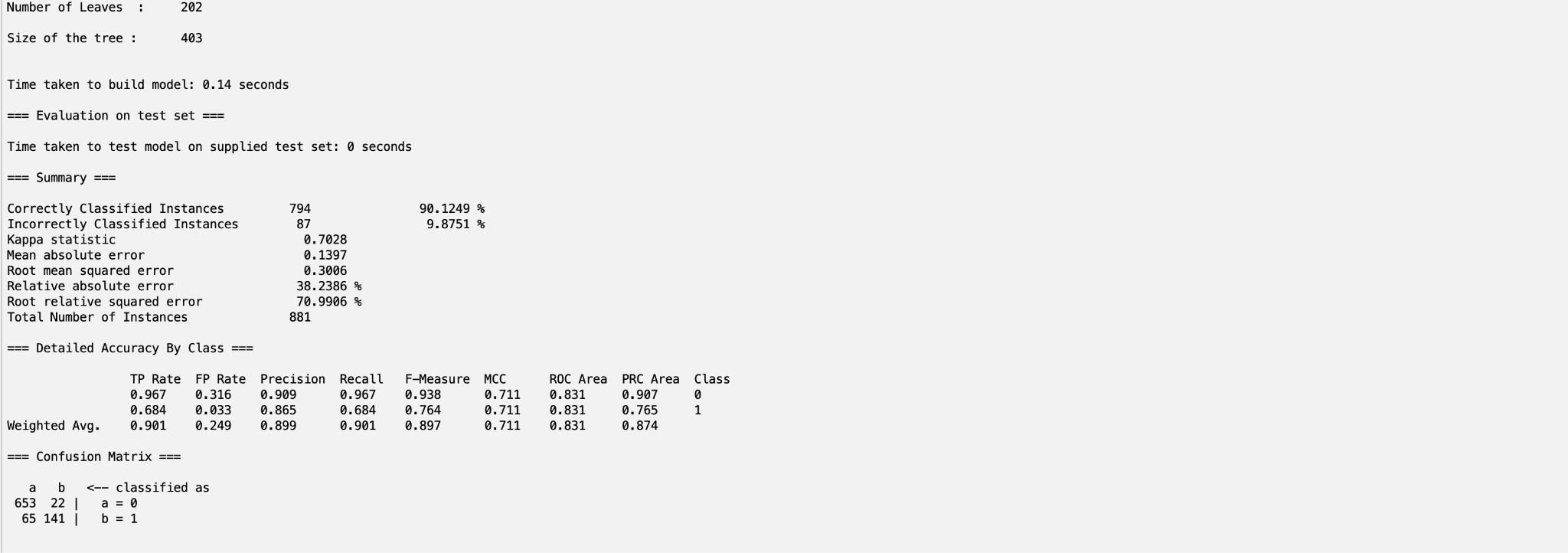
*6.2 CorrelationAttributeEval with J48 Classification*

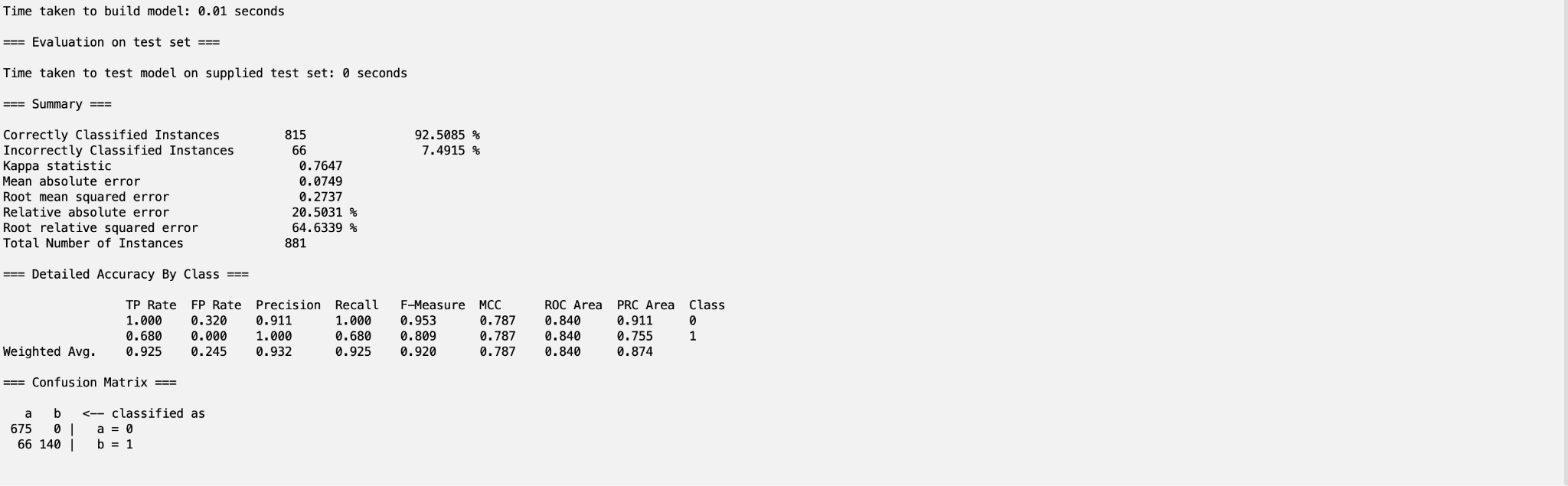
*6.3 CorrelationAttributeEval with OneR Classification*

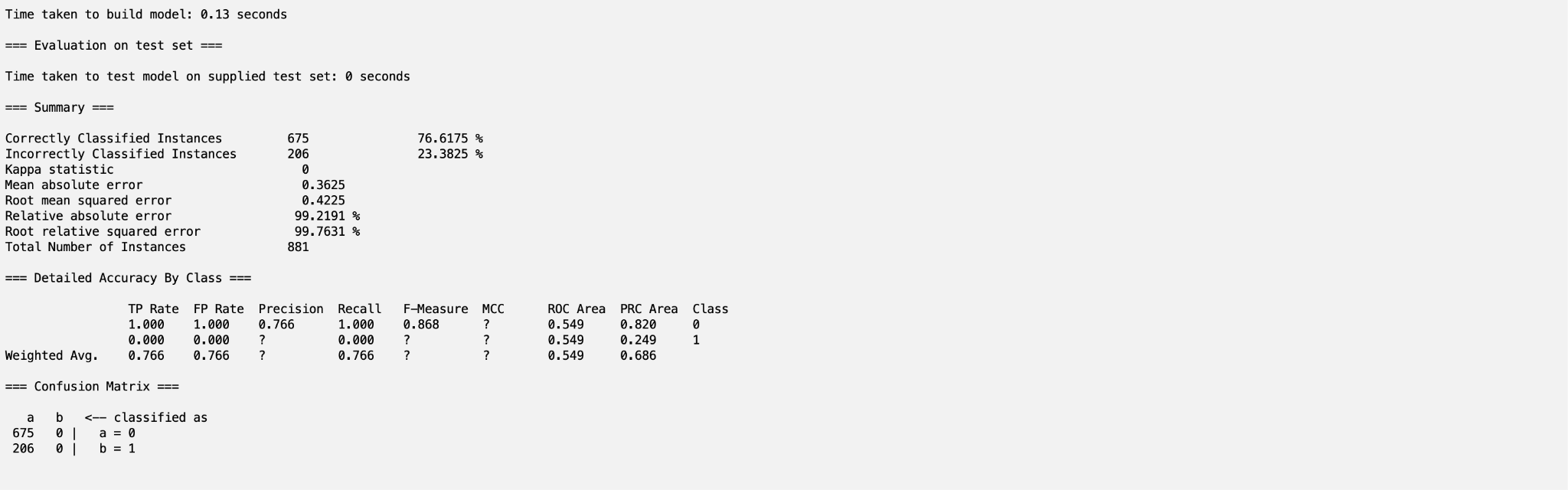
*6.4 CorrelationAttributeEval with Logistic Classification*

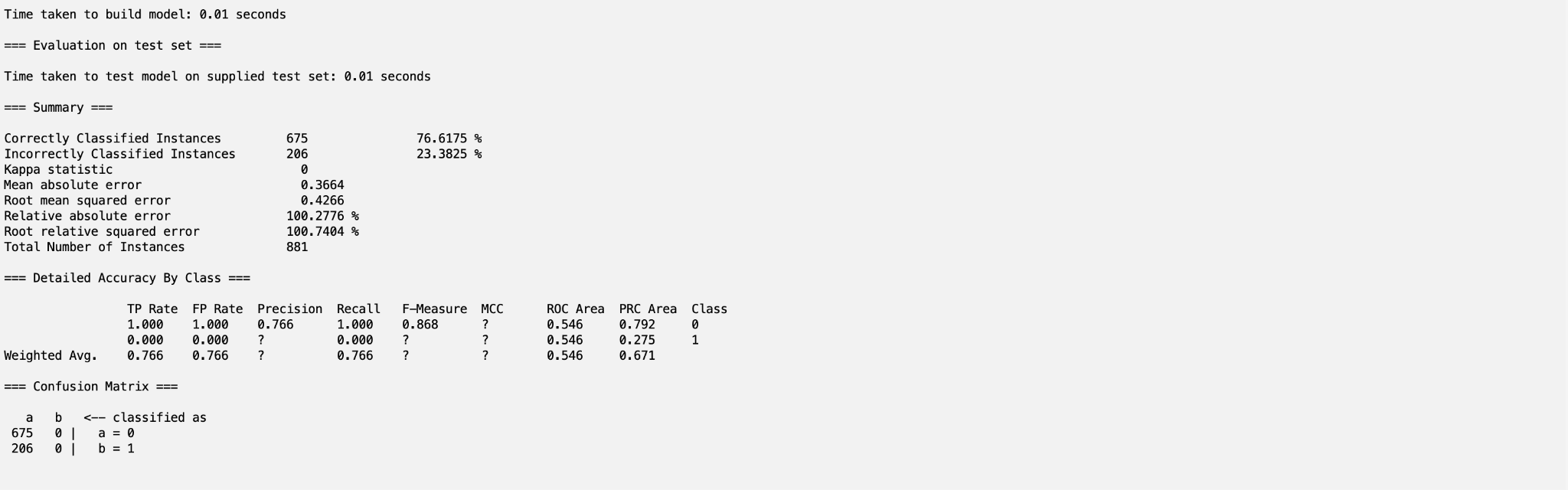
**

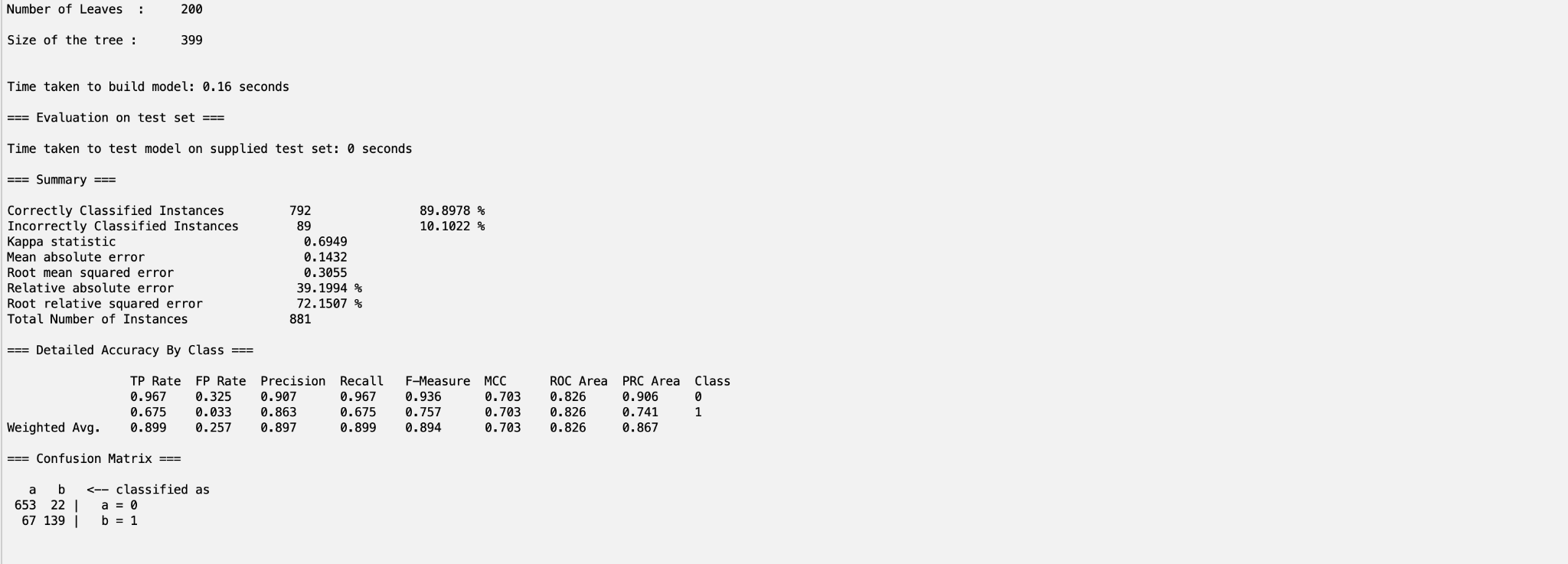
*6.5 CfsSubsetEval with NaiveBayes Classification*

*6.6 CfsSubsetEval with J48 Classification*

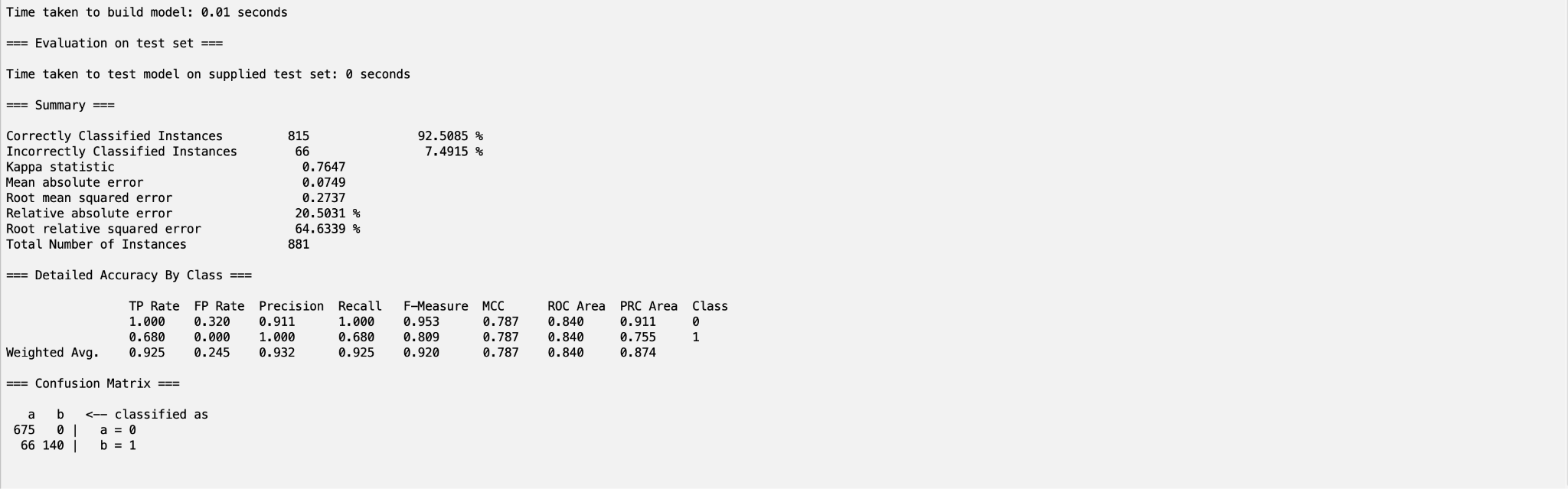
*6.7 CfsSubsetEval with OneR Classification*

*6.8 CfsSubsetEval with Logistic Classification*

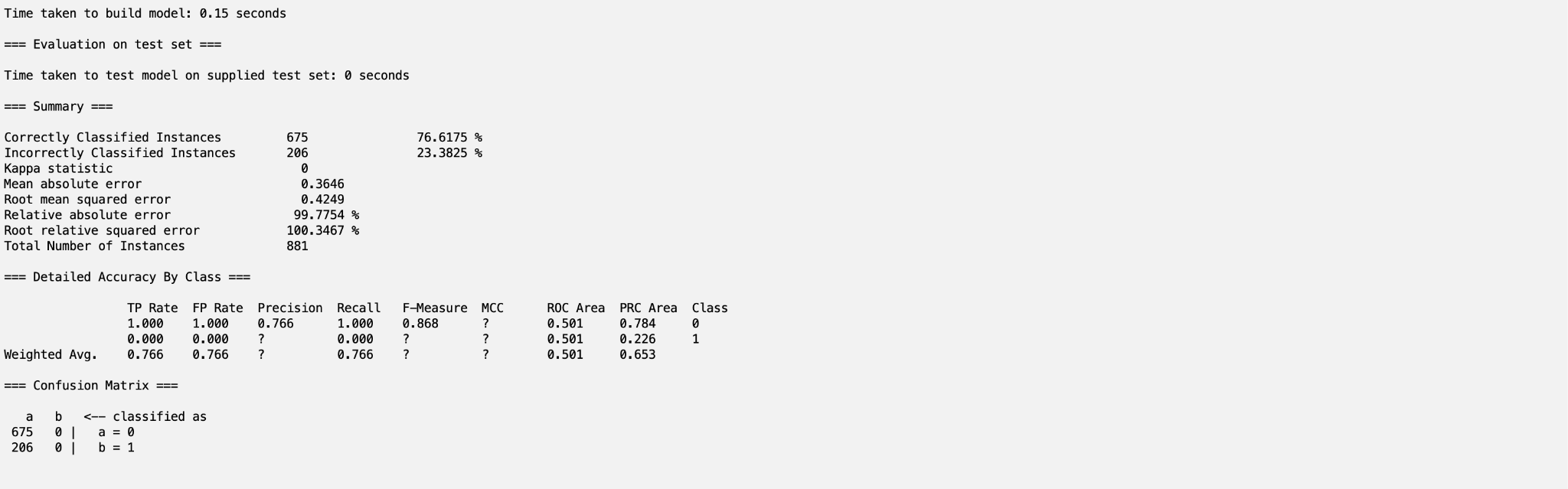
*6.9 InfoGainAttributeEval with NaiveBayes Classification*

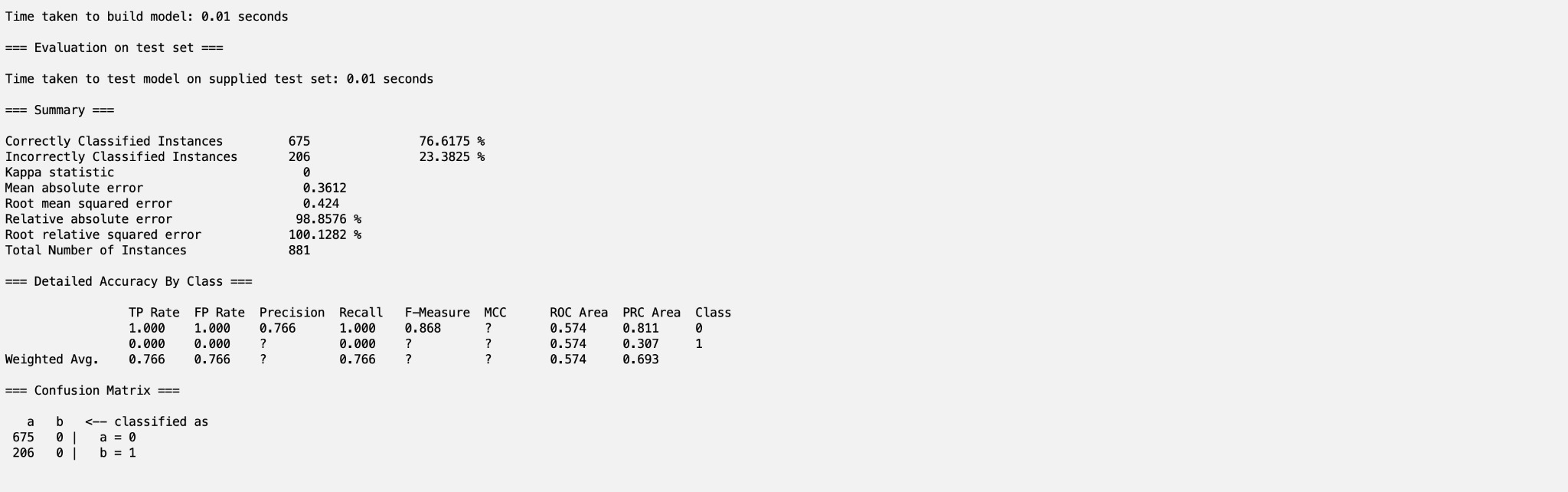
*6.10 InfoGainAttributeEval with J48 Classification*

*6.11 InfoGainAttributeEval with OneR Classification*

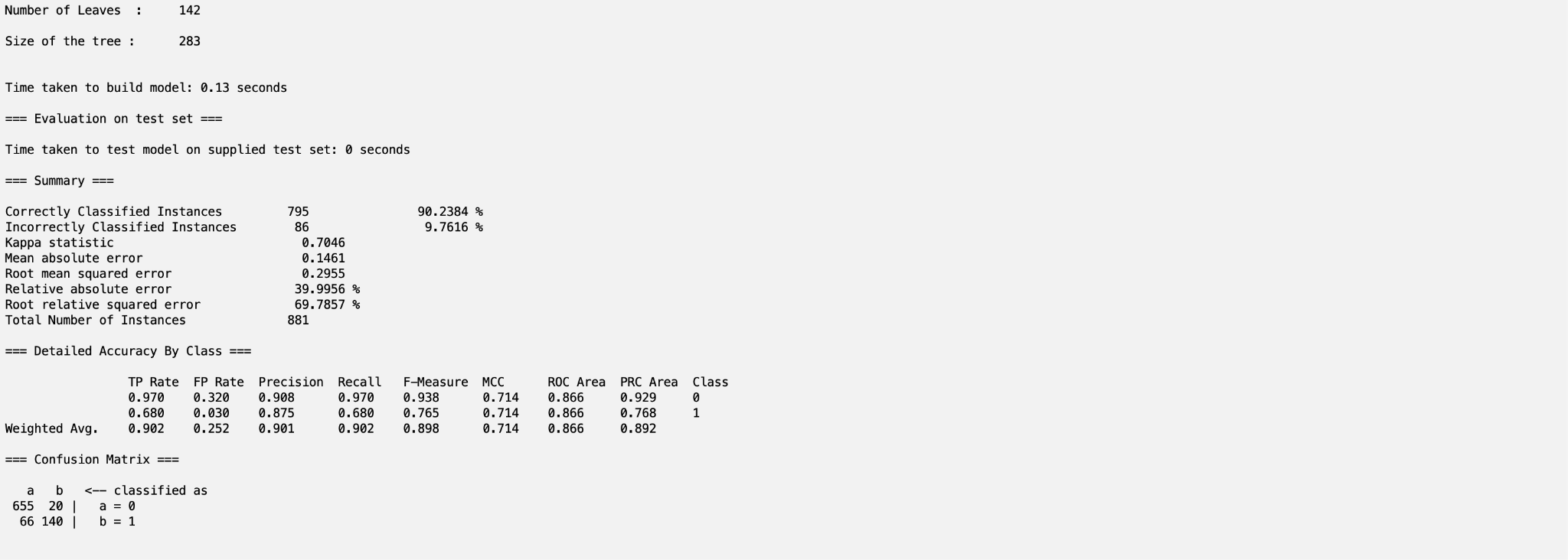
**

*6.12 InfoGainAttributeEval with Logistic Classification*

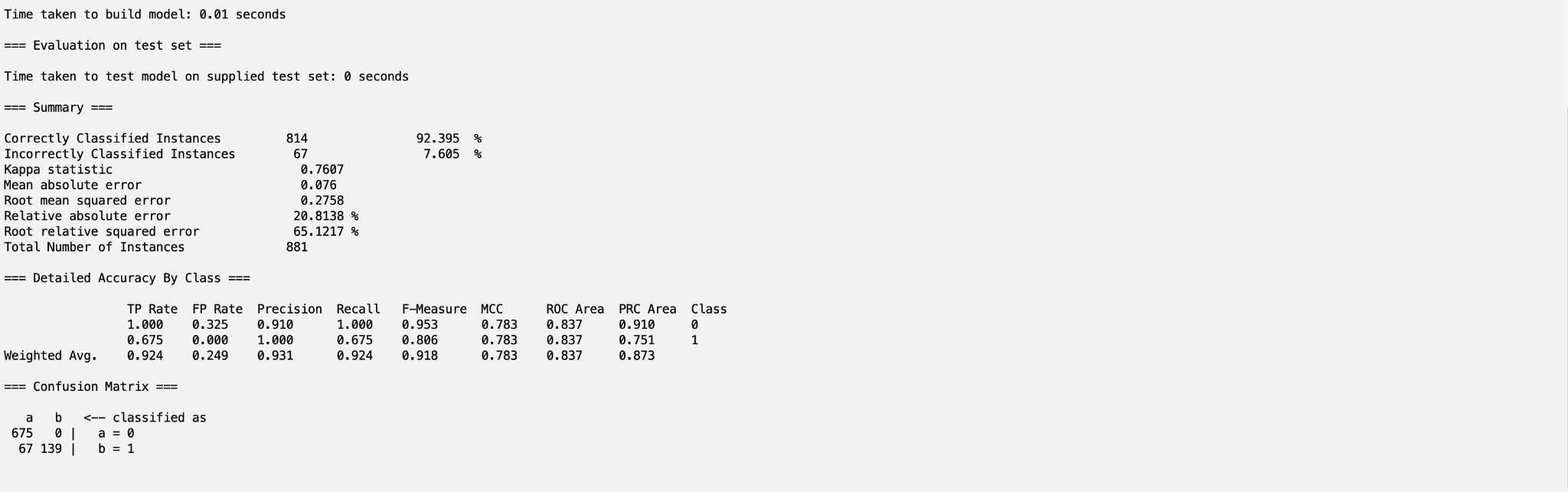
**

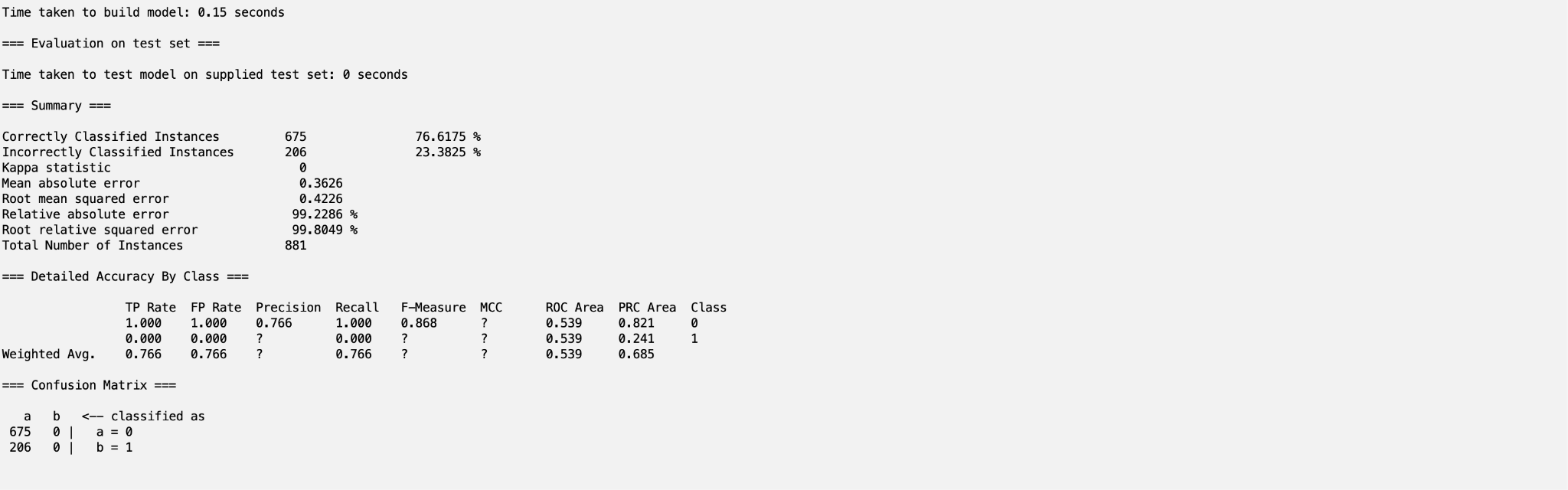
*6.13 OneRAttributeEval with NaiveBayes Classification*

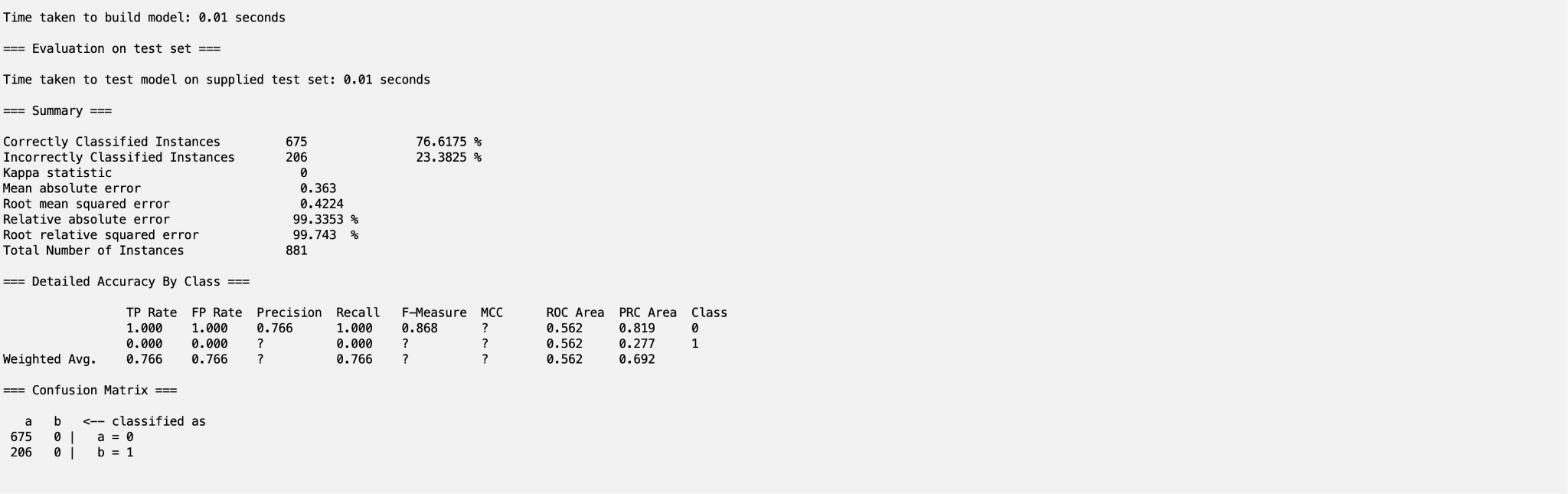
*6.14 OneRAttributeEval with J48 Classification*

**

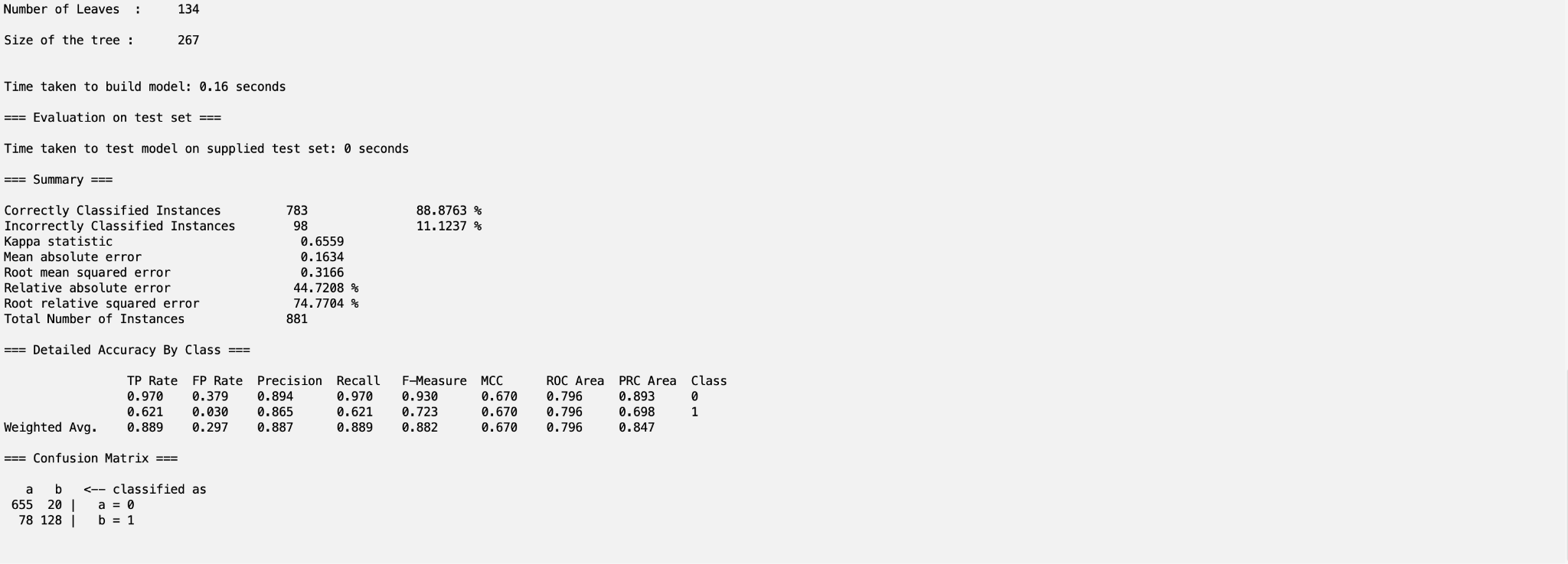
*6.15 OneRAttributeEval with OneR Classification*

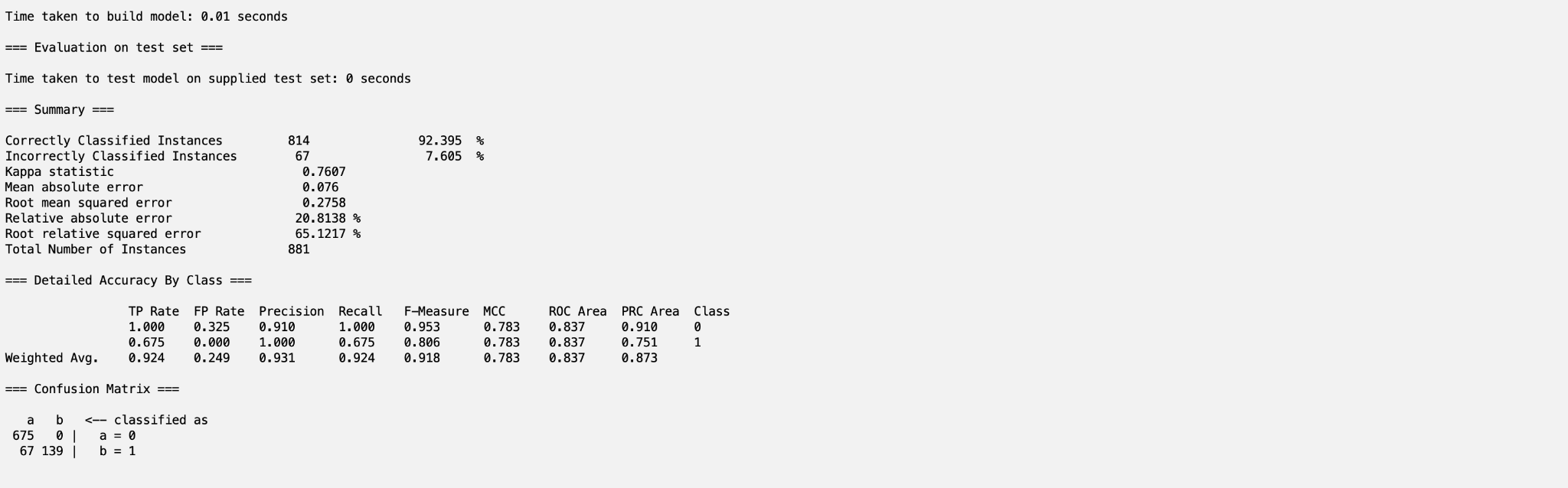
**

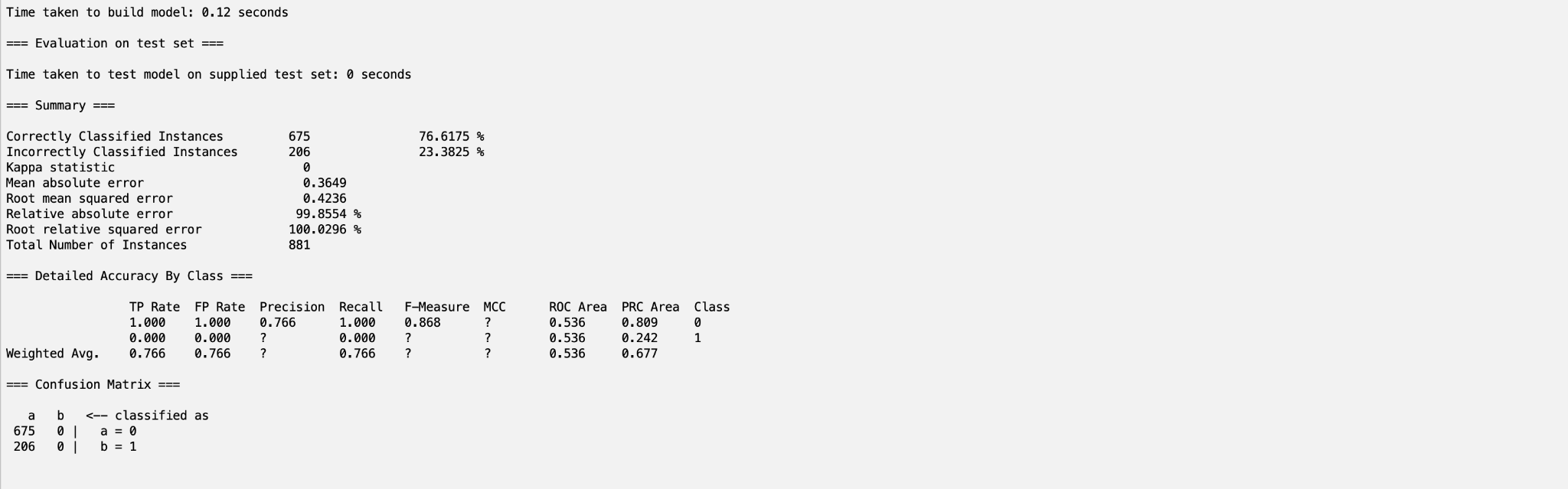
*6.16 OneRAttributeEval with Logistic Classification*

*6.17 Our Chosen Attributes with NaiveBayes Classification*

*6.18 Our Chosen Attributes with J48 Classification*

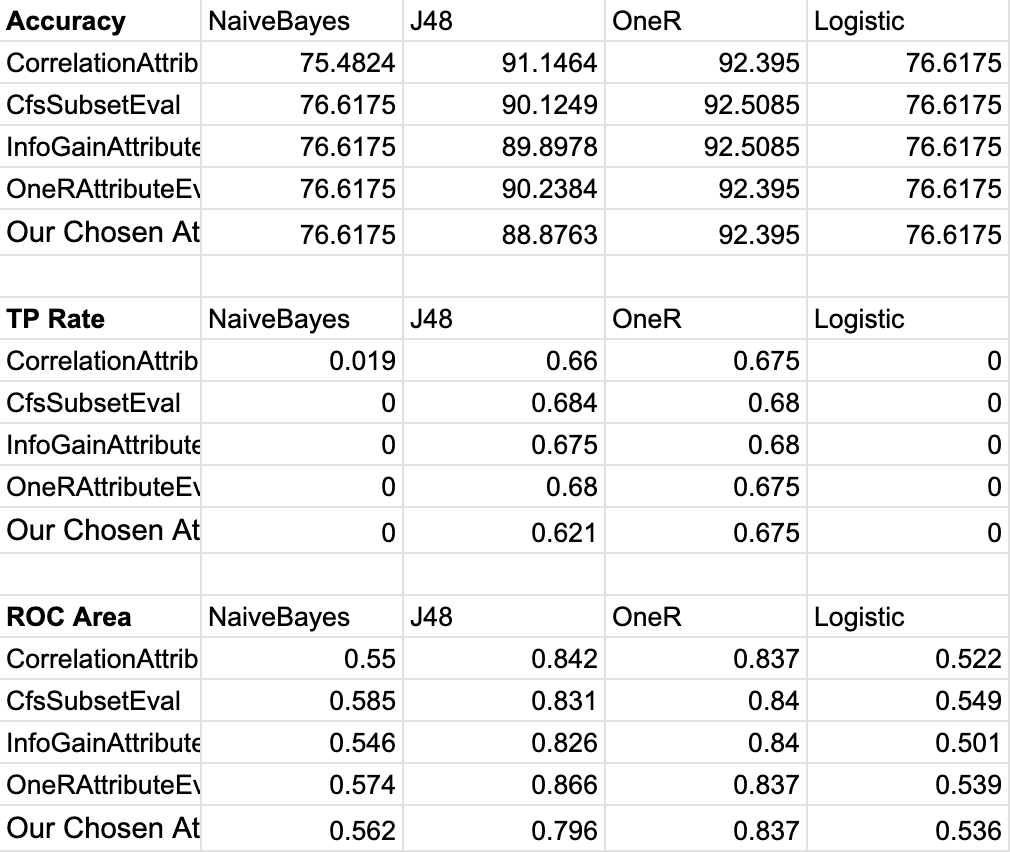
**

*6.19 Our Chosen Attributes with OneR Classification*

*6.20 Our Chosen Attributes with Logistic Classification*

### **Part 7 - Performance Comparison/Results**

When examining the results of the classifiers, we can generate the following table based upon their performance in Accuracy, True positive rate, and ROC Area.



From this table, we generated a list of the 5 highest accuracies, true positive rates, and ROC areas:

Five Highest Accuracies:

1. CfsSubsetEval with OneR Classification - 92.5085%
2. InfoGainAttributeEval with OneR Classification - 92.5085%
3. CorrelationAttributeEval with OneR Classification - 92.395%

2. OneRAttributeEval with OneR Classification - 92.395%

2. Our Chosen Attributes with OneR Classification - 92.395%

Five Highest True Positive Rates:

1. CfsSubsetEval with J48 Classification - 0.684
2. CfsSubsetEval with OneR Classification - 0.68

2. InfoGainAttributeEval with OneR Classification - 0.68

2. OneRAttributeEval with J48 Classification - 0.68

5. InfoGainAttributeEval with J48 Classification - 0.675

Five Highest ROC Areas:

1. OneRAttributeEval with J48 Classification - 0.866
2. CorrelationAttributeEval with J48 Classification - 0.842
3. CfsSubsetEval with OneR Classification - 0.84

3. InfoGainAttributeEval with OneR Classification - 0.84

5. Our Chosen Attributes with One R Classification - 0.837

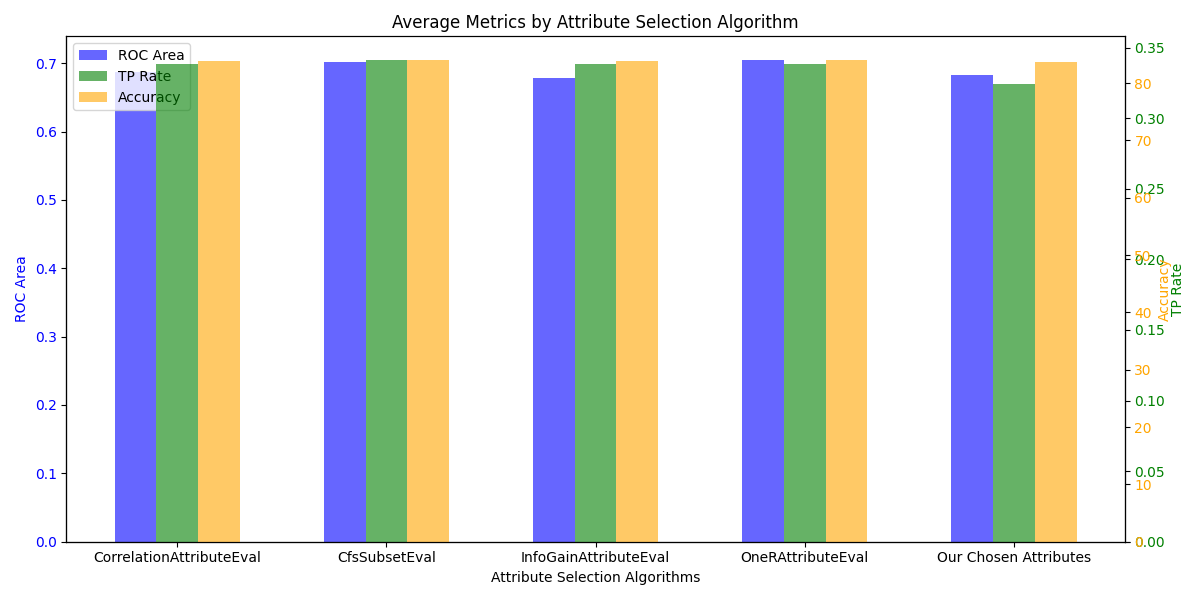
We decided to weight true positive the highest when comparing the results of the models, as it’s very important that the model could accurately predict that named all-stars are all-stars.

Comparing all of the results, we determined the best model to be CfsSubsetEval with J48 Classification. It has the largest true positive rate of 0.684, an accuracy of 90.238%, and an ROC Area of 0.831. Although it does not make the top five in Accuracy and ROC Area, it’s accuracy and ROC area are very close to the top 5, and having the highest true positive Rate proves its applicability.

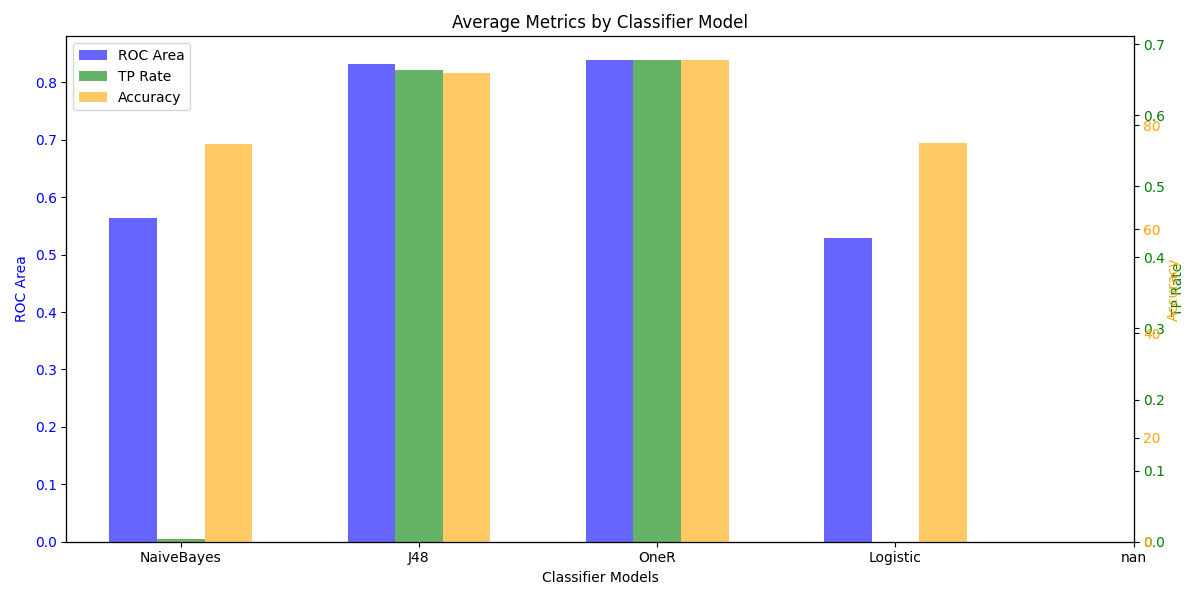
Tied for second place are CfsSubsetEval with OneR Classification and InfoGainAttributeEval with OneR Classification, who have the highest accuracy, the second highest true positive rate, and the third highest ROC Areas.

### **Part 8 - Interesting Trends**

Examining the performance, we compared the average performance of the attribute selection algorithms to generalize trends. In the bar graph below, we can see that CfsSubsetEval performs the best on all metrics and OneRAttributeEval performs strongly as well. It was interesting to see that the attributes from the set we chose performed the worst, as from our basketball knowledge we assumed it would perform strongly.



We did the same for the classifier models, and it was clear to see that OneR resoundingly performed the best, with J48 performing strong as well. The NaiveBayes and Logistic models performed almost equally poorly on the data, and when examining their results we can see that both almost always classified every instance as being a non-allstar (0 TP rate).

The poor performance of NaiveBayes and the Logistic classification models can be attributed to their inability to handle the large class imbalance in the dataset. Both tend to default towards the majority class in cases of imbalanced data, as in this case, which attributed too their very low true positive identification. Additionally, both NaiveBayes, which assumes feature independence, and Logistic Regression, which relies on a linear decision boundary, may not capture the complex patterns necessary for correct predictions. 

Another very important trend is that OneR, the best performing classifier model, selected Age or Wins as the rule attribute (Wins 3 times, Age twice). This shows that these are the two most useful attributes when determining if someone will be named an all-star. Logically, age makes sense, as strong players stay in the league longer, meaning that most likely an old player will be an all-star. A key example of this is Lebron James, who is 40 years old, has been in the league for 22 seasons, and is still being named an all-star. The Wins trend is a bit more difficult to explain, because all of the players who touched the court during the game get a credit for the win, meaning that all-stars would not necessarily have more wins in a season than the other starters on their team. More research is needed to determine the cause of wins being such a strong determinant of all-star selection. However, we have a theory that Wins is important due to fan base popularity. As we mentioned above fans have a huge part in selecting an all-star, we believe that the more wins a team has the more popular the team is. Therefore, the player on the winning team is more likely to get selected as an all-star.

Other notable trends were that although J48 did not lead in accuracy, it showed a strong performance in TP rates across multiple attribute evaluation methods, and that models like OneR and J48 maintained a high ROC Area across multiple attribute selection techniques.

### **Part 9 - Contribution of Team Members**

The work was done collaboratively every step of the way either working on call with each other or delegating tasks and splitting up workload evenly amongst the group members.

Generally, Gus preprocessed the data and created the models in Weka, while Lalit synthesized that information and created the slideshow.

### **Part 10 - Data/Sources**

Link for initial data: [nba\_player\_stats\_with\_all\_star\_selection](https://docs.google.com/spreadsheets/d/1CRG5cMvPcBPPIG5V1fQ7Isc8kLkFtbSyJSoOOhe6zKU/edit?usp=sharing)

Link for preprocessed data: [preprocessed\_nba\_player\_stats\_with\_all\_star\_selection](https://docs.google.com/spreadsheets/d/1n2qeZ10pE9Bowklm70t70wJmunYzVmXByIjvSySJ3WM/edit?usp=sharing)

Link for train/test/validation splits: [Train/Test/Validation Splits](https://drive.google.com/drive/folders/1CpVvNZAUjiZ-QRj3UjEO0xE5J-LbOgqC?usp=sharing)

We received our data from [stats.nba](https://www.nba.com/stats/players/traditional) and [basketball-reference](https://www.basketball-reference.com/allstar/).