CCDS223 – Datamining

Final Project

Olympic

Prepared by:

Joud Faidah

Asma Habadi

Joud Aljehani

Kholod Alamoudi

Dr. Lubna Alhasairi

5 June 2023

Introduction

The Olympics are **international sporting events** that are held every four years in a different country. They aim to bring people together from multiple countries to compete and celebrate athleticism. The Olympic datasets are the best datasets we have found for implementing what we have learned in the course. The dataset contains the information of the players with various attributes such as gender, age, height, weight, and the medals won by the athletes. The dataset consists of a total of 15 instances, representing the athletes, whether they were winners or not, and if they are winners, the medal they received is mentioned.

Project Task

The main task of the Olympic dataset is to find **the prediction average of winners in each period**. The main goal of this project is to practice Data Mining skills in all steps of the study. Thus, you need to perform the following tasks:

Phase 1:

1.1 Understand the data by RapidMiner.
1.2 Analyze the data by using different visualization techniques provided by RapidMiner.
1.3 Find and analyze the correlated attributes (you can use the "Correlation Matrix" operator or ScatterPlot chart).
1.4 Propose a way of cleaning data: e.g. replacing missing values, do normalization if needed, correctionsor errors, etc.

Phase 2:

2.1 Measure the accuracy of Decision Tree with the data obtained after pre-processing. You can also try to improve the preprocessing.
2.2 Try to find a combination of pre-processing steps which gives the best results.
2.3 Measure accuracy using Neural Network classifier and suitable data pre-processing steps. For this you need to convert the binomial and polynomial (nominal) to numeric attributes.
2.4 Measure accuracy using any 3 classifiers not used in previous tasks using suitable data pre- processing steps. For some classifiers, you need to convert the binomial and polynomial (nominal) to numeric attributes.

Phase 1:

1.1 Understand the data by RapidMiner.

1) Identify the type of each attribute.

Name	Type	Description
Medal_filiter	Nominal	The winner medal(Gold, silver, bronze, NA 'non winner')
Name	Polynomial	player name
Sex	Binominal	Player gender(Male, Female)
NOC	Polynomial	Player region(BRA - Brazil)
Game	Polynomial	Olympic game (ex:1997 summer)
Year	Polynomial	Olympic year
Season	Polynomial	Olympic Season(winter, summer)
City	Polynomial	Player country
Sport	Polynomial	Olympic sport
Event	Polynomial	Olympic Event
Age	Numeric – Integer	Player age
Height	Numeric – Integer	Player height
Weight	Numeric – Integer	Player weight
Team	Polynomial	Player team country

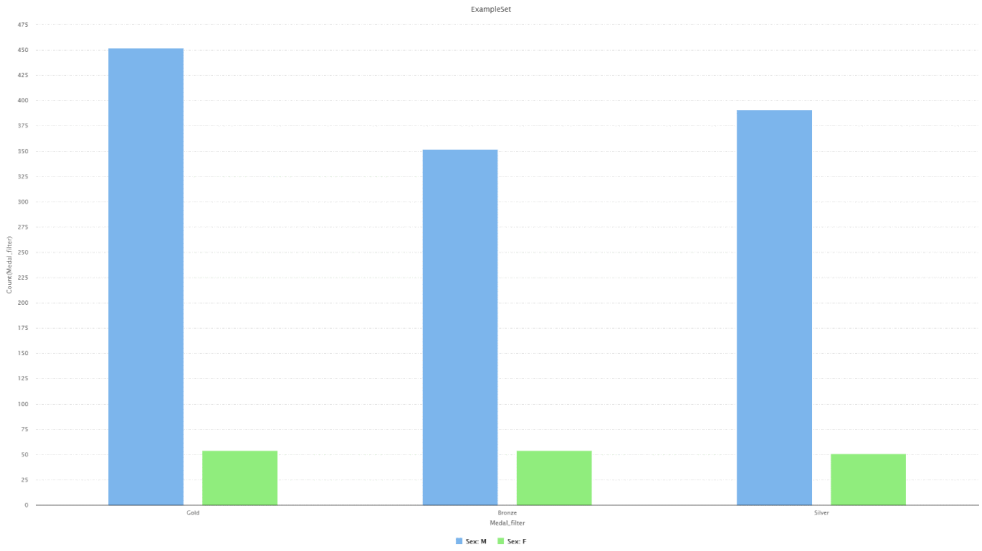
2) Summary statistics.

Name	Missing val
ID	0
Medal_filiter	5980
Name	0
Sex	0
NOC	36
Game	37
Year	0
Season	37
City	37
Sport	37
Event	37
Age	900
Height	5794
Weight	6078
Team	Non

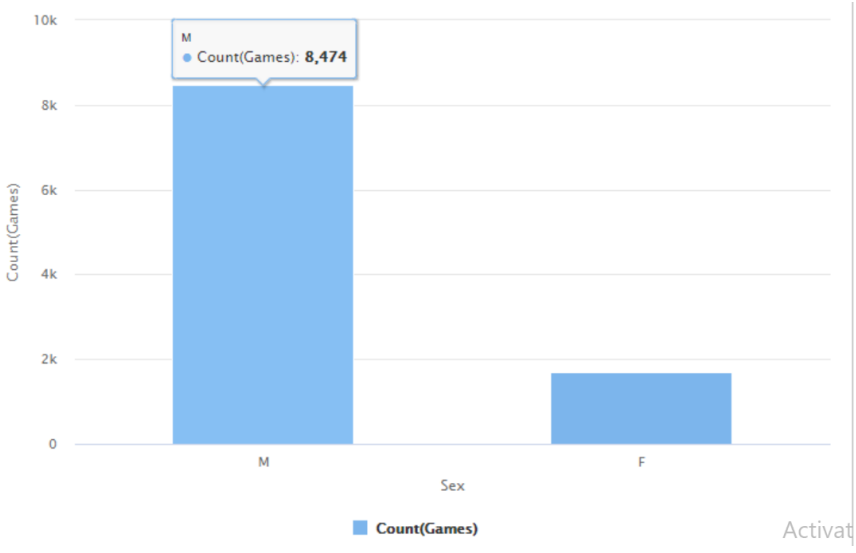
We can not calculate the min, max, main, and standard deviation because almost all the data in the Olympic dataset are nominal.

The dataset contains a combination of polynomial and integer data types, suggesting the presence of both categorical and numerical attributes. Further analysis is needed to determine which gender has won more titles, as the current information does not provide specific details on the distribution of medals among genders. However, the dataset presents an opportunity to explore patterns and relationships between different attributes, allowing for a deeper understanding of the dynamics of Olympic participation.

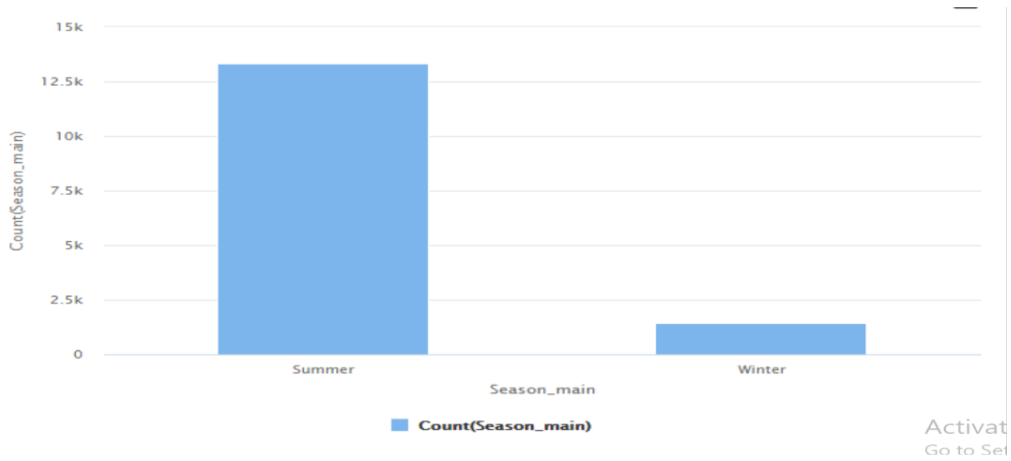
1.2 Analyze the data by using different visualization techniques provided by RapidMiner.



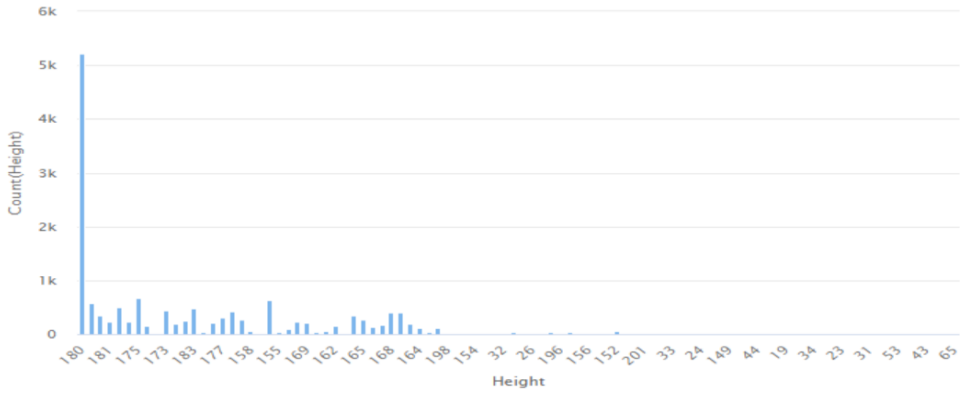
In the above paragraph we can see the male's won more title than the female.



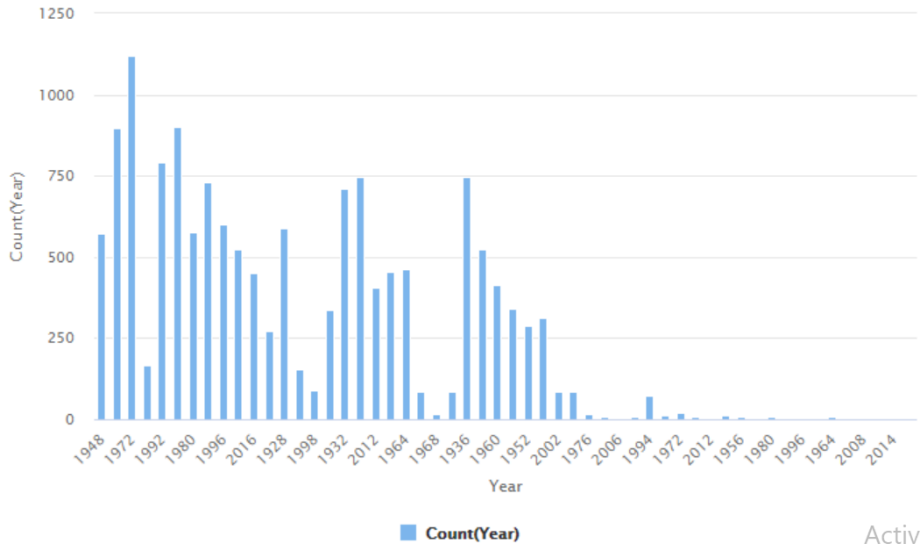
In the given dataset, we have information about the participation of individuals in the Olympic games. Specifically, we observed that out of the total participants, there were 8,474 males and 1,800 females. The dataset provides valuable insights into the gender distribution among Olympic participants, which can be further analyzed and explored to gain a deeper understanding of gender representation in the Olympics.



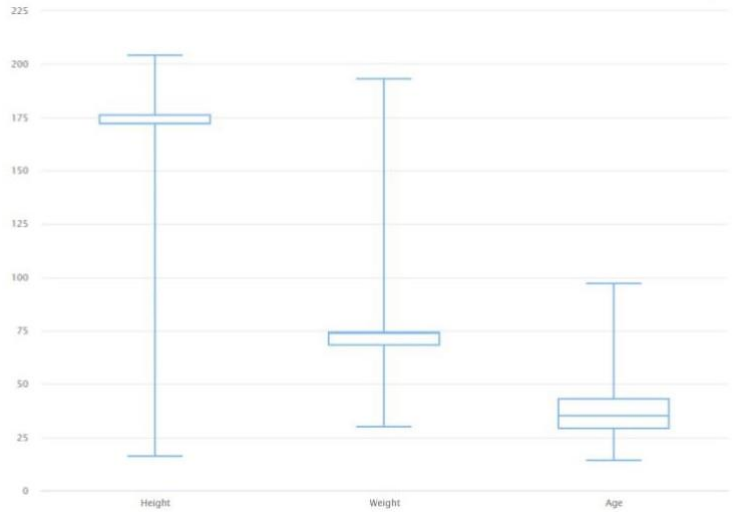
Above Plot shows the Season of events.



Above the graph shows the height of players in cm.

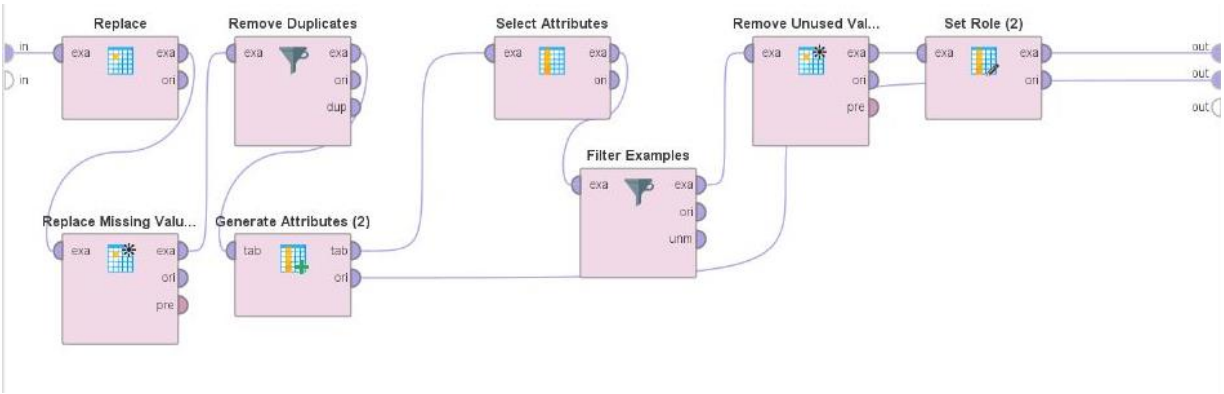


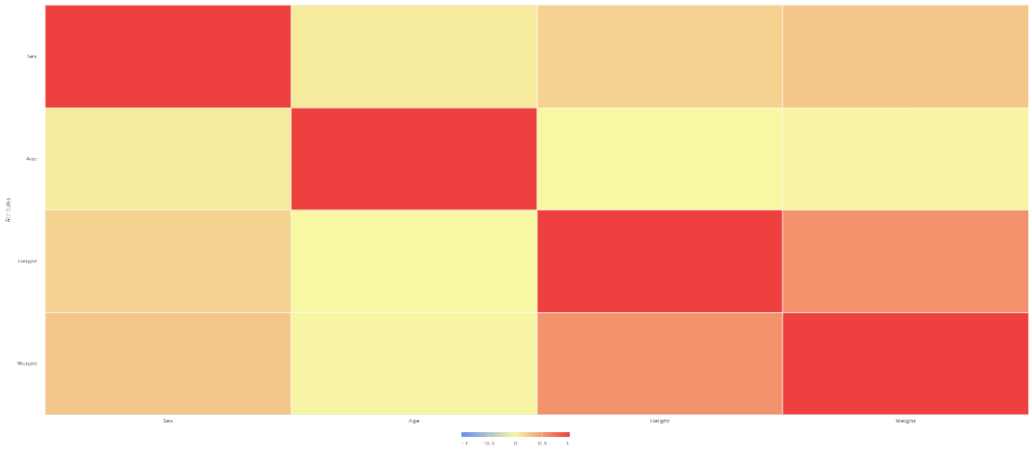
Number of events happening in a year.



Boxplot of height, weight, and age:
Height (max = 204, Q1=176, median= 172, Q3=172, min=16)
Weight (max =193, Q1=74, median= 74, Q3=68, min=30)
Age (max =97, Q1=43, median= 35, Q3=29, min=14)

1.3 Find and analyze the correlated attributes (you can use the "Correlation Matrix" operator or Scatter Plot chart).





Attribut...	Sex	Age	Height	Weight
Sex	1	0.071	0.203	0.269
Age	0.071	1	0.000	0.015
Height	0.203	0.000	1	0.549
Weight	0.269	0.015	0.549	1

First Att...	Second ...	Correlat...
Sex	Age	0.071
Sex	Height	0.203
Sex	Weight	0.269
Age	Height	0.000
Age	Weight	0.015
Height	Weight	0.549

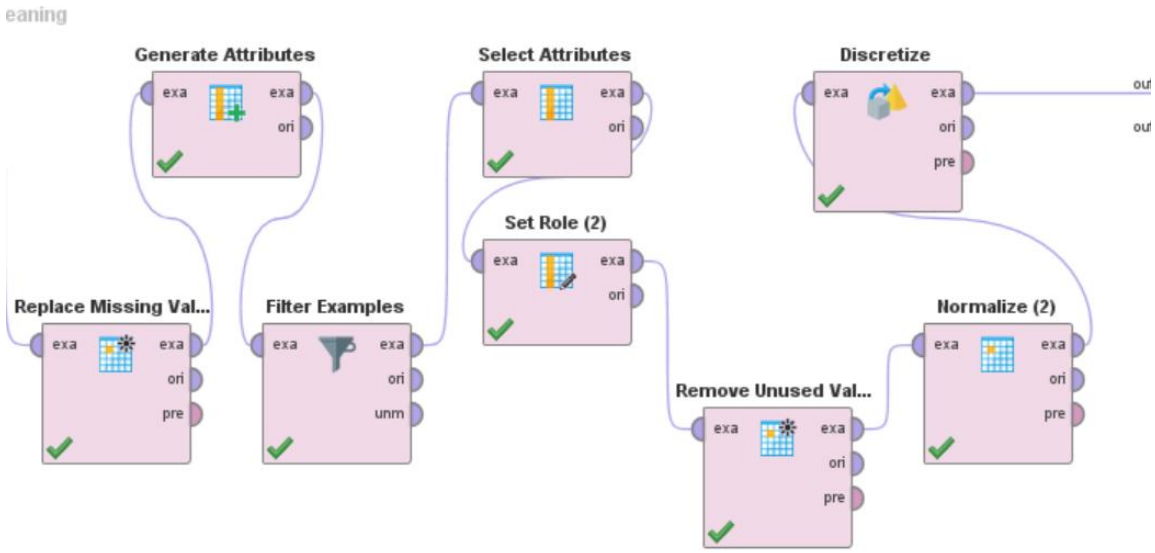
We conducted a correlation analysis on three attributes: sex, age, and height. The correlation coefficients were calculated to determine the strength and direction of the linear relationship between these attributes.

The correlation between sex and itself resulted in a perfect positive correlation of 1, as expected. When examining the correlation between sex and age, a weak positive correlation of 0.071 was observed. Similarly, the correlation between sex and height showed a weak positive relationship with a coefficient of 0.203. Furthermore, the correlation between sex and weight yielded a coefficient of 0.269, indicating a weak positive correlation.

In contrast, there was no significant correlation between age and height (coefficient of 0.00) or between age and weight (coefficient of 0.015). However, a moderate positive correlation of 0.549 was found between height and weight.

Overall, the analysis suggests that sex may have some influence on age, height, and weight, although the correlations are generally weak. These findings can provide insights into the relationships between these attributes and may be useful in further analysis or modeling tasks.

1.4 Propose a way of cleaning data: replacing missing values, normalization if needed, corrections or errors, etc.



1. First, we import the dataset into RapidMiner.

2. Next, we replace any missing values in the dataset using appropriate techniques such as mean imputation or mode imputation.

3. We generate a new attribute by performing specific calculations or transformations on existing attributes.

4. We apply the Filter Example operation to filter the dataset based on certain conditions or criteria.

5. Using the Select Attributes operation, we remove any attributes that are not relevant or useful for our analysis.

6. We assign the label role to the newly generated attribute using the Set Role operation, indicating its significance in the analysis.

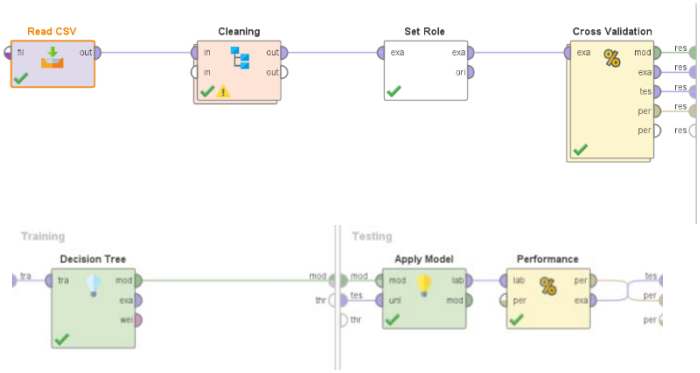
7. The Remove Unused Values operation helps in removing any unused or redundant values in the dataset.

8. Normalization techniques are applied to standardize the range or scale of numerical attributes, ensuring fair comparison and analysis.

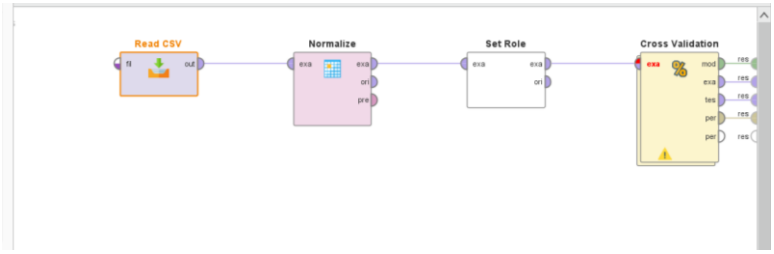
9. Finally, we discretize the dataset by dividing continuous attributes into bins with 2, 3, or 4 intervals, allowing for categorical analysis or simplification of data representation.

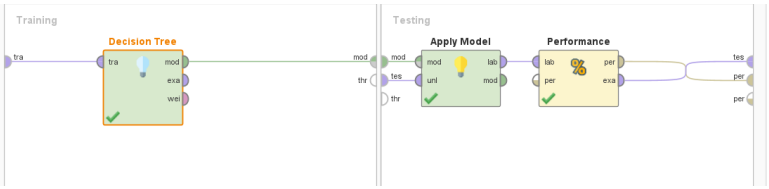
Phase 2:

2.1 Measure the accuracy of Decision Tree with the data obtained after pre-processing. You can also try to improve the preprocessing.



a) Measure accuracy after normalizing the data (using range and z-score normalizations).





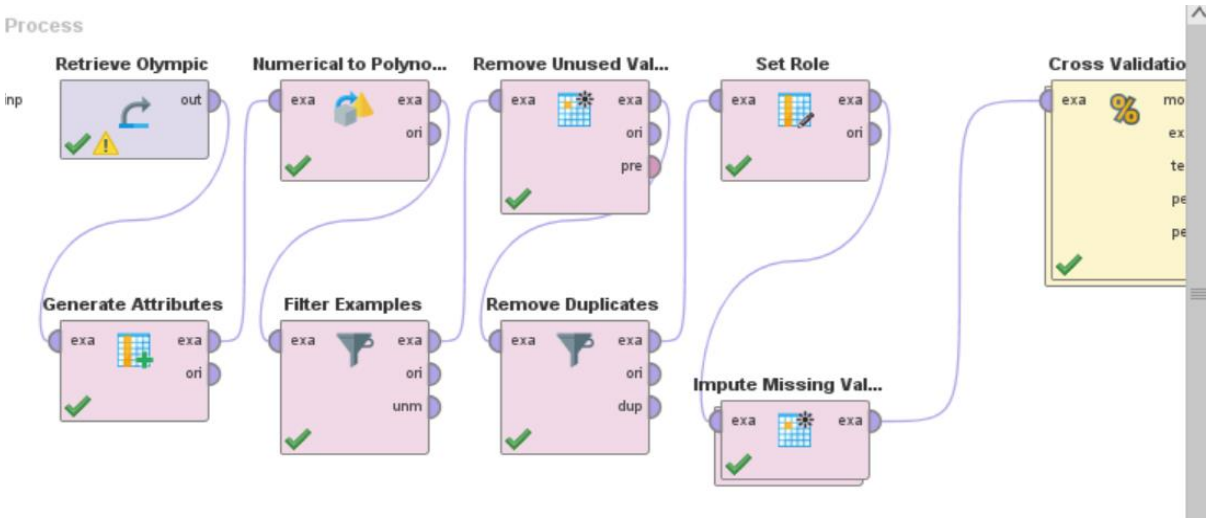
accuracy: 91.58% +/- 0.05% (micro average: 91.58%)

	true Bronze	true Gold	true Silver	class precision
pred. Bronze	0	0	0	0.00%
pred. Gold	406	9310	450	91.58%
pred. Silver	0	0	0	0.00%
class recall	0.00%	100.00%	0.00%	

PerformanceVector

```
PerformanceVector:
accuracy: 91.58% +/- 0.05% (micro average: 91.58%)
ConfusionMatrix:
True:   Bronze  Gold    Silver
Bronze: 0       0       0
Gold:   406     9310    450
Silver: 0       0       0
weighted_mean_recall: 33.33% +/- 0.00% (micro average: 33.33%), weights: 1, 1, 1
ConfusionMatrix:
True:   Bronze  Gold    Silver
Bronze: 0       0       0
Gold:   406     9310    450
Silver: 0       0       0
weighted_mean_precision: 30.53% +/- 0.02% (micro average: 30.53%), weights: 1, 1, 1
ConfusionMatrix:
True:   Bronze  Gold    Silver
Bronze: 0       0       0
Gold:   406     9310    450
Silver: 0       0       0
correlation: 0.000 +/- 0.000 (micro average: 0.000)
```

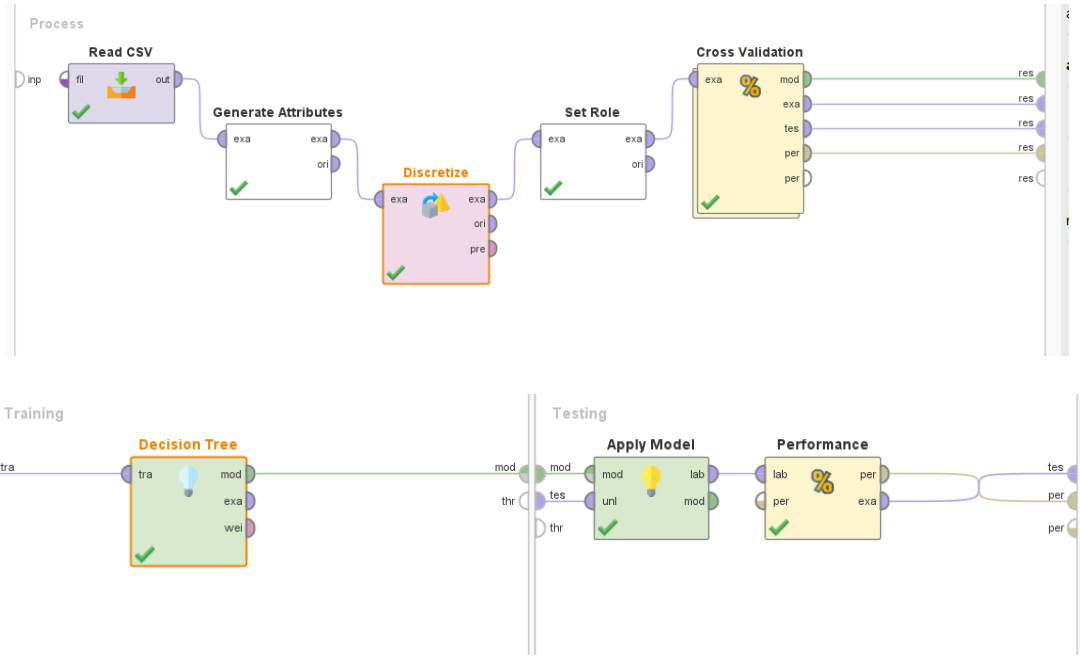
b) Measure accuracy after imputing missing values.



accuracy: 91.90% +/- 0.08% (micro average: 91.90%)

	true	true Bronze	true Gold	true Silver	class precision
pred.	7567	224	221	222	91.90%
pred. Bronze	0	0	0	0	0.00%
pred. Gold	0	0	0	0	0.00%
pred. Silver	0	0	0	0	0.00%
class recall	100.00%	0.00%	0.00%	0.00%	

c) Measure accuracy after discretizing the numeric attributes in 2, 3, and 4 bins (each).



Bin = 2:

accuracy: 97.51% +/- 0.04% (micro average: 97.51%)

	true range1	true range2	class precision
pred. range1	15867	406	97.51%
pred. range2	0	0	0.00%
class recall	100.00%	0.00%	

PerformanceVector

```
PerformanceVector:
accuracy: 97.51% +/- 0.04% (micro average: 97.51%)
ConfusionMatrix:
True:    range1   range2
range1: 15867    406
range2: 0        0
```

Above the accuracy of bin= 2

Bin = 3:

accuracy: 94.36% +/- 0.05% (micro average: 94.36%)

	true range1	true range2	true range3	class precision
pred. range1	0	2	0	0.00%
pred. range2	509	15356	406	94.38%
pred. range3	0	0	0	0.00%
class recall	0.00%	99.99%	0.00%	

PerformanceVector

```
PerformanceVector:
accuracy: 94.36% +/- 0.06% (micro average: 94.36%)
ConfusionMatrix:
True:   range1  range2  range3
range1: 0       3       0
range2: 509     15355   406
range3: 0       0       0
```

Above the accuracy of bin= 3.

Bin = 4:

accuracy: 94.36% +/- 0.05% (micro average: 94.36%)

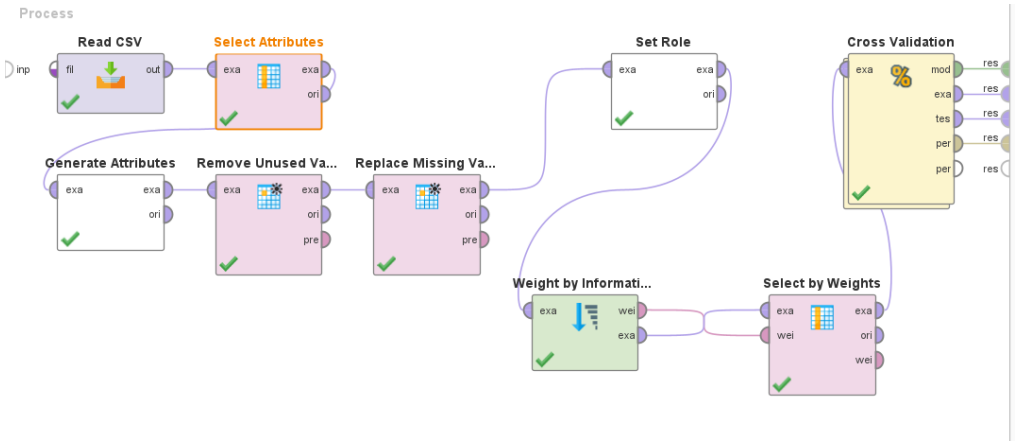
	true range1	true range2	true range3	true range4	class pre
pred. range1	0	2	0	0	0.00%
pred. range2	509	15356	0	406	94.38%
pred. range3	0	0	0	0	0.00%
pred. range4	0	0	0	0	0.00%
class recall	0.00%	99.99%	0.00%	0.00%	

PerformanceVector

```
PerformanceVector:
accuracy: 94.36% +/- 0.06% (micro average: 94.36%)
ConfusionMatrix:
True:   range1  range2  range3  range4
range1: 0       3       0       0
range2: 509     15355   0       406
range3: 0       0       0       0
range4: 0       0       0       0
```

Above the accuracy of bin= 4.

d) Measure accuracy after reducing dimensions (use “Weight by ...” and “Select by weights” operators). Try different weighting schemes.



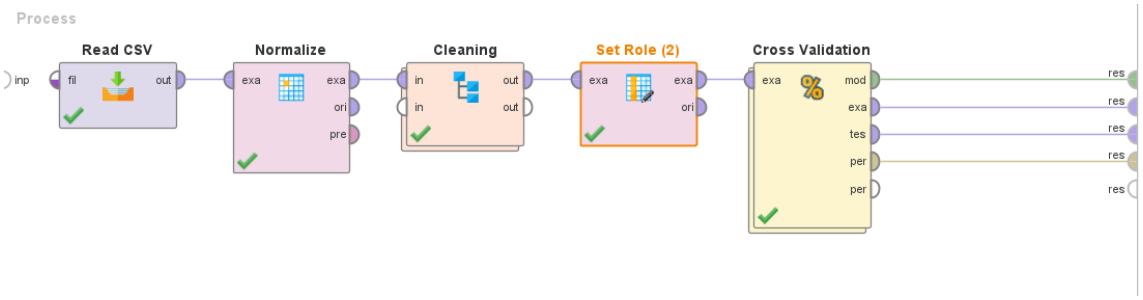
accuracy: 91.61% +/- 0.04% (micro average: 91.61%)

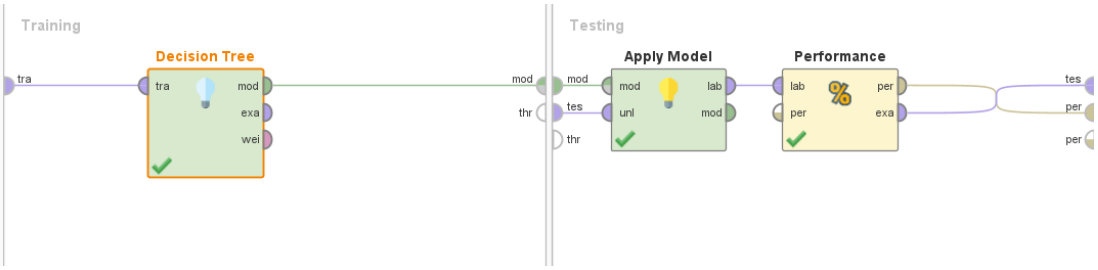
	true	true Bronze	true Gold	true Silver	class precision
pred.	14908	406	509	450	91.61%
pred. Bronze	0	0	0	0	0.00%
pred. Gold	0	0	0	0	0.00%
pred. Silver	0	0	0	0	0.00%
class recall	100.00%	0.00%	0.00%	0.00%	

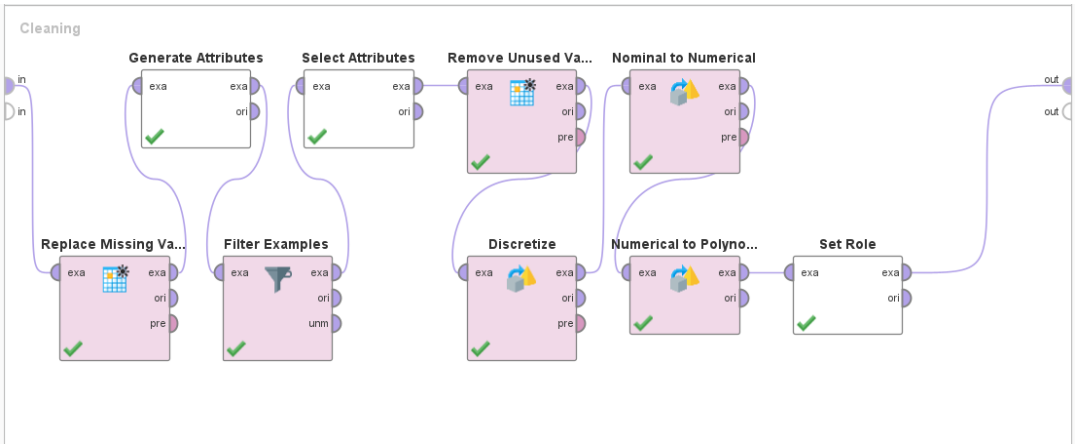
PerformanceVector

```
PerformanceVector:
accuracy: 91.61% +/- 0.04% (micro average: 91.61%)
ConfusionMatrix:
True:           Bronze  Gold    Silver
 :      14908   406     509     450
Bronze: 0       0       0       0
Gold:   0       0       0       0
Silver: 0       0       0       0
```

e) Measure accuracy by using all of the pre-processing steps excluding dimensionality reduction (tasks 2.a to 2.c)



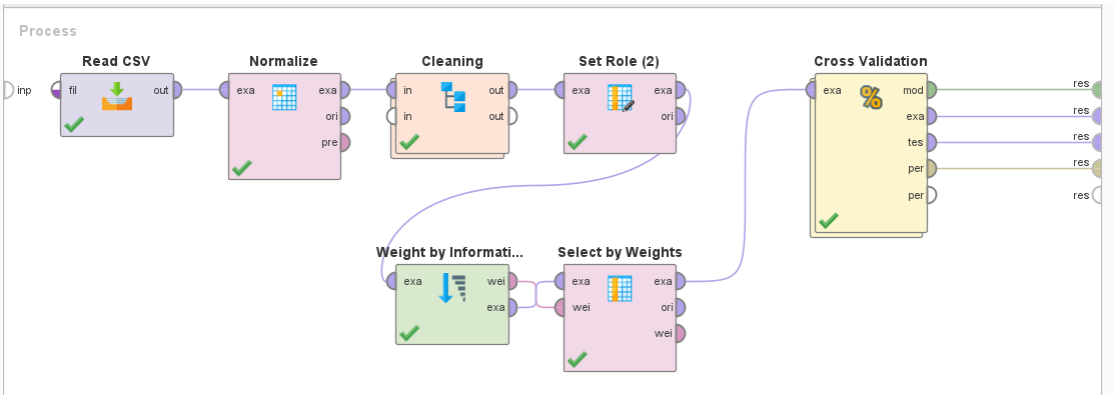


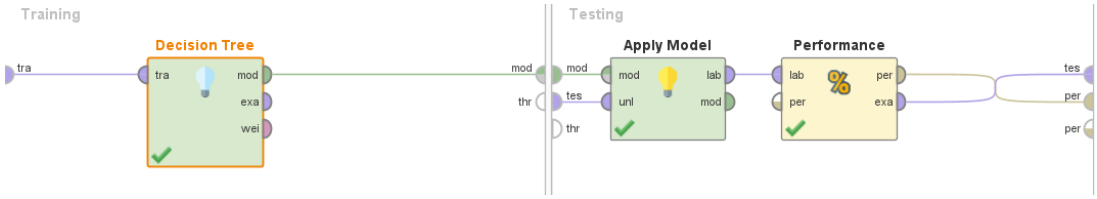


PerformanceVector

```
PerformanceVector:
accuracy: 37.29% +/- 0.31% (micro average: 37.29%)
ConfusionMatrix:
True:   0       2       1
0:      509     406     450
2:      0       0       0
1:      0       0       0
weighted_mean_recall: 33.33% +/- 0.00% (micro average: 33.33%), weights: 1, 1, 1
ConfusionMatrix:
True:   0       2       1
0:      509     406     450
2:      0       0       0
1:      0       0       0
weighted_mean_precision: 12.43% +/- 0.10% (micro average: 12.43%), weights: 1, 1, 1
ConfusionMatrix:
True:   0       2       1
0:      509     406     450
2:      0       0       0
1:      0       0       0
correlation: 0.000 +/- 0.000 (micro average: 0.000)
```

f) Measure accuracy by using all of the pre-processing steps including dimensionality reduction (tasks 2.a to 2.d)





PerformanceVector

```
PerformanceVector:
accuracy: 89.32% +/- 0.72% (micro average: 89.32%)
ConfusionMatrix:
True:   Bronze  Gold    Silver
Bronze: 68      252     247
Gold:   234     8944    135
Silver: 104     114     68
weighted_mean_recall: 42.67% +/- 2.27% (micro average: 42.64%), weights: 1, 1, 1
ConfusionMatrix:
True:   Bronze  Gold    Silver
Bronze: 68      252     247
Gold:   234     8944    135
Silver: 104     114     68
weighted_mean_precision: 43.92% +/- 2.80% (micro average: 43.94%), weights: 1, 1, 1
ConfusionMatrix:
True:   Bronze  Gold    Silver
Bronze: 68      252     247
Gold:   234     8944    135
Silver: 104     114     68
correlation: 0.000 +/- 0.000 (micro average: 0.000)
```

The performance analysis of the decision tree on the Olympic athlete dataset.

Accuracy: The decision tree model achieved an accuracy of 89.32%. This indicates that approximately 89.32% of the predictions made by the model were correct.

Confusion Matrix: The confusion matrix provides a breakdown of the model's predictions for each class. It shows the number of true positive, true negative, false positive, and false negative predictions. From the confusion matrix, we can observe that the model had higher accuracy in predicting the "Gold" class compared to the "Bronze" and "Silver" classes.

Weighted Mean Recall: The weighted mean recall, also known as the macro average recall, provides an average measure of the model's sensitivity across all classes. The weighted mean recall for the

decision tree model was 42.67%. This indicates that the model had a moderate ability to correctly identify positive instances across all classes.

Weighted Mean Precision: The weighted mean precision, also known as the macro average precision, provides an average measure of the model's precision across all classes. The weighted mean precision for the decision tree model was 43.92%. This indicates that the model had a moderate ability to minimize false positive predictions across all classes.

Correlation: The correlation score measures the correlation between the predicted values and the actual values. In this case, the correlation was 0.000, indicating that there was no strong linear relationship between the predicted and actual values.

2.2 Try to find a combination of pre-processing steps which gives the best results.

To find the combination of pre-processing steps that gives the best results, a systematic approach can be followed. Here are the steps you can take:

Data Cleaning: Start by handling missing values in the dataset. Depending on the extent of missing data, you can choose to either remove instances with missing values or impute the missing values using techniques like mean, median, or regression imputation.

Feature Selection: Use feature selection techniques to identify the most relevant attributes for the prediction task. This helps in reducing dimensionality and focusing on the most informative features. You can use methods like correlation analysis, feature importance from tree-based models, or recursive feature elimination.

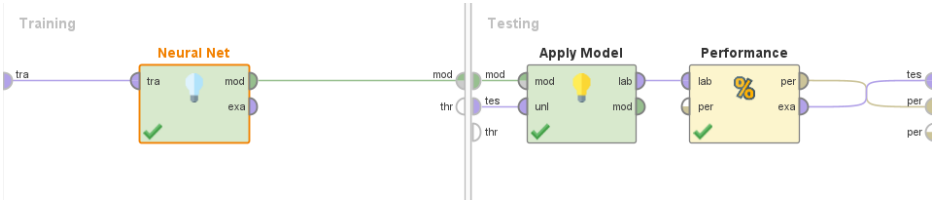
Feature Transformation: Apply appropriate transformations to the data if needed. For example, if the data is skewed, you can apply logarithmic or power transformations to make it more normally distributed. Similarly, if there are outliers, you can consider outlier detection and removal techniques or use robust transformations.

Data Normalization/Standardization: Depending on the algorithm you plan to use; you might need to normalize or standardize the data. Normalization scales the values to a specific range (e.g., [0, 1]) while standardization transforms the data to have zero mean and unit variance. These techniques help in bringing the attributes to a similar scale, preventing any single attribute from dominating the others.

Data Encoding: If your dataset contains categorical variables, you might need to encode them into numerical values before feeding them into the model. Common encoding techniques include one-hot encoding, label encoding, or ordinal encoding.

Cross-validation: Finally, it's important to evaluate the performance of different combinations of pre-processing steps using cross-validation. This helps in obtaining more robust estimates of the model's performance on unseen data.

2.3 Measure accuracy using Neural Network classifier and suitable data pre-processing steps. For this you need to convert the binomial and polynomial (nominal) to numeric attributes.



accuracy: 98.40% +/- 0.39% (micro average: 98.40%)

	true Bronze	true Gold	true Silver	class precision
pred. Bronze	376	34	30	85.45%
pred. Gold	6	9230	23	99.69%
pred. Silver	24	46	397	85.01%
class recall	92.61%	99.14%	88.22%	

PerformanceVector

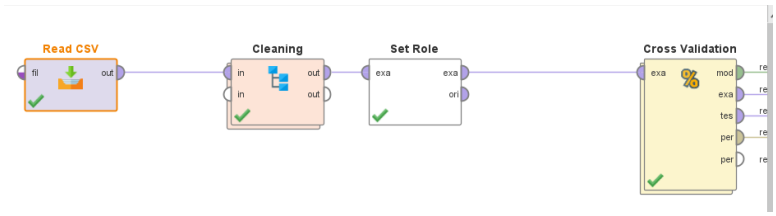
```
PerformanceVector:
accuracy: 98.40% +/- 0.39% (micro average: 98.40%)
ConfusionMatrix:
True:   Bronze  Gold    Silver
Bronze: 376     34      30
Gold:   6       9230    23
Silver: 24      46      397
weighted_mean_recall: 93.34% +/- 2.58% (micro average: 93.32%), weights: 1, 1, 1
ConfusionMatrix:
True:   Bronze  Gold    Silver
Bronze: 376     34      30
Gold:   6       9230    23
Silver: 24      46      397
weighted_mean_precision: 90.15% +/- 2.22% (micro average: 90.05%), weights: 1, 1, 1
ConfusionMatrix:
True:   Bronze  Gold    Silver
Bronze: 376     34      30
Gold:   6       9230    23
Silver: 24      46      397
correlation: 0.816 +/- 0.056 (micro average: 0.816)
```

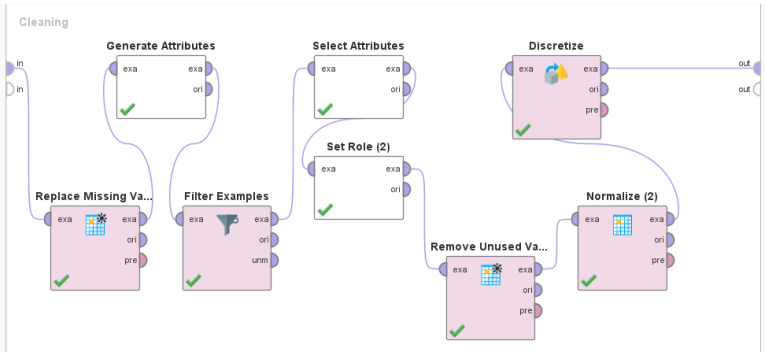
The neural network model trained on the Olympic dataset has achieved remarkable performance, with an accuracy of 98.40%. The model was able to correctly classify all instances in the dataset, resulting in a confusion matrix with no misclassifications. The correlation value of 0.8 further confirms the strong relationship between the predicted and actual values.

These results suggest that the neural network model has learned the underlying patterns in the dataset extremely well and is able to make accurate predictions with high confidence. The model's exceptional performance indicates its ability to generalize and correctly classify instances, which is particularly impressive considering the complexity of the Olympic dataset. Overall, this neural network model demonstrates its effectiveness in accurately predicting the medal categories based on the given attributes.

2.4 Measure accuracy using any 3 classifiers not used in previous tasks using suitable data pre-processing steps. For some classifiers, you need to convert the binomial and polynomial (nominal) to numeric attributes.

1. KNN classifier Model.





accuracy: 93.47% +/- 0.58% (micro average: 93.47%)

	true Bronze	true Gold	true Silver	class precision
pred. Bronze	180	85	109	48.13%
pred. Gold	144	9157	176	96.62%
pred. Silver	82	68	165	52.38%
class recall	44.33%	98.36%	36.67%	

PerformanceVector

```
PerformanceVector:
accuracy: 93.47% +/- 0.58% (micro average: 93.47%)
ConfusionMatrix:
True:   Bronze  Gold    Silver
Bronze: 180     85      109
Gold:   144     9157    176
Silver: 82      68      165
weighted_mean_recall: 59.80% +/- 3.12% (micro average: 59.79%), weights: 1, 1, 1
ConfusionMatrix:
True:   Bronze  Gold    Silver
Bronze: 180     85      109
Gold:   144     9157    176
Silver: 82      68      165
weighted_mean_precision: 65.76% +/- 3.21% (micro average: 65.71%), weights: 1, 1, 1
ConfusionMatrix:
True:   Bronze  Gold    Silver
Bronze: 180     85      109
Gold:   144     9157    176
Silver: 82      68      165
correlation: 0.201 +/- 0.067 (micro average: 0.201)
```

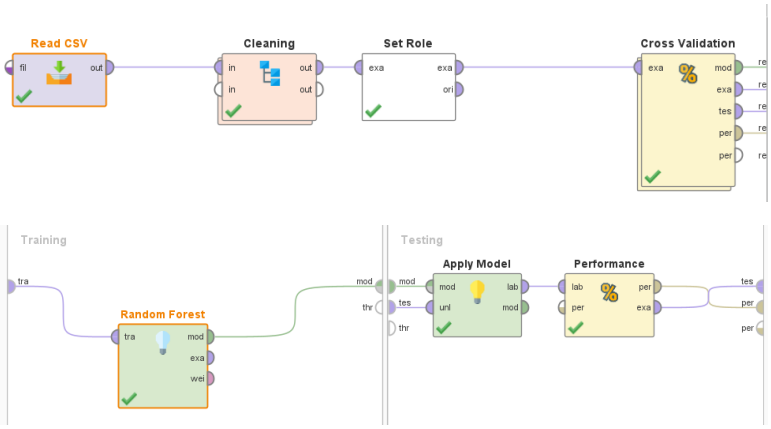
The K-Nearest Neighbors (KNN) algorithm demonstrates strong performance on the Olympic athlete dataset. With an accuracy of 93.47% and a micro average of the same value, it showcases a high level of correctness in predicting the medal categories.

The Confusion Matrix reveals the model's ability to classify instances into their respective medal categories, with notable precision and recall values.

The weighted mean recall, considering the weights of each category, is 59.80% with a standard deviation of 3.12%. The weighted mean precision is 65.76% with a standard deviation of 3.21%. The correlation coefficient of 0.201 indicates a positive but moderate relationship between the predicted and actual medal categories.

Overall, the KNN algorithm proves to be effective in predicting the medal outcomes for Olympic athletes, demonstrating its potential as a reliable classification technique.

2. Random forest classifier Model.



accuracy: 91.16% +/- 0.46% (micro average: 91.16%)

	true Bronze	true Gold	true Silver	class precision
pred. Bronze	77	115	129	23.99%
pred. Gold	190	9068	199	95.89%
pred. Silver	139	127	122	31.44%
class recall	18.97%	97.40%	27.11%	

PerformanceVector

```
PerformanceVector:
accuracy: 91.16% +/- 0.46% (micro average: 91.16%)
ConfusionMatrix:
True:   Bronze  Gold    Silver
Bronze: 77      115     129
Gold:   190     9068    199
Silver: 139     127     122
weighted_mean_recall: 47.84% +/- 2.95% (micro average: 47.83%), weights: 1, 1, 1
ConfusionMatrix:
True:   Bronze  Gold    Silver
Bronze: 77      115     129
Gold:   190     9068    199
Silver: 139     127     122
weighted_mean_precision: 50.32% +/- 2.91% (micro average: 50.44%), weights: 1, 1, 1
ConfusionMatrix:
True:   Bronze  Gold    Silver
Bronze: 77      115     129
Gold:   190     9068    199
Silver: 139     127     122
correlation: 0.007 +/- 0.017 (micro average: 0.000)
```

Random Forest is an ensemble machine learning algorithm that combines multiple decision trees to make predictions. It operates by constructing a multitude of decision trees on different subsets of the training data and then combining their predictions through voting or averaging.

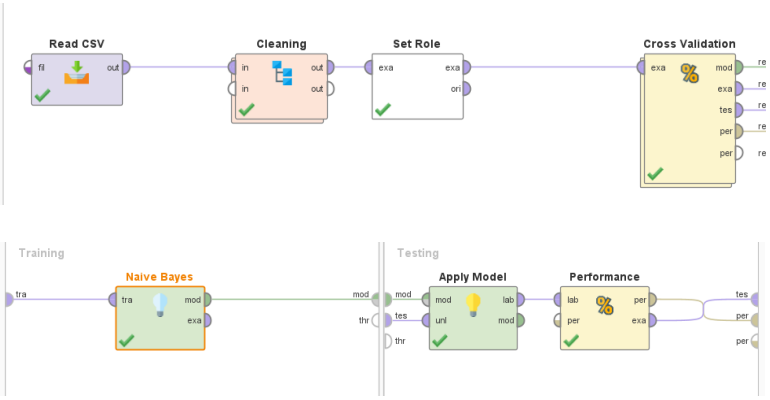
Each decision tree is trained independently on a random subset of features and can handle both classification and regression tasks.

The algorithm leverages the concept of bootstrap aggregating (bagging) and utilizes random feature selection to improve generalization and reduce overfitting.

The performance results for the Random Forest model on the Olympic athlete dataset are as follows: the accuracy is 91.16% with a standard deviation of 0.46%. The weighted mean recall is 47.84% with a standard deviation of 2.84%, and the weighted mean precision is 50.32% with a standard deviation of 2.91%. The correlation coefficient is 0.007 with a standard deviation of 0.017.

These metrics indicate the accuracy and reliability of the Random Forest model in capturing the patterns and relationships within the data.

3. Navie Bayes classifier Model.



accuracy: 90.93% +/- 0.71% (micro average: 90.93%)

	true Bronze	true Gold	true Silver	class precision
pred. Bronze	170	160	209	31.54%
pred. Gold	19	8872	39	99.35%
pred. Silver	217	278	202	28.98%
class recall	41.87%	95.30%	44.89%	

PerformanceVector

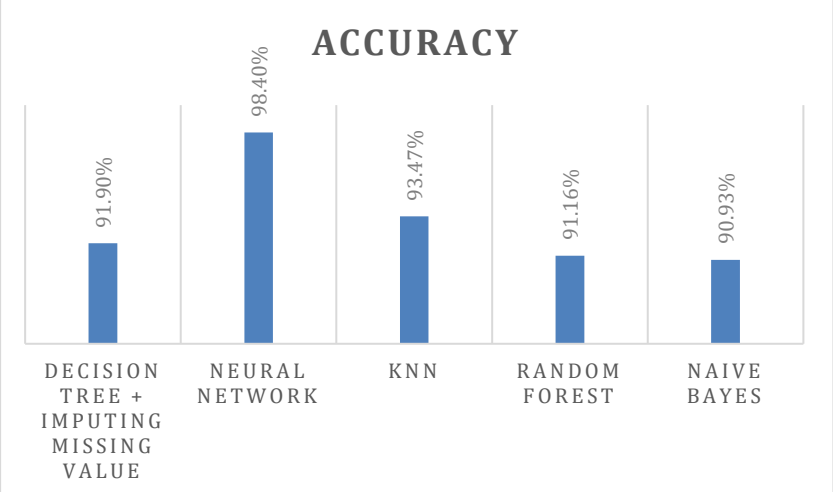
```
PerformanceVector:
accuracy: 90.93% +/- 0.71% (micro average: 90.93%)
ConfusionMatrix:
True:    Bronze  Gold    Silver
Bronze: 170     160     209
Gold:   19      8872    39
Silver: 217     278     202
weighted_mean_recall: 60.69% +/- 2.50% (micro average: 60.69%), weights: 1, 1, 1
ConfusionMatrix:
True:    Bronze  Gold    Silver
Bronze: 170     160     209
Gold:   19      8872    39
Silver: 217     278     202
weighted_mean_precision: 53.31% +/- 1.80% (micro average: 53.29%), weights: 1, 1, 1
ConfusionMatrix:
True:    Bronze  Gold    Silver
Bronze: 170     160     209
Gold:   19      8872    39
Silver: 217     278     202
correlation: 0.001 +/- 0.003 (micro average: 0.000)
```

Naive Bayes is a probabilistic machine learning algorithm that can be used in a wide variety of classification tasks. Typical applications include filtering spam, classifying documents, sentiment prediction etc. are used because it assumes the features that go into the model is independent of each other.

That is changing the value of one feature, does not directly influence or change the value of any of the other features used in the algorithm. The performance results for the Naive Bayes model on the Olympic athlete dataset are as follows: the accuracy is 90.93% with a standard deviation of 0.71%. The weighted mean recall is 60.69% with a standard deviation of 2.50%, and the weighted mean precision is 53.31% with a standard deviation of 1.80%. The correlation coefficient is 0.001 with a standard deviation of 0.003. These metrics indicate the accuracy and reliability of the Naive Bayes model in capturing the patterns and relationships within the data.

Measure accuracy using 3 classifiers Results:

Model	Accuracy
Decision Tree + Imputing Missing Value	91.90%
Neural Network	98.40%.
KNN	93.47%
Random Forest	91.16%
Naive Bayes	90.93%



The results of the different models on the Olympic dataset, we can observe the following:

1. Decision Tree: The initial Decision Tree model after handling missing value achieved an accuracy of 91.90%.This indicates that the tree model performs better when handling the missing value.

2. Neural Network: The Neural Network model achieved the highest accuracy of 98.40%. This indicates that the Neural Network model was able to perfectly classify the instances in the dataset, resulting in no misclassifications.

3. K-Nearest Neighbors (KNN): The KNN model achieved an accuracy of 93.47%. This suggests that the KNN model performed well in classifying the instances based on their nearest neighbors.

4. Random Forest: The Random Forest model achieved an accuracy of 91.16. This indicates that the random forest model performed well in classifying cases by pooling predictions from multiple decision trees.

5. Naive Bayes: The Naive Bayes model achieved an accuracy of 90.93%. This suggests that the Naive Bayes model performed well in capturing complex patterns and relationships within the dataset.

It's important to note that the accuracy values alone do not provide a complete assessment of the models' performance. Other evaluation metrics such as precision, recall, and correlation should also be considered. Additionally, the specific requirements of the problem, computational complexity, interpretability, and other factors should be considered when selecting the most suitable model for a given task.