Krista Miller

COMP 4448: Data Science Tools II Assignment 2

**Directions:** Do this assignment in Jupyter Notebook and provide screenshots of code and output in this word document wherever required. You will upload this word document containing screenshots of code and answers as well as your Jupyter Notebook to Canvas. All assignments will be submitted and graded through canvas and grades will be transferred to the 2U platform.

**Goal:** The goal of this assignment is to give you the opportunity to develop an intuition for classification using rule-based classification.

**Packages:** Core packages you may need for this assignment include numpy, pandas, sklearn. Matplotlib.pyplot and/or seaborn.

**Points:** 50

Question 1

You are provided with the following training data set

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | refund | marital\_status | income\_above\_80k | cheat |
| 1 | yes | single | yes | no |
| 2 | no | married | yes | no |
| 3 | no | single | no | no |
| 4 | yes | married | yes | no |
| 5 | no | divorced | yes | yes |
| 6 | no | married | no | no |
| 7 | yes | divorced | yes | no |
| 8 | no | single | yes | yes |
| 9 | no | married | no | no |
| 10 | no | single | yes | yes |

The data is to be used to predict whether individuals will cheat in filing their taxes or not. The attributes are **refund**, indicating whether an individual received tax refund or not, **marital\_status** indicating whether the individual is *married*, *single*, or *divorced*, the income\_above\_80k, indicating whether an individual’s taxable income is *above $80,000* or not. The output variable, **cheat**, is a binary variable indicating whether an individual cheated in filing taxes or not.

The following decision tree can be used to extract the classification rule for the data provided.

Predict No

No

Yes

Married

Single or divorced

Predict No

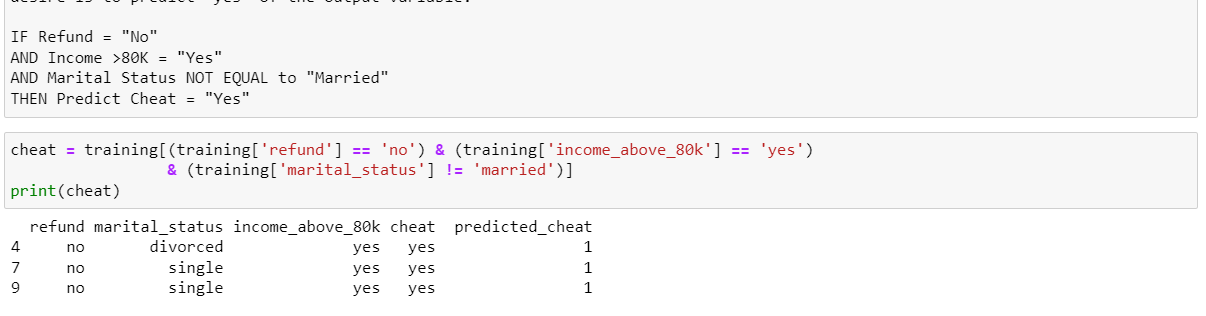
Predict No

Predict Yes

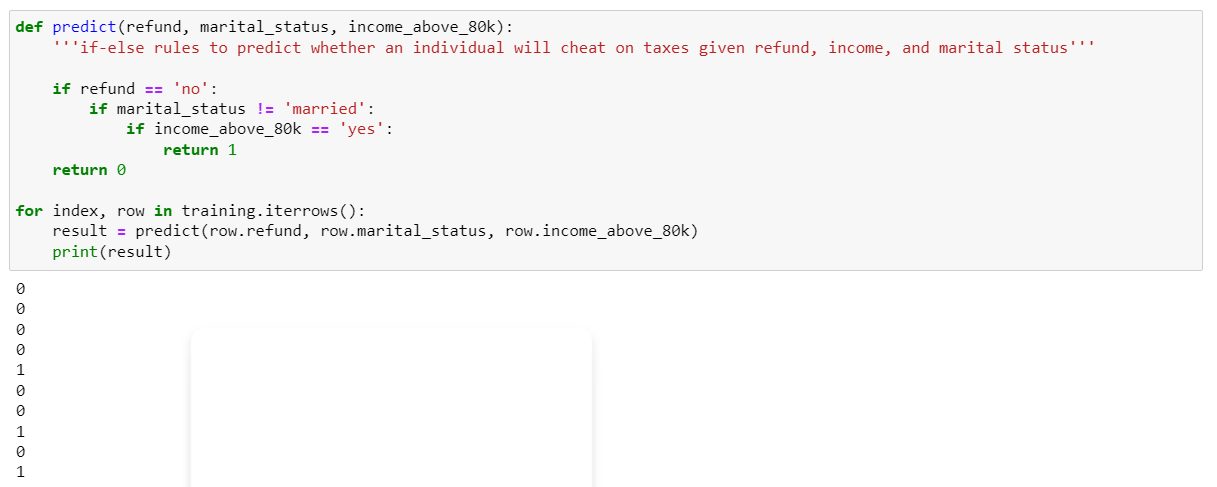
Yes

No

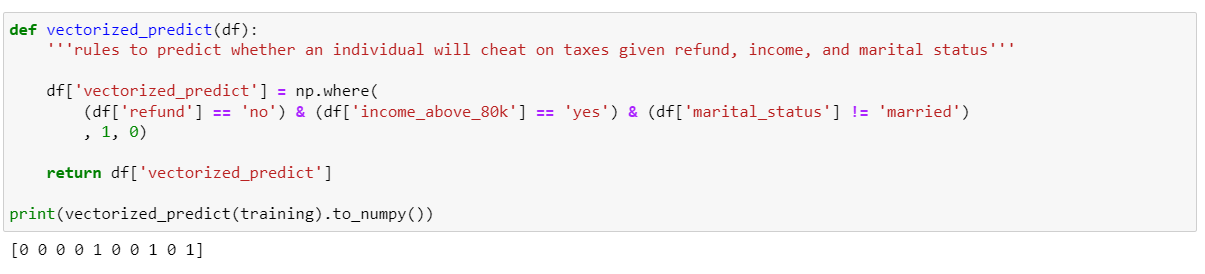
1. Write an IF-THEN rule derived from this decision tree, that can be used to classify instances in the data suppose your desire is to predict “yes” for the output variable.



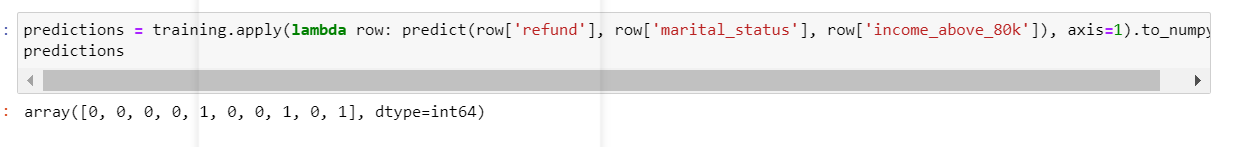
1. In code, create a function with an if-else statement to implement the rule you wrote above in question 1a. You can name the function **predict.** The function takes the input data and returns a vector of predicted output values. You can decide if you want your function’s input data argument to be a numpy array or a data frame, ,then structure your function body to process the input data accordingly. Provide some brief document of your function. You can loop through each instance in the input data and assess whether the instance satisfies the rule for predicting “yes” or not. If the instance satisfies the rule for predicting “yes”, then predict 1, otherwise, predict 0.



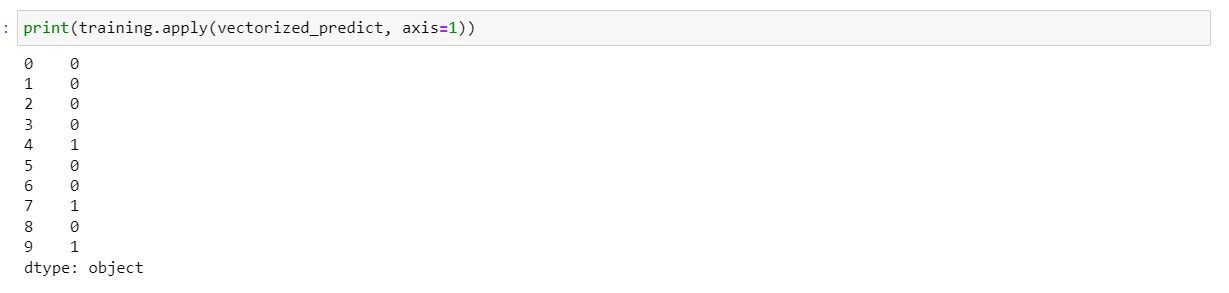
1. Create another function that implements the rule for predicting “yes” for the output, but this time, you will not use if-else statement or a loop. Rather use numpy arrays, matrices or vector to vectorize your code for faster and more efficient implementation. If the instance satisfies the rule for predicting “yes”, then predict 1, otherwise, predict 0. You can use logical operators for such implementation to compare values of test instances to values of attributes in the rule. This function still takes the input data and returns a vector of predicted values. You can call this function **vectorized\_predict.**



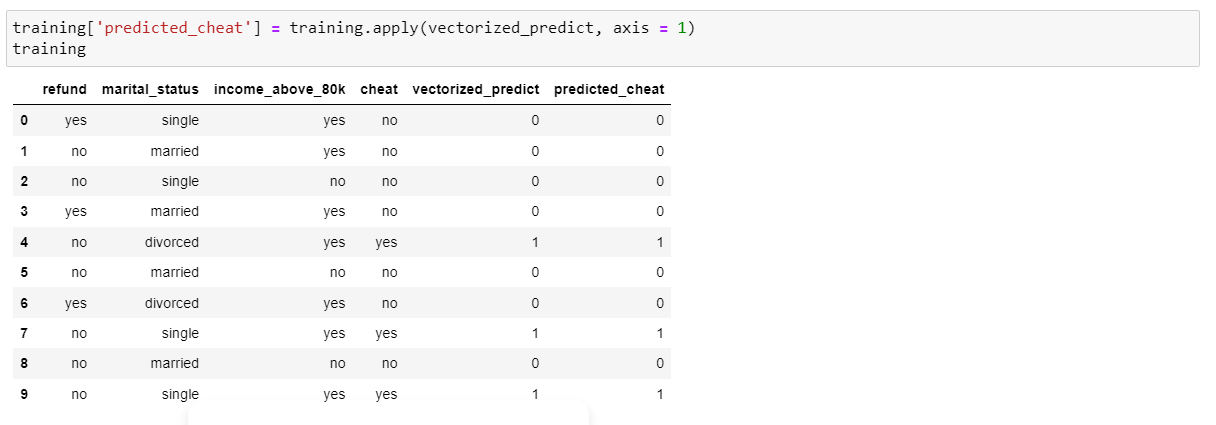
1. Select the input data (refund, marital\_status, income\_above\_80k) in the training data and apply your **predict** function to the input data to predict the outcome values of the input data. Your function should return a vector of predicted values.



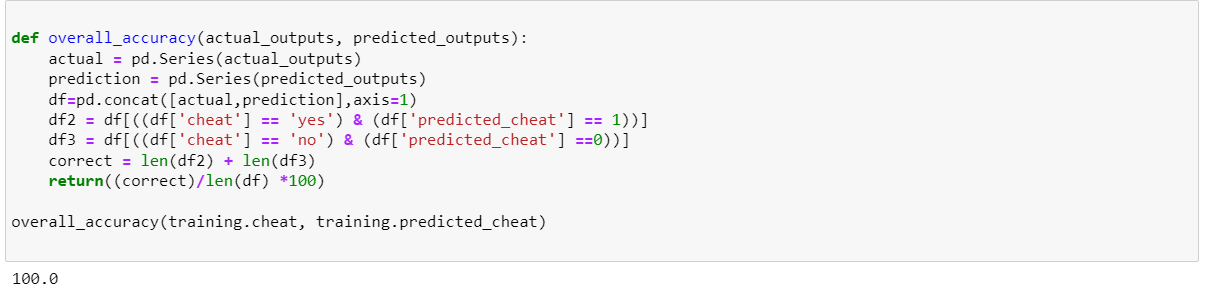
1. Select the input data (refund, marital\_status, income\_above\_80k) in the training data and apply your **vectorized\_predict** function to the input data to predict the outcome values of the input data. Your function should return a vector of predicted values. Are the predicted values with the **vectorized\_predict** function the same as the predicted values obtained through the **predict** function? (you should have the same results).



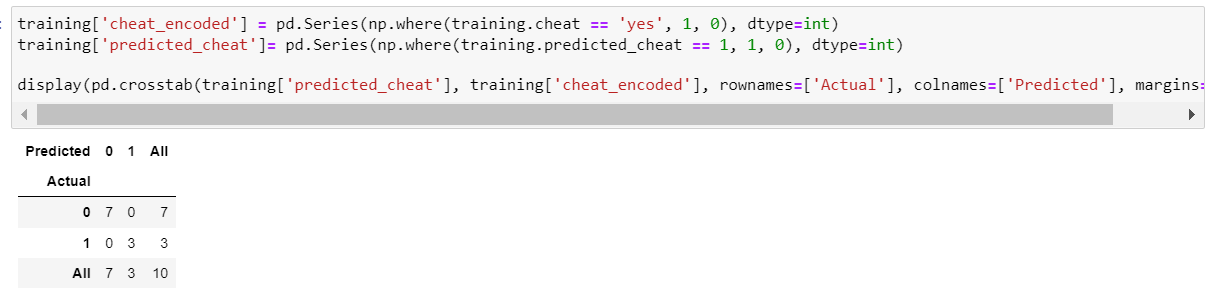
1. Include the predicted values as a column to the training data and name that column **predicted\_cheat**.



1. Create a function that computes the overall accuracy of the classification. You can call this function **overall\_accuracy**. The function should take two arguments, a vector of actual output values and a vector of predicted output values, then returns the classification accuracy. Inside the function, you can use boolean logic to compare the actual and predicted values, then count the proportion of how many predicted values are equal to the actual values to get the overall accuracy.



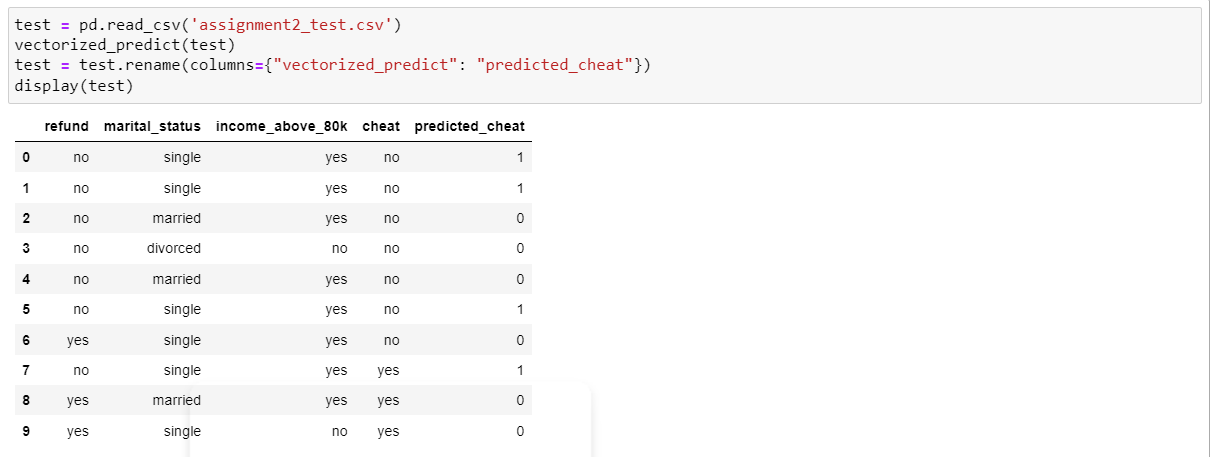
1. Create a confusion matrix using the predicted and actual output values. You can use pandas crosstab function.



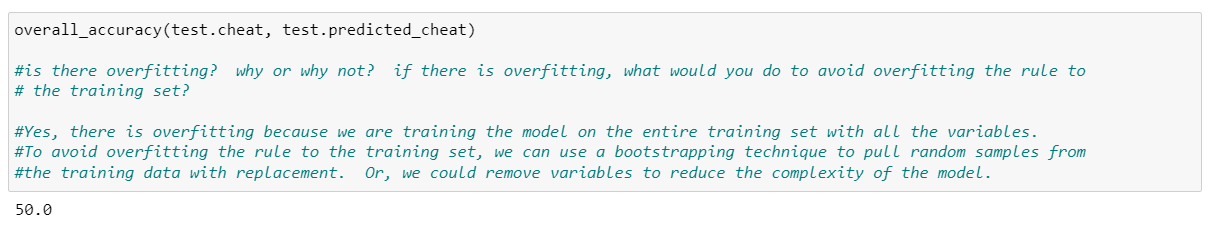
1. Supose the test set data is as follows.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | refund | marital\_status | income\_above\_80k | cheat |
| 1 | no | single | yes | no |
| 2 | no | single | yes | no |
| 3 | no | married | yes | no |
| 4 | no | divorced | no | no |
| 5 | no | married | yes | no |
| 6 | no | single | yes | no |
| 7 | yes | single | yes | no |
| 8 | no | single | yes | yes |
| 9 | yes | married | yes | yes |
| 10 | yes | single | no | yes |

Apply the **vectorized\_predict** function to the input of the test dataset to predict the output values for the test dataset. Include these predicted values as a column to the test dataset and called this column **predicted\_cheat.**



1. Apply the **overall\_accuracy** function to the predicted out values and actual output values of the test set to compute the overall accuracy. Compare the overall accuracy of the test set and that of the training set. Is there overfitting? Why or why not? If there is overfitting, what would you do to avoid overfitting the rule to the training set?



**Question 2**

You will use the same training dataset provided in question 1. Suppose we wanted to create one rule using only one attribute that best classifies the input data, you will need to write a function or an algorithm to find that best attribute. There are different approaches we can use to evaluate the best attritubute that will classify our data with the highest accuracy. One of the accuracy measures we can use to find the best attribute is information gain, which uses entropy.

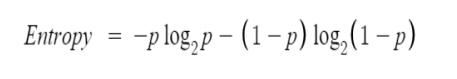
1. Given the training dataset in question 1, write a function that computes the entropy of output variable. The function should take the entire training dataset, and the name of the output variable as arguments and return the entropy value. You can call the function, **entropy.**

Note that entropy H(S) of the target variable T with possible values for a set of S examples is given by:

H(S) =

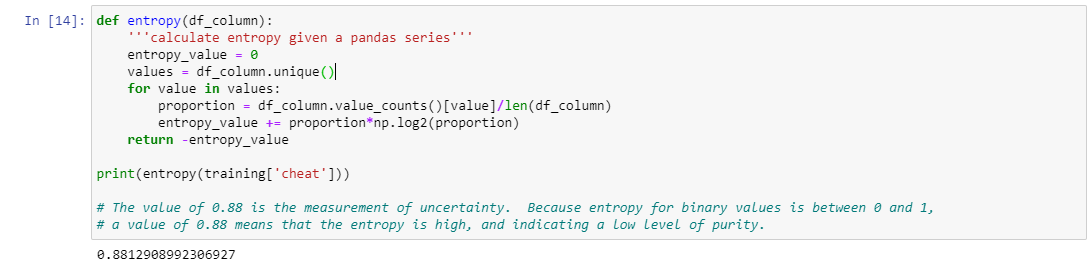
where

Alternatively, you could also define entropy for a binary target variable as:

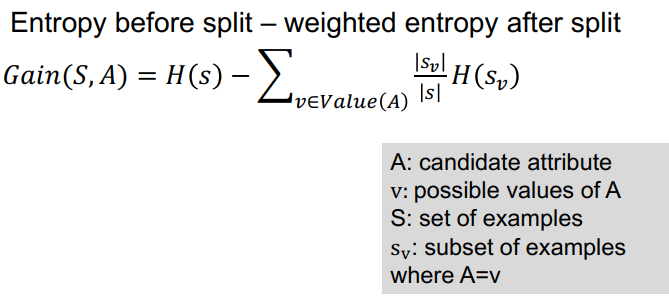


Where p = proportion of the desired class or proportion of “yes” in the given data.

Apply the entropy function to the training dataset to compute the entropy. Do you think the data is more pure, less pure or more/less pure?



1. Create another function that takes the dataset, the name of the output variable, and a specific input variable as arguments and returns the information gain for a split of the data on that specific input variable. You can call the function, **information\_gain.** Information gain is defined as:



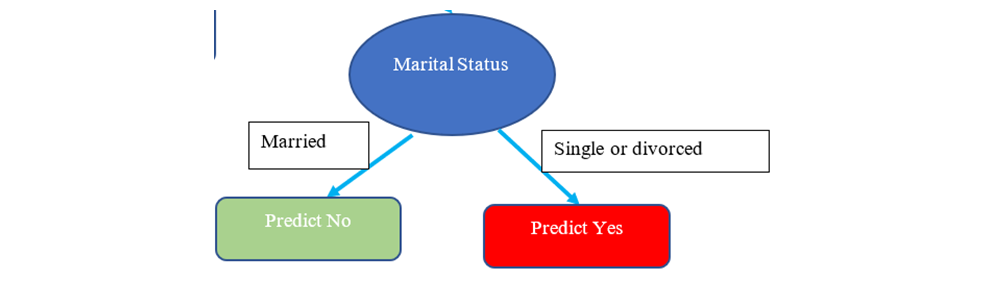
Apply the information\_gain function to each input variable to compute the information gain for each input variable. Which input variables is the best (has the highest information gain) for creating one-rule?



1. Do a cross tabulation using the best attribute obtained in 2b above and the output variable in the training dataset.



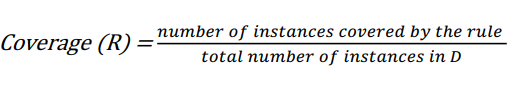
1. Draw a decision tree (use shapes in the word document) to show how the best variable can be used for classification. This is a decision tree with a single node or single variable.



1. Create a one-rule from the decision tree in 2d. The rule should containing an antecedent and a consequent. The antecedent should use only the best attribute and it’s value or values.

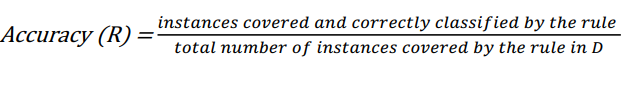


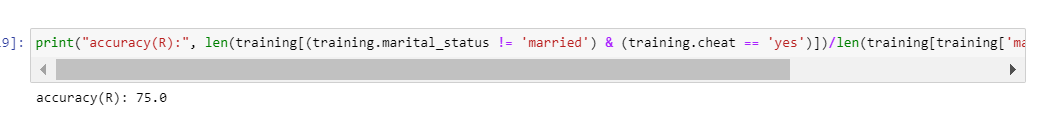
1. Using the antecedent of your rule, extract the data covered by the rule and compute the coverage of the rule.



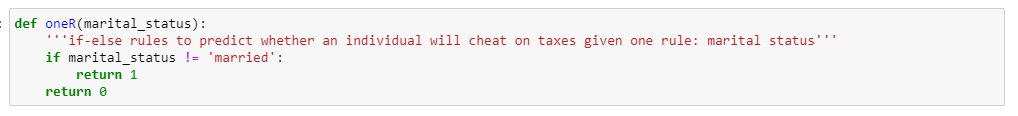


1. Using the antecedent and precedent of your rule, extract the data covered by the rule and compute the accuracy of the rule.

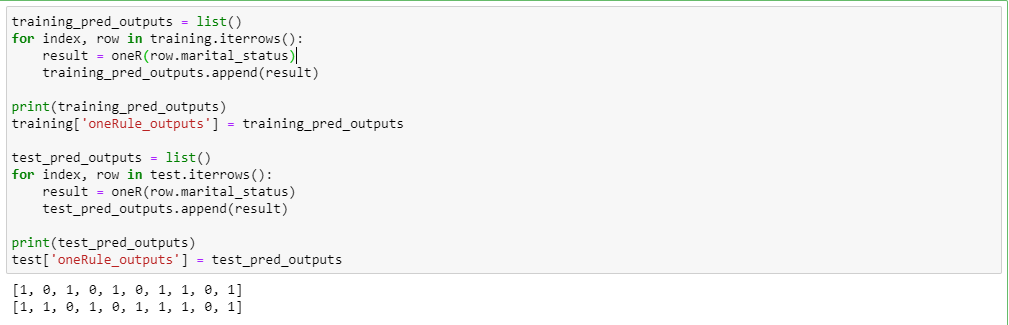




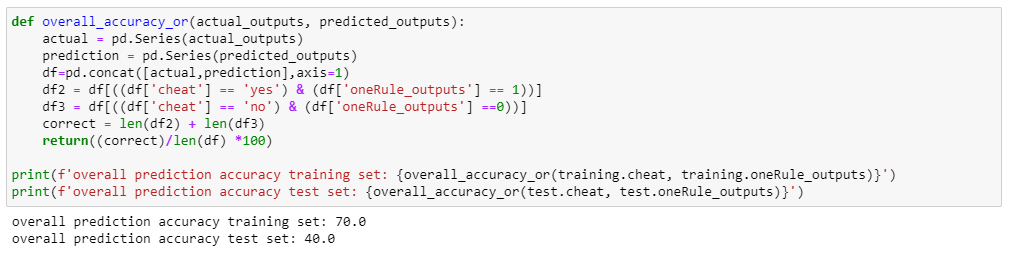
1. Create a function that implents the rule as an if-else statement to predict the outcomes of any instance (a vector of values associated with the input variables for a specific individual). The function should be able to take one or more instances as an argument in the form of a dataframe or numpy arrary.



1. Implemement the function on the training set and test set in question 1 to get the predicted outputs for the training set and test set.



1. What is the overall prediction accuracies for the training set and test set? You can use the overall\_accuracy function you initially defined.



1. Do you think there is overfitting or underfitting? Why or why not?

