CSED/AIGS 526: Data Mining

Your name: Jaeyoon Sim

Student ID: 20222421

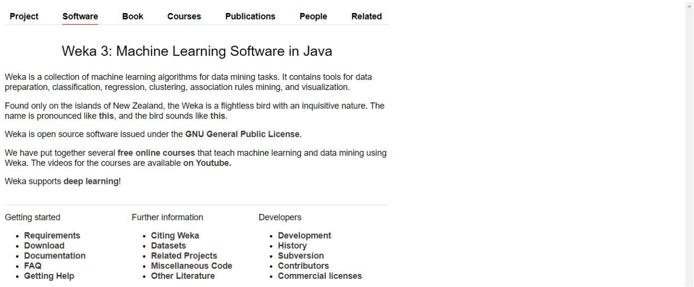
Use late days quota(Yes/No): No

Problem 1

Weka

The purpose of this assignment is to learn about a program called ”Weka” and to get used to the data given. ”Weka” is a collection of machine learning algorithms for data mining tasks. It contains tools for data preparation, classification, regression, clustering, association rules mining, and visualization. That’s why we should install ”Weka” on our computer.

1. We access https://www.cs.waikato.ac.nz/ml/weka/.



Weka 3: Machine Learning Software in Java

Weka is a collection of machine learning algorithms for data mining tasks. It contains tools for data preparation, classification, regression, clustering, association rules mining, and visualization.

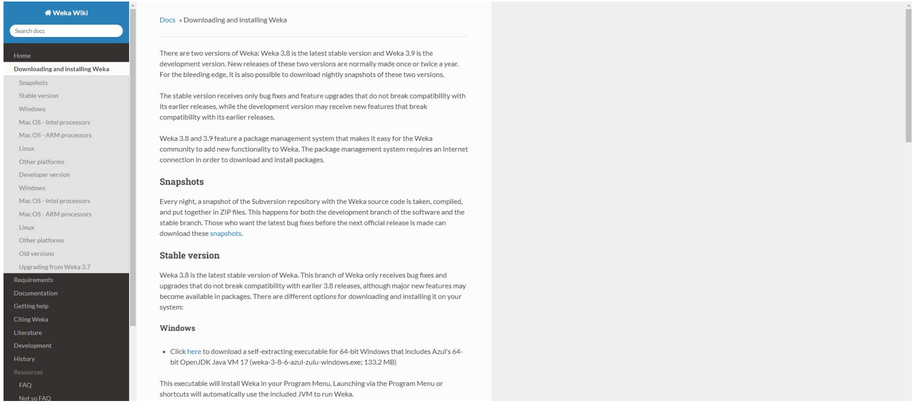
Found only on the islands of New Zealand, the Weka is a flightless bird with an inquisitive nature. The name is pronounced like **this**, and the bird sounds like **this**.

Weka is open source software issued under the **GNU General Public License**.

We have put together several **free online courses** that teach machine learning and data mining using Weka. The videos for the courses are available **on Youtube.**

Weka supports **deep learning**!

2. We click Download to go to the page where the ”Weka” document is located.



Docs » Downloading and installing Weka

There are two versions of Weka: Weka 3.8 is the latest stable version and Weka 3.9 is the development version. New releases of these two versions are normally made once or twice a year. For the bleeding edge, it is also possible to download nightly snapshots of these two versions.

The stable version receives only bug fixes and feature upgrades that do not break compatibility with its earlier releases, while the development version may receive new features that break compatibility with its earlier releases.

Weka 3.8 and 3.9 feature a package management system that makes it easy for the Weka community to add new functionality to Weka. The package management system requires an internet connection in order to download and install packages.

Snapshots

Every night, a snapshot of the Subversion repository with the Weka source code is taken, compiled, and put together in ZIP files. This happens for both the development branch of the software and the stable branch. Those who want the latest bug fixes before the next official release is made can download these snapshots.

Stable version

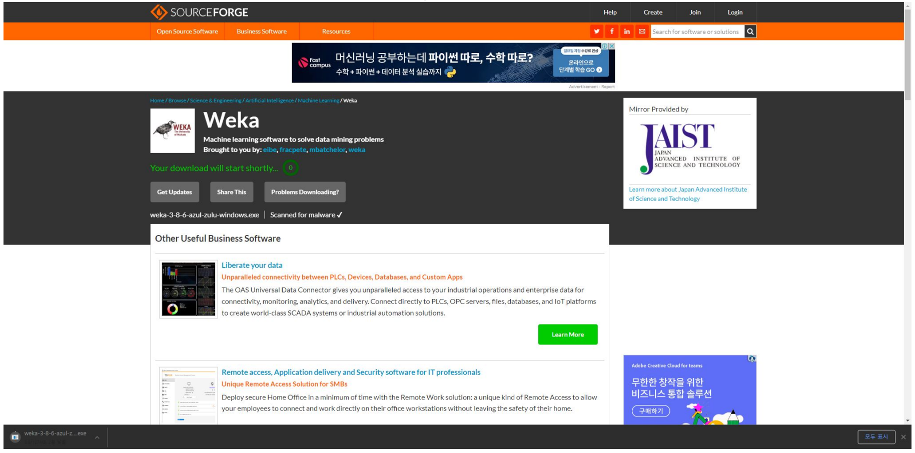
Weka 3.8 is the latest stable version of Weka. This branch of Weka only receives bug fixes and upgrades that do not break compatibility with earlier 3.8 releases, although major new features may become available in packages. There are different options for downloading and installing it on your system:

Windows

- Click here to download a self-extracting executable for 64-bit Windows that includes Azul's 64-bit OpenJDK Java VM 17 (weka-3-8-6-azul-zulu-windows.exe; 133.2 MB)

This executable will install Weka in your Program Menu. Launching via the Program Menu or shortcuts will automatically use the included JVM to run Weka.

3. We select the window OS to install the program on the computer and download the program.


SOURCEFORGE
Help
Create
Join
Login
Open Source Software
Business Software
Resources
Search for software or solutions
Fast campus
머신러닝 공부하는데 파이썬 따로, 수학 따로?
수학 + 파이썬 + 데이터 분석 실습까지
온라인으로 단계별 학습 GO
Advertisement - Report
Home / Browse / Science & Engineering / Artificial Intelligence / Machine Learning / Weka
WEKA
Get Updates
Share This
Problems Downloading?
Mirror Provided by
JAIST
JAPAN ADVANCED INSTITUTE OF SCIENCE AND TECHNOLOGY
Learn more about Japan Advanced Institute of Science and Technology
Learn More
Adobe Creative Cloud for teams
무한한 창작을 위한
비즈니스 통합 솔루션
구매하기
weka-3-8-6-azul-z....exe
모두 표시

Weka

Machine learning software to solve data mining problems

Brought to you by: eibe, fracpete, mbatchelor, weka

Your download will start shortly... 0

weka-3-8-6-azul-zulu-windows.exe | Scanned for malware ✓

Other Useful Business Software

Liberate your data

Unparalleled connectivity between PLCs, Devices, Databases, and Custom Apps

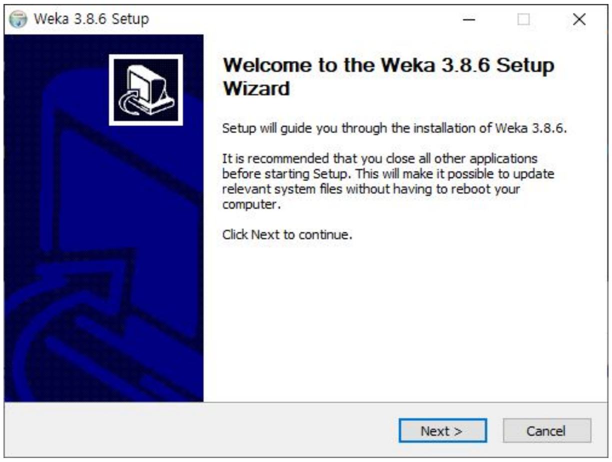
The OAS Universal Data Connector gives you unparalleled access to your industrial operations and enterprise data for connectivity, monitoring, analytics, and delivery. Connect directly to PLCs, OPC servers, files, databases, and IoT platforms to create world-class SCADA systems or industrial automation solutions.

Remote access, Application delivery and Security software for IT professionals

Unique Remote Access Solution for SMBs

Deploy secure Home Office in a minimum of time with the Remote Work solution: a unique kind of Remote Access to allow your employees to connect and work directly on their office workstations without leaving the safety of their home.

4. From now on, we use the .exe file to install ”Weka”.

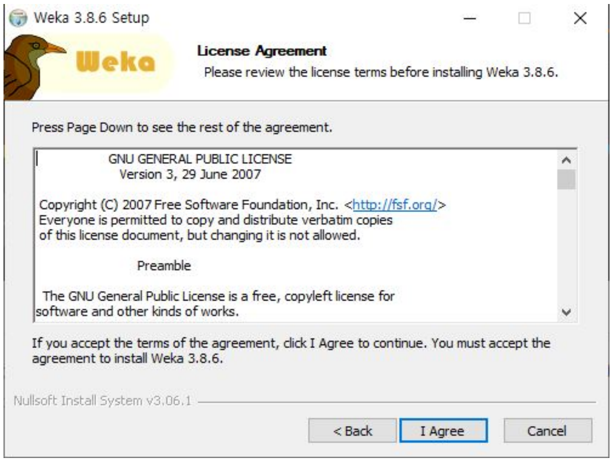


Welcome to the Weka 3.8.6 Setup Wizard

Setup will guide you through the installation of Weka 3.8.6.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.



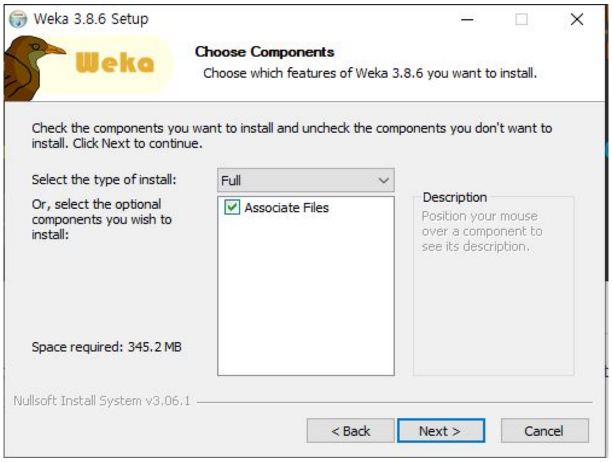
License Agreement

Please review the license terms before installing Weka 3.8.6.

Press Page Down to see the rest of the agreement.

```
            GNU GENERAL PUBLIC LICENSE
               Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

                    Preamble

 The GNU General Public License is a free, copyleft license for
software and other kinds of works.
```

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Weka 3.8.6.



Choose Components

Choose which features of Weka 3.8.6 you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

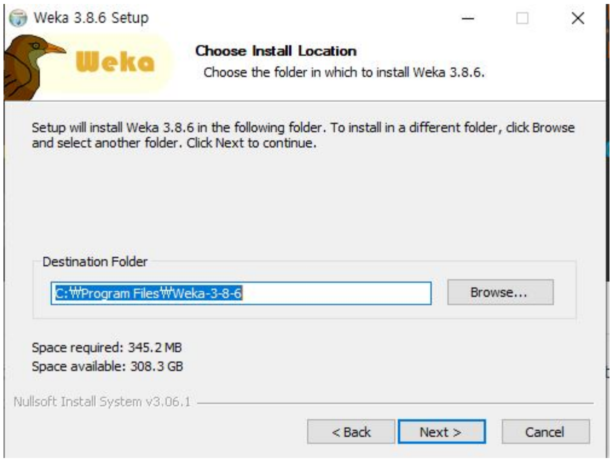
Select the type of install: Full

Or, select the optional components you wish to install: ☑ Associate Files

Description

Position your mouse over a component to see its description.

Space required: 345.2 MB



Choose Install Location

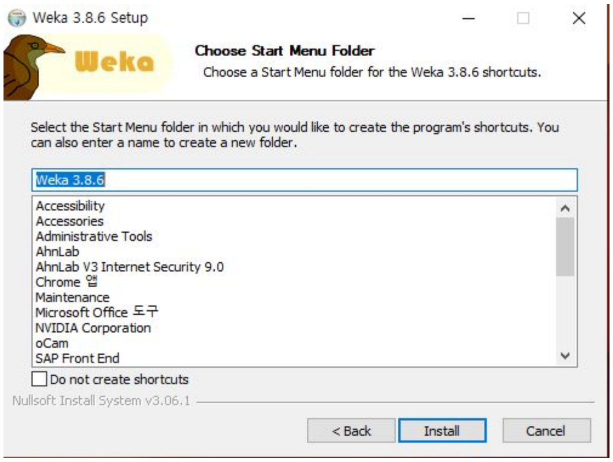
Choose the folder in which to install Weka 3.8.6.

Setup will install Weka 3.8.6 in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder: C:₩Program Files₩Weka-3-8-6

Space required: 345.2 MB

Space available: 308.3 GB



Choose Start Menu Folder

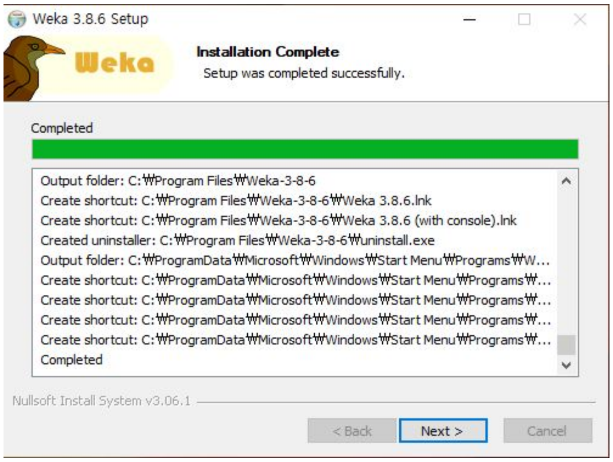
Choose a Start Menu folder for the Weka 3.8.6 shortcuts.

Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.

Weka 3.8.6

Accessibility
Accessories
Administrative Tools
AhnLab
AhnLab V3 Internet Security 9.0
Chrome 앱
Maintenance
Microsoft Office 도구
NVIDIA Corporation
oCam
SAP Front End

□ Do not create shortcuts



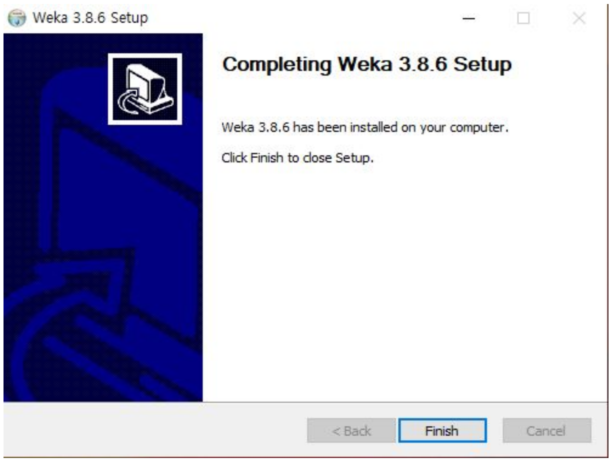
Installation Complete

Setup was completed successfully.

Completed

```
Output folder: C:₩Program Files₩Weka-3-8-6
Create shortcut: C:₩Program Files₩Weka-3-8-6₩Weka 3.8.6.lnk
Create shortcut: C:₩Program Files₩Weka-3-8-6₩Weka 3.8.6 (with console).lnk
Created uninstaller: C:₩Program Files₩Weka-3-8-6₩uninstall.exe
Output folder: C:₩ProgramData₩Microsoft₩Windows₩Start Menu₩Programs₩W...
Create shortcut: C:₩ProgramData₩Microsoft₩Windows₩Start Menu₩Programs₩...
Create shortcut: C:₩ProgramData₩Microsoft₩Windows₩Start Menu₩Programs₩...
Create shortcut: C:₩ProgramData₩Microsoft₩Windows₩Start Menu₩Programs₩...
Create shortcut: C:₩ProgramData₩Microsoft₩Windows₩Start Menu₩Programs₩...
Completed
```

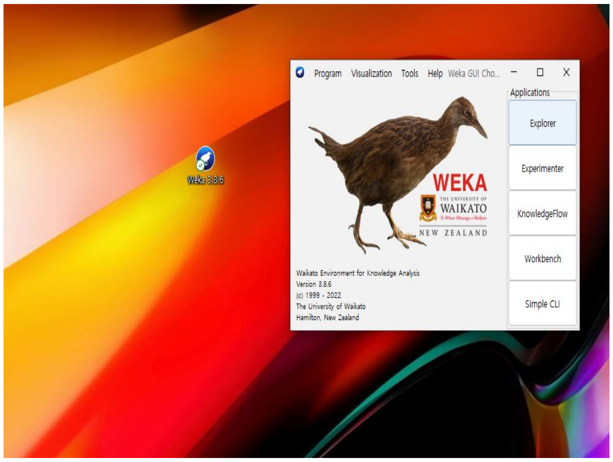
5. When we complete the installation of the ”Weka” program, we can run the ”Weka” program as follows.



Completing Weka 3.8.6 Setup

Weka 3.8.6 has been installed on your computer.

Click Finish to close Setup.

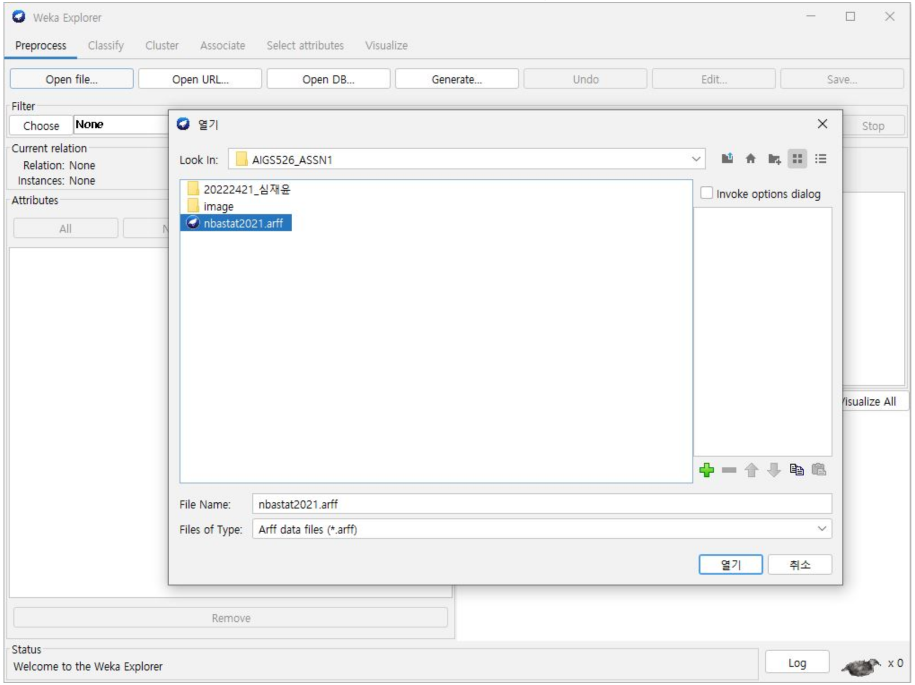


Waikato Environment for Knowledge Analysis
Version 3.8.6
(c) 1999 - 2022
The University of Waikato
Hamilton, New Zealand

Problem 2

Preprocess, and Clustering

We received an .arff (attribute-relation file format) file used by ”Weka” for this assignment. In order to use this .arff file, we open ”Weka” and click ”Explorer” applications. Data files for ”Weka” is .arff, and we can open this file in ”Weka” by following the below.

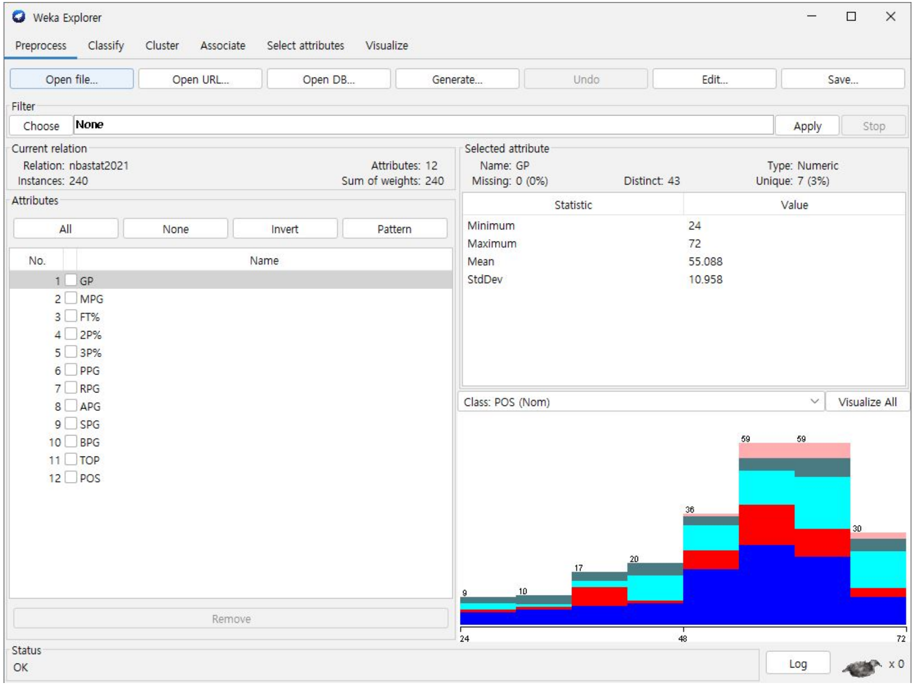


Filter: None

Relation: None
Instances: None

File Name: nbastat2021.arff
Files of Type: Arff data files (*.arff)

Welcome to the Weka Explorer



Filter: None

Relation: nbastat2021
Instances: 240
Attributes: 12
Sum of weights: 240

Name: GP
Missing: 0 (0%)
Distinct: 43
Type: Numeric
Unique: 7 (3%)

Statistic	Value
Minimum	24
Maximum	72
Mean	55.088
StdDev	10.958

No.	Name
1	□ GP
2	□ MPG
3	□ FT%
4	□ 2P%
5	□ 3P%
6	□ PPG
7	□ RPG
8	□ APG
9	□ SPG
10	□ BPG
11	□ TOP
12	□ POS

Class: POS (Nom)

OK

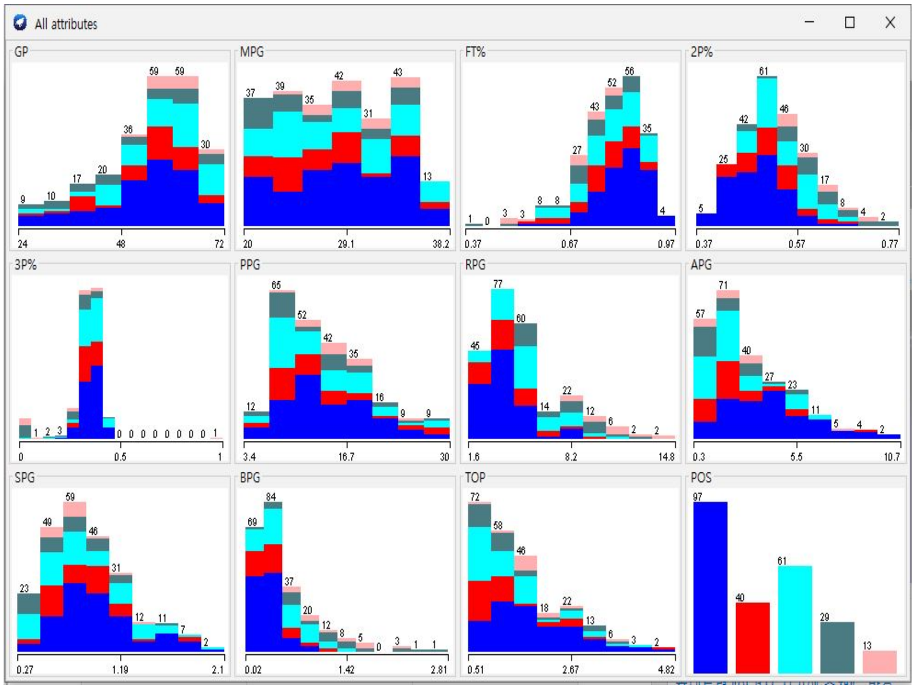
With ”nbastats2021.arff” file, the dataset contains basketball related records from 240 players from National Basketball League(NBA). We will look at the data set made based on the various information commonly used in basketball games. Before that, we would like to briefly look at the attributes used in the ”Weka” program.

1. GP: Game Played
2. MPG: Minutes played per game
3. FT%: Free throw percentage
4. 2P%: 2-points field goal percentage
5. 3P%: 3-points field goal percentage
6. PPG: Points per game
7. RPG: Rebounds per game
8. APG: Assists per game
9. SPG: Steals per game
10. BPG: Blocks per game
11. TOPG: Turnovers per game
12. POS: Position

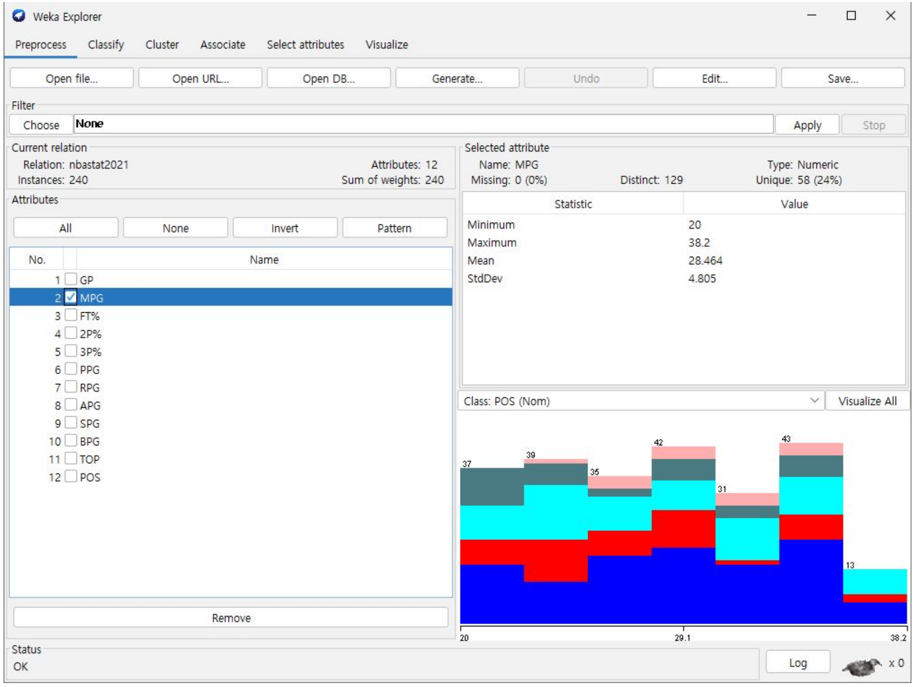
Given nbastat2021 dataset contains statistics from National Basketball League(NBA). Each player is characterized by various features seen above. In particular, attribute 1 to 11 are features of players related to basketball games, and feature 12 is about positions in which players are played in basketball games.

In actual basketball games, roles are divided from 1 to 5 depending on the position. Number one is a point guard, number two is a shooting guard, number three is a small forward, number four is a power forward, and number five is a center. Each role has different things to do, and we would like to use positions such as {G, G-F, F, F-C, C} in this assignment.

First of all, we want to look at the data in the ”Preprocess” tab of the ”Weka” program. The ”Preprocess” tab currently contains information about the dataset. We can find out what data sets are, how many attributes are in total, and how many instances are in total. The following is the result of visualization by creating a histogram from all attributes of the data.



Of the total 12 histograms, we will look closely at the histograms of MPG. If we look at the MPG histogram, we can see how long 240 players played the game.


Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Open file... | Open URL... | Open DB... | Generate... | Undo | Edit... | Save...

Filter
Choose None | Apply | Stop

Current relation
Relation: nbastat2021 Attributes: 12
Instances: 240 Sum of weights: 240

Selected attribute
Name: MPG Type: Numeric
Missing: 0 (0%) Distinct: 129 Unique: 58 (24%)

Statistic	Value
Minimum	20
Maximum	38.2
Mean	28.464
StdDev	4.805

Attributes
All | None | Invert | Pattern

No.	Name
1	□ GP
2	☑ MPG
3	□ FT%
4	□ 2P%
5	□ 3P%
6	□ PPG
7	□ RPG
8	□ APG
9	□ SPG
10	□ BPG
11	□ TOP
12	□ POS

Remove

Class: POS (Nom) | Visualize All

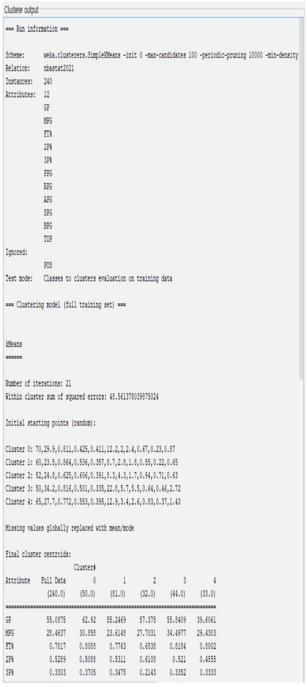
Status
OK | Log | x 0

It expresses 240 players in histograms every 2.6 minutes from a minimum of 20 minutes to a maximum of 38.2 minutes. The players played basketball for an average of 28.464 minutes, with a standard deviation of 4.805 minutes.

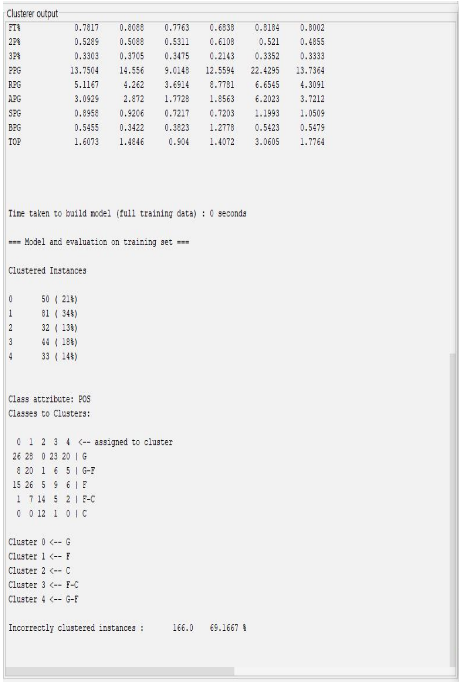
Now, we want to look at the data in the "Cluster" tab of the "Weka" program. Considering the results of visualizing histograms from all attributes of data and the roles given for each position in basketball games, we would like to proceed with clustering using {3P%, RPG, APG, BPG}. And we would like to exclude {GP, MPG, FT%, 2P%, PPG, SPG, TOP, POS}. The reason is that once we think about our personal experience and role by position in basketball games, rebounds and blocks match the position of the center, while 3P% and assist match the position of the guard. In general, this trend is shown, but considering the given data, it tends to be consistent to some extent, and overall, there is no significant difference in histograms for each position in the number of games played, time of game, free throw, 2-points field goal, scoring, still, and turnover.

This time, we're going to use the "SimpleKMeans" clustering method stored in the "Weka" Cluster tab, ignoring the attributes we mentioned earlier. In the case of the KMean algorithm, it is sensitive to the distribution of data and has weak adaptability to the Outlier. The "SimpleKMeans" algorithm in "Weka" uses Euclidean distance measure to compute distances between instances and clusters. Here, the "POS" attribute will be excluded when attempting to perform the clustering method. Along with these conditions, the total number of clusters is set to five to cluster the positions of basketball players.

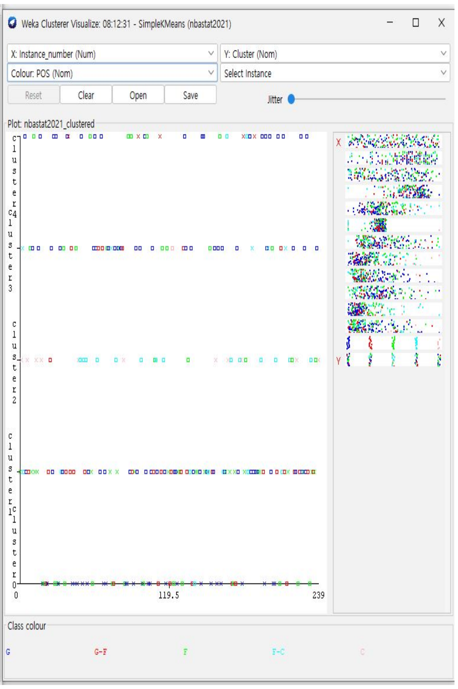
The following result is the result of excluding only "POS" and executing from the attributes. As a result of SimpleKMeans, it can be seen that 166 out of 240 instances are not accurately clustered. So it can be seen that 69.1667% of the total instances did not properly form a cluster excluding only "POS".



```
=== Run information ===

Scheme:       weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density
Relation:     nbastat2021
Instances:    240
Attributes:   12
              GP
              MPG
              FT%
              2P%
              3P%
              PPG
              RPG
              APG
              SPG
              BPG
              TOP
Ignored:
              POS
Test mode:    Classes to clusters evaluation on training data

=== Clustering model (full training set) ===


kMeans
======

Number of iterations: 21
Within cluster sum of squared errors: 48.561378039875024

Initial starting points (random):

Cluster 0: 70,29.9,0.811,0.425,0.411,12.2,2,2.4,0.67,0.23,0.87
Cluster 1: 60,23.5,0.864,0.536,0.357,8.7,2.8,1.8,0.55,0.22,0.65
Cluster 2: 52,24.8,0.625,0.606,0.391,8.3,4.3,1.7,0.94,0.71,0.63
Cluster 3: 50,34.2,0.816,0.501,0.335,22.8,5.7,5.5,0.64,0.46,2.72
Cluster 4: 65,27.7,0.772,0.553,0.395,12.9,3.4,2.6,0.83,0.37,1.43

Missing values globally replaced with mean/mode

Final cluster centroids:
                         Cluster#
Attribute    Full Data          0          1          2          3          4
               (240.0)     (50.0)     (81.0)     (32.0)     (44.0)     (33.0)
===============================================================================
GP             55.0875      62.92    55.2469     57.375    55.8409    39.6061
MPG            28.4637     30.858    23.6148    27.7031    34.4977    29.4303
FT%             0.7817     0.8088     0.7763     0.6838     0.8184     0.8002
2P%             0.5289     0.5088     0.5311     0.6108      0.521     0.4855
3P%             0.3303     0.3705     0.3475     0.2143     0.3352     0.3333
```



```
FT%             0.7817     0.8088     0.7763     0.6838     0.8184     0.8002
2P%             0.5289     0.5088     0.5311     0.6108      0.521     0.4855
3P%             0.3303     0.3705     0.3475     0.2143     0.3352     0.3333
PPG            13.7504     14.556     9.0148    12.5594    22.4295    13.7364
RPG             5.1167      4.262     3.6914     8.7781     6.6545     4.3091
APG             3.0929      2.872     1.7728     1.8563     6.2023     3.7212
SPG             0.8958     0.9206     0.7217     0.7203     1.1993     1.0509
BPG             0.5455     0.3422     0.3823     1.2778     0.5423     0.5479
TOP             1.6073     1.4846      0.904     1.4072     3.0605     1.7764




Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       50 ( 21%)
1       81 ( 34%)
2       32 ( 13%)
3       44 ( 18%)
4       33 ( 14%)


Class attribute: POS
Classes to Clusters:

  0  1  2  3  4  <-- assigned to cluster
 26 28  0 23 20 | G
  8 20  1  6  5 | G-F
 15 26  5  9  6 | F
  1  7 14  5  2 | F-C
  0  0 12  1  0 | C

Cluster 0 <-- G
Cluster 1 <-- F
Cluster 2 <-- C
Cluster 3 <-- F-C
Cluster 4 <-- G-F

Incorrectly clustered instances :       166.0    69.1667 %
```


Weka Clusterer Visualize: 08:12:31 - SimpleKMeans (nbastat2021)

X: Instance_number (Num)

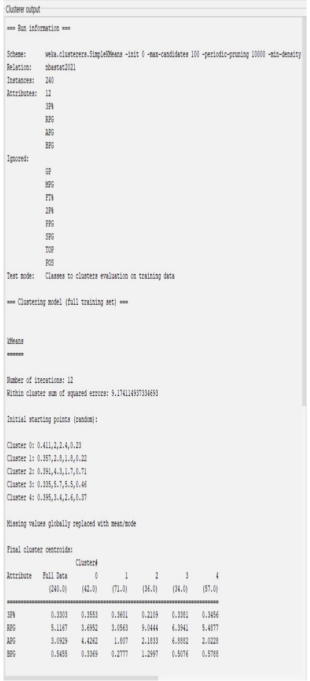
Y: Cluster (Nom)

Colour: POS (Nom)

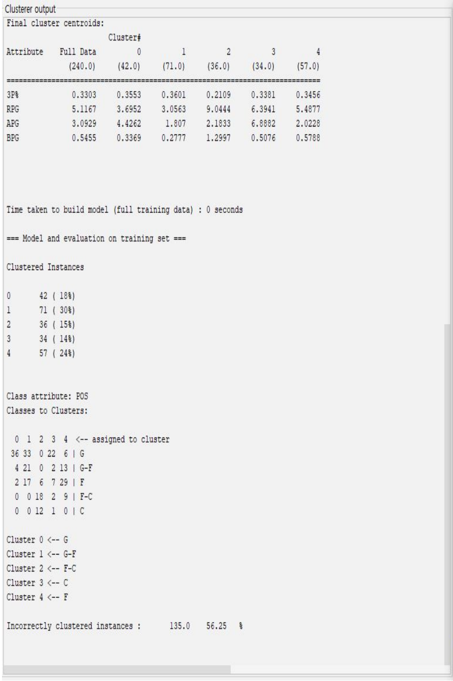
Select Instance

Plot: nbastat2021_clustered

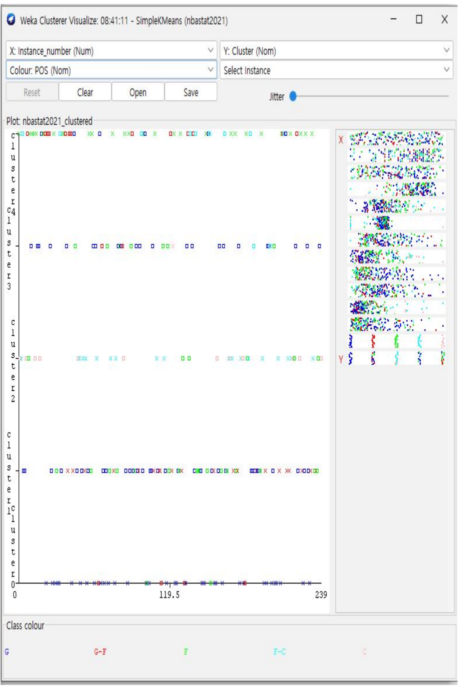
This is the result of clustering using {3P%, RPG, APG, BPG} attributes as we initially thought. This time, as a result, it can be seen that 135 out of 240 instances are not accurately clustered. Therefore, 56.25% of the total instances did not properly form clusters. It can be seen that this resulted in better clustering results based on existing thoughts and analysis.


Clusterer output

```
=== Run information ===

Scheme:       weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density
Relation:     nbastat2021
Instances:    240
Attributes:   12
              3P%
              RPG
              APG
              BPG
Ignored:
              GP
              MPG
              FT%
              2P%
              PPG
              SPG
              TOP
              POS
Test mode:    Classes to clusters evaluation on training data

=== Clustering model (full training set) ===


kMeans
======

Number of iterations: 12
Within cluster sum of squared errors: 9.174114937334693

Initial starting points (random):

Cluster 0: 0.411,2,2.4,0.23
Cluster 1: 0.357,2.8,1.8,0.22
Cluster 2: 0.391,4.3,1.7,0.71
Cluster 3: 0.335,5.7,5.5,0.46
Cluster 4: 0.395,3.4,2.6,0.37

Missing values globally replaced with mean/mode

Final cluster centroids:
                         Cluster#
Attribute    Full Data          0          1          2          3          4
               (240.0)     (42.0)     (71.0)     (36.0)     (34.0)     (57.0)
================================================================================
3P%             0.3303     0.3553     0.3601     0.2109     0.3381     0.3456
RPG             5.1167     3.6952     3.0563     9.0444     6.3941     5.4877
APG             3.0929     4.4262      1.807     2.1833     6.8882     2.0228
BPG             0.5455     0.3369     0.2777     1.2997     0.5076     0.5788
```


Clusterer output

```
Final cluster centroids:
                         Cluster#
Attribute    Full Data          0          1          2          3          4
               (240.0)     (42.0)     (71.0)     (36.0)     (34.0)     (57.0)
================================================================================
3P%             0.3303     0.3553     0.3601     0.2109     0.3381     0.3456
RPG             5.1167     3.6952     3.0563     9.0444     6.3941     5.4877
APG             3.0929     4.4262      1.807     2.1833     6.8882     2.0228
BPG             0.5455     0.3369     0.2777     1.2997     0.5076     0.5788




Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       42 ( 18%)
1       71 ( 30%)
2       36 ( 15%)
3       34 ( 14%)
4       57 ( 24%)


Class attribute: POS
Classes to Clusters:

  0  1  2  3  4  <-- assigned to cluster
 36 33  0 22  6 | G
  4 21  0  2 13 | G-F
  2 17  6  7 29 | F
  0  0 18  2  9 | F-C
  0  0 12  1  0 | C

Cluster 0 <-- G
Cluster 1 <-- G-F
Cluster 2 <-- F-C
Cluster 3 <-- C
Cluster 4 <-- F

Incorrectly clustered instances :       135.0    56.25   %
```


Weka Clusterer Visualize: 08:41:11 - SimpleKMeans (nbastat2021)

X: Instance_number (Num)

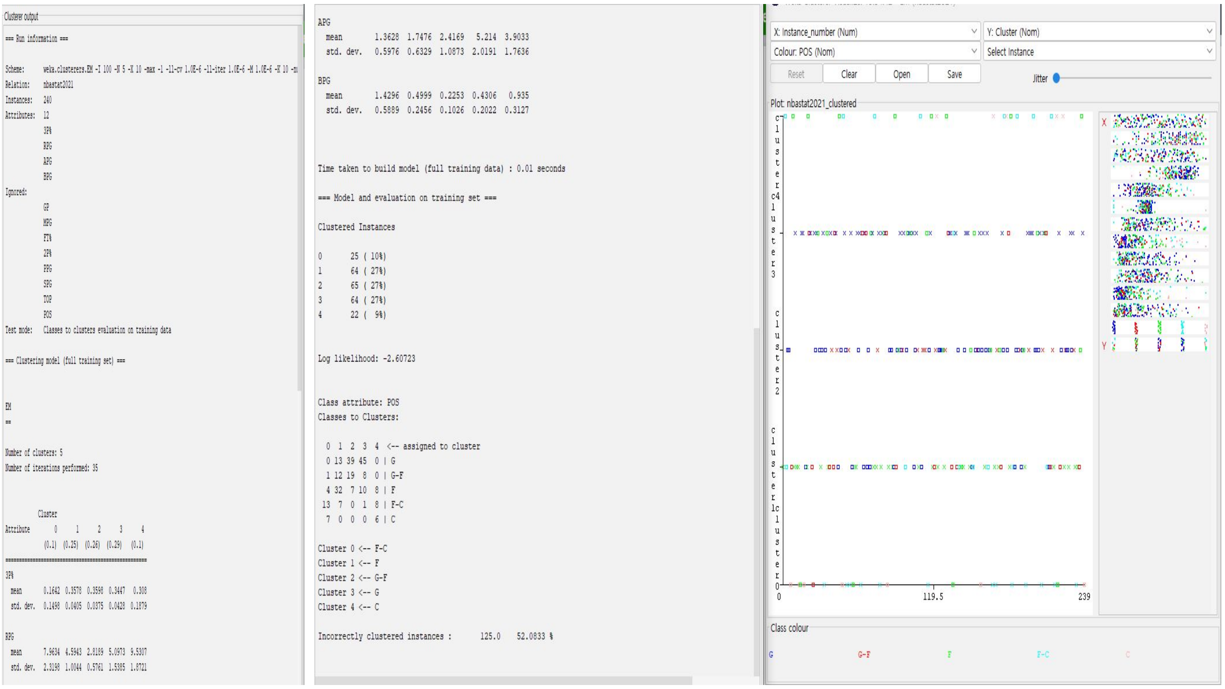
Y: Cluster (Nom)

Colour: POS (Nom)

Select Instance

The final problem of Problem 2 is to create five clusters again and visualize them using the "EM" clustering algorithm. In statistics, an expectation–maximization (EM) algorithm is an iterative method to find (local) maximum likelihood or maximum a posteriori(MAP) estimates of parameters in statistical models, where the model depends on unobserved latent variables. The EM iteration alternates between performing an expectation(E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and a maximization(M) step, which computes parameters maximizing the expected log-likelihood found on the E step. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step.

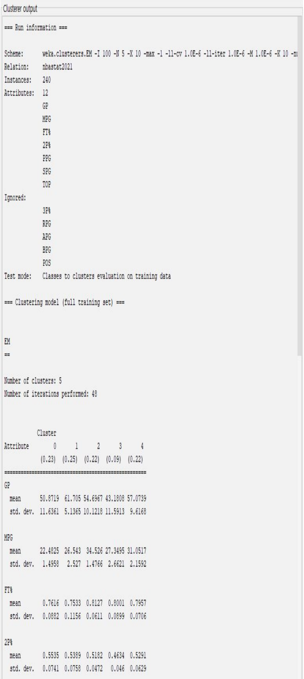
In this problem, using the EM algorithm, we can also change the hyperparameters of EM cluster. First of all, we want to set the EM clustering method to different 'ignored attributes' and execute it. EM clustering was performed by including {3P%, RPG, APG, BPG} attributes in the first case, and EM clustering was performed by including {GP, MPG, FT%, 2P%, PPG, SPG, TOP} attributes. In the first case, {GP, MPG, FT%, 2P%, PPG, SPG, TOP, POS} attributes were ignored, and in the second case, {3P%, RPG, APG, BPG, POS} attributes were ignored.



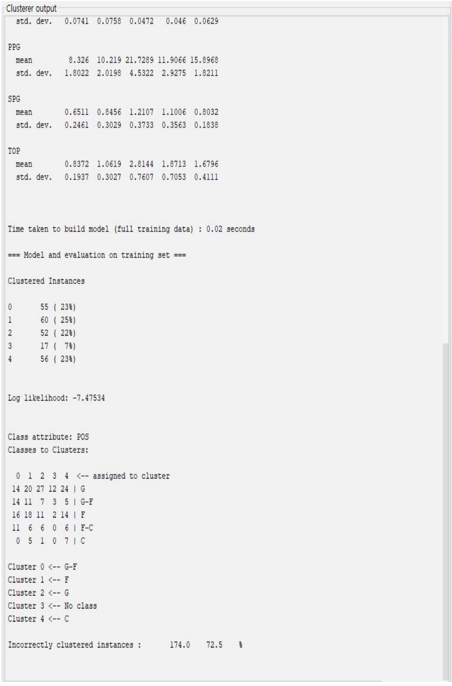
Clusterer output

```
=== Run information ===

Scheme:       weka.clusterers.EM -I 100 -N 5 -X 10 -max -1 -ll-cv 1.0E-6 -ll-iter 1.0E-6 -M 1.0E-6 -K 10 -nu
Relation:     nbastat2021
Instances:    240
Attributes:   12
              3P%
              RPG
              APG
              BPG
Ignored:
              GP
              MPG
              FT%
              2P%
              PPG
              SPG
              TOP
              POS
Test mode:    Classes to clusters evaluation on training data

=== Clustering model (full training set) ===


EM
==

Number of clusters: 5
Number of iterations performed: 35


             Cluster
Attribute         0       1       2       3       4
              (0.1)  (0.25)  (0.26)  (0.29)   (0.1)
====================================================
3P%
  mean       0.1642  0.3578  0.3598  0.3447   0.308
  std. dev.  0.1498  0.0405  0.0375  0.0428  0.1879

RPG
  mean       7.9634  4.5943  2.8189  5.0973  9.5307
  std. dev.  2.3198  1.0044  0.5761  1.5385  1.8721
```

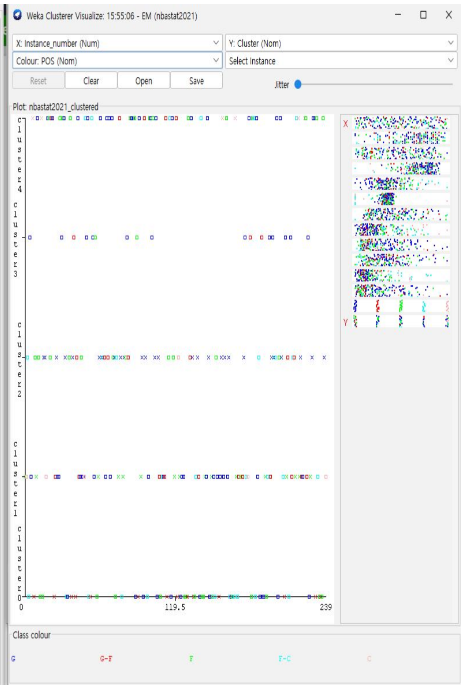
```
APG
  mean       1.3628  1.7476  2.4169   5.214  3.9033
  std. dev.  0.5976  0.6329  1.0873  2.0191  1.7636

BPG
  mean       1.4296  0.4999  0.2253  0.4306   0.935
  std. dev.  0.5889  0.2456  0.1026  0.2022  0.3127



Time taken to build model (full training data) : 0.01 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       25 ( 10%)
1       64 ( 27%)
2       65 ( 27%)
3       64 ( 27%)
4       22 (  9%)


Log likelihood: -2.60723


Class attribute: POS
Classes to Clusters:

  0  1  2  3  4  <-- assigned to cluster
  0 13 39 45  0 | G
  1 12 19  8  0 | G-F
  4 32  7 10  8 | F
 13  7  0  1  8 | F-C
  7  0  0  0  6 | C

Cluster 0 <-- F-C
Cluster 1 <-- F
Cluster 2 <-- G-F
Cluster 3 <-- G
Cluster 4 <-- C

Incorrectly clustered instances :       125.0    52.0833 %
```



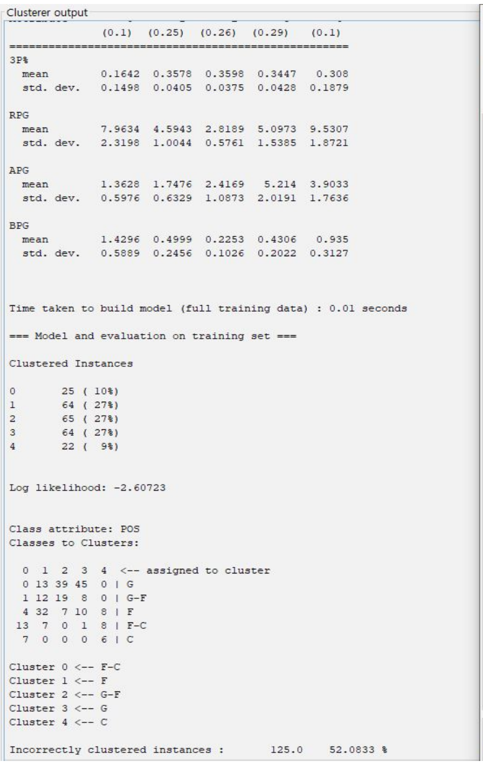
```
Clusterer output
=== Run information ===

Scheme:       weka.clusterers.EM -I 100 -N 5 -X 10 -max -1 -ll-cv 1.0E-6 -ll-iter 1.0E-6 -M 1.0E-6 -K 10 -nu
Relation:     nbastat2021
Instances:    240
Attributes:   12
              GP
              MPG
              FT%
              2P%
              PPG
              SPG
              TOP
Ignored:
              3P%
              RPG
              APG
              BPG
              POS
Test mode:    Classes to clusters evaluation on training data

=== Clustering model (full training set) ===


EM
==

Number of clusters: 5
Number of iterations performed: 48


            Cluster
Attribute         0       1       2       3       4
             (0.23)  (0.25)  (0.22)  (0.09)  (0.22)
===================================================
GP
  mean      50.8719  61.705 54.6967 43.1808 57.0739
  std. dev. 11.6361  5.1365 10.1218 11.5913  9.6168

MPG
  mean      22.4825  26.543  34.526 27.3495 31.0517
  std. dev.  1.4958   2.527  1.4766  2.6621  2.1592

FT%
  mean       0.7616  0.7533  0.8127  0.8001  0.7957
  std. dev.  0.0882  0.1156  0.0611  0.0899  0.0706

2P%
  mean       0.5535  0.5389  0.5182  0.4634  0.5291
  std. dev.  0.0741  0.0758  0.0472   0.046  0.0629
```



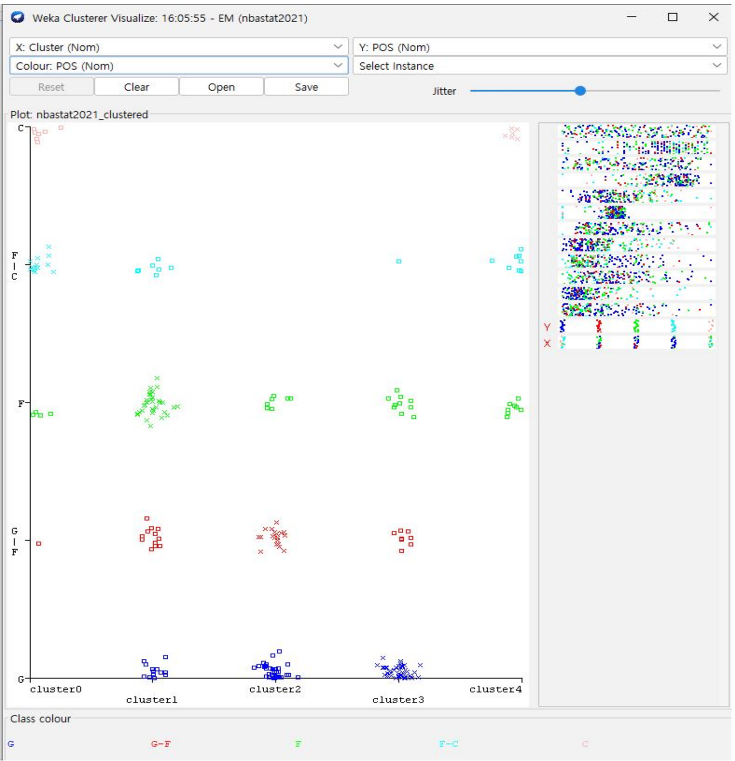
```
Clusterer output
  std. dev.   0.0741  0.0758  0.0472   0.046  0.0629

PPG
  mean         8.326  10.219 21.7289 11.9066 15.8968
  std. dev.   1.8022  2.0198  4.5322  2.9275  1.8211

SPG
  mean        0.6511  0.8456  1.2107  1.1006  0.8032
  std. dev.   0.2461  0.3029  0.3733  0.3563  0.1838

TOP
  mean        0.8372  1.0619  2.8144  1.8713  1.6796
  std. dev.   0.1937  0.3027  0.7607  0.7053  0.4111



Time taken to build model (full training data) : 0.02 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       55 ( 23%)
1       60 ( 25%)
2       52 ( 22%)
3       17 (  7%)
4       56 ( 23%)


Log likelihood: -7.47534


Class attribute: POS
Classes to Clusters:

  0  1  2  3  4  <-- assigned to cluster
 14 20 27 12 24 | G
 14 11  7  3  5 | G-F
 16 18 11  2 14 | F
 11  6  6  0  6 | F-C
  0  5  1  0  7 | C

Cluster 0 <-- G-F
Cluster 1 <-- F
Cluster 2 <-- G
Cluster 3 <-- No class
Cluster 4 <-- C

Incorrectly clustered instances :       174.0    72.5    %
```

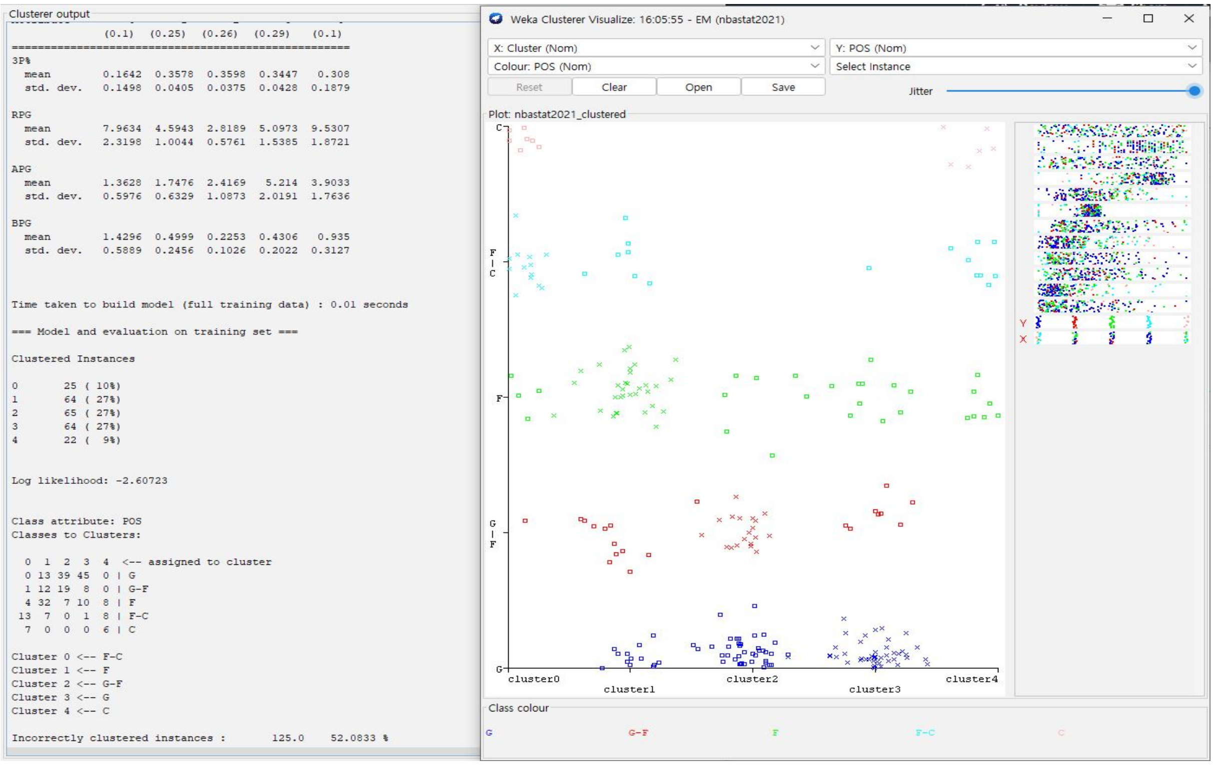


Now, we will try to make five clusters each representing a different position-{G, G-F, F, F-C, C} and visualize one of them. Like the SimpleKMeans above, clustering was performed with {3P%, RPG, APG, BPG} attributes left. The results were compared while adjusting jitter, which is the degree to which data is scattered. In the case of hyperparameter, it was used as default except for changing the number of clusters to five.



```
Clusterer output
                 (0.1)  (0.25)  (0.26)  (0.29)   (0.1)
======================================================
3P%
  mean          0.1642  0.3578  0.3598  0.3447   0.308
  std. dev.     0.1498  0.0405  0.0375  0.0428  0.1879

RPG
  mean          7.9634  4.5943  2.8189  5.0973  9.5307
  std. dev.     2.3198  1.0044  0.5761  1.5385  1.8721

APG
  mean          1.3628  1.7476  2.4169   5.214  3.9033
  std. dev.     0.5976  0.6329  1.0873  2.0191  1.7636

BPG
  mean          1.4296  0.4999  0.2253  0.4306   0.935
  std. dev.     0.5889  0.2456  0.1026  0.2022  0.3127



Time taken to build model (full training data) : 0.01 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       25 ( 10%)
1       64 ( 27%)
2       65 ( 27%)
3       64 ( 27%)
4       22 (  9%)


Log likelihood: -2.60723


Class attribute: POS
Classes to Clusters:

  0  1  2  3  4  <-- assigned to cluster
  0 13 39 45  0 | G
  1 12 19  8  0 | G-F
  4 32  7 10  8 | F
 13  7  0  1  8 | F-C
  7  0  0  0  6 | C

Cluster 0 <-- F-C
Cluster 1 <-- F
Cluster 2 <-- G-F
Cluster 3 <-- G
Cluster 4 <-- C

Incorrectly clustered instances :       125.0    52.0833 %
```





Clusterer output

```
                (0.1)  (0.25)  (0.26)  (0.29)   (0.1)
====================================================
3P%
  mean         0.1642  0.3578  0.3598  0.3447   0.308
  std. dev.    0.1498  0.0405  0.0375  0.0428  0.1879

RPG
  mean         7.9634  4.5943  2.8189  5.0973  9.5307
  std. dev.    2.3198  1.0044  0.5761  1.5385  1.8721

APG
  mean         1.3628  1.7476  2.4169   5.214  3.9033
  std. dev.    0.5976  0.6329  1.0873  2.0191  1.7636

BPG
  mean         1.4296  0.4999  0.2253  0.4306   0.935
  std. dev.    0.5889  0.2456  0.1026  0.2022  0.3127



Time taken to build model (full training data) : 0.01 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       25 ( 10%)
1       64 ( 27%)
2       65 ( 27%)
3       64 ( 27%)
4       22 (  9%)


Log likelihood: -2.60723


Class attribute: POS
Classes to Clusters:

  0  1  2  3  4  <-- assigned to cluster
  0 13 39 45  0 | G
  1 12 19  8  0 | G-F
  4 32  7 10  8 | F
 13  7  0  1  8 | F-C
  7  0  0  0  6 | C

Cluster 0 <-- F-C
Cluster 1 <-- F
Cluster 2 <-- G-F
Cluster 3 <-- G
Cluster 4 <-- C

Incorrectly clustered instances :       125.0    52.0833 %
```

Weka Clusterer Visualize: 16:05:55 - EM (nbastat2021)

X: Cluster (Nom)

Y: POS (Nom)

Colour: POS (Nom)

Select Instance

Plot: nbastat2021_clustered

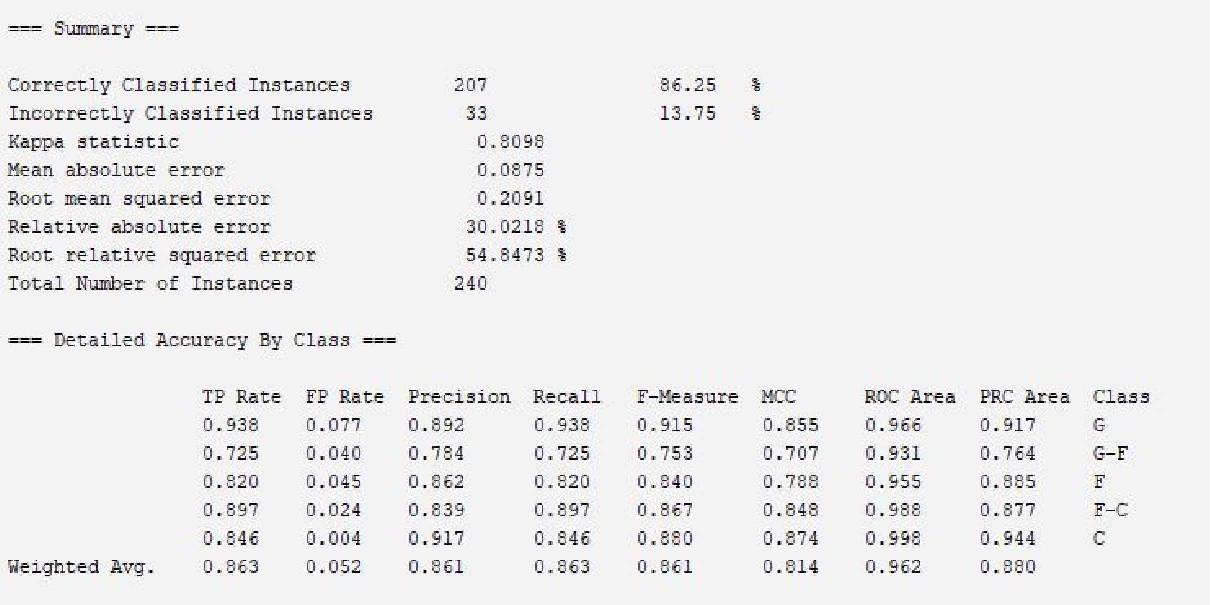
The results of clustering are not so satisfactory. The best result is that there are only instances corresponding to the position for each cluster, but our result was not. However, unlike SimpleKMeans clustering, which clustered 135 instances incorrectly, EM clustering, which clustered 125 instances incorrectly, showed better results.

Problem 3

Classification

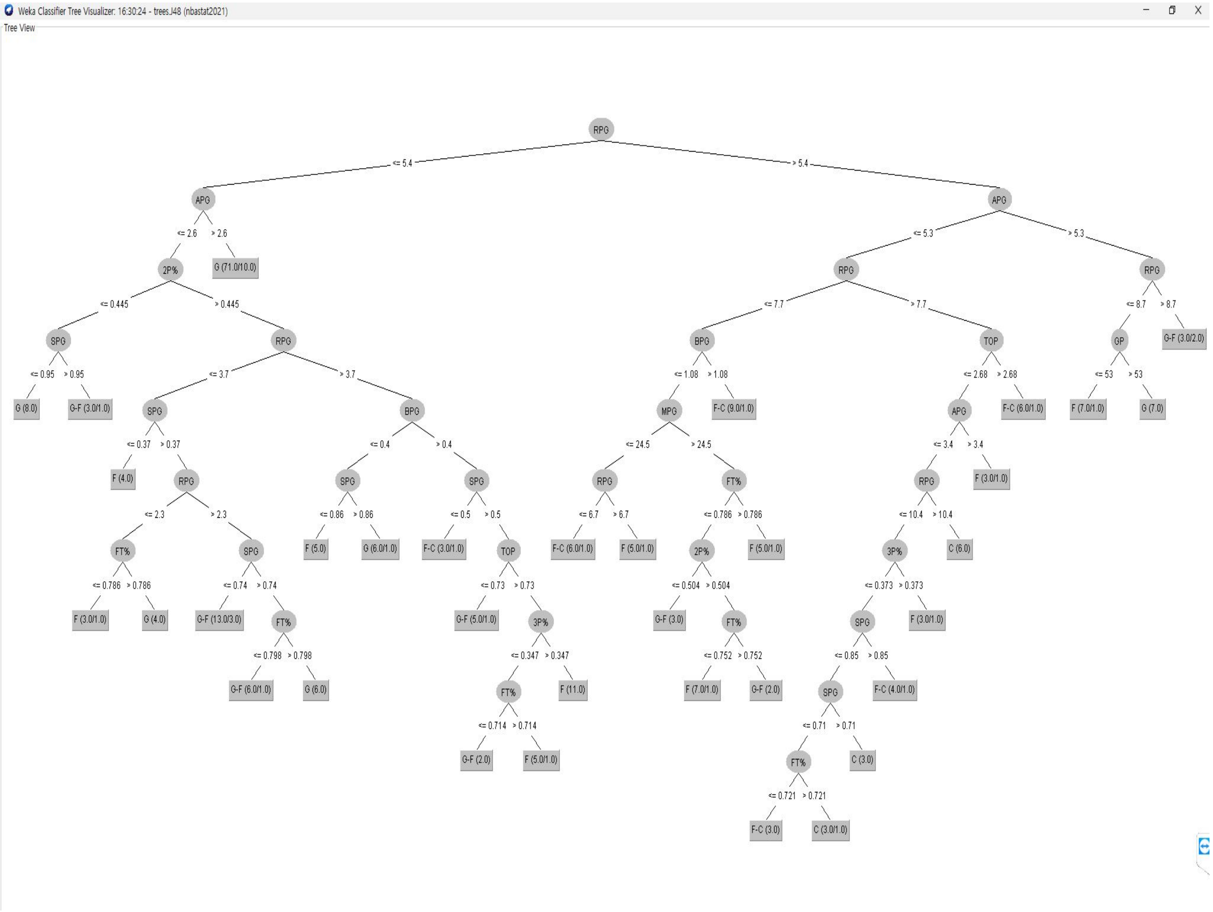
As mentioned in Problem 1, each player is characterized by 11 features and a position. Now, we can classify the players by their positions using the given features in ”Weka”. And the ”Weka” program provides several decision tree-based algorithm. In this problem, we will classify the data using C4.5 decision tree(J48 in ”Weka”) with default parameters. With using parameters, we will train the tree on the data and test on the same data.

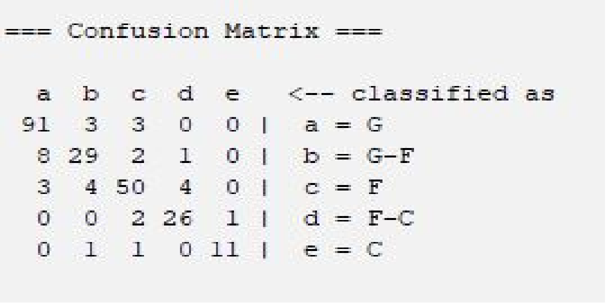
The main feature of the C4.5 decision tree is that it first accepts both continuous and discrete variables. It can handle incomplete data, and solve the overfitting problem through the pruning process of cutting off the end of the decision tree. In addition, different weights may be applied to the variables constituting the training data. We will classify the data through this C4.5 decision tree algorithm. The following is the screenshot of summary and detailed accuarcy by class.



```
=== Summary ===

Correctly Classified Instances         207               86.25   %
Incorrectly Classified Instances        33               13.75   %
Kappa statistic                          0.8098
Mean absolute error                      0.0875
Root mean squared error                  0.2091
Relative absolute error                 30.0218 %
Root relative squared error             54.8473 %
Total Number of Instances              240

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.938    0.077    0.892      0.938    0.915      0.855    0.966     0.917     G
                 0.725    0.040    0.784      0.725    0.753      0.707    0.931     0.764     G-F
                 0.820    0.045    0.862      0.820    0.840      0.788    0.955     0.885     F
                 0.897    0.024    0.839      0.897    0.867      0.848    0.988     0.877     F-C
                 0.846    0.004    0.917      0.846    0.880      0.874    0.998     0.944     C
Weighted Avg.    0.863    0.052    0.861      0.863    0.861      0.814    0.962     0.880
```

The following is a visualization of the results using decision tree. The size of the tree is the total number of nodes in the tree, which is terminal nodes + nonterminal nodes. In this case, the size of tree is 67.

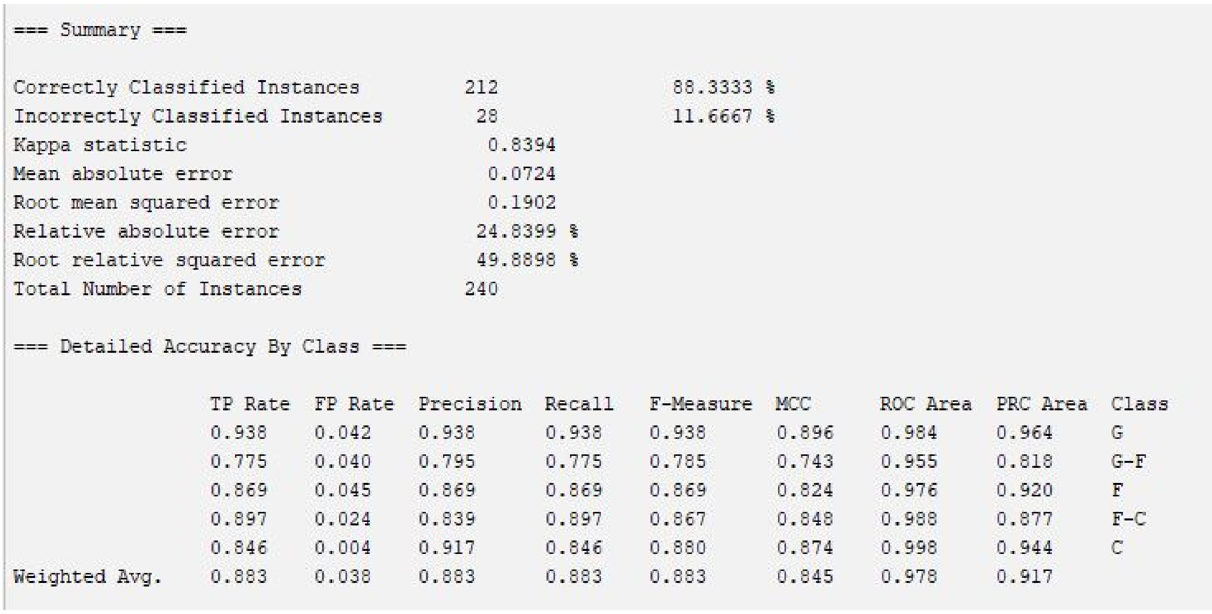




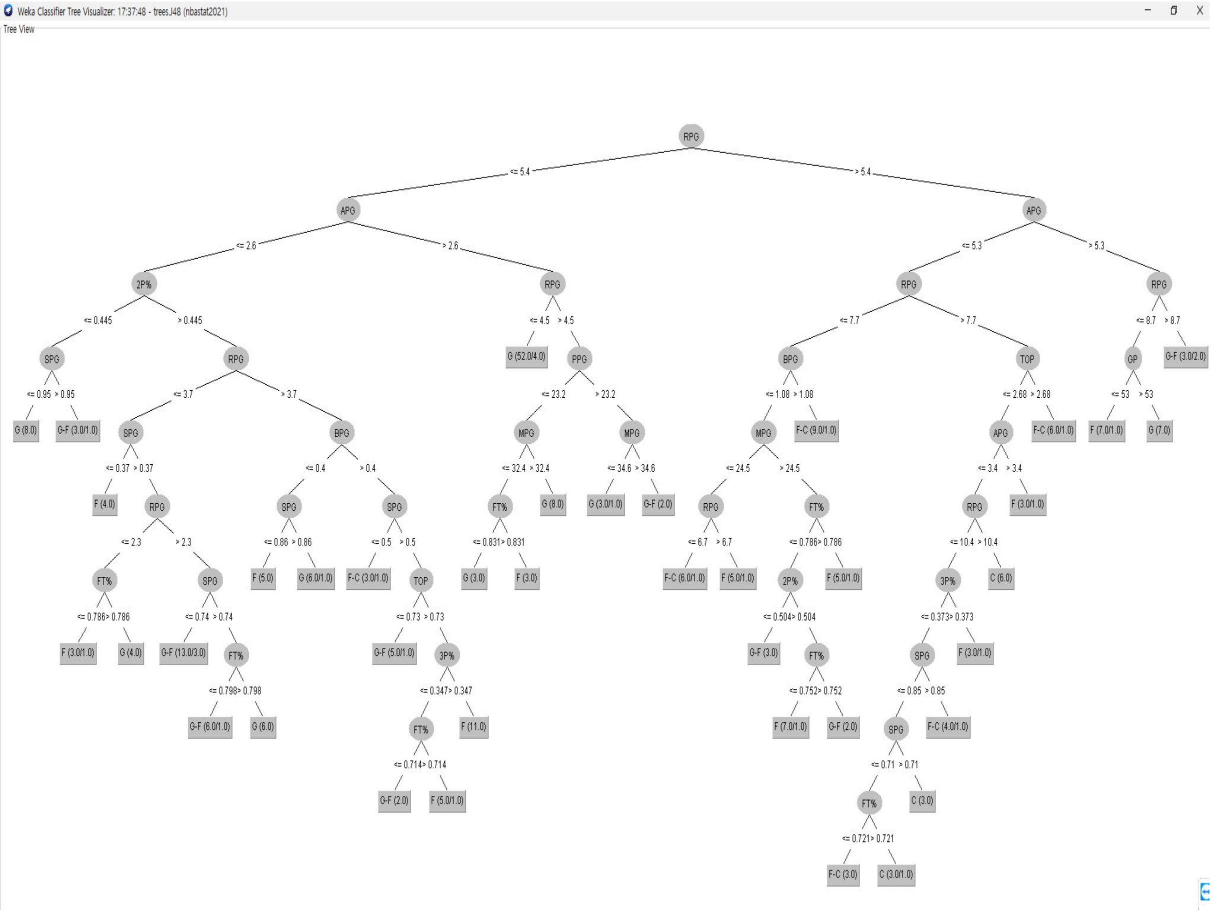
```
=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 91  3  3  0  0 |  a = G
  8 29  2  1  0 |  b = G-F
  3  4 50  4  0 |  c = F
  0  0  2 26  1 |  d = F-C
  0  1  1  0 11 |  e = C
```

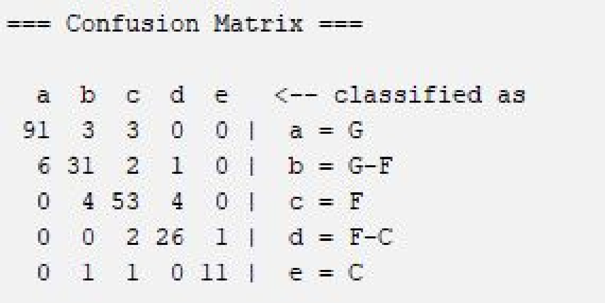
The above is the confusion matrix provided by ”Weka”, where row shows the true class and column shows the predicated class. Of the instances with a total of 97 guard(G) positions, 91 were correctly classified. And 3 of the instances with a total of 61 forward(F) positions were incorrectly classified as guard(G).

Finally, in this problem, we can change the parameters of the C4.5 decision tree. We will classify using C4.5 decision tree with 50% pruning rate. The following is the screenshot of summary and detailed accuarcy by class.



```
=== Summary ===

Correctly Classified Instances         212               88.3333 %
Incorrectly Classified Instances        28               11.6667 %
Kappa statistic                          0.8394
Mean absolute error                      0.0724
Root mean squared error                  0.1902
Relative absolute error                 24.8399 %
Root relative squared error             49.8898 %
Total Number of Instances              240

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.938    0.042    0.938      0.938    0.938      0.896    0.984     0.964     G
                 0.775    0.040    0.795      0.775    0.785      0.743    0.955     0.818     G-F
                 0.869    0.045    0.869      0.869    0.869      0.824    0.976     0.920     F
                 0.897    0.024    0.839      0.897    0.867      0.848    0.988     0.877     F-C
                 0.846    0.004    0.917      0.846    0.880      0.874    0.998     0.944     C
Weighted Avg.    0.883    0.038    0.883      0.883    0.883      0.845    0.978     0.917
```





```
=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 91  3  3  0  0 |  a = G
  6 31  2  1  0 |  b = G-F
  0  4 53  4  0 |  c = F
  0  0  2 26  1 |  d = F-C
  0  1  1  0 11 |  e = C
```

I think RPG is the most important attribute and FT% is the least important attribute. In the Decision tree, the overall tree shape can be complicated or simplified depending on the attribute selected first. As the tree shape becomes more complex, the attribute that can create a simple tree becomes more important because overfitting occurs more easily. We can also find informative attributes directly in the same way as information gain, gini impurity, etc. However, it was judged that the optimal result was generated when the decision tree was used with training data in ”Weka”, although it was omitted because there were many attributes and computations.