Smart Snake - Classification and Prediction

Index

First Iteration

Phase 1: Instances Collection

Although we had already made a decision about which attributes would be more interesting in Tutorial 3, after some more thought we decided to change the representation a bit.

For this first iteration, we chose the following features of the actual game:

- snake_body : We choose to represent the body as a grid, containing 0s for empty positions,1s for positions where some part of the snake_body was lying at the moment, and a 2 to represent the head of the snake. We also used 1s to represent walls or cells that were out of the boundaries.

  However, after some tries, we discovered that performing this 48*48 representation was too heavy for Weka. That's why we made the decision of doing a close up to the snake, we decided on a "radius" so that we would only look in a square centered on the snake's head. Because matrices are not an appropriate type of attribute for Weka, we decided to use a numeric attribute, appending the different numbers, row by row.

- head_pos_x: This is the number that represents the x-coordinate of the snake's head.
- head_pos_y: This is the number that represents the y-coordinate of the snake's head.
- food_pos_x: This is the number that represents the x-coordinate of the food.
- food_pos_y: This is the number that represents the y-coordinate of the food.
- score: The score the agent has obtained up to this moment.
- actual_direction: The action the snake has just taken. We decided to represent this information numerically, with this one-to-one correspondence: UP = 1, DOWN = 2, LEFT =3, RIGHT = 4

We also decided to compute the following instances related to the future game:

- next_score: We computed the next_score by looking at the direction that will be taken and the state of the game on the next tick. Hence, if the food was eaten by the snake, we add 100 to the score, and else we subtract 1 from the score,

Phase 2: Classification

For this phase, we had in mind two strategies. Firstly, trying with different data sets to see which mix of attributes would be worth keeping. Then, we would try to create new attributes based on the attributes we already had and see how this affected the different model's performance.

In this phase, we decided to not only try with the supplied test data set, but also with the training set and the cross-validation. We were worried about overfitting and wanted to see how the evaluation would look in these three different situations for each of the attributes. Nevertheless, the final table in which we summed up all the information we gathered, only contains information about the supplied data set.

Explanation about the chosen models (ZeroR, J48, Random Forest):

We evaluated ZeroR as we did in the previous tutorials, due to its simplicity, it gives as a baseline for comparison.

We opted for J48 due to its good performance in previous tutorials and its capability to visually represent decision trees. This enabled us to comprehend the classification process, evaluate the occurrence of overfitting. Also considered our particular case since as seen in class C4.5 algorithm present robustness to noise and we considered this relevant for the large amount of attributes we have managed along the entire assignment.

We decided to pick Random Forest as the new algorithm to introduce. It builds many decision trees during training. Each tree looks at a random subset of the data and features. When classifying, all the trees vote, and the most popular choice becomes the final classification. That's why it's good at handling large numbers of features, and present robustness to overfitting.

Classification with all attributes (except next_score)

ZeroR -Using training set

When using the training set, our model consistently labels all instances as category 2, or "DOWN". However, it accurately classifies only 27.7695 % of the instances, which amounts to 1499 out of 5398 instances. The confusion matrix reveals that the classifications for UP, RIGHT, and LEFT are empty, while the DOWN column contains the counts for each class. The mean absolute error is calculated at 0.3743, and the root mean squared error at 0.4326.

-Cross-Validation (with 10 folds)

During cross-validation with 10 folds, the model once again assigns all instances to category 4, resulting in the same accuracy of 27.7695 %. The confusion matrix remains unchanged, as do the error metrics. -**Supplied Test**

Applying the model to the supplied test data yields a similar outcome, but with a slight decrease in accuracy to 27.4757 %. This signifies that only 283 out of 1030 test instances were correctly classified as "DOWN". The confusion matrix reflects the distribution of classifications in the test data. The error metrics remain similar with previous evaluations.

J48 -Using training set

The model constructed a tree with 65 leaves and 135 nodes. Despite an impressive accuracy of 98.9441 %, only 57 instances out of 5398 were misclassified, indicating potential overfitting. The mean absolute error is 0.0092, with a root mean squared error of 0.0677. The classifications for "UP," "DOWN," "LEFT," and "RIGHT" show mostly accurate results, with few misclassifications. **-Cross-Validation (with 10 folds)**

During cross-validation, the accuracy slightly decreased to 97.3509 %, with 143 misclassified instances out of 5398. The mean absolute error went up to 0.1096, and the root mean squared error to 0.0262. While the confusion matrix resembles the previous analysis, some instances exhibit different classifications.

-Supplied Test

Testing the model on supplied test data revealed an accuracy of 94.2718 %, with 59 misclassified instances out of 1030. The mean absolute error decreased to 0.0308, while the root mean squared error is 0.1635.

Random Forest -Using training set

Achieving a perfect accuracy of 100% on the training set, all 5398 instances were correctly classified, raising concerns about potential overfitting. The mean absolute error is minimal at 0.0083, and the root mean squared error is 0.0363. The confusion matrix displays a perfect diagonal pattern, indicating precise classification across all instances.

-Cross-Validation (with 10 folds)

During cross-validation, the model maintains exceptional accuracy at 97.3138 %, with only 145 misclassified instances out of 5398. The precision extends across all classes, supported by minimal error values: a mean absolute error of 0.0243 and a root mean squared error of 0.1012. The confusion matrix further confirms the precise classification, aligning mostly along the diagonal. -**Supplied Test**

Testing the model on supplied test data reveals an equally high accuracy of 94.1748 %, with 970 out of 1030 instances correctly classified. The mean absolute error increases to 0.05, and the root mean squared error to 0.1456.

Classification with filtered attributes

We wanted to try what the impact of leaving out the body was going to be. Hence, we tried with the other attributes.

ZeroR

Due to ZeroR's methodology, we anticipated obtaining identical results as before, as it only considers the most prevalent class in the dataset.

J48 -Using training set

A tree with 989 nodes and 495 leaves was generated, indicating potential overfitting, especially considering the limited use of only 6 attributes. Despite a seemingly high accuracy of 96.2764 %, with 5197 instances correctly classified out of 5398, there were 201 misclassifications. The confusion matrix showed a concentration of values along the diagonal.

-Cross-Validation (with 10 folds)

During cross-validation, the model decreased its accuracy at 87.4213 %, with 679 misclassified instances out of 5398. The error values increase: a mean absolute error of 0.0766 and a root mean squared error of 0.2387.

-Supplied Test

As suspected, the model suffered from severe overfitting. Testing on the supplied data revealed a significant decrease in accuracy to 32.6214 %, with only 336 out of 1030 instances correctly classified and 694 misclassified. The mean absolute error rose to 0.337, and the root mean squared error to 0.5586.

Random Forest -Using training set

Achieving a perfect accuracy of 100%, all 5398 instances were correctly classified. The mean absolute error stood at 0.0255, with a root mean squared error of 0.0639. The confusion matrix displayed a flawless diagonal pattern.

-Cross-Validation (with 10 folds)

During cross-validation, the model maintained a high accuracy of 91.0337 %, with 4914 instances correctly classified and 484 misclassified out of 5398. The mean absolute error increased to 0.075, and the root mean squared error to 0.1766. Similar to previous analyses, the confusion matrix primarily showed values along the diagonal.

-Supplied Test

Testing on unseen data resulted in a significant decrease in accuracy to 47.2816 %. Only 487 out of 1030 instances were correctly classified, leaving 543 misclassified. The mean absolute error rose notably to 0.3059, with a root mean squared error of 0.4062. The confusion matrix appeared disorganized, with values dispersed throughout.

Table of results for supplied data set.

Attributes	**Model**	**Correct**	**Incorrect**	**MAE**	**RMSE**	**Accuracy**	**Time taken**
head_x, head_y, food_x, food_y, snake_body score, direction	ZeroR	283	747	0.3749	0.4334	27.4757 %	0.01 seconds
	J48	971	59	0.0308	0.1635	94.2718 %	0.01 seconds
	Random Forest	970	60	0.05	0.1456	94.1748 %	0.04 seconds
head_x, head_y, food_x, food_y, score, direction	J48	336	694	0.337	0.5586	32.6214 %	0.39 seconds
	Random Forest	487	543	0.3059	0.4062	47.2816 %	0.52 seconds

Seeing this results, we believed using the two of them highlighted in green would be better because of the accuracy achieved in each of them

Phase 3: Building an automatic agent

We made our first try of building an automatic object with the two models with higher accuracy. However, we found out the snake didn't seem to catch on to the current situation and didn't look for food, nor did it avoid the walls. It was then evident that we needed to add new attributes to make this work.

Table of results for the different models.

Attributes	**Model**	**Description**	**Score**
head_x, head_y, food_x, food_y, snake_body	J48	The snake moved to the right until it crashed.	-37
score, direction	Random Forest	The snake moved to the right until it crashed.	-37

Phase 4: Prediction

Once again, because it was our first time, we wanted to see the evolution of the model's accuracy when evaluated in the three different ways.

Prediction with all attributes:

Linear regression -Using training set

The model demonstrates a good correlation coefficient of 0.9995, indicating a robust linear relationship between predictor variables and target scores. With a mean absolute error (MAE) of 3.1277 and a root mean squared error (RMSE) of 12.5677.

-Cross-Validation (with 10 folds)

The model achieved a high correlation coefficient of 0.9995, indicating a strong linear relationship between the predictor variables and the target variable. The mean absolute error (MAE) was 3.1306, indicating the average absolute difference between the predicted and actual scores. The root mean squared error (RMSE) was 12.5843. Overall, the model showed promising performance in predicting the score in the next tick based on game state and action features.

-Supplied Test

When tested on the supplied test set, the Linear Regression model maintains a strong correlation coefficient of 0.9987, indicating a robust linear relationship between predictor variables and target scores. However, there's a slight increase in both mean absolute error (MAE) to 3.5286 and root mean squared error (RMSE) to 14.2797 compared to the training evaluation. This suggests a slightly lower level of prediction accuracy when applied to unseen data, but overall, the model still performs well in predicting the score in the subsequent game tick based.

Random Forest -Using training set

The RandomForest model exhibits a high correlation coefficient of 0.9999, indicating a strong relationship between predictor variables and target scores. While the mean absolute error (MAE) is slightly higher at 3.9033 compared to Linear Regression, the root mean squared error (RMSE) is significantly lower at 6.6717. This suggests that while the average absolute difference between predicted and actual scores is slightly higher, the model's predictions are generally closer to the actual values.

-Cross-Validation (with 10 folds)

The model achieved a high correlation coefficient of 0.9994. However, the mean absolute error (MAE) is relatively higher at 5.7388 compared to the evaluation on the training set. Similarly, the root mean squared error (RMSE) is also higher at 13.7043, indicating larger variations in prediction errors.

-Supplied Test

When evaluated on the supplied test set, a correlation coefficient of 0.9757 is shown. The mean absolute error (MAE) is considerably higher at 44.8905, and the root mean squared error (RMSE) is also elevated at 61.9768 compared to previous evaluations. Indicating significant discrepancies between predicted and actual scores over unseen data.

Prediction with filtered attributes:

Following classification structure we wanted to test how accurate the precision would be without the snake's body, to see its impact. **Linear Regression**

-Training Set

The model exhibits a strong correlation coefficient of 0.9995, highlighting a robust linear relationship between predictor variables and target scores. It achieves a mean absolute error (MAE) of 3.1277 and a root mean squared error (RMSE) of 12.5677.

-Cross-Validation (10 Folds):

The model achieved a high correlation coefficient of 0.9995. However, the mean absolute error (MAE) is similar at 3.1306 compared to the evaluation on the training set. Similarly, the root mean squared error (RMSE) is 12.5843.

-Supplied Test:

Evaluation on the supplied test set reveals a correlation coefficient of 0.9987. Nevertheless, the model demonstrates slightly higher mean absolute error (MAE) at 3.5286 and root mean squared error (RMSE) at 14.2797 compared to previous assessments.Obtaining the same result without filtering attributes.

Random Forest -Training Set

The model exhibits a strong correlation coefficient of 0.9999, highlighting a robust linear relationship between predictor variables and target scores. It achieves a mean absolute error (MAE) of 3.8431 and a root mean squared error (RMSE) of 6.7659.

-Cross-Validation (10 Folds):

In cross-validation, the model maintains its high correlation coefficient of 0.9995, signifying a consistent linear relationship between predictor variables and the target. The mean absolute error (MAE) is 5.3687, indicating the average absolute difference between predicted and actual scores, with a root mean squared error (RMSE) of 13.4476.

-Supplied Test:

When tested on an external test set, the Linear Regression model upholds a robust correlation coefficient of 0.9838, suggesting a reliable linear relationship between predictors and scores. Despite a considerably large increase in both mean absolute error (MAE) to 36.5629 and root mean squared error (RMSE) to 50.4881 compared to training.

Table of results of supplied data set.

Attributes	**Model**	**MAE**	**RMSE**	**Correlation**	**Time taken**
head_x, head_y, food_x, food_y, snake_body score, next_score,	Linear Regression	3.5286	14.2797	0.9987	0.58 seconds
	Random	44.8905	61.9768	0.9757	0.76 seconds
direction	Forest				
head_x, head_y, food_x, food_y, score, next_score, direction	Linear Regression	3.1277	14.2797	0.9987	0.74 seconds
	Random Forest	36.5629	50.4881	0.9838	1.54 seconds

The data presents notable high error values, indicating potential inaccuracies within the dataset. These errors could stem from various sources, such as feature selection, data collection issues, or sampling errors. Despite exhibiting a high accuracy correlation coefficient, the observed increase in errors when predicting unseen data indicates that the performance is below the expected.

Second Iteration

Phase 1: Instances Collection

After the completion of the first iteration, we realized that the way we were representing the snake's body was not being taken into account. That's why we decided to represent each of the "cells" surrounding the head as a different attribute. After performing this change, out of the 56 attributes in the .arff files, 49 of them are used to somehow represent the body. To do this, we changed the function from the iteration before, to create instead of one numeric value with all of the information, which was not very helpful, specially because when the first positions were all zeros, to create one value per attribute.

With respect to the other attributes, we decided to keep them as they were on the iteration before.

Phase 2: Classification

We chose to use the same models as in the iteration before as they had given us good accuracy and we thought they made sense in this context. We wanted to compare both iterations so it made sense to use the same models to do so.

Once again, wanting to really understand how the models worked with the data, we chose to try not only the model on the testing data, but also perform cross validation and see what the data looked like in the training set.

Classification with all attributes (once again, except next_score)

ZeroR -Using training set

The model classifies all instances as 4, that is, "RIGHT". Only 27.4927% of instances are classified correctly, that is, 1511 out of 5496 instances. The confusion matrix consists of the first three columns related to UP, DOWN and LEFT being empty, and the last column, corresponding to RIGHT containing the number of instances belonging to each of the classes.We got a mean absolute error of 0.3743 and a root mean squared error of 0.4326.

-Cross-Validation (with 10 folds)

Once again, the model classifies all instances as 4, because it is the class that shows up the most. Hence, the results are the same, only 27.4927% of accuracy. The confusion matrix is the same. We got the same errors as before.

-Supplied Test

Doing the classification on the test data we extract by playing the result obtained is similar, the accuracy in this case decreases to 27.2524%. This fact means that only 487/1787 of the test instances were classified as "RIGHT". The confusion matrix is similar to the ones above, but with other numbers, representing the number of instances in the test data that have each of the classifications. We got the same errors as before.

J48 -Using training set

The model created a tree of 19 leaves and 28 nodes. Given the great number of attributes (55), we believe this tree might not be severely fit to our data, bearing in mind the tree has been pruned. We see that the accuracy of the tree is 99.89%, that is, 5490/5496 instances were correctly classified, leaving only 6 of them misclassified. This makes us a bit more suspicious about whether we might be looking at some overfitting in the data. The mean absolute error is of 0.001 and the root mean squared error is of 0.221. 1251/1252 of the instances of class "UP" were correctly classified, with only two of them being classified as "RIGHT". 13097/1309 were correctly classified as "DOWN", with only two of them being classified as "LEFT". Only two of the 1423 instances belonging to "LEFT" were misclassified, while all of the 1511 belonging to "RIGHT" were correctly classified.

-Cross-Validation (with 10 folds)

This time, the accuracy was reduced to 99.8544%. Only 8 out of 5496 instances were misclassified, leaving the other 5488 correctly classified. The mean absolute error is 0.0009 and the root mean squared error is 0.0262. The confusion matrix looks quite similar to the one above with only some of the instances classified differently to before.

-Supplied Test

After testing the accuracy of our model with the applied test, we were surprised to find out it was 97.3139%. That means 1739 out of 1787 instances of the test data were correctly classified, leaving 48 misclassified. The mean absolute error was 0.0136 and the root mean squared error of 0.1152. By looking at the confusion matrix, we see most of the misclassification (39/48) comes from instances belonging to the "RIGHT" class that were classified as "DOWN".

Random Forest -Using training set

We get an accuracy of 100%, meaning the 5496 instances were correctly classified, which once again makes us suspicious about overfitting with the data. The mean absolute error is 0.0013 and the root mean squared error 0.0092. The confusion matrix looks like a perfectly diagonal one, as all instances were correctly classified.

-Cross Validation (with 10 folds)

The results demonstrate a high accuracy of 99.9454%, with only 3 instances misclassified out of 5496. The model shows strong precision across all classes. Additionally, the error values are minimal, with a mean absolute error of 0.0036 and a root mean squared error of 0.0241. The confusion matrix shows a very precise classification, with the majority of instances correctly identified along the diagonal.

-Supplied Test

The accuracy was once again extremely high, of 99.8321%, with 1784 out of 1787 instances correctly classified, and 3 of them being misclassified. This means that the created model worked extremely well with the unseen data. The mean absolute error was 0.0103 and the root mean squared error was 0.0518. The confusion matrix is almost perfectly diagonal, with the three misclassified instances reflected on different positions in the matrix.

<u>Classification with filtered attributes</u>

We wanted to try what the impact of leaving out the body (the 49 instances that represented the cells around the head of it) was going to be. Hence, we tried with the other 6 attributes.

ZeroR

Because of the way ZeroR works, we knew we would be getting once again the same results as above, as only the most predominant class is taken into account.

J48 -Using training set

A tree was created with 891 nodes and 446 leaves. This is a clear sign that the model has been overfit, bearing in mind there were only 6 attributes being used. Hence, we do not really think the 96.1245%

accuracy is significant. 5283/5496 instances were correctly classified, leaving only 213 misclassified, and the confusion matrix had greater values on the diagonal than in other parts of it.

-Cross-Validation (with 10 folds)

The accuracy of the model decreased to 88.2096%, correctly classifying 4848 instances and incorrectly classifying 648, out of the 5496. The mean absolute error was 0.0724 and the root mean squared error of 0.2338. The confusion matrix now had more instances away from the main diagonal.

-Supplied Test

What we suspected happened. The model was terribly overfit. After running the supplied test, we saw how the accuracy decreased to 42.2496% only 755 out of the 1787 instances were correctly classified, leaving 1032 misclassified. The mean absolute error was 0.2929 and the root mean squared error of 0.5295.

Random Forest -Using training set

The accuracy was 100%, with 5496 instances correctly classified. The mean absolute error was 0.023 and the root mean squared error was 0.0626. The confusion matrix was a perfect diagonal.

-Cross Validation (with 10 folds)

The accuracy was once again high, of 91.3574%, leaving 5021 correctly classified instances, 475 misclassified, out of the 5496 total instances. The mean absolute error is 0.0666 and the root mean squared error is 0.1714. The confusion matrix, once again, has most of the values on the diagonal, with the 475 misclassified instances spread out over the rest of the matrix.

-Supplied Test

The accuracy was once again drastically decreased, being 42.4175% on unseen data. Only 758 out of the 1787 instances were correctly classified, leaving 1029 misclassified. The mean absolute error was 0.3308 and the root mean squared error of 0.4247. The confusion matrix looked messy, with the values mixed up throughout it.

<u>Classification with created attributes</u>

We decided to create a new attribute that we think might be beneficial: distance to food. This way, the machine learning tool will not need to infer this information, but have it directly at hand. We choose to express the distance in two different attributes, distance_food_x and distance_food_y. This way, maybe some decision about what move to make first would be considered. Also, we didn't use the absolute value because we were curious about if the different models would be able to catch up on which direction to take depending on the sign.

After seeing that erasing the snake_body was worsening the accuracy, we decided to try with all the original attributes and this new one.

ZeroR

Because of the way ZeroR works, we knew we would be getting once again the same results as above, as only the most predominant class is taken into account. **J48**

-Using training set

A tree was created with 16 leaves and 24 nodes. This gives us the impression that it won't be very overfit to the training data, as there are a big number of attributes in comparison with the number of nodes in the model. The accuracy was 99.9454%, which is very high, with 5493 out of the 5496 instances correctly classified, and only 3 of them misclassified. The mean absolute error as of 0.0005 and the root mean squared error of 0.0156. The confusion matrix looks almost diagonal.

-Cross-Validation (with 10 folds)

Once again, the accuracy was quite high, 99.8908%. Out of the 5496 instances, 5490 were correctly classified and only 6 misclassified. The mean absolute error was 0.0007 and the root mean squared error of 0.0234. Once again, the confusion matrix looks almost diagonal.

-Supplied Test

It was also surprising to find that we got a high accuracy of 97.4259%. Out of the 1787 instances, 1741 were correctly classified and only 46 were misclassified. The mean absolute error was 0.013 and the root mean squared error of 0.1129. Looking at the confusion matrix, almost all of the misclassifications are "RIGHT" instances that were classified as "DOWN". Apart from this, it looks almost diagonal.

Random Forest -Using training set

We got 100% accuracy, with 5496 of the instances being correctly classified. The mean absolute error of 0.0015 and the root mean squared error of 0.0091. The confusion matrix was perfectly diagonal.

-Cross Validation (with 10 folds)

We got an accuracy of 99.9454 %, with 5493 instances correctly classified and 3 misclassified. The mean absolute error was 0.0042 and the root mean squared error of 0.024. The confusion matrix was almost perfectly diagonal.

-Supplied Test

We got an accuracy of 99.6642 %, with 1781 instances correctly classified and 6 misclassified. The mean absolute error was 0.0134 and the root mean squared error 0.0597. The confusion matrix, once again, was almost perfectly diagonal.

Finally, we created a table containing the results of the testing on the supplied test, which gives us the most information, as it evaluates the created model on unseen data.

Table of results for supplied test.

Attributes	**Model**	**Correct**	**Incorrect**	**MAE**	**RMSE**	**Accuracy**	**Time taken**
head_x, head_y, food_x, food_y, snake_body,score, direction	ZeroR	487	1300	0.3743	0.4326	27.2524%	1.34s
	J48	1739	48	0.0136	0.1152	97.3139%	1.27s

	Random Forest	1784	3	0.0103	0.0518	99.8321%	2.83s
head_x, head_y, food_x, food_y, score, direction	J48	755	1032	0.2929	0.5295	42.2496%	2.02s
	Random Forest	758	1029	0.3308	0.4247	42.4175 %	0.06s
head_x, head_y, food_x, food_y, score, direction, distance_food_x, distance_food_y	J48	1741	46	0.013	0.1129	97.4259%	0.01s
	Random Forest	1781	6	0.0134	0.0597	99.6642%	0.05s

Because of the results obtained, we chose to use the first representation with both its highest accuracy. We changed our strategy a bit because instead of using the two situations in which we had a higher score, we chose to try the same representation with the models that had a greater accuracy, as before the accuracy was not related to a good performance. Later we learned this was related to the direction we were choosing to feed the data. This will be explained in more depth in iteration 4.

Phase 3: Building an automatic agent

Once again, we encountered the snake moving to the right and crashing against the wall without making any meaningful movement. Despite attempting both models, and subsets of it, neither of them proved effective in solving our encountered problem.

Table of results for the different models.

Attributes	**Model**	**Description**	**Score**
head_x, head_y, food_x, food_y, snake_body,score, direction	J48	The snake moved to the right until it crashed.	-37
	Random Forest	The snake moved to the right until it crashed.	-37

Phase 4: Prediction

Because we were sure we had to change the model completely, we decided to skip making predictions here, and just look at predictions in the following iteration, as we thought they would be sufficiently similar.

Third Iteration

Phase 1: Instances Collection

After completing the development of the intelligent agent for iteration 2 and observing no improvement in performance, we identified that the agent was consistently choosing a direction leading to collision with the wall. To correct this issue, we decided to improve the attributes used to train the model. Specifically, we incorporated the following additional attributes:

Distance to the Wall: This attribute represents the distance between the snake's head and the walls in each direction (up, down, left, right). Taking into account that head position already considers distance to the upper wall and distance to the leftmost wall, we included the distance to the right, and down walls. By including this information, the agent can better predict its proximity to obstacles and avoid collisions.

Distance to the Food: This attribute indicates the distance between the snake's head and the food item. We created it earlier using weka but it was easier to implement in python in order to complete phase 3.

Distance to Snake's Body: This attribute represents the distance between the snake's head and the different parts of the body. We chose to always use the 30 first parts of the body, as we needed a fixed amount. When the body had a body length of less than 30, we decided to introduce a 480 in that position, with the aim to represent that there was plenty of distance (infinite in fact) before collision. We created two vectors, snake_body_dis_h and snake_body_dis_v, each of them representing the vertical and horizontal distance with respect to the head of the snake. Then, each of the positions of the vector became an attribute of its own, so we ended up with 60 arguments representing in its own way the body of the snake.

Previous Direction: This attribute captures the direction in which the snake moved in the previous turn. Considering the snake's recent movement history can help the agent make more informed decisions about its next action and avoid repetitive patterns that lead to failure.

We also thought that we could improve the results we were getting in the Phase 4 of previous iterations, and decided to add some new values related to the future state of the game.

We reused a function we had used for Tutorial 1, called compute_next_state, which, given the direction, computed what the game would look like in the following tick, by computing where the head, the body, the score and the food would be. While it's true that the next food was sometimes impossible to compute, because of the randomness when a new food appears, we thought that still this information would be valuable as it would almost always be accurate.

Hence, we added the following attributes by using the compute_next_state and simple subtraction:

- next_food_x : x coordinate of the position of the food on next tick
- next_food_y : y coordinate of the position of the food on next tick
- next_head_x : x coordinate of the position of the snake's head on next tick
- next_head_y : y coordinate of the position of the snake's head on next tick
- next_dis_wall_down : distance to the lower wall on next tick
- next_dis_wall_right : distance to the rightmost wall on next tick

Phase 2: Classification

We decided to use different sets of attributes and test their accuracy. We were surprised to find out how high the accuracy was in almost all of them, and how the results were quite similar no matter the subset

chosen. We only saw that erasing both representations of the body was not good for any of the models, but with the rest of combinations, we achieved really high results.We also wanted to try with new models that given their characteristics we believed were appropriate for this situation.

Explanation about the chosen models (JRip, Random Tree):

JRip:

JRip is a rule based classifier which creates propositional rules used to classify elements.
RIP is short for RIPPER: Repeated Incremental Pruning to Produce Error Reduction. We decided it would be interesting to try this algorithm because we were worried about overfitting and hence believed using pruning would be wise. Also, we also wanted to try one classifier based on rules and not in trees to compare it.

Random Tree:

Because we saw it was similar to Random Forest, which was working really well for us. The contrast between Random Forest and Random Tree is that the Random Tree typically refers to a single decision tree instead of many.

Table of results for supplied data set:

Attributes	**Model**	**Correct**	**Incorrect**	**MAE**	**RMSE**	**Accuracy**	**Time taken**
(NO NEXT) snake_body_pos (representing the grid), head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x,	J48	1423	0	0	0	100%	0.08 seconds
	Random Forest	1423	0	0.0048	0.0174	100%	0.14 seconds
	JRip	1423	0	0	0	100%	0.04 second
distance_food_y, distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), last_direction, actual_direction	Random Tree	1414	9	0.0032	0.0562	99.3675 %	0.07 seconds

(NO GRID) head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), last_direction, actual_direction	J48	1423	0	0	0	100%	0.08 seconds
	Random Forest	1423	0	0.0062	0.0216	100%	0.09 seconds
	JRip	1423	0	0	0	100%	0.03 seconds
	Random Tree	1357	66	0.0232	0.1523	95.3619 %	0.05 seconds
(NO BODY DIS) head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), last_direction, actual_direction	J48	1423	0	0.0003	0.0007	100%	0.06 seconds
	Random Forest	1423	0	0.0069	0.0234	100%	0.11 seconds
	JRip	1423	0	0.0003	0.0004	100%	0.03 seconds
	Random Tree	1392	31	0.0109	0.1044	97.8215 %	0.04 seconds
(ANY BODY) head_pos_x, head_pos_y,food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_food_x_abs, distance_food_y_abs,	J48	1231	192	0.08	0.2272	86.5074 %	0.04 seconds
	Random Forest	1229	194	0.0999	0.2123	86.3668 %	0.14 seconds
	JRip	1276	147	0.0813	0.2135	89.6697 %	0.01 seconds

distance_wall_down, distance_wall_right, last_direction, actual_direction	Random Tree	1064	359	0.1267	0.3548	74.7716 %	0.02 seconds
(NOBODYGRID_NOLASTDIR) snake_body_pos (representing the grid), head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), actual_direction	J48	1423	0	0	0	100%	0.05 seconds
	Random Forest	1423	0	0.0085	0.0251	100%	0.08 seconds
	JRip	1423	0	0	0	100%	0.05 seconds
	Random Tree	1379	44	0.0155	0.1243	96.9079 %	0.04 seconds
(NOBODYDIS_NOLASTDIR) head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), actual_direction	J48	1423	0	0.0003	0.0007	100%	0.04 seconds
	Random Forest	1423	0	0.0084	0.0264	100%	0.11 seconds
	JRip	1423	0	0.0003	0.0004	100%	0.05 seconds
	Random Tree	1422	1	0.0004	0.0187	99.9297 %	0.02 seconds

Phase 3: Building an automatic agent

In the same way as previous iterations, because we were really confused about why our models were not working correctly, we chose to try building the agent with all of the models, wanting to check whether any of them finally worked. Once again, we found out most of them didn't, as they run right to the wall, but we started to see how some models started to work better than others:

When attempting with the entire set, we still did not achieve a meaningful result. The snake consistently moved right until it collided. Similarly, we initially tried without the body grid, then without the distance

to the body. Again, the snake solely moved right.In all cases mentioned above we tried with J48 and RandomForest, and got the same result in both models.
After removing both representations of the body, we noticed that the snake made decisions just based on its body representation across various tree models. With J48, it simply moved downward until it crashed.

However, when using the RandomForest model for prediction, a slight variation occurred: it moved downward, then turned left for 2 ticks, and then resumed its downward movement until it crashed. After achieving this initial progress, we opted to experiment with different models on the same dataset, even though they might not have had high accuracy. Unfortunately, when using Decision Table, Jrip, IBK with k=3, or Random Tree, we still observed the snake following a straight downward path until it crashed.
We noticed that sometimes the model was predicting a direction that the snake could not take (such as moving to the right when it was already moving right). This seemed odd, considering we had included the attribute "last_direction" to enable the model to learn such behaviors. However, we decided to prohibit those movements in the code and test it without the "lastdir" attribute to see if it was causing confusion during learning. Despite these adjustments, we tried different models such as Random Tree, J48, and PART, but did not obtain any interesting results.
Interestingly, JRip was the first model that allowed the snake to eat once before crashing, reaching a score of 28. However, our best results were achieved with Random Forest on this dataset. With Random Forest, the snake managed to eat several times and lasted a considerable amount of time playing compared to the other models, reaching a score of 240.

Table of results for the different models:

Attributes	**Model**	**Description**	**Score**
snake_body_pos (representing the grid), head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), last_direction, actual_direction	J48	The snake moved to the right until it crashed.	-37
	Random Forest	The snake moved to the right until it crashed.	-37

head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), last_direction, actual_direction	J48	The snake moved to the right until it crashed.	-37
snake_body_pos (representing the grid), head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_wall_down, distance_wall_right, last_direction, actual_direction	Random Forest	The snake moved to the right until it crashed.	-37
head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_wall_down, distance_wall_right, last_direction, actual_direction	J48	The snake moved down until it crashed.	-47
	Random Forest	The snake moved down then moved to the left and continued down until it crashed..	-47
	Decision Table	The snake moved down until it crashed.	-47
	JRip	The snake moved down until it crashed.	-47
	Random Tree	The snake moved down until it crashed.	-47
head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_wall_down, distance_wall_right, actual_direction	J48	The snake moved down until it crashed.	-47
	Random Forest	Worked alright for a few iterations, ate several times until I went straight to a wall and crashed.	240

Decision Table	The snake moved down until it crashed.	-47
JRip	Eat once and died	60
Random Tree	The snake moved down until it crashed.	-47

After this experimentation, the lesson we could extract is that the highest accuracies were related to the highest scores, but that other models with these same accuracies were working horribly.At least in this iteration, the snake was not running directly to the wall, and sometimes it made complicated moves, although it seemed as it was not looking for food most of the times. We were really surprised to find out we were getting these scores when the accuracy of the models was so high in phase 2. We explained later what was happening and how we fixed it.

Phase 4: Prediction

Prediction with all attributes:

Linear Regression:

-Training Set:

The model showcases a strong correlation coefficient of 0.9996, reflecting a robust linear relationship between predictor variables and target scores. It achieves a mean absolute error (MAE) of 3.8708 and a root mean squared error (RMSE) of 13.3876.

-Cross-Validation (10 Folds):

During cross-validation, the model maintains a high correlation coefficient of 0.9996, indicating a strong linear relationship between predictor variables and the target. The mean absolute error (MAE) became slightly higher at 4.0302 compared to the training evaluation, with a root mean squared error (RMSE) of 13.799.

-Supplied Test:

When tested on the supplied test set, the Linear Regression model sustains a strong correlation coefficient of 0.9997. However, there is a slight decrease in the mean absolute error (MAE) to 3.841 and a slight increase in the root mean squared error (RMSE) to 13.3905 compared to the training evaluation. This suggests a good performance in predicting the score in unseen data.

Random Forest:

-Training Set:

The RandomForest model demonstrates a strong correlation coefficient of 0.9999, indicating a robust relationship between predictor variables and target scores. Despite a higher mean absolute error (MAE) of 5.6388 compared to Linear Regression, the root mean squared error (RMSE) significantly decreases to 8.4425, suggesting generally closer predictions to actual values.

-Cross-Validation (10 Folds):

In cross-validation, the model achieves a high correlation coefficient of 0.9994. However, both the mean absolute error (MAE) and root mean squared error (RMSE) increase to 9.368 and 17.2792, respectively,

compared to the evaluation on the training set. These elevated errors indicate larger variations in prediction accuracy.

-Supplied Test:

Evaluation on the supplied test set yields a correlation coefficient of 0.9986. However, the model exhibits considerably higher mean absolute error (MAE) at 31.22 and root mean squared error (RMSE) at 38.1224 compared to previous evaluations, suggesting significant discrepancies between predicted and actual scores over unseen data.

Prediction with filtered attributes

In this scenario, we conducted various exclusions. In the first case, we excluded the snake's grid. In the second case, we excluded both the snake's grid, the snake body distance, and the last action as well.

CASE 1 (without body grid)

Linear Regression

-Training Set

The model maintains a strong correlation coefficient of 0.9996, indicating a robust linear relationship between predictor variables and target scores. It achieves a mean absolute error (MAE) of 3.6895 and a root mean squared error (RMSE) of 13.4995.

-Cross-Validation (10 Folds)

During cross-validation, the model achieves a high correlation coefficient of 0.9996. The mean absolute error (MAE) remains similar at 3.7588 compared to the evaluation on the training set, with a root mean squared error (RMSE) of 13.6735.

-Supplied Test

Evaluation on the supplied test set shows a correlation coefficient of 0.9997. However, the model demonstrates slightly higher mean absolute error (MAE) at 3.6513 and root mean squared error (RMSE) at 13.218.

Random Forest:

-Training Set

The model maintains a strong correlation coefficient of 0.9999, indicating a robust linear relationship between predictor variables and target scores. It achieves a mean absolute error (MAE) of 5.3312 and a root mean squared error (RMSE) of 8.1299.

-Cross-Validation (10 Folds)

During cross-validation, the model sustains a high correlation coefficient of 0.9995, signifying a consistent linear relationship between predictor variables and the target. The mean absolute error (MAE) is 8.0521, with a root mean squared error (RMSE) of 15.5859.

-Supplied Test

When tested on an external test set, the RandomForest model maintains a robust correlation coefficient of 0.9988, indicating a reliable linear relationship between predictors and scores. However, there is a

considerable increase in both mean absolute error (MAE) to 28.6631 and root mean squared error (RMSE) to 34.9014 compared to training.

CASE 4 (without body grid, body distance, last direction)

Linear Regression

-On the Training Set

The model exhibits a strong correlation coefficient of 0.9996, indicating a reliable linear relationship between predictor variables and target scores. It achieves a mean absolute error (MAE) of 3.6393 and a root mean squared error (RMSE) of 13.5567.

-During Cross-Validation (10 Folds)

The model maintains a high correlation coefficient of 0.9996 during cross-validation. The mean absolute error (MAE) remains consistent at 3.645 compared to the evaluation on the training set, with a root mean squared error (RMSE) of 13.5683.

-On the Supplied Test

Evaluation on the supplied test set reveals a correlation coefficient of 0.9997. However, the model demonstrates a slightly higher mean absolute error (MAE) at 3.5367 and root mean squared error (RMSE) at 13.1391.

Random Forest -On the Training Set

The model sustains a strong correlation coefficient of 0.9999, affirming a robust linear relationship between predictor variables and target scores. It achieves a mean absolute error (MAE) of 5.4613 and a root mean squared error (RMSE) of 8.4971.

-During Cross-Validation (10 Folds)

During cross-validation, the model maintains a high correlation coefficient of 0.9995, indicating a consistent linear relationship between predictor variables and the target. The mean absolute error (MAE) is 8.7884, with a root mean squared error (RMSE) of 17.4165.

-On the Supplied Test

When tested on an external test set, the RandomForest model maintains a robust correlation coefficient of 0.9903, suggesting a reliable linear relationship between predictors and scores. However, there is a notable increase in both mean absolute error (MAE) to 71.4648 and root mean squared error (RMSE) to 95.1876 compared to training.

Table of results for supplied test

Attributes	**Model**	**MAE**	**RMSE**	**Correlation**	**Time taken**
snake_body_pos (representing the grid), head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y,	Linear Regression	3.841	13.3905	0.9997	6.91 seconds

distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), last_direction, actual_direction	Random Forest	31.22	38.1224	0.9986	7.21 seconds
head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), last_direction, actual_direction	Linear Regression	3.6513	13.218	0.9997	5.21 seconds
	Random Forest	28.6631	34.9014	0.9988	5.12 seconds
head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_wall_down, distance_wall_right, last_direction, actual_direction	Linear Regression	3.5367	13.1391	0.9997	1.59 seconds
	Random Forest	71.4648	95.1876	0.9988	1.72 seconds

Fourth iteration

Phase 1: Instances Collection

After Phase 3 in the third iteration, we were left with a feeling that something was indeed wrong. It didn't make sense that we could only achieve a score of 250 when everything seemed to be right. We thought the information we were giving our model was right, and that the actual_direction was indeed the decision to be made on the next tick, because we had spent lots of time choosing the information and especially this attribute, as we knew it was really important. But when everything went wrong, we spent a bit more time and saw that the point in which we were collecting the data was after the move had already

taken place. This meant that the direction we were giving to each attribute was misplaced by one time step, and that it was not the information we wanted to feed the model.

After revisiting the code, we decided to move the lines print game state and print_line_data, as this was the source of the problem. When we studied in depth the outputs after performing this change we were relieved to finally see the direction was correctly written for the instances, and eager to try this new data set in our models.

Then, we also thought about the way we were representing the distance to the food. Before, the value could be positive or negative, as we aimed to represent not only the distance to the food but also the direction in which the snake had to move in some way. We were not sure if the model would be able to catch up on this, so we decided to also feed the model with these values in absolute value to see if this would make any changes.

Finally, thinking about Phase 4, we thought providing the next position of the head, by the two coordinates which represent it, would be beneficial as well.

To sum up, the new attributes were actual_direction, which was corrected, next_head_x, next_head_y, dist_food_x_abs, dist_food_y_abs

Phase 2: Classification

We first started to try the classifiers with all the attributes (no attributes related to the future). We say that trying on the test data we got the greatest accuracy of 87.6667% with Random Forest. Then, we tried to erase the attributes corresponding to the grid representation of the body, because we thought giving the body in two ways might affect the model. We saw an improvement in the accuracy of every model, once again, the Random Forest gives the best accuracy, with 88.055%. Then, we tried leaving the grid and erasing the representation of the body distance. This improved the accuracy in comparison with the last set of attributes, but only for J48 and JRip, and decreased the accuracy for Random Tree and Random Forest. From this, we infer that the second representation is better for the J algorithms but worse for the Random Tree and Random Forest. We took this into account in the selection of the features.

We then tried erasing both bodies, and we saw it worked a lot worse than leaving the two bodies for all of the models with the exception of JRip, which grew its accuracy by 5%.

Afterwards, we decided to try another approach and compare them both. We kept the original data set and erased the last direction and food_distance_x and food_distance_y, because we already had distance_food_x and distance_food_y, and we tried to avoid having repeated attributes. We saw that J48 and Random Forest increased in accuracy, while the two others remained the same. Because we saw this was interesting, we tried to combine this approach with the second approach, erasing the body grid, to see if this improved the accuracy.

Finally, we found out this last proposed approach was not beneficial for any of the models, so we decided to stop the experimentation here.

In this table, we summed up the results of the experimentation explained above. All this data corresponds to evaluating the model on the supplied test data file.

Table of results for supplied test

Attributes	**Model**	**Correct**	**Incorrect**	**MAE**	**RMSE**	**Accuracy**	**Time taken**
snake_body_pos (representing the grid), head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), last_direction, actual_direction	J48	1498	302	0.0983	0.2709	83.2222 %	1.09 seconds
	Random Forest	1578	222	0.1077	0.2224	87.6667 %	0.96 seconds
	JRip	1559	241	0.0981	0.2458	86.6111 %	0.94 seconds
	Random Tree	1344	456	0.1264	0.355	74.6667 %	0.92 seconds
(NO GRID) head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x,	J48	1523	277	0.0919	0.2641	84.6111%	0.18 seconds
	Random	1585	215	0.1116	0.2241	88.0556%	0.46
distance_food_y, distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), last_direction, actual_direction	Forest						seconds
	JRip	1553	247	0.0953	0.2472	86.2778%	0.07 seconds
	Random Tree	1379	421	0.117	0.342	76.6111 %	0.08 seconds

(NO BODY DIS) head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), last_direction, actual_direction	J48	1537	263	0.0964	0.2566	85.3889 %	0.28 seconds
	Random Forest	1570	230	0.1131	0.2289	87.2222 %	2.2 seconds
	JRip	1507	293	0.1061	0.2622	83.7222 %	2.24 seconds
	Random Tree	1407	393	0.1092	0.3304	78.1667 %	0.04 seconds
(NOLASTDIR_NOFOODDIS) snake_body_pos (representingthe grid), head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), actual_direction	J48	1550	250	0.0926	0.2527	86.1111 %	0.9 seconds
	Random Forest	1576	224	0.1105	0.2293	87.5556 %	4.1 seconds
	JRip	1574	226	0.0955	0.2389	87.4444 %	3.49 seconds
	Random Tree	1349	451	0.1253	0.3539	74.9444 %	0.07 seconds
(NOGRID_NOLASTDIR_NOFOODDIS) head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x_abs,	J48	1525	275	0.0995	0.2574	84.7222 %	2.54 seconds
	Random Forest	1575	225	0.1162	0.2309	87.5%	6.17 seconds
	JRip	1571	229	0.0969	0.241	87.2778 %	5.52 seconds

distance_food_y_abs,distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), actual_direction	Random Tree	1382	418	0.1161	0.3408	76.7778 %	0.11 seconds

Phase 3: Building an automatic agent

Once again, we decided to try all of the models because we were not entirely sure that accuracy was related to a better performance all the time. After trying Random Forest and J48 on the first set of attributes and seeing how we got a worse score in Random Forest when it had better accuracy, we were convinced that if we tried them all we would find the best model for the training, because accuracy was not being the best factor in making the decision. We saw this pattern in every single set of attributes, in which the model that achieved the highest accuracy was not necessarily the one to achieve the highest score.

Table of results for the different models

Attributes	**Model**	**Description**	**Score**
snake_body_pos (representing the grid), head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), last_direction, actual_direction	J48	Moves a bit but doesn't manage to eat. Ends up hitting a wall.	-47
	Random Forest	Keeps moving without making much sense. Avoids all walls but doesn't wat at all.	-360
	JRip	Only moves to the right, and ends up dying because it hits the wall.	-37
	RandomTree	This keeps moving a lot. Eats some times but ends up dying.	-19

(NO GRID) head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x,	J48	Moves without sense and finally dies.	-59

distance_food_y, distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), last_direction, actual_direction	Random Forest	We see how this algorithm starts to make more sense. This algorithm managed to eat 8 pieces of food, but it took too long to get to the food each time. Hence, it ended up getting less score than others just because of this reason. We think this might be related with the way we trained the snake	-100
	JRip	It managed to eat one or two pieces of food, but finally run into a wall.	64
	RandomTree	It managed to eat one or two pieces of food, but finally run into a wall.	-41
(NO BODY DIS) head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), last_direction, actual_direction	J48	Moved to the right and finally down	-41
	Random Forest	Didn't eat.	-88
	JRip	Managed to wat two food and died. Died because the food was located on the border so maybe the training instances we gave him also made this error.	129
	RandomTree	Didn't eat once. Moved in an odd way.	-65

(ANY BODY) head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right, last_direction, actual_direction	J48	Moves in an odd way. Ends up running into a wall.	-67
	Random Forest	Moves in an odd way. Ends up running into a wall.	-61
	JRip	The snake moved to the right every time. Ended up hitting the wall.	-37
	RandomTree	Moves in an odd way. Ends up running into a wall.	-74
(NOLASTDIR_NOFO	J48	Every time it only goes up.	-31
ODDIS) snake_body_pos (representing the grid), head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), actual_direction	Random Forest	Doesn't die because it doesn't hit any wall, but it doesn't move correctly in terms of ending up eating the food.We had to end the game with the escape.	-458 (and countin g)
	JRip	It manages to eat but ends up hitting the rightmost wall.	43
	RandomTree	Doesn't die because it doesn't hit any wall, but it doesn't move correctly in terms of ending up eating the food.We had to end the game with the escape.	-500 (and countin g)
(NOGRID_NOLASTDIR_NOFOODDIS) head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right,	J48	Moves differently each time but with no sense.	-75
	Random Forest	Moves differently each time but with no sense.	-129
	JRip	Eats two times and runs into a wall. But moves following some sense.	136

snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), actual_direction	RandomTree	Didn't even manage to eat one piece of food. Moved without making much sense. Again, we believe this might be related with the training. We had to end the game with the escape.	-779 (and countin g)

During this phase, we took comfort in the fact that the snake successfully evaded death, even though its score decreased over time. There were instances where it approached the food and made attempts to consume it. However, we further explored and analyzed better results in the next iteration. This testing showed us the significance of representing the snake's body in the data. Interestingly, we observed distinct behaviors across models, with J48 and JRip exhibiting similarities, as well as Random Forest and Random Tree models on its performance.

Phase 4: Prediction

At this point, we knew the problems we were encountering had nothing to do with our model, and everything to do with our data. It turned out we were nowhere near being good at playing the snake game and the models were learning from our poor decision making. That's why we decided to start another iteration, which would be similar enough to this one in the accuracy of the models, but more consistent with the way the snake should move. We decided to try our "intelligent agent" from Tutorial 1 and see what happened.

Fifth iteration

Phase 1: Instances Collection

After numerous iterations, with a particular focus on improving the performance of our intelligent agent from the previous iteration, we discussed the quality of our data for effectively learn by the models. Our gameplay strategy showed inconsistencies; at times, we missed the food's position and circled around it until capturing it. Moreover, as the snake grew larger, it tended to move away from the food to avoid collisions with itself, for then going back to catch it. We thought that these behaviors might ruin the models' learning process. Therefore, in this instance, we collected the attributes and used our smart agent created in tutorial 1 to enable the models to learn optimized solutions. As suspected, the results showed significant improvement.

Phase 2: Classification

We opted to conduct the classification process once more, utilizing the same model as in previous iterations to ensure consistency in comparison. The variation in attributes or instance collection between this iteration and the previous one is not other than the key difference in how the instances were collected, with our smart agent from tutorial 1 making the decisions on where to move. Consequently, it is reasonable to expect that the classifications maintain their structure across different models but exhibit

higher levels of accuracy. Furthermore, we observe that the maximum accuracy values, highlighted in green, remain consistent with the same model as in the previous iteration.

Table of results for supplied test data

Attributes	**Model**	**Correct**	**Incorrect**	**MAE**	**RMSE**	**Accuracy**	**Time taken**
snake_body_pos (representing the grid), head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), last_direction, actual_direction	J48	1609	35	0.0107	0.1004	97.871 %	0.25 seconds
	Random Forest	1624	20	0.0411	0.0946	98.7835 %	1.13 seconds
	JRip	1615	29	0.0087	0.0851	98.236 %	0.63 seconds
	Random Tree	1522	122	0.0371	0.1926	92.5791 %	0.02 seconds
(NO GRID)	J48	1613	31	0.0098	0.0971	98.1144	0.13
head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), last_direction, actual_direction						%	seconds
	Random Forest	1626	18	0.0292	0.0823	98.9051 %	0.86 seconds
	JRip	1613	31	0.0104	0.0973	98.1144 %	0.46 seconds
	Random Tree	1542	102	0.031	0.1761	93.7956 %	0.02 seconds

(NO BODY DIS) head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), last_direction, actual_direction	J48	1609	35	0.0107	0.1004	97.871 %	0.02 seconds
	Random Forest	1628	16	0.0256	0.0755	99.0268 %	0.32 seconds
	JRip	1615	29	0.0089	0.0939	98.236 %	0.17 seconds
	Random Tree	1534	110	0.0335	0.1829	93.309 %	0.01 seconds
(ANY BODY) head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x,distance_food_y,distance_food_x_abs, distance_food_y_abs,distance_wall_down, distance_wall_right, last_direction, actual_direction	J48	1607	37	0.012	0.1057	97.7494 %	0.03 seconds
	Random Forest	1608	36	0.0214	0.1048	97.8102 %	0.06 seconds
	Random Tree	1583	61	0.0186	0.1362	96.2895 %	0.02 seconds
	JRip	1608	36	0.0125	0.1045	97.8102 %	0.02 seconds

Phase 3: Building an automatic agent

Table of results for the different models

Attributes	**Model**	**Description**	**Score**
snake_body_pos (representing the grid), head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y,	J48	This snake works really well, but it ends up hitting its body. Manages to eat plenty of food.	1578
	Random Forest	Dies by hitting a wall.Manages to eat plenty of food.	1275

distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), last_direction, actual_direction	JRip	Dies by hitting a wall.Manages to eat plenty of food.	862
	RandomTree	Dies by hitting a wall.Manages to eat plenty of food.	552
(NO GRID) head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), last_direction, actual_direction	J48	Moves effectively and stops due to a predicting error from weka. Traceback (most recent call last): File "/home/mi1/proyecto/SnakeGame.py", line 373, in <module> a = weka.predict("./J48_v5_nogrid.model", x, "./training_keyboard_v5_final_nogrid.arff") File "/home/mi1/proyecto/wekal.py", line 60, in predict x[i] = attribute.index_of(x[i]) File "/home/mi1/miniconda3/envs/julia/lib/python3.9/site-packages/weka/core/dataset.py", line 1123, in index_of return javabridge.call(self.jobject, "indexOfValue", "(Ljava/lang/String;)I", label) File "/home/mi1/miniconda3/envs/julia/lib/python3.9/site-packages/javabridge/jutil.py", line 891, in call nice_args = get_nice_args(args, args_sig) File "/home/mi1/miniconda3/envs/julia/lib/python3.9/site-packages/javabridge/jutil.py", line 1148, in get_nice_args return [get_nice_arg(arg, subsig) File "/home/mi1/miniconda3/envs/julia/lib/python3.9/site-packages/javabridge/jutil.py", line 1148, in <listcomp> return [get_nice_arg(arg, subsig) File "/home/mi1/miniconda3/envs/julia/lib/python3.9/site-packages/javabridge/jutil.py", line 1184, in get_nice_arg return env.new_string_utf(arg) File "_javabridge.pyx", line 1663, in _javabridge.JB_Env.new_string_utf AttributeError: 'int' object has no attribute 'encode'	2090
	Random Forest	Dies by hitting a wall. Didn't reach as many apples as previous iterations.	581

	JRip	Dies by hitting a wall.Manages to eat plenty of food.	904
	RandomTree	Dies by hitting itself. Just able to eat twice.	124
(NO BODY DIS) head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x)snake_body_dis_v (distance to the body positions on y), last_direction, actual_direction	J48	This works really well. It avoids walls and eats plenty of food. It ended up dying because it crashed into itself. We believe this is related to the data we are using, that comes from a model that wasn't perfect.	3365
	Random Forest	Dies by hitting itself, however it works nicely avoiding itself and eating on several occasions.	2122
	JRip	Dies by hitting a wall.Manages to eat plenty of food.	1789
	RandomTree	Dies by hitting itself. Just able to eat twice.	186
(ANY BODY) head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_food_x_abs, distance_food_y_abs, distance_wall_down, distance_wall_right, last_direction, actual_direction	J48	Dies by hitting a wall.Manages to eat plenty of food.	1219
	Random Forest	Dies by hitting a wall.Manages to eat plenty of food.	1279
	JRip	Dies by hitting a wall.Manages to eat plenty of food.	841
	RandomTree	Dies by hitting a wall.Manages to eat plenty of food.	863

Finally, we started getting the results we expected. Some technical issues related to weka made some games stop before entering in game over. They are explained in the conclusion below.

We saw how we got the best performance with the J48 algorithm on the representation without body distance. Once again, we were surprised that the best performance was not related to the best accuracy.

Phase 4: Prediction

Table of results for the supplied data

Attributes	**Model**	**MAE**	**RMSE**	**Correlation**	**Time taken**
snake_body_pos (representing the grid), head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_wall_down, distance_wall_right, snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), last_direction, next_score, next_food_x, next_food_y, next_head_x, next_head_y, next_dis_wall_down, next_dis_wall_right actual_direction	Linear Regression	4.4666	12.9204	0.9995	5.42 seconds
	Random Forest	35.3663	45.056	0.9949	1.88 seconds
(NO BODY GRID), head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_wall_down, distance_wall_right,	Linear Regression	4.4486	12.8762	0.9995	0.5 seconds

snake_body_dis_h (distance to the body positions on x) snake_body_dis_v (distance to the body positions on y), last_direction, next_score, next_food_x, next_food_y, next_head_x, next_head_y, next_dis_wall_down, next_dis_wall_right actual_direction	Random Forest	34.4808	46.2411	0.9949	1.63 seconds
(NO BODY) head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_wall_down, distance_wall_right, last_direction, next_score, next_food_x, next_food_y, next_head_x, next_head_y, next_dis_wall_down, next_dis_wall_right actual_direction	Linear Regression	4.1964	12.8127	0.9995	0.04 seconds
	Random Forest	73.5417	95.1701	0.979	1.39 seconds
(NO BODY, NO FUTURE) head_pos_x, head_pos_y, food_pos_x, food_pos_y, score, distance_food_x, distance_food_y, distance_wall_down,	Linear Regression	4.1964	12.8127	0.9995	0.02 seconds
	Random Forest	63.6131	85.2364	0.9832	1.72 seconds
distance_wall_right, last_direction, next_score, actual_direction					

We noticed that during the prediction process, we obtained the same correlation coefficient in the different subsets we considered. However, the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE) decreased as the linear regression model had fewer attributes, opposite to what happened on Random Forest where the correlation coefficient remained constant, but the errors tended to increase when removing attributes.

This could be attributed to the distinct characteristics of linear regression and random forest models. In linear regression, reducing the number of attributes decreases multicollinearity and variance, potentially improving model stability and reducing errors. On the other hand, in random forest removing attributes can limit the model's ability to capture complex patterns, resulting in increased errors despite a constant correlation coefficient. This discrepancy showed how model behaviors influence their performance when attributes are altered.

We incorporated different future attributes such as next_food_x, next_food_y, next_head_x, next_head_y... Since we thought this might be useful when predicting the next score, but as we can see in the table results, after its performance without them we can check that they are just noise for linear regression model.

Questions

1. What is the difference between learning these models with instances coming from a human-controlled agent and an automatic one?

The model learns from the data. Hence, if the data comes from an automatic agent that has been developed with a specific algorithm, the model will eventually learn the game from the algorithm itself. For example, if the algorithm of the automatic agent avoids certain situations, the learned model will do so too. On the other hand, when the model learns from a human-controlled agent, because we do not work with specific algorithms, but to the best of our capabilities, the model will learn from our successes and from our errors. Nevertheless, it is important to highlight how we humans can sometimes be quite inconsistent with our decisions, especially in a fast paced situation like playing this game. We can miss the axis in which we should move in order to get faster to the food, and round the food many times trying to aim, but failing to do so. The model will learn from these inconsistencies and behave in an odd manner.

In our experience, it is better to let the model learn from an “intelligent” agent, as it will be more consistent with its decisions, but maybe if the model learned from someone who is really good at playing the game, then this might change.

2. If you wanted to transform the regression task into classification, what would you have to do? What do you think could be the practical application of predicting the score?
In order to transform the regression task into a classification one, we would need to discretize the score attribute into a certain number of categories such as low, medium, and high scores. Using cutoff points when analyzing the data is helpful to create these new categories.

When thinking about the practical applications score prediction might have, we think this could be used in competitions, in order to group players into categories related to their score when it is not clear the level each of them have. Also, this prediction models could be incorporated into mobile games, improving the game experience for the users by adjusting the difficulty automatically based on past performance of the user.

3. **What are the advantages of predicting the score over classifying the action? Justify your answer.**
   Predicting the score provides more detailed information about the game and the player's performance compared to classifying actions, which is continuous rather than discrete. Classifying actions categorizes things into groups rather than providing specific details, hence losing information.

4. **Do you think that some improvement in the ranking could be achieved by incorporating an attribute that would indicate whether the score at the current time has dropped?**

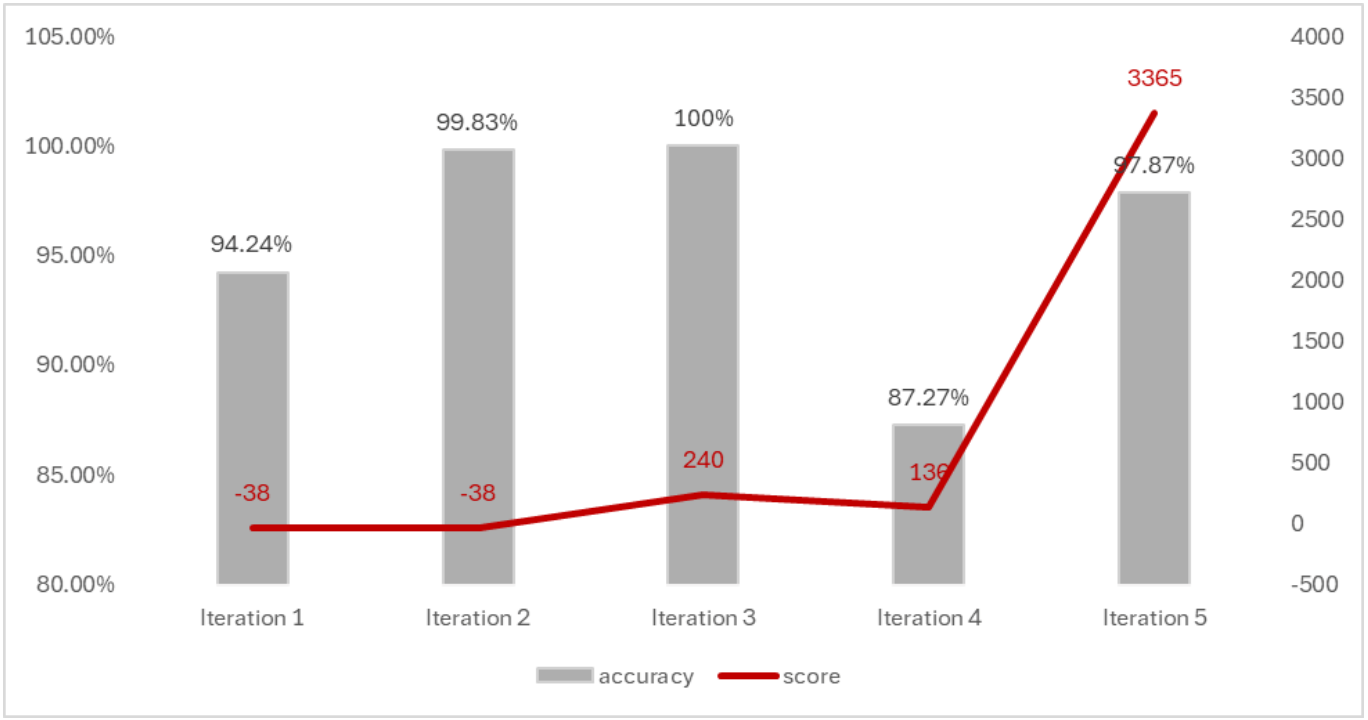
Yes, incorporating an attribute indicating whether the score at the current time has dropped could potentially improve the ranking. This additional attribute could provide valuable information about the current state of the snake game, indicating whether the snake is eating or not. Through various iterations, it has been observed that present meaningful attributes are helpful for both classification and prediction tasks. We may compare adding this attribute with a similar learning experience we had during the creation of this assignment, in which adding explicitly the distance to the food was better than just stating where the head of the snake was and where the food was. In a similar way, by indicating to the snake that we are not eating the food in the way we are moving (hence a drop of -1 in the score) might be better than just stating the position of the food and the current score.

By incorporating this information into the model, it could better capture the dynamics of the game and potentially lead to improved performance in ranking.

Conclusions

The following graph summarizes our progress through the different iterations. It can be observed that during the first three iterations, the accuracy was extremely high, yet there were no significant results when building the intelligent agent. This discrepancy once again highlights issues with the data. It wasn't until the 4th and 5th iterations that we adjusted our approach towards prediction, focusing on the present data rather than the past. It was during these iterations that we began to observe meaningful results, with high accuracy in classification during iteration 4 that increased in the next iteration due to the interaction of the tutorial 1 agent collecting data, and translating in a good performance when building the agent.

Graph: Comparison of accuracy and performance achieved on the different iterations



Once again, this practice showed us the importance of choosing good attributes. After Tutorial 3 we had given some thought to the way the body of the snake should be represented, and the new ideas we had in this Assignment really proved to work a lot better than the ones we had before. Also, we have seen the power of iterating through different ideas, as some of the representations you initially choose don't end up working as well as expected. That is something we have seen in the changes made between iteration 1 and iteration 2 of the process, and during iteration 3 and 4. In the first case, choosing to represent the body in a different way made the body significant, as when erased, the model dropped its accuracy. In the second case, actually complying with the time steps and choosing the direction chosen for the next iteration rather than the direction that the agent had just taken proved to solve our problems and our bad performance on phase 3 of iteration 3.

We could experiment once more with the need to try different sets of attributes. Finding the balance between feeding the model with too much information and too little was difficult, and we had to try many different ways to end up getting something that worked for us. This practice convinced us that it’s not always about giving the model all the information available, but giving the model the most relevant information in the most concise way. Also, including some new attributes might not always have the expected effect. For example, in Iteration 2, we introduced the distance to the food, thinking that maybe it would be easier for the models to have the information rather than infer it, but we saw that it actually was not better, but it slightly reduced the accuracy of the created models. In the following iterations, we found out that the problem with this argument is that we thought that making the distance to the food negative or positive, as to indicate not only how close or far away from the food the snake was, was not working for us. But when we created the absolute value of this attribute, the model improved its accuracy.

Regarding the third phase, it was the most difficult one in all the iterations. Setting up the system to run the code was challenging, but what really threw us off was learning that many times, the fact that the model has high accuracy does not mean it will work. For example, on iteration 3, we thought we had achieved the “perfect” model, and, to our surprise, when we tried making predictions in real time, we only managed to achieve a score of 230. The snake avoided the walls sometimes, but it didn’t look for food most of the time. When we realized the snake was learning about the direction it had just taken, and not the one to perform, everything made sense. This was the reason why, although our models were highly accurate, they were not working the way we intended them to.

We also wanted to highlight a problem we had both during the phases of instance collection and creation of our intelligent agent. When we managed to achieve a high score at the time of collecting both training and test data, opening the files in weka was not possible. We are not sure why, but we always got errors that no matter how we tried to fix them, by using fewer attributes or checking the format of the .arff files manually, we didn’t manage to solve them. It’s for this reason that we believe that although our final automatic agent did sufficiently well, this issue kept us from achieving higher scores, as our model was not trained in how to act when the body was bigger or the score was higher. Also, when weka makes predictions in real time, it also runs into the same problem. When the automatic agent had achieved a score of 2090 in iteration 5, using the J48 model, the game stopped by giving us an error related to the same reason as before.

In the fourth phase, after encountering most of the difficulties from the previous two phases, we found that predicting didn't pose any challenges aside from analyzing the results. Similar to previous iterations, we maintained consistency in the structure of datasets for comparison purposes. We used the same models in different iterations: linear regression, which we considered suitable for prediction tasks, and RandomForest, chosen for its effectiveness in classification and to evaluate its predictive capabilities. Across the iterations, we observed high correlation coefficients among the different models and how the data influenced them. We obtained better results with fewer attributes in linear regression, while the opposite was found for RandomForest. Once again, we reaffirmed the importance of data in specific models, as it can significantly impact the output.

Overall, we believe this practice has been challenging but also has been a great experience of learning hands on everything discussed in class, and the real problems people that work in this field face everyday, like the decision making involved in the process of choosing features and models. It has also been really interesting how this practice has taught us the importance of feeding the model with good data.

There is a lot of discussion today about Artificial Intelligence substituting humans in their day to day life, and although we are not experts in the field just yet, we are now more convinced than ever about how AI and human will continue to work hand to hand for a long time yet, and how AI will be a tool instead than a substitute for mankind. While doing the practice, we saw that the model did need to learn from high quality data in order to make good decisions. Throughout the different iterations, no matter how good we thought we had played, the fact was that without a preprogrammed algorithm, the model learned from our inconsistencies. Hence, good examples are vital for a good performance of the model. We see how Machine Learning can have helpful applications in fields as delicate as medicine, in which in a similar

way to the Snake Game, important decisions should be made quickly ,but we are sure that good doctors will still be needed to make the models learn from the best and avoid making mistakes related to inconsistencies.