Sean Wendlandt

4/23/23

COMP 4448: Data Science Tools II Assignment 3

**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.

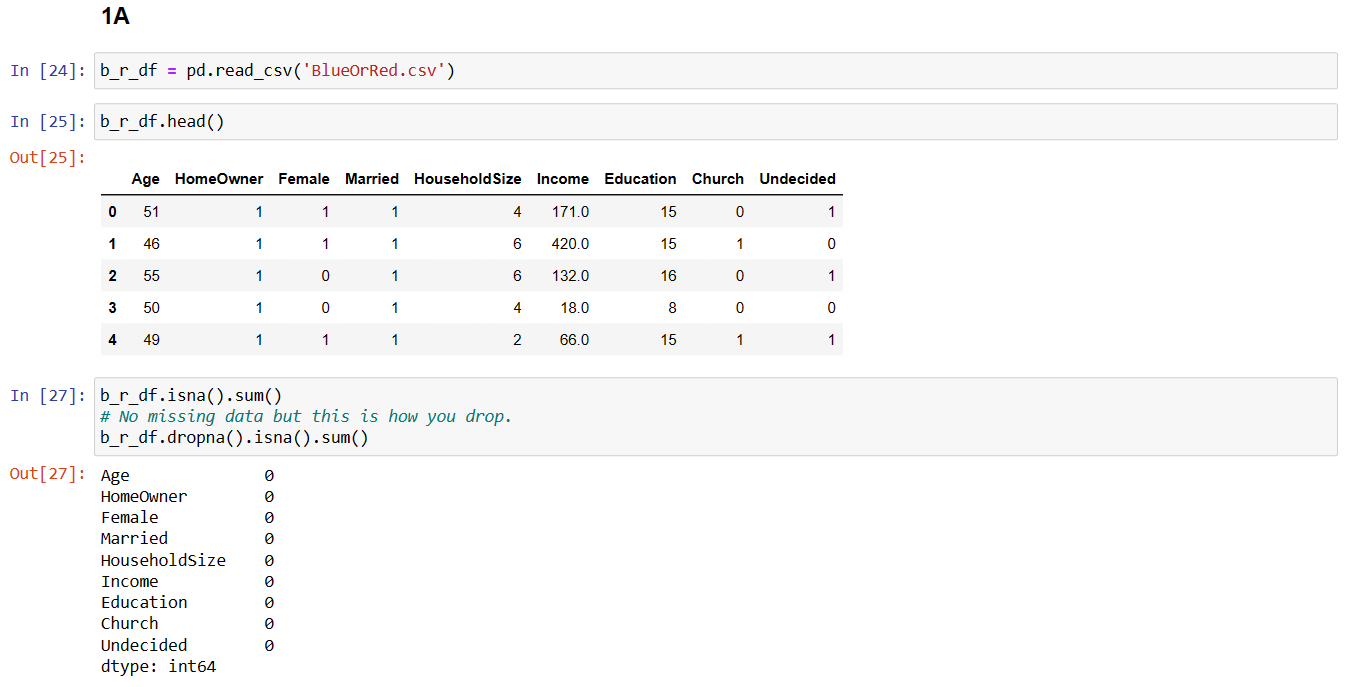
**Goal:** The goal of this assignment is to give you the opportunity to get acquainted with the modeling in scikit-learn using decision trees.

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

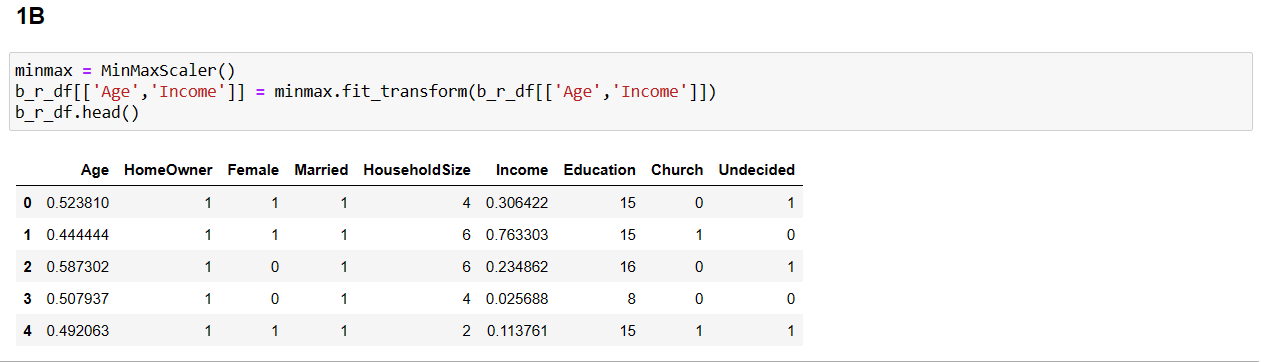
**Question 1**

For a presidential election, campaign organizers for both the Republican and Democrat parties are interested in identifying undecided voters who would consider voting for their party. The file **BlueOrRed.csv** (provided on canvas)contains data on a sample of voters with tracked variables including: whether or not they are undecided regarding their candidate preference, age, whether they own a home, gender, marital status, household size, income, years of education, and whether they attend church. For this dataset, you would use the **Undecided** variable as your output variable and the rest of the variables as your input variables. You want to predict whether a candidate is undecided(Undecided=1) or decided(Undecided=0).

1. Upload the data into Python and check for missing data. If there is any missing data, drop the cases that have the missing data.



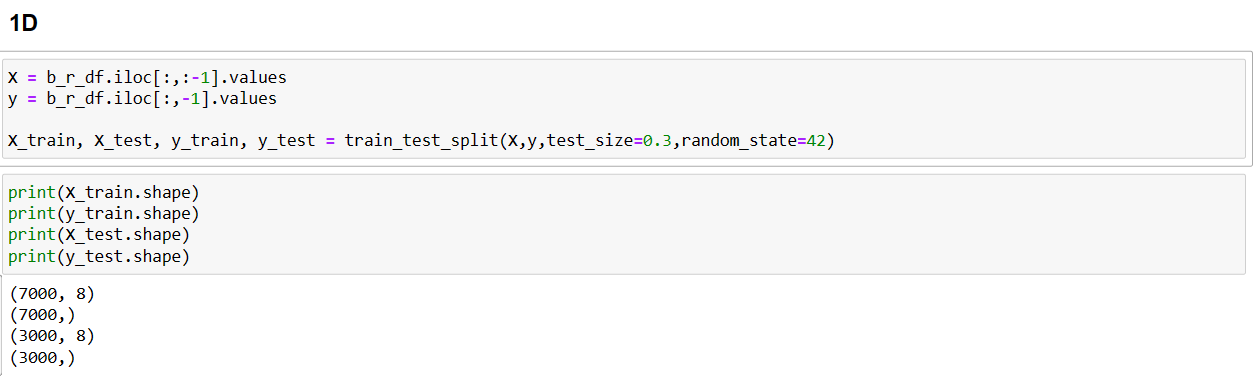
1. Normalize the continuous variables using min-max normalization. You should obtain a new dataset that has your normalized continuous input variables, the categorical input variables (the categorical input variables are fine and don’t need to be normalized), and your output variable.



1. Why do you think using min-max normalization is a better choice for this situation compared to using other scalers such as the Standard Scaler which standardizes the data?

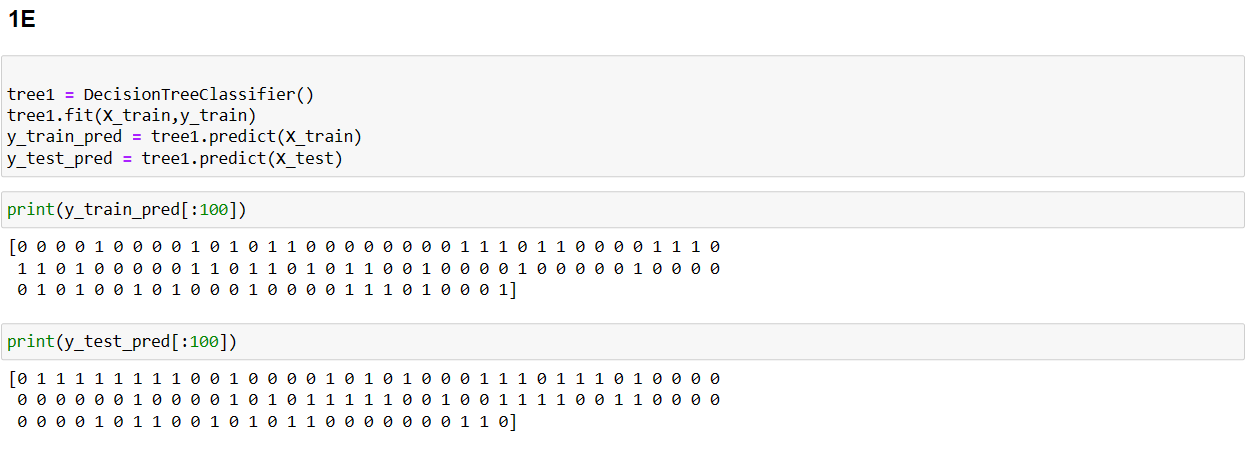
I think minmax scalar is better in this situation as the income variable may be skewed or have outliers which are handled better with this method.

1. Use the train\_test\_split() function in the sklearn.model\_selection module in scikit-learn to split the data into training and test sets. Your test set should be 30% of the entire data. Print the shape of the X\_train, y\_train, X\_test and y\_test data. You can use the default value of the random state in splitting the data, which is random\_state=42.



1. Construct a decision tree and fit the tree into the data. Then use your decision tree model to make predictions on the training set as well as on the test set. Assign your results to **y\_train\_pred** and **y\_test\_pred** respectively. Print out the first 100 predictions for each of the **y\_train\_pred** and **y\_test\_pred**.

Paste a screenshot of your code and output here

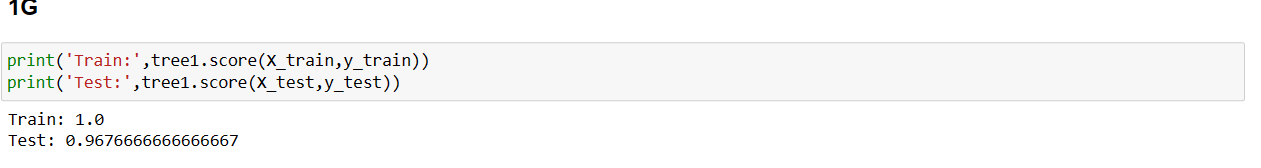


1. Find the overall accuracy of the tree model on the training set and on the test set. Use the accuracy\_score() function in the sklearn.metrics module. Does the tree overfit the model? Why or why not?

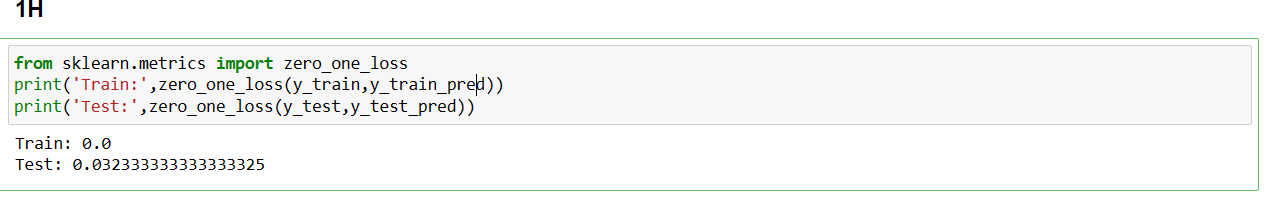


I would argue that the tree is not overfitting as it still is able to predict the test dataset with high accuracy. This could be due to some strongly correlated variables to the output that works well with this model.

1. Also, use the .score() method of the tree estimator to find the overall accuracy on the training set and test set.



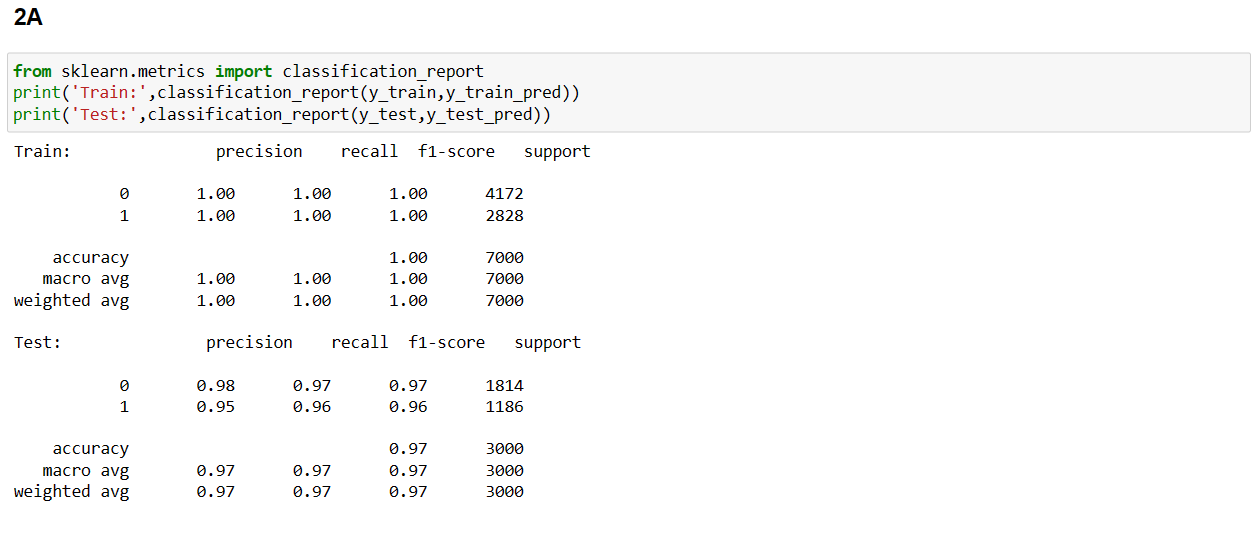
1. Use the zero\_one\_loss() function inside the sklearn.metric module to compute the overall prediction error of the tree model on the test set and training set.



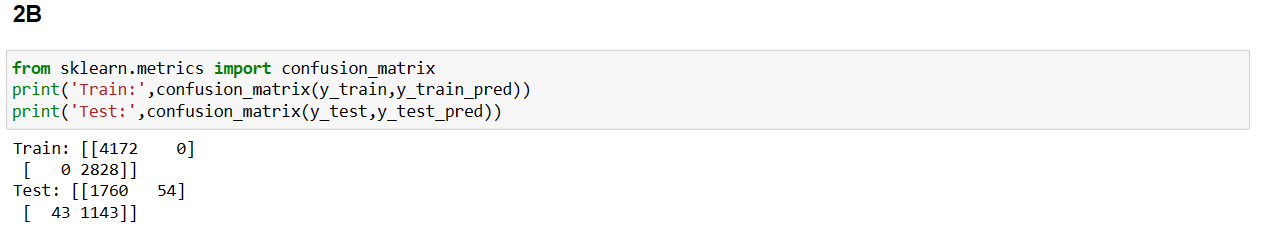
**Question 2**

Do other assessment of your model using the following tools in sklearn.metrics module

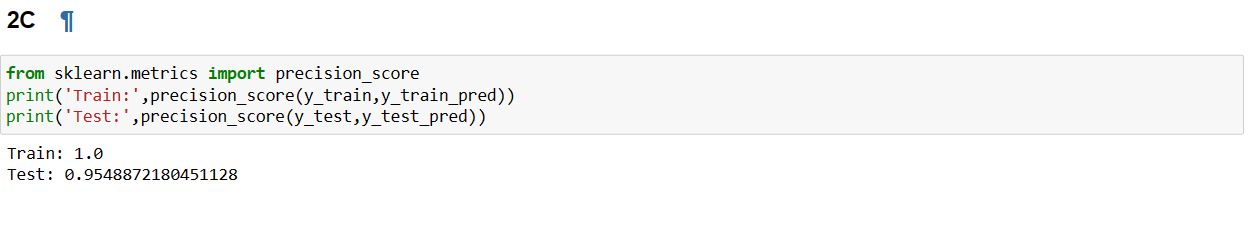
1. Generate the classification report using the classification\_report ()



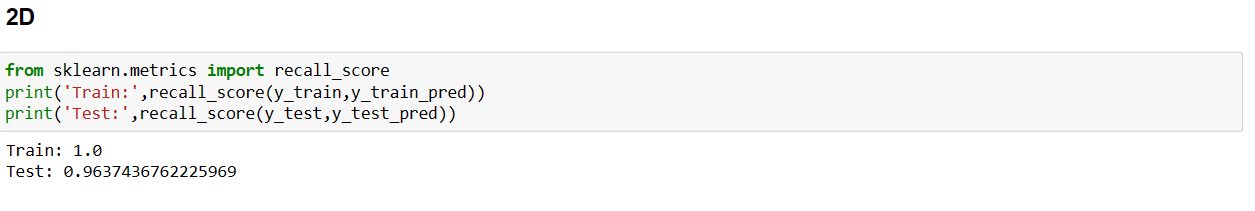
1. Generate the confusion matrix using confusion\_matrix()



1. Generate the precision score using precision\_score()



1. Generate the recall score using recall\_score()



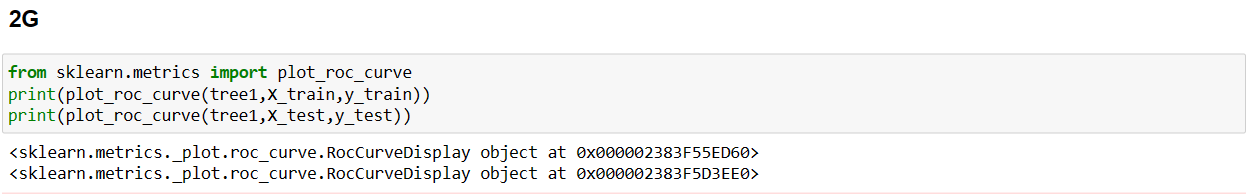
1. Generate the f1 score using f1\_score()

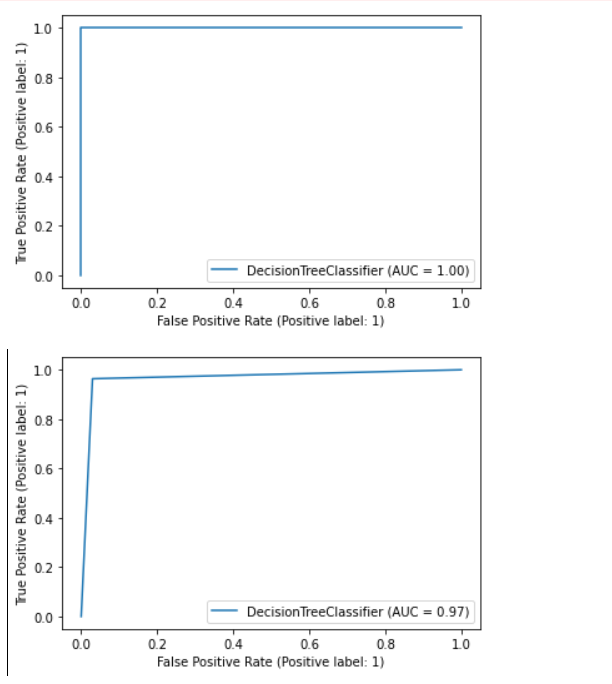


1. When is it more appropriate to use f1 score (or precision score and recall) compared to using the overall accuracy to evaluate your model?

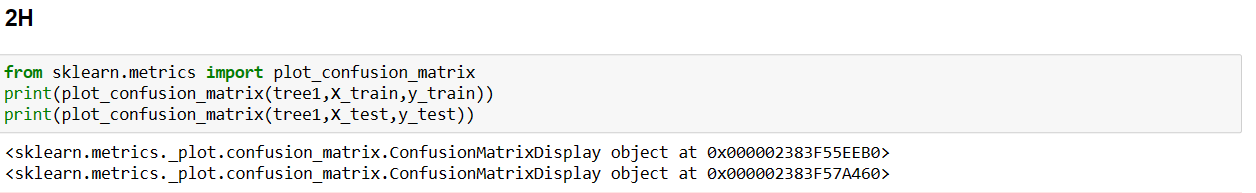
F1 score, precision, and recall are more appropriate than overall accuracy when dealing with imbalanced classes, misclassification costs, or when optimizing the trade-offs between precision and recall. They are also more commonly used in binary classification problems. Depending on the problem at hand, it is important to carefully consider the specific characteristics and requirements in order to choose appropriate evaluation metrics for a model.

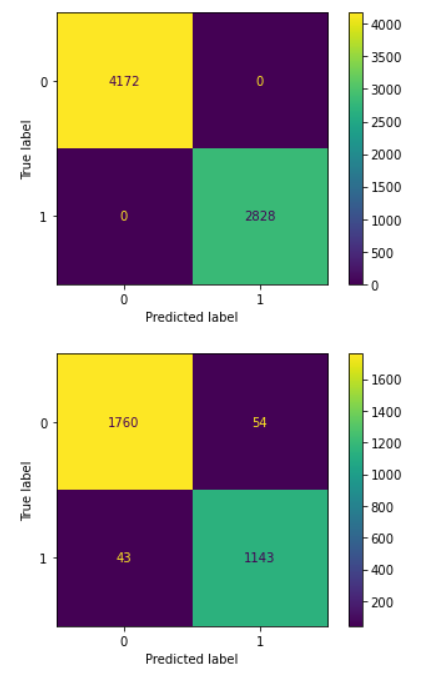
1. Generate a roc curve using plot\_roc\_curve()





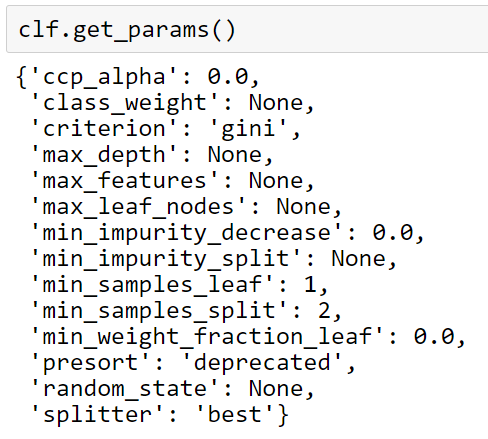
1. Plot the confusion matrix using plot\_confusion\_matrix





**Question 3**

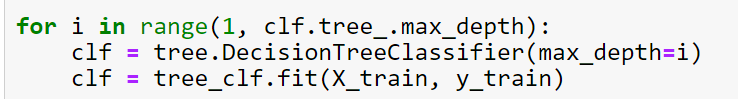
You will now use cross validation to find better accuracy scores as well as to tune your hyperparameters. The hyperparameters of a decision tree as shown below. We are more interested in optimizing the depth of the tree (max\_depth).



1. First retrieve the maximum depth of this decision tree. Use **clf.tree\_.max\_depth** where clf is the name of your estimator and **tree** is the module in sklearn.

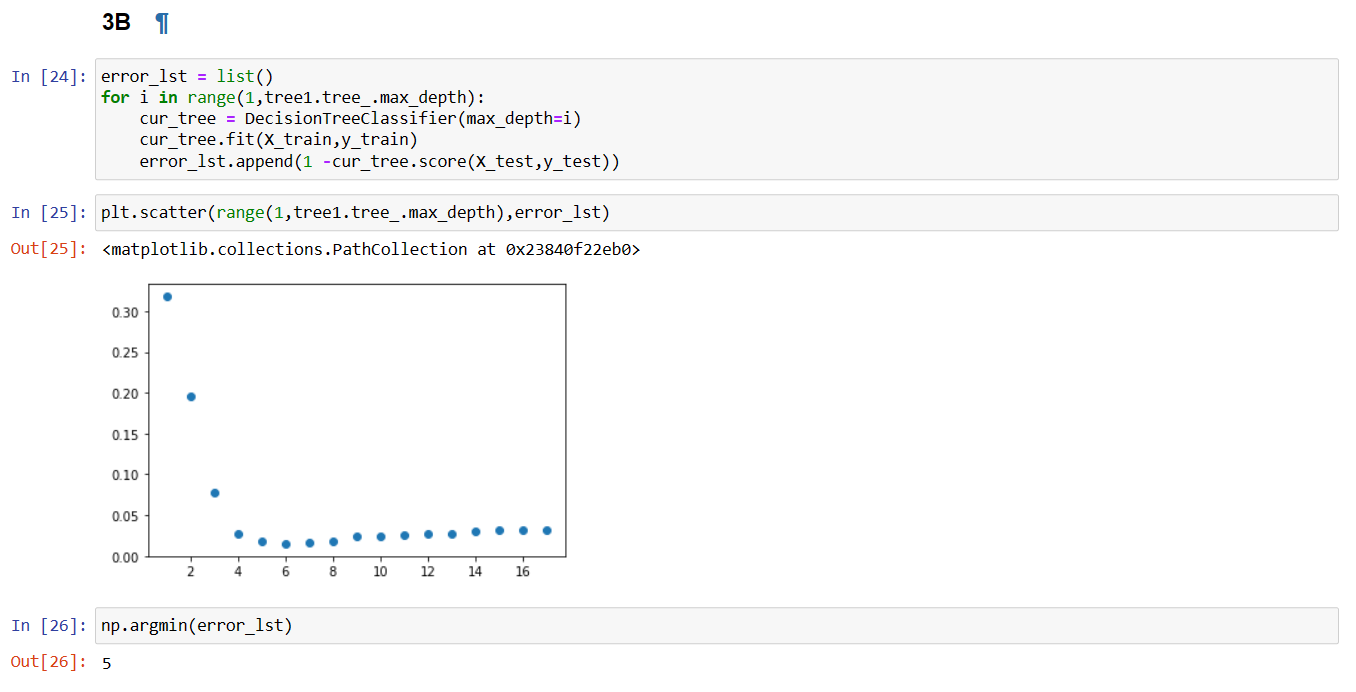


1. You will find the optimal depth of the tree by tuning the max\_depth hyperparameter. Write a for loop to iteratively compute the overall error rate of the model on the test set only for the possible values of the maximum depth of the tree. Here are ideas for your code (See class notes on **Modeling in Scikit-learn** for more ideas):



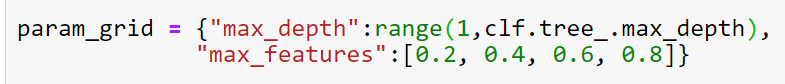
You will need to initialize a variable outside the loop to track the error rates for each maximum depth value.

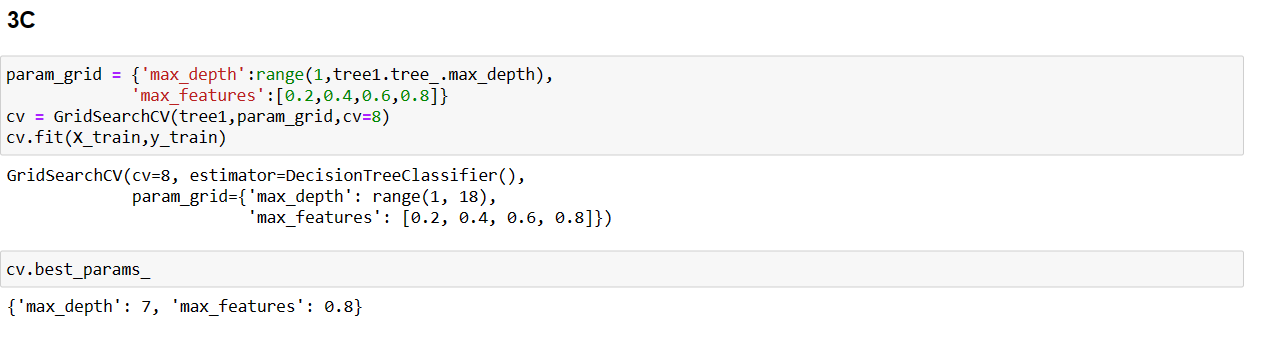
Then plot the overall error rate on the y-axis and the maximum depth of the tree on the x-axis. Also use np.argmin() on the error rates you obtained to retrieve the optimal maximum depth of the tree with the least error rate. Does this value match with what you see on your plot?



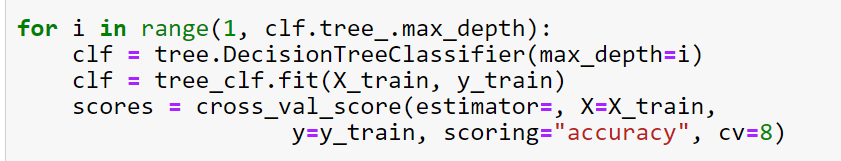
The values match with the plot as you can see the error rate obtains the lowest point on the 6th point on the graph and the 5th(or 6th since it starts at 0) on the error\_list.

1. Now, you will still find the optimal maximum depth of the decision tree model using grid search cross validation. Additionally, you will be simultaneously optimizing the **max\_features** hyperparameter as well. You can check the documentation to see what these hyperparameters represent. You will use the GridSearchCV() constructor inside the sklearn.model\_selection module. Make sure to pass the arguments (**estimator** and **param\_grid** into the GridSearchCV() constructor. Set the parameter cv=8. Do you have the same optimal maximum depth as you got in the question above? (See class notes on **Modeling in Scikit-learn** for more ideas). You can use these parameter grid as one of the arguments:

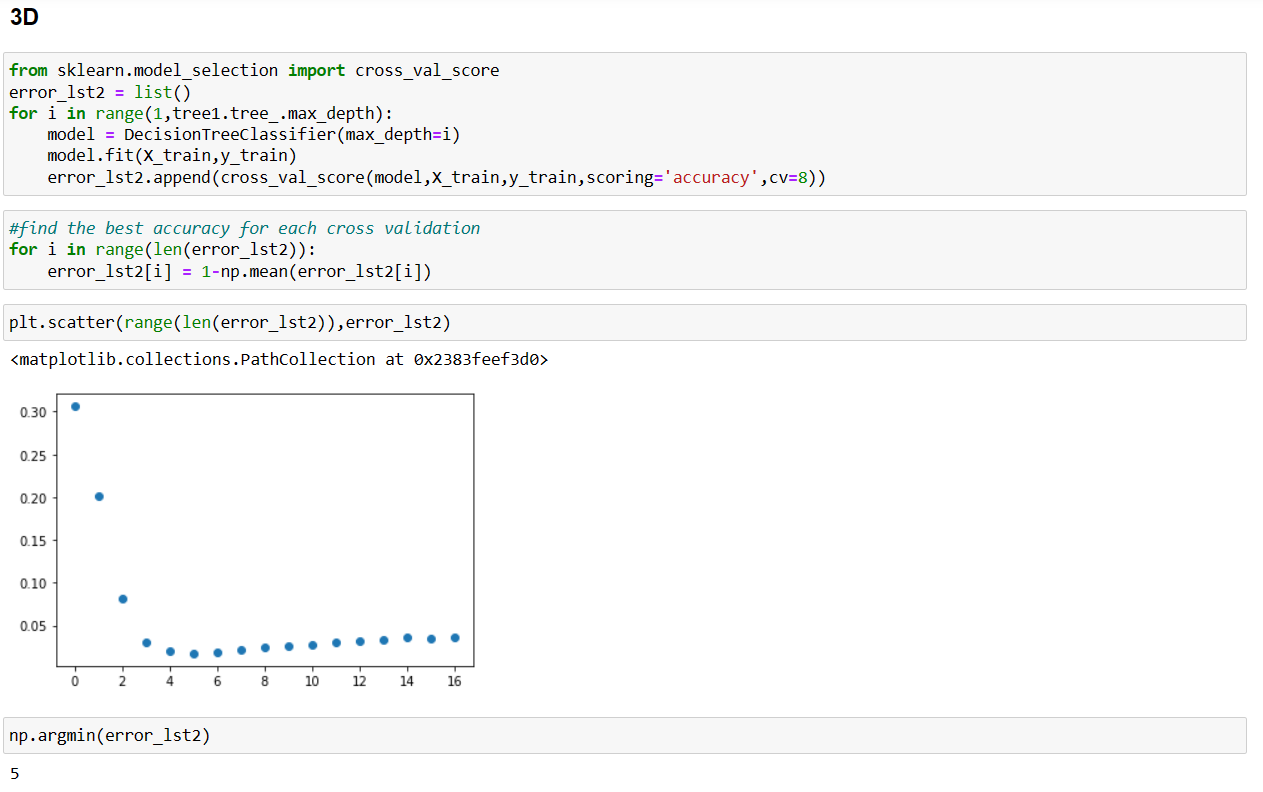




1. You will tune the maximum depth hyperparameter again using cross validation but this time, you will use the **cross\_val\_score()** function inside the sklearn.model\_selection module. You will use this cross\_val\_score() function in combination with a for loop to find the optimal maximum depth of the tree. Here is a code snippet idea:



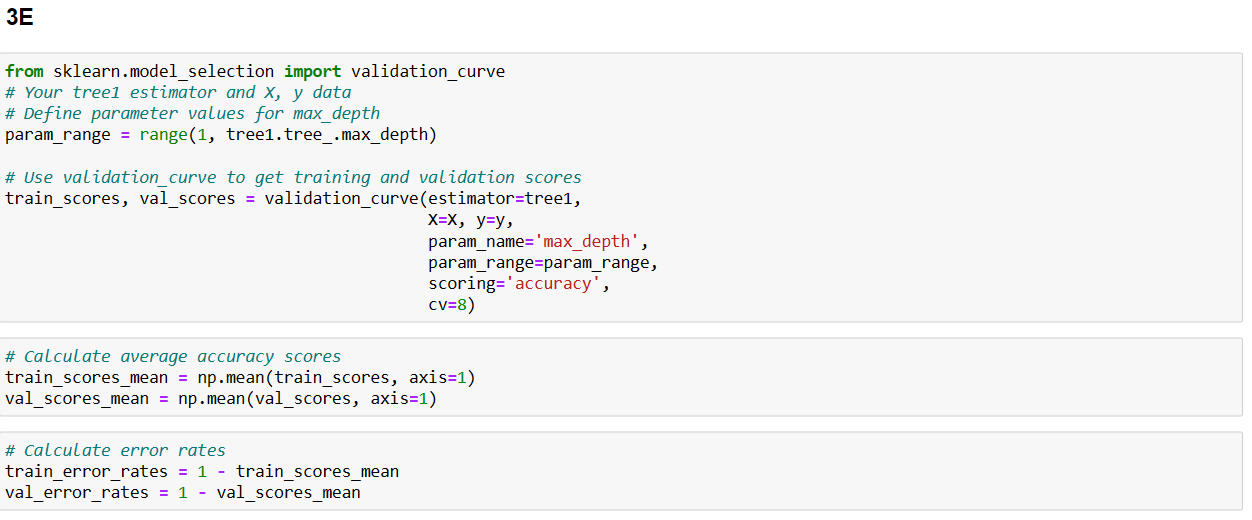
Track the error rate for each depth and plot the error rates versus the corresponding maximum depths of the tree. Also use the np.argmin() on the error rates to retrieve the optimal maximum depth , corresponding to the lowest error rate. Does the retrieved optimal maximum depth look like what you see on your plot? ? (See class notes on **Modeling in Scikit-learn** for more ideas).

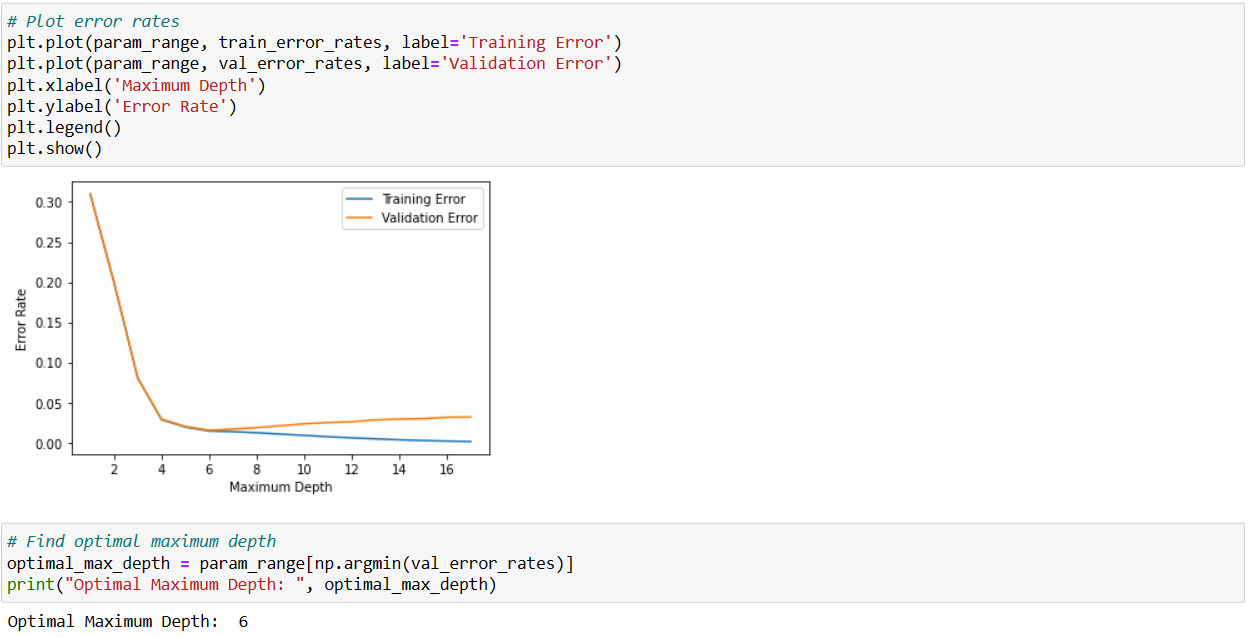


Does the retrieved optimal maximum depth look like what you see on your plot?

The values match with the plot as you can see the error rate obtains the lowest point on the 6th point on the graph and the 5th(or 6th since it starts at 0) on the error\_lst2.

1. You will use another approach to plot the error rates versus the maximum depth of the tree. Here, you would use the validation\_curve() function inside the sklearn.model\_selection module. This function implements cross validation and returns the training accuracy scores and test (validation) accuracy scores for each iteration in the k-fold cross validation, for each of the hyperparameter values. You will need to aggregate the scores using np.mean(axis=1) to get the average of the accuracy scores for all iterations, for each hyperparameter value. Use these average accuracy scores to obtain corresponding error rates for the training and test (validation) sets. Then plot the error rates on the training set versus the maximum depth values. Also plot on the same figure, the error rates of the test (or validation) set versus the maximum depth values. What is the optimal maximum depth? (See class notes on **Modeling in Scikit-learn** for more ideas).





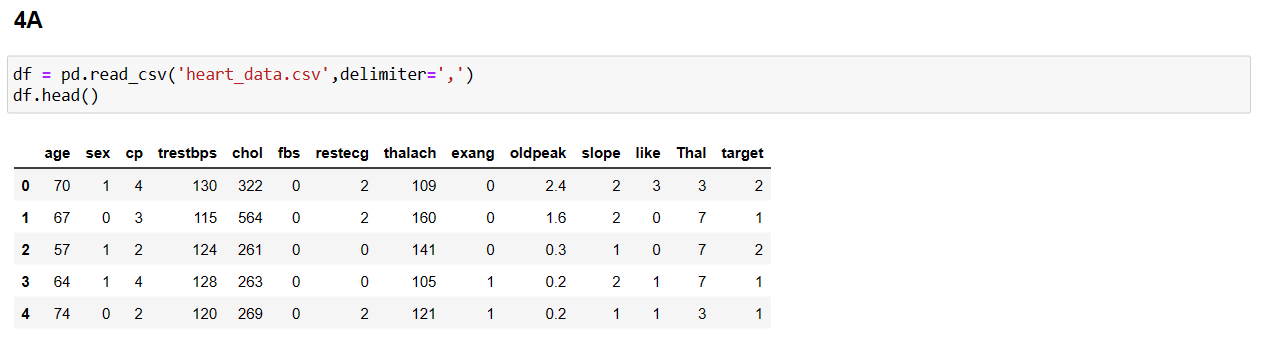
Optimal max depth: 6

Question 4:

Find your own dataset suitable for classification with at least three input variables and 200 cases: You will build a decision tree classifier and a random forest classifier. Find some interesting dataset instead of the popular iris data, etc. Feel free to use a dataset suitable for classification from this link provided below or some other source of your choice: <https://vincentarelbundock.github.io/Rdatasets/articles/data.html>

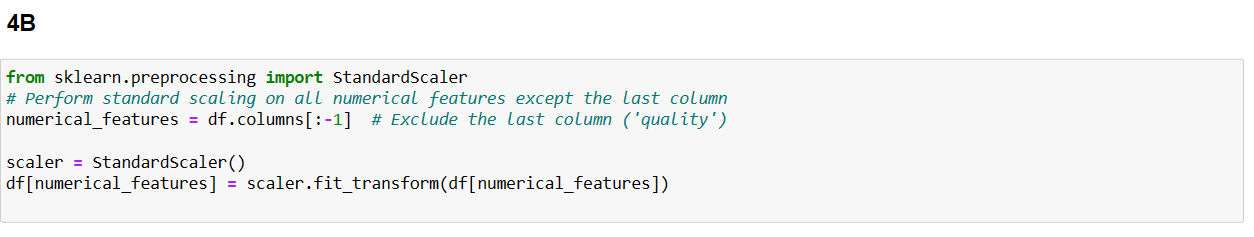
Address the following and include code/output snippets from b) to f). Include the response under each sub question.

1. State your research question, for example: **Are decision trees and random forest good models for predicting whether someone will default on a loan or not based on their age and income level?** This is just an example, your dataset does not have to be (or should not be) about loans.



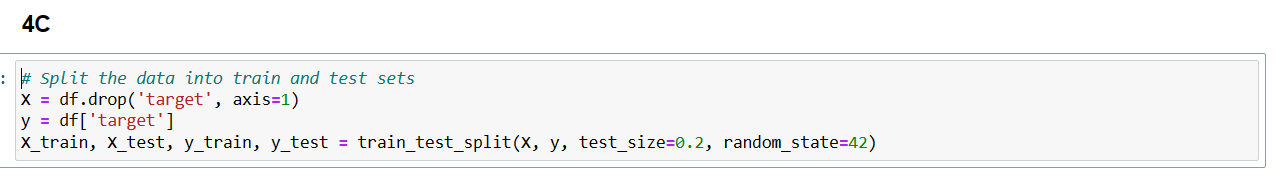
**Using a commonly seen classification dataset that is prediction heart disease (1=no, 2=yes)**

1. Data pre-processing (to the extent deemed necessary)



Using standard scalar as no crazy outliers.

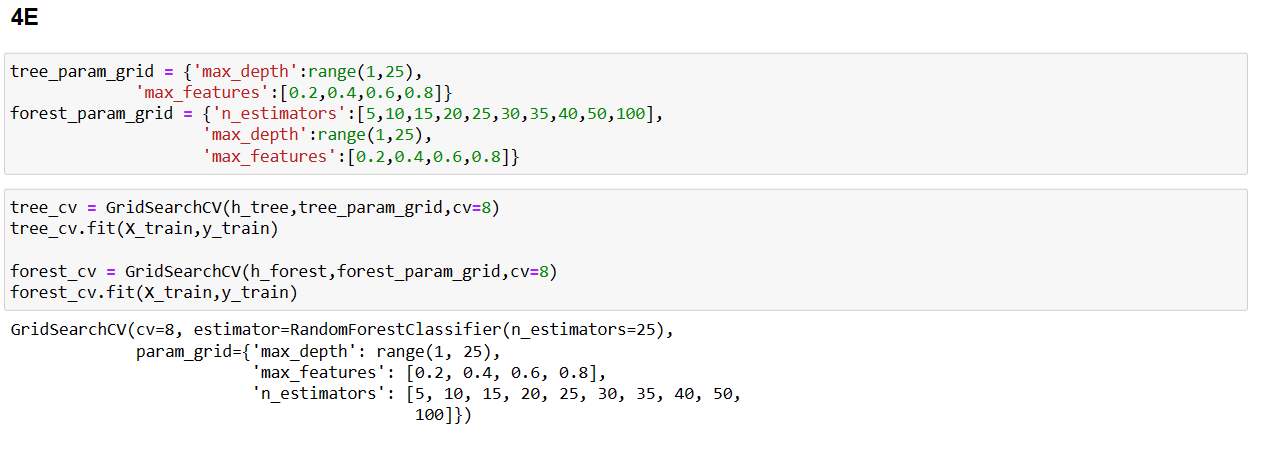
1. Data splitting



1. Model construction (a decision tree and a random forest)

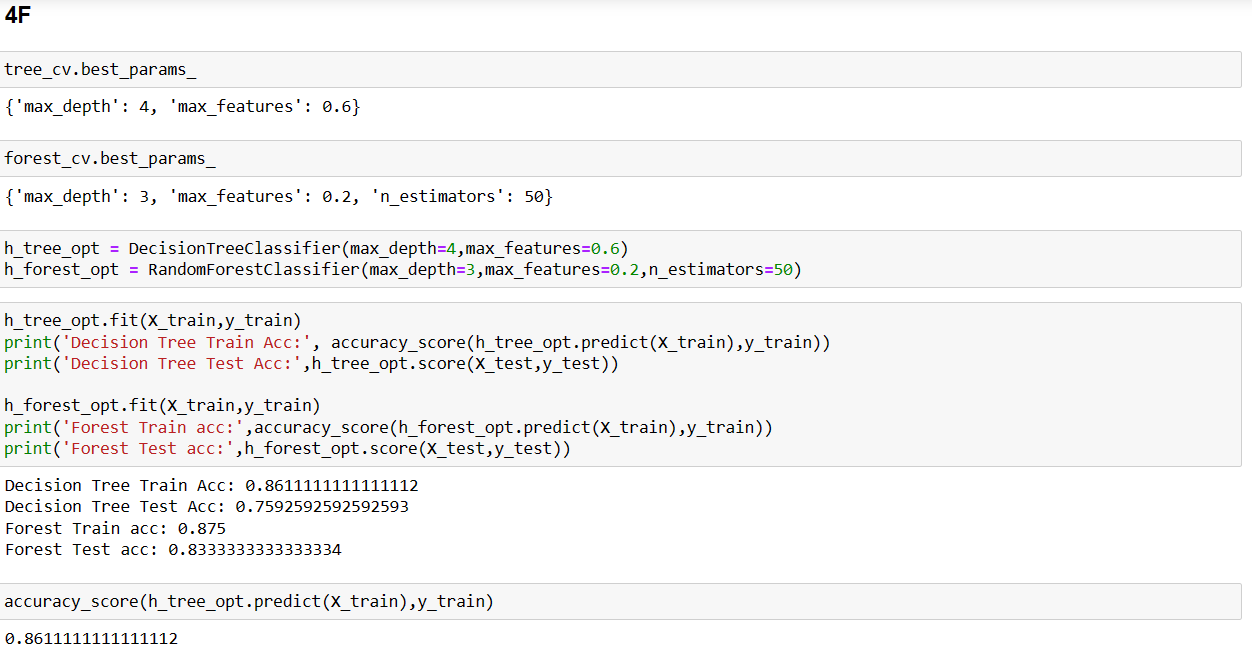


1. Hyperparameter turning (for each model, tune the hyperparameter that is important to you and use any of the methods for hyperparameter tuning learned in class such as cross validation with for loop, gridsearch cross validation, etc. You could tune more than a single parameter for each model if you want).



Grid search to find best params.

1. Use the best or optimal parameter values to build a model, then compute the accuracy score for the decision tree and for the random forest).



1. Discuss about overfitting for both models and, also discuss which model is better for classification for your dataset and why?

The Random Forest model is the better performing model in this case, as it shows higher accuracy on both the training and test data compared to the Decision Tree model. Furthermore, the difference between training and test accuracy is smaller in the Random Forest model, indicating better generalization performance. Neither model is overfitting as they only achieve an accuracy of .861 tree and .875 forest accuracies on the training data. If this were a real-world issue, especially in the medical field, further evaluation would be done for evaluation like the previous metrics performed in this assignment (F1, recall, precision, ROC, ect).

**Note:**

To construct a decision tree, you could use

**>>> from** **sklearn** **import** tree

**>>>** clf = tree.DecisionTreeClassifier()

**>>>** clf = clf.fit(X, Y)

To construct a random forest, you could use:

**>>> from** **sklearn.ensemble** **import** RandomForestClassifier

**>>>** clf\_rf = RandomForestClassifier(max\_depth=2, random\_state=0)

**>>>** clf\_rf.fit(X, y)