ARTIFICIAL NEURAL NETWORKS FINAL EXAM

Fatma Betül Fişne-040200204

Repository: https://github.com/betulfis/final-ANN

Data Collection

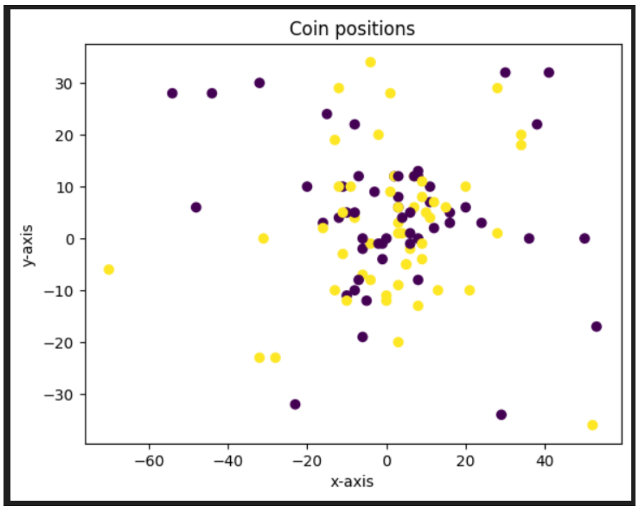
First of all, the environment needed to collect data had to be created. For this, the tablet camera was used and placed facing the floor. In order to determine the position where the coin fell, the 2-dimensional coordinate system was printed on 4 A4 sheets of paper, with 1 region of the coordinate system on each sheet. A 2D coordinate system was pasted on the ground. The same coin was dropped 105 times in a row at the origin level from a height of 1 meter. The first 35 of them were dropped with the coin's tails side up, the next 35 were dropped with the coin's heads side up, and the last 35 were dropped with the coin in a vertical position. After each toss, a photo of the coin on the ground was taken. After the photos were taken, heads or tails were manually detected and the result was written on each photo. Since there is a coordinate plane in the photograph, the x and y coordinates of the points where the coin fell were determined from the photograph and its distance from the origin was written on the photograph.

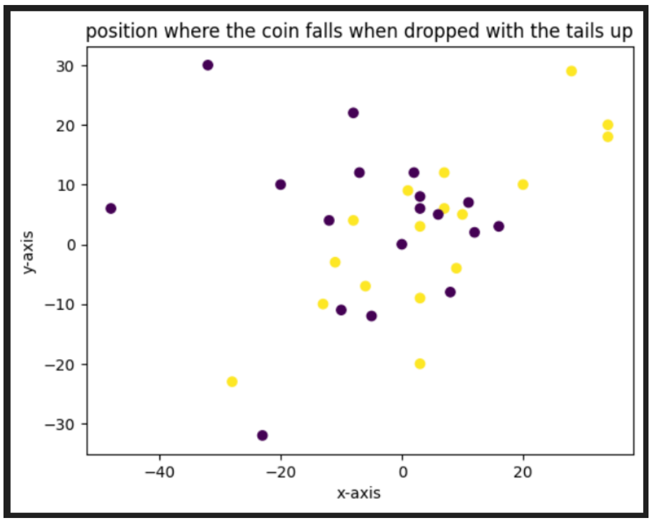
Data Labeling

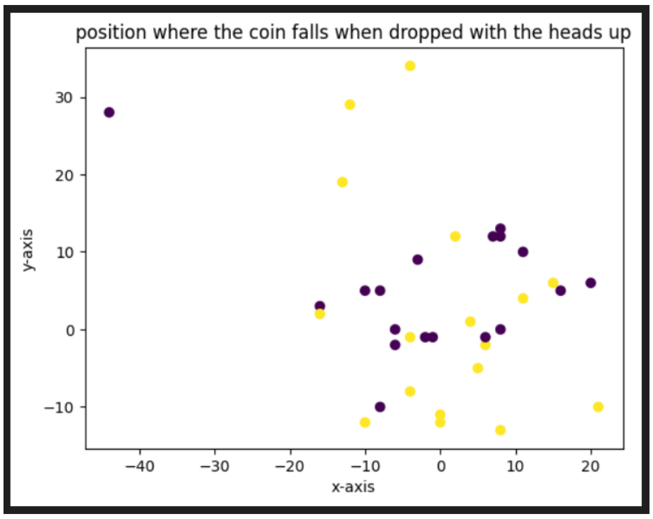
The collected data was transferred to an Excel table. The position of the coin when it was dropped was filled in the first line, the x-coordinate was filled in the second line, the y-coordinate was filled in the 3rd line, the distance from the origin was filled in the 4th line, and which side of the coin was on top (heads or tails) was filled in the 5th line with the data taken from each photograph.

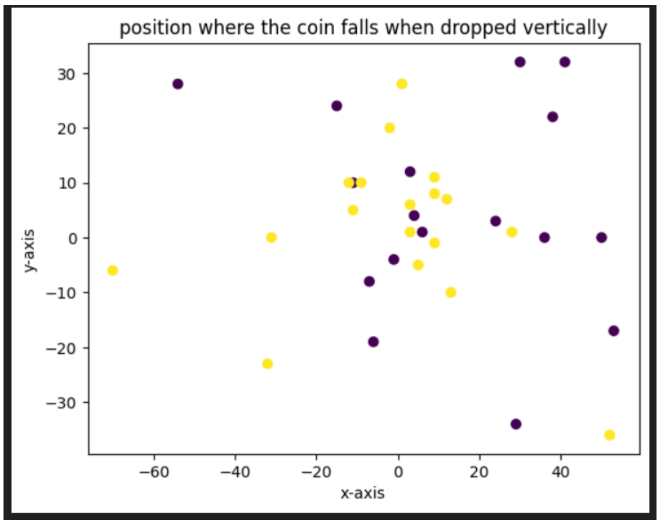
Preprocessing

First of all, it was decided to draw a scatter plot to see the distribution of the data. For this, each column of the table was taken as ndarray. Non-numeric (categorical) data (initial orientation and final orientation) were coded numerically. A scatter plot was created using x-coordinates and y-coordinates, and the colors of the data points were determined according to the result of heads or tails. Thus, the distribution of all data points was seen. It was observed that the data was concentrated around the origin and the position distribution did not change much depending on whether it was heads or tails. In addition, the data was divided into three according to initial orientation and a separate distribution graph was drawn for each initial orientation to observe how the money was distributed according to initial orientation. While collecting the data, photographs of the data in cases where the falling money was out of the camera's perspective could not be taken, so it was not added as data. Thus, the outliers were eliminated from the beginning.









Model Evaluation for Heads or Tails Prediction

Since our result had two options (heads or tails), binary classification models had to be used. Models were researched for this and, based on homework and research, it was determined that RBF SVM, linear SVM, decision tree or logistic regression could be used. It was predicted that RBF SVM could provide more accurate results and this model was tested. In addition, other models were also tested and their accuracy was measured, but the accuracy of the RBF SVM result was higher as expected.

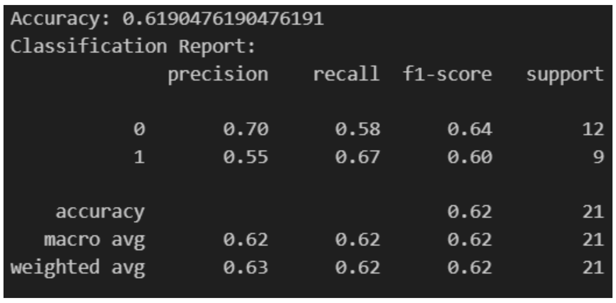
Coding

The data was taken from the table as ndarray. Three properties were determined as independent variables: x-coordinate, y-coordinate and initial orientation. Output was chosen as the final orientation. The data was divided into 80% train and 20% test. The independent variable matrix (X) was normalized (It was also tried in its non-normalized form, but the accuracy was significantly lower). The model was created with train data.

Evaluation

The test data was given as input to the created model and a prediction was taken from the model. Accuracy result was 0.62. Considering that the coin toss event is random and the probability of getting heads or tails is 0.5, I think accuracy is sufficient for this situation. When I tried other models, it was seen that the accuracy of the decision tree model was 0.52, logistic regression was 0.43, and linear SVM was 0.38.

Accuracy of RBF SVM model:

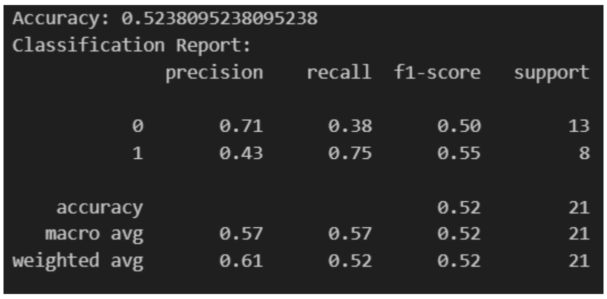


```
Accuracy: 0.6190476190476191
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.58      0.64        12
           1       0.55      0.67      0.60         9

    accuracy                           0.62        21
   macro avg       0.62      0.62      0.62        21
weighted avg       0.63      0.62      0.62        21
```

Accuracy of decision tree model:

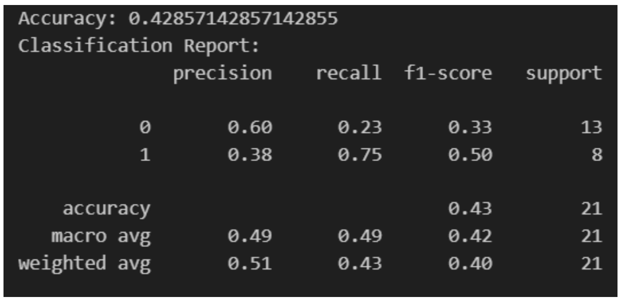


```
Accuracy: 0.5238095238095238
Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.38      0.50        13
           1       0.43      0.75      0.55         8

    accuracy                           0.52        21
   macro avg       0.57      0.57      0.52        21
weighted avg       0.61      0.52      0.52        21
```

Accuracy of logistic regression model:

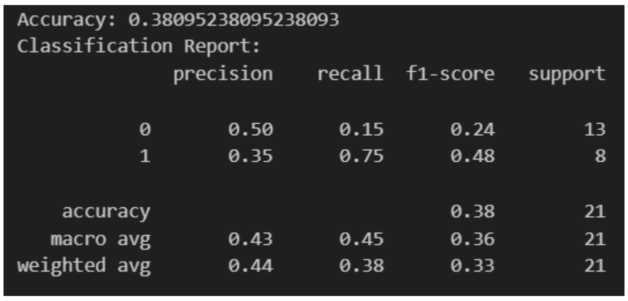


```
Accuracy: 0.42857142857142855
Classification Report:
              precision    recall  f1-score   support

           0       0.60      0.23      0.33        13
           1       0.38      0.75      0.50         8

    accuracy                           0.43        21
   macro avg       0.49      0.49      0.42        21
weighted avg       0.51      0.43      0.40        21
```

Accuracy of linear SVM model:



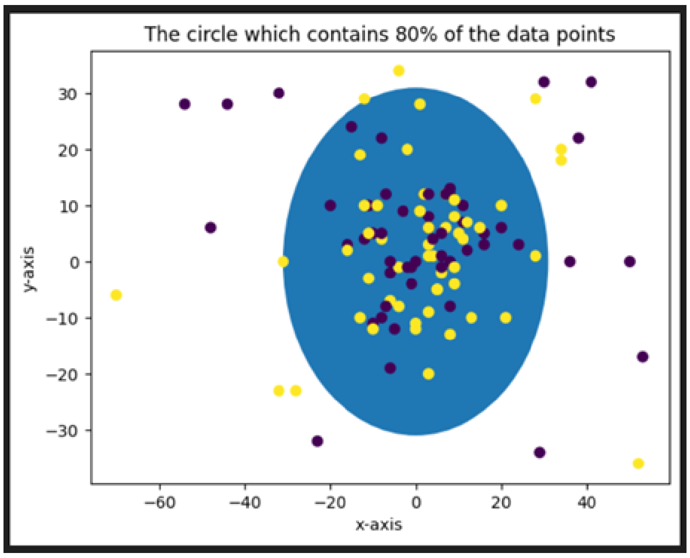
```
Accuracy: 0.38095238095238093
Classification Report:
              precision    recall  f1-score   support

           0       0.50      0.15      0.24        13
           1       0.35      0.75      0.48         8

    accuracy                           0.38        21
   macro avg       0.43      0.45      0.36        21
weighted avg       0.44      0.38      0.33        21
```

Distance Prediction

For position prediction, initial orientation was used as input and distance from the origin was used as output. Since input is discrete and output is continuous, a suitable model was required. When researched on the internet, it was seen that there was a technique called Random Forest Regression. This technique is capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation. This algorithm was applied to the train dataset and prediction was made for the test data set and error was found. The prediction of the model was found according to 3 different initial conditions. Accordingly, if the coin is dropped with the heads side up, distance from the origin is predicted as 19 cm. If the coin is dropped with the tails side up, the distance from the origin is predicted as 14 cm. If the coin is dropped vertically distance from the origin is predicted as 24 cm. Of course, it is not realistic to make a direct distance estimate. It can be said that by adding a reasonable size margin to these areas for an accurate prediction, the probability of the coin dropping into the circle segment is high. Apart from this, an area covering 80% of the data was drawn to visualize in which regions the money was concentrated. We can say that the probability of a thrown coin falling into this area is quite high.



Challenges

Determining input properties was quite confusing. Because we don't have many features. Since the drop height of the coin was determined as 1 meter, no experiments were made from different heights. Moreover, since the word 'dropping' was mentioned in the question, the money was always left as a free fall. If it was released by applying force, force could be added as a parameter. Accuracy could thus be increased. Since I am not very experienced in Python, I also had difficulties in coding, but I managed to overcome it by using resources on the internet. Also, to simplify the coding, I made manual recording instead of video.