**Assignment 1** (15%)

CSE 5120 – Introduction to Artificial Intelligence – Fall 2024

*Submitted to*

Department of Computer Science and Engineering  
California State University, San Bernardino, California

*by*

Alexis Naranjo (008347411)

(Your collaborator in this homework (if any))

Date: *November 15th, 2024*

*Email:*

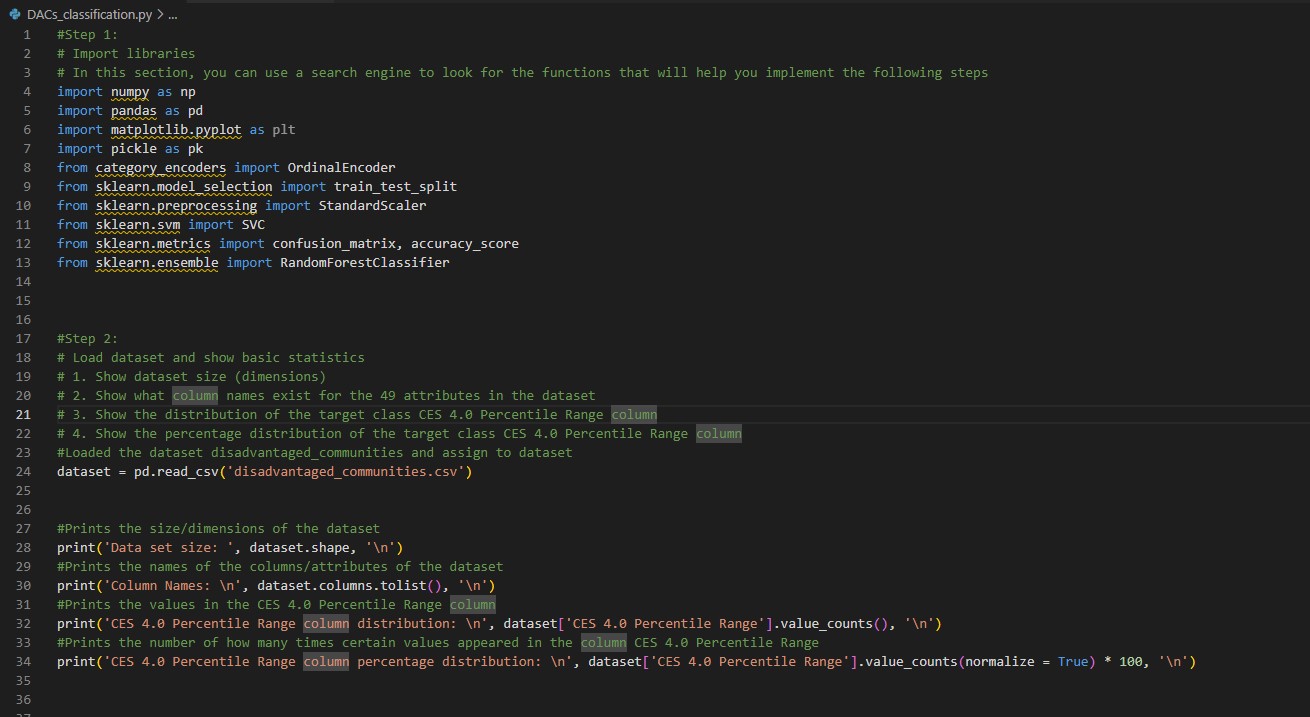
* *008347411@coyote.csusb.edu*
* *Your collaborator’s email (if you collaborated with any)*

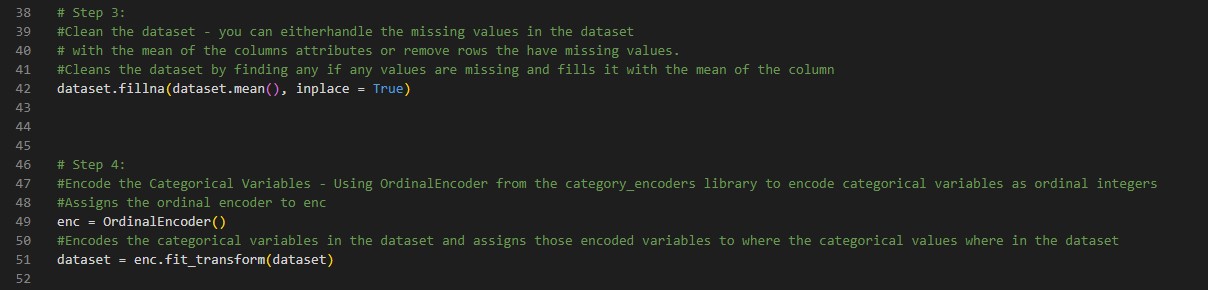
**Assignment Report**

A brief description of your work here acknowledging your collaboration with your class fellow (or a friend from other CSE 5120 sections), and the capacity at which he/she collaborated with you, followed by the algorithms you implemented.

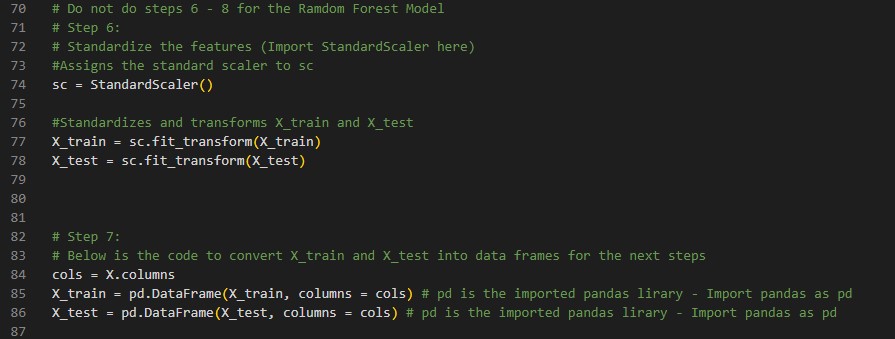
1. **DACs\_classification.py for Disadvantaged Communities dataset**

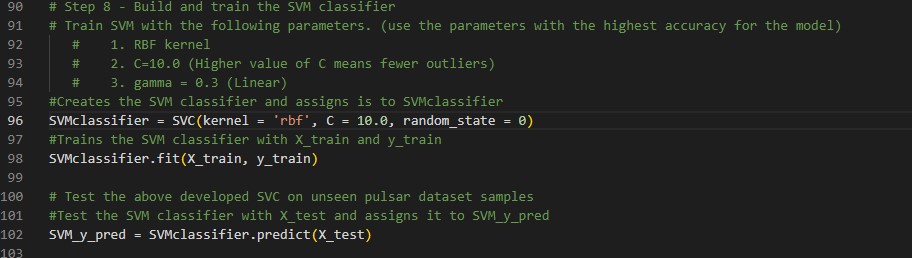
Your explanation of the dataset, your code solution, and any documentation with screenshots of your code Evaluation (results from **DACs\_classification**.py)

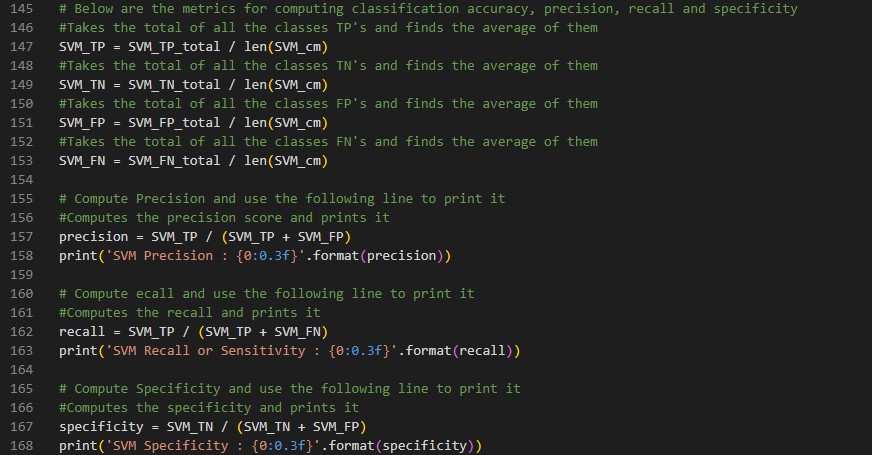
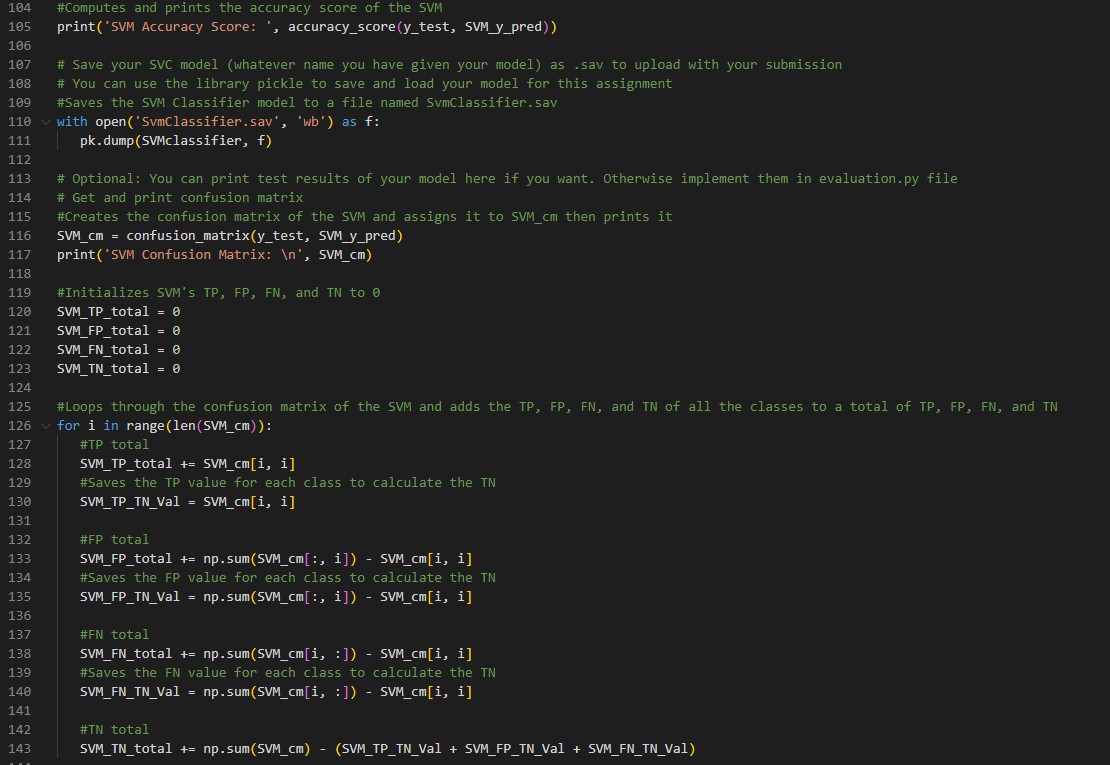
The dataset we are working with is information regarding the Disadvantaged Communities of California that shows communities in California that are suffering from some forms of challenges and disadvantages. The goal of the dataset is to highlight these disadvantages in these communities so that improvements can be made in these areas through things such as resources. With this dataset we can load it into our file and build classifier models of both SVM (Support Vector Machine) and RF (Random Forest) to predict and analyze the data in the dataset. The first step I did to accomplish this in the DACs\_classification.py was to first import any libraries I would need to create the classifiers and to print any important information. Then I loaded the dataset and printed out some of the key information of our dataset such as the dimensions of our data, the features, and the values of our target variable for this dataset which is CES 4.0 Percentile Range.

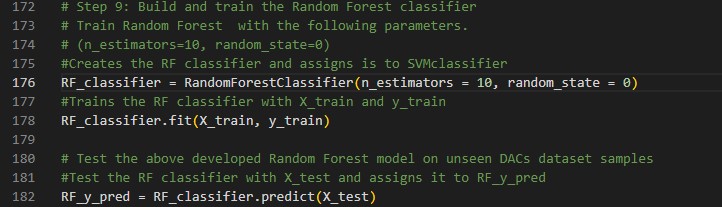
Once I had done that, I needed to prepare the dataset so that I could train test split it. To prepare the dataset I needed to first clean it so that our classifier could take our data to build a model. I did this by finding any missing values in our data and replacing those missing sections with the mean value that was seen in that column. This is crucial step as the train test split won’t work if sees that there are values that missing in our dataset, this could also create an inaccurate dataset due to certain values missing from our data. Another step I needed to take in the preparation of the dataset was to take the any categorical variables in our dataset and encode them. This involved columns such as California County, Approximate Location, CES 4.0 Percentile Range, and DAC category. The reason for this is that train test split won’t work unless it’s given integer values and these four columns in the dataset contain non-integers.

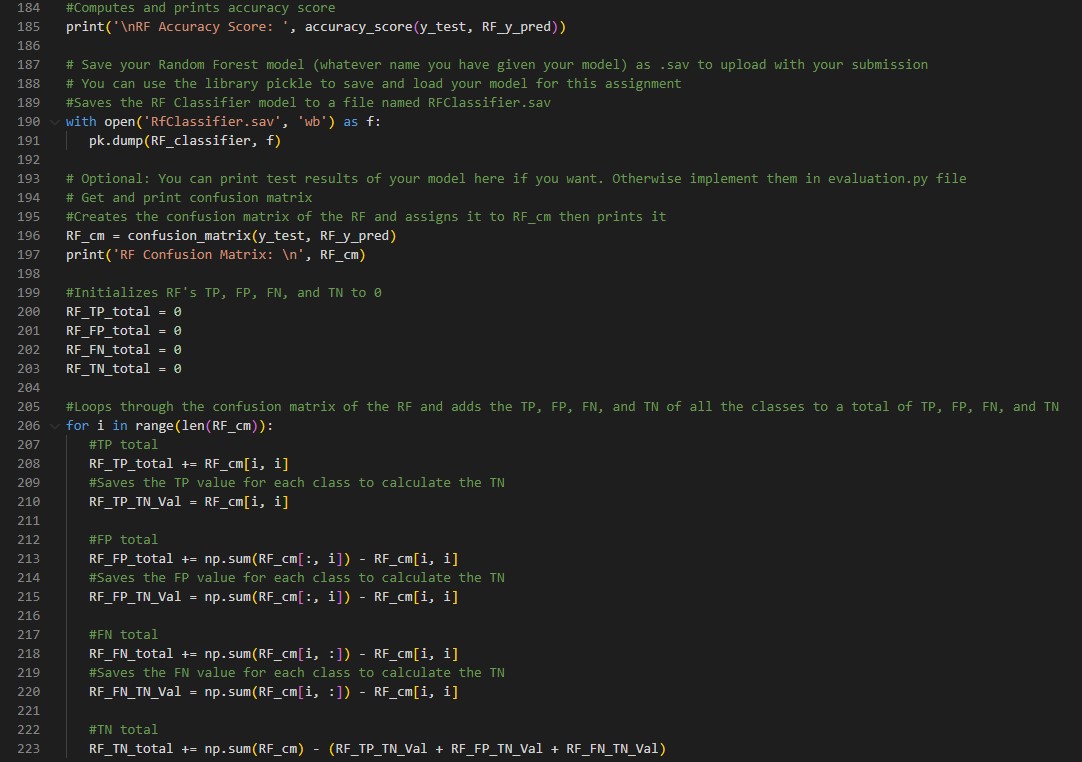
After this the dataset was ready to be train test split so the first step, I took was to split the data into predictor variables and the target variable. For our case I set X (predictor variables) to all columns except for CES 4.0 Percentile Range and then set y (target variable) to CES 4.0 Percentile Range column. Once I had split the data, I trained and tested data with 80% and 20% and stratified to a get a balance train test split to then get my X\_train, X\_test, y\_train, and y\_test for the next steps of training and testing it on the classifier.

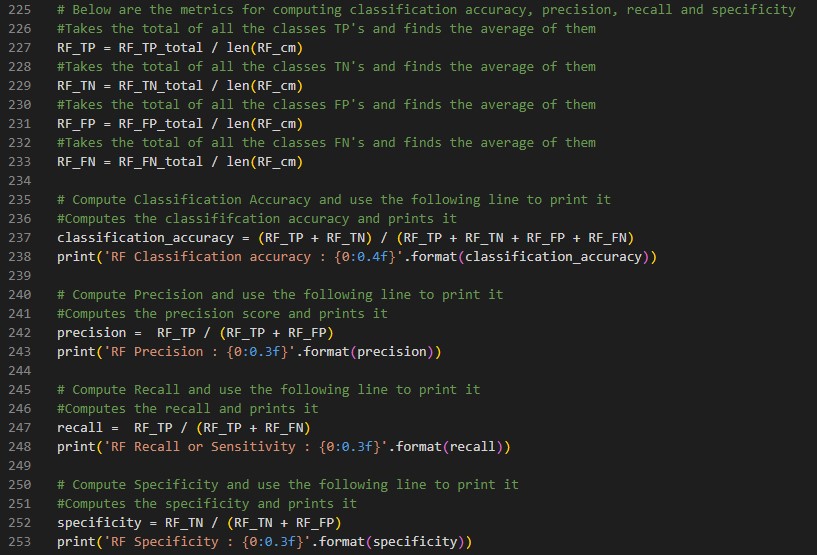
Before creating the classifiers, training them, and testing them I had to standardize my new X\_train and X\_test sets so that my classifiers models could be more accurate and not have variables that were not standard throughout the whole set.

After these steps we are finally able to create our SVM and RF models to then be able to train and test them. The first step here is to create our models and for SVM we needed to use SVC with the parameters of kernel = 'rbf', C = 10.0, random\_state = 0 to get a model accuracy that was at 85%. Once that model was created, I was then able to train the model with set created from the train test split called X\_train and y\_train, and then I was also able to test the model with X\_test.

With our SVM classifier created and trained I then saved the model to a .sav file so that it can be used by others and future cases relating to this data. After creating, training and testing the classifier I was also then able to print out valuable information on it such as the accuracy score, confusion matrix, precision, recall, and specificity which I did by calculating the TP, TN, FP, and FN values of the confusion matrix.

The same thing was then done with the RF model by first creating the model with the parameters n\_estimators = 10, random\_state = 0 to achieve an accuracy score of the model above 90%.

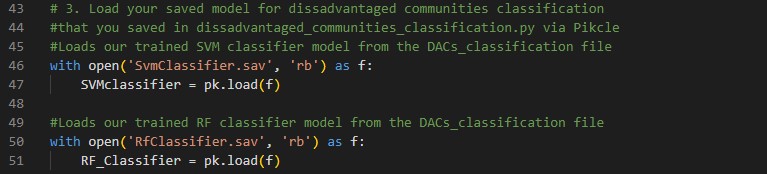
I then output and computed the same values that I did for the SVM model for the RF model and included the classification accuracy.

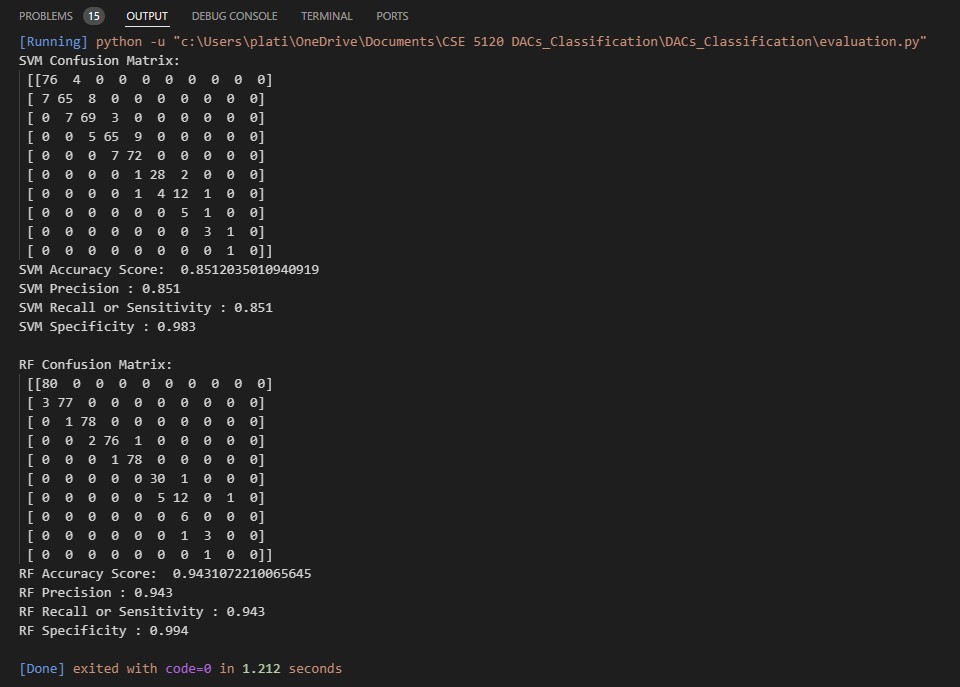


After implementing and running the code above we are given information on both models to see which of the two performs the best with the dataset given.

1. **Evaluation (evaluation.py) for your model performance evaluation - optional**

You can also provide the description of your code written in evaluation.py to load the saved model that can be readily used on a test dataset for the staff. This section is optional, and you can skip it

The code for both the DACs\_classification file and evaluation file are very similar with slight changes in them. In the evaluation file we still load the dataset, clean the dataset, and train test split the dataset the same way. However, the main difference is that since I have already created both classifier models in the DACs\_classification file and saved them, then all I need to do is load those files with the classifier models and skip having to build and train them in the evaluation file.After this the code continues to be the same except the output of this file is a little cleaner when related to the models as it only shows the valuable information that we need from the models and their compatibility with the dataset and not the extra stuff such as dimension when relating to the dataset. The output of the file is as follows:

This gives us some very valuable information about what model is best when relating to the dataset of Disadvantaged Communities. As we can see from the output above, the best model to use in this case would be the RF model as it provides us with an accuracy score of 94% while the SVM model only has a score of 85%. We can also see that the RF model has an easier time identifying and predicting the classes of the dataset than that of the SVM. This means that when accessing which communities need more aid with things such as resources, that the government or any other agency or company should apply and follow the results of a RF model as it is a better predictor of what communities will potentially be needing more aid.