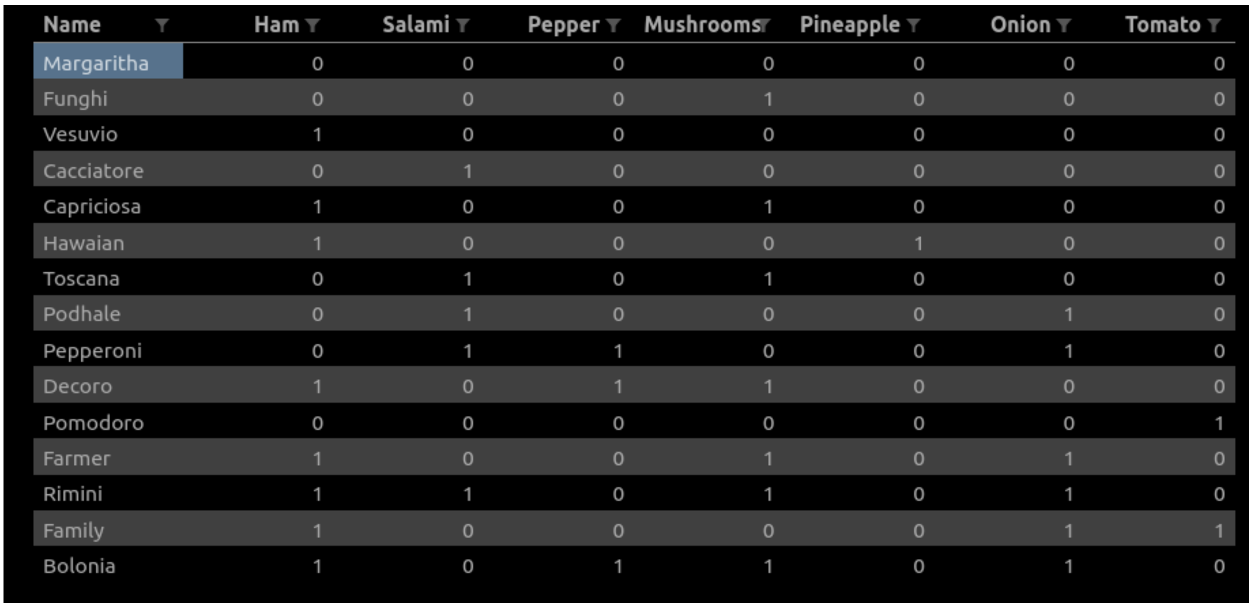
wAIter Subproject Report

Decision Tree

Szymon Kuleczka

I. Dataset

The dataset used in the subproject is the list of 15 pizzas from **Mamma Mia** pizzeria with the corresponding list of 7 pizza toppings: ham, salami, pepper, mushrooms, pineapple, onion, tomato. If the topping is present on the pizza, corresponding cell value is 1, otherwise it's 0. The data is initially stored in *CSV* file, then loaded to **Pandas** *DataFrame*.



Name	Ham	Salami	Pepper	Mushrooms	Pineapple	Onion	Tomato
Margaritha	0	0	0	0	0	0	0
Funghi	0	0	0	1	0	0	0
Vesuvio	1	0	0	0	0	0	0
Cacciatore	0	1	0	0	0	0	0
Capriciosa	1	0	0	1	0	0	0
Hawaian	1	0	0	0	1	0	0
Toscana	0	1	0	1	0	0	0
Podhale	0	1	0	0	0	1	0
Pepperoni	0	1	1	0	0	1	0
Decoro	1	0	1	1	0	0	0
Pomodoro	0	0	0	0	0	0	1
Farmer	1	0	0	1	0	1	0
Rimini	1	1	0	1	0	1	0
Family	1	0	0	0	0	1	1
Bolonia	1	0	1	1	0	1	0

This pizza *DataFrame* is further used during generation of random order history for the customer. Rules are the following: firstly, random number between 6 and 11 is being chosen – it's the number of standard pizzas from menu in the order history. Then, two random numbers are generated, lets call them *pizzas_1* (*0 <= pizzas_1 <= number_of_standard_pizzas)* and *pizzas_2* (*number_of_standard_pizzas* - *pizzas_1*). Data is sampled from pizza DF twice – first time using pizzas_1 and second time using *pizzas_2*. Thanks to this procedure, we can have up to two repetitions of pizzas in order history (for example two Margarithas). Afterwards, 25 - number_of_standard_pizzas custom pizzas with completely random toppings are generated. Finally, algorithm assigns 0 or 1 in column *'Good_Pizza?'* to all of the records, with 0 corresponding to *'Bad Pizza'*, and 1 to *'Good Pizza'*. In this way, in each execution of the program we are dealing with different, randomized order history.

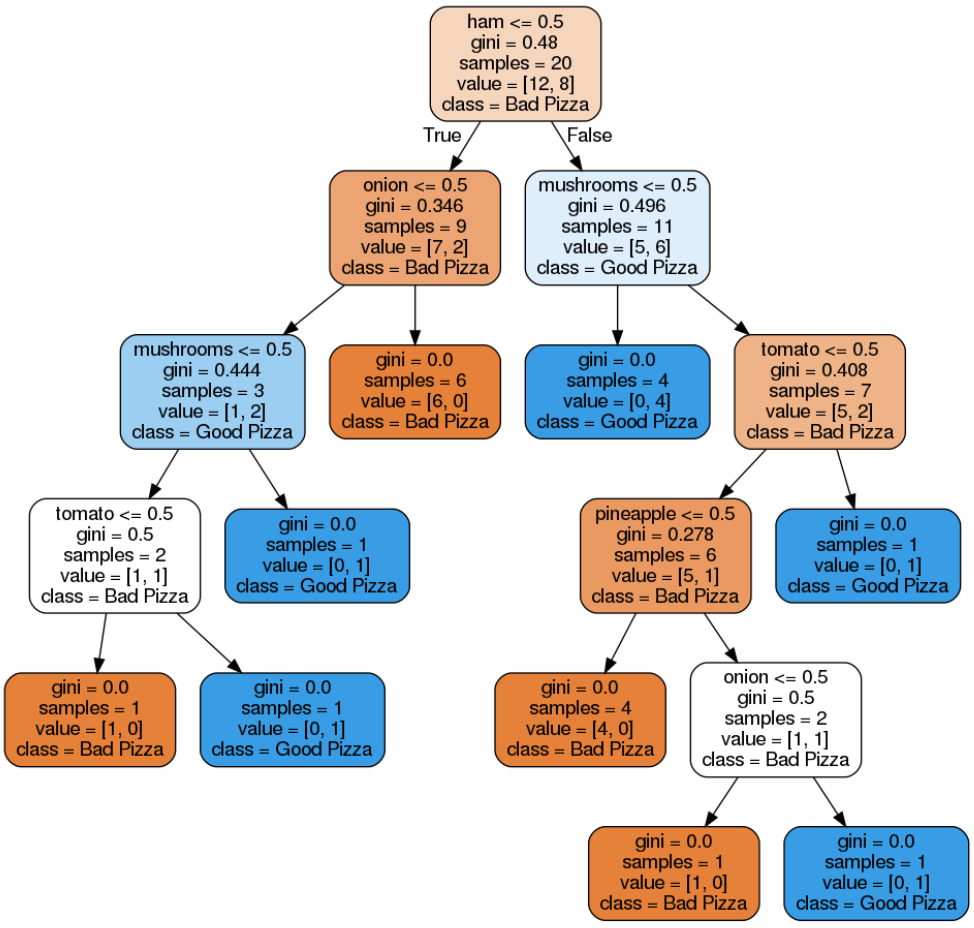
II. Decision Tree

The tree in the subproject is trained using *DecisionTreeClassifier* class, which comes from **scikit-learn** library. It is created using **CART** algorithm, which differs from **ID3** in the usage of **Gini impurity** instead of **Entropy**. We can express Gini impurity with formula

```
Gini = 1 – Σ (Pi)² for i=1 to number of classes
```

In short, gini impurity tells us what is the probability of a new data point being incorrectly classified. Instead of **information gain**, we calculate **gini gain** by taking a weighted sum of the gini impurities.

Tree is visualized by *pydotplus.*



III. Integration

After training decision tree for certain order history, it is used to predict whether the customer will like the pizza from menu or not. Basing on this information, waiter suggests the customer, which pizza will he like. The customer enters the restaurant, waiter approaches him, and they have a short conversation, directed for the purpose of presenting the algorithm.