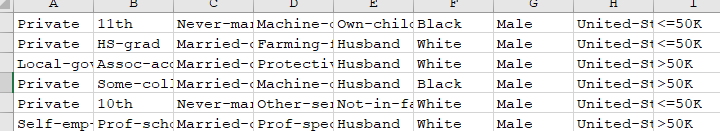
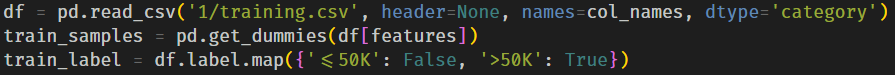
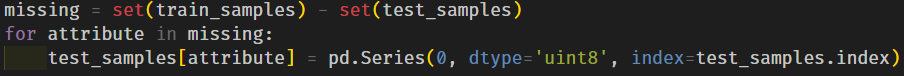
**Part 1**

Removal of the unknown values and continuous attributes was done in excel. The first five entries of testing.csv is shown here:



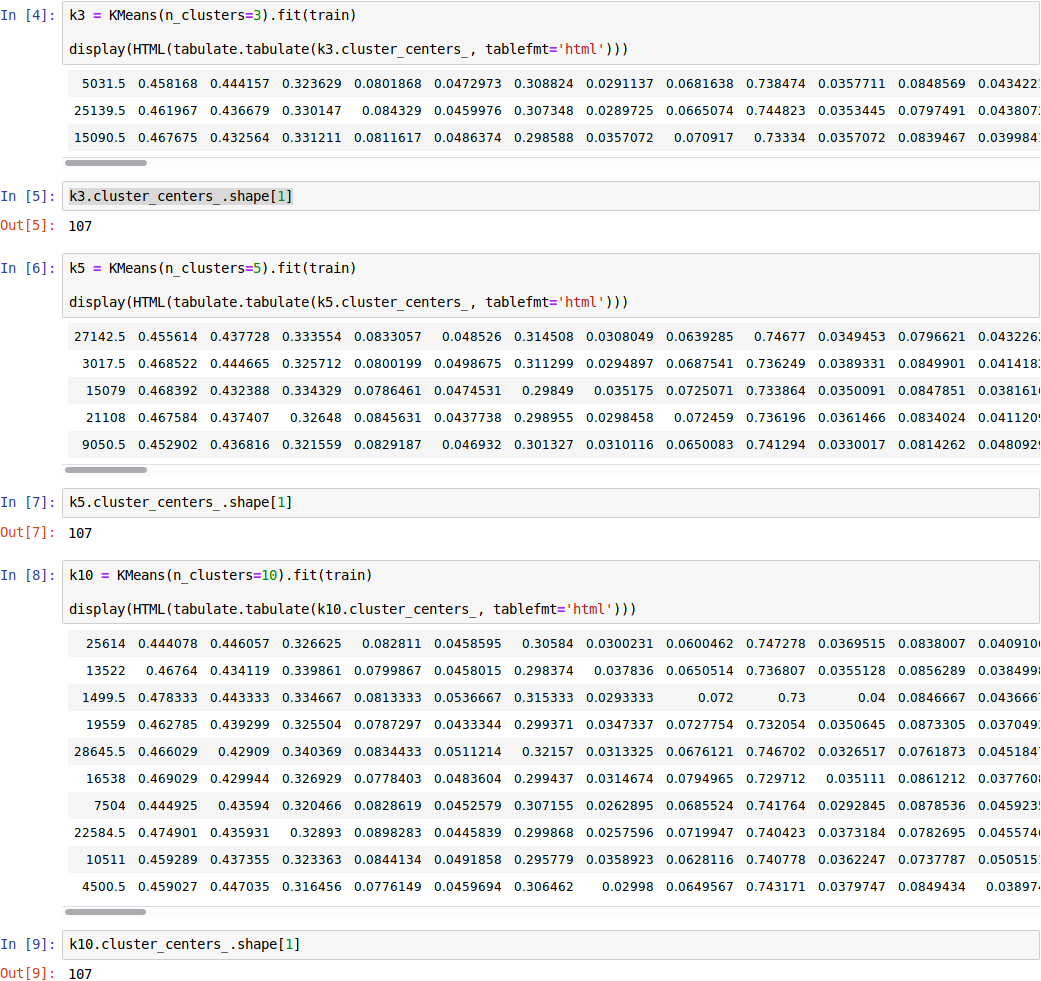
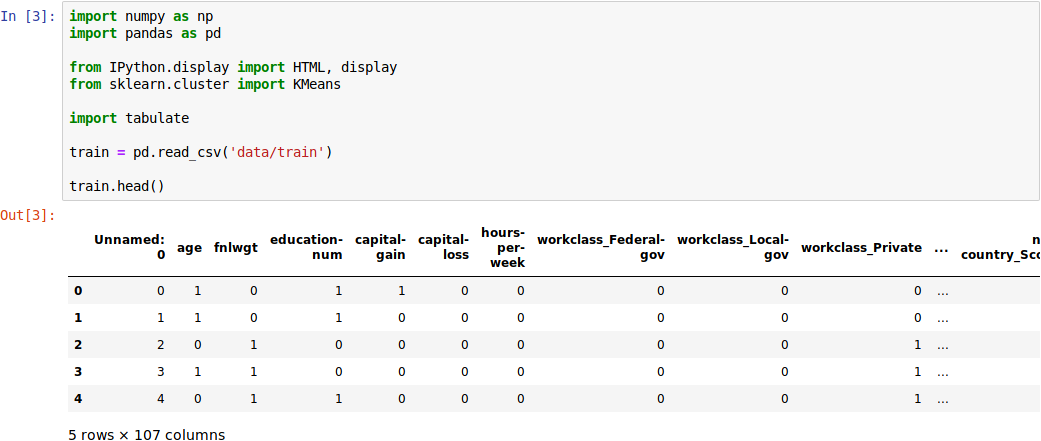
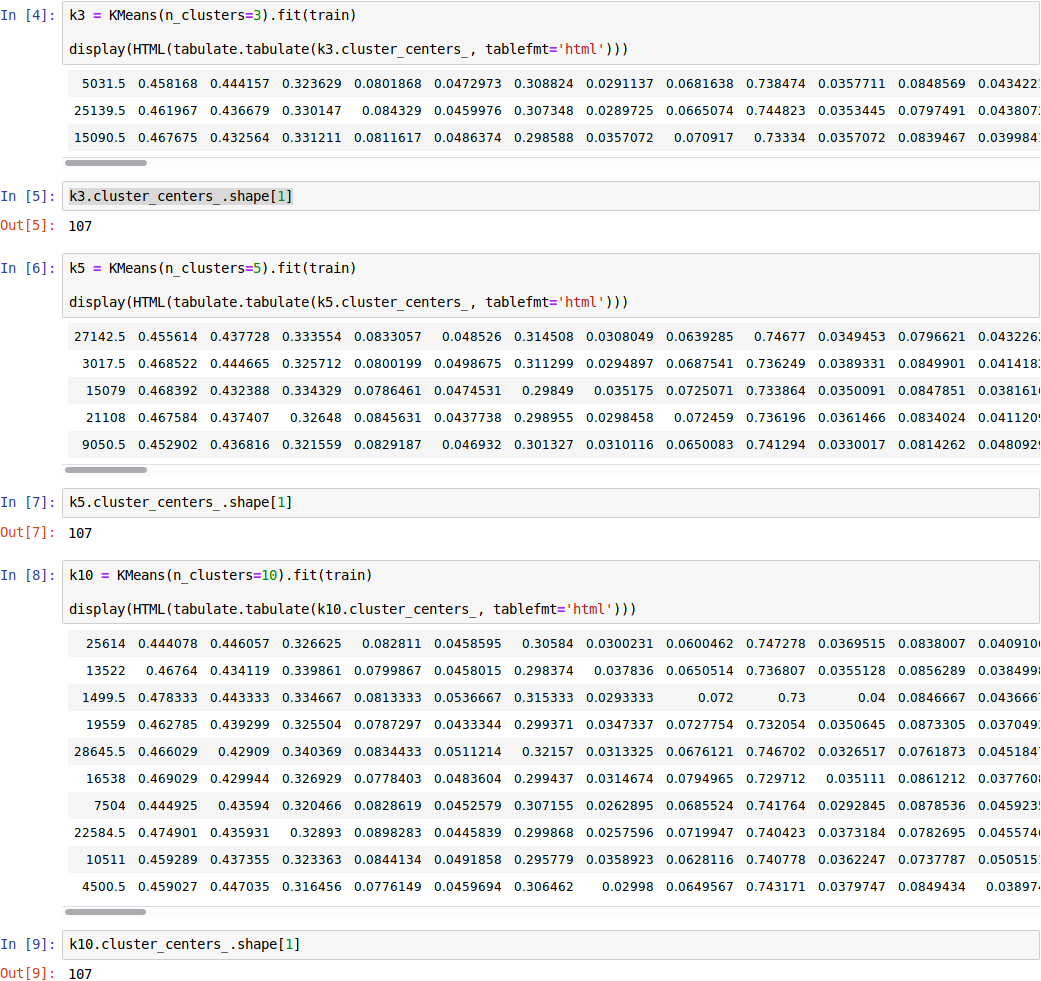
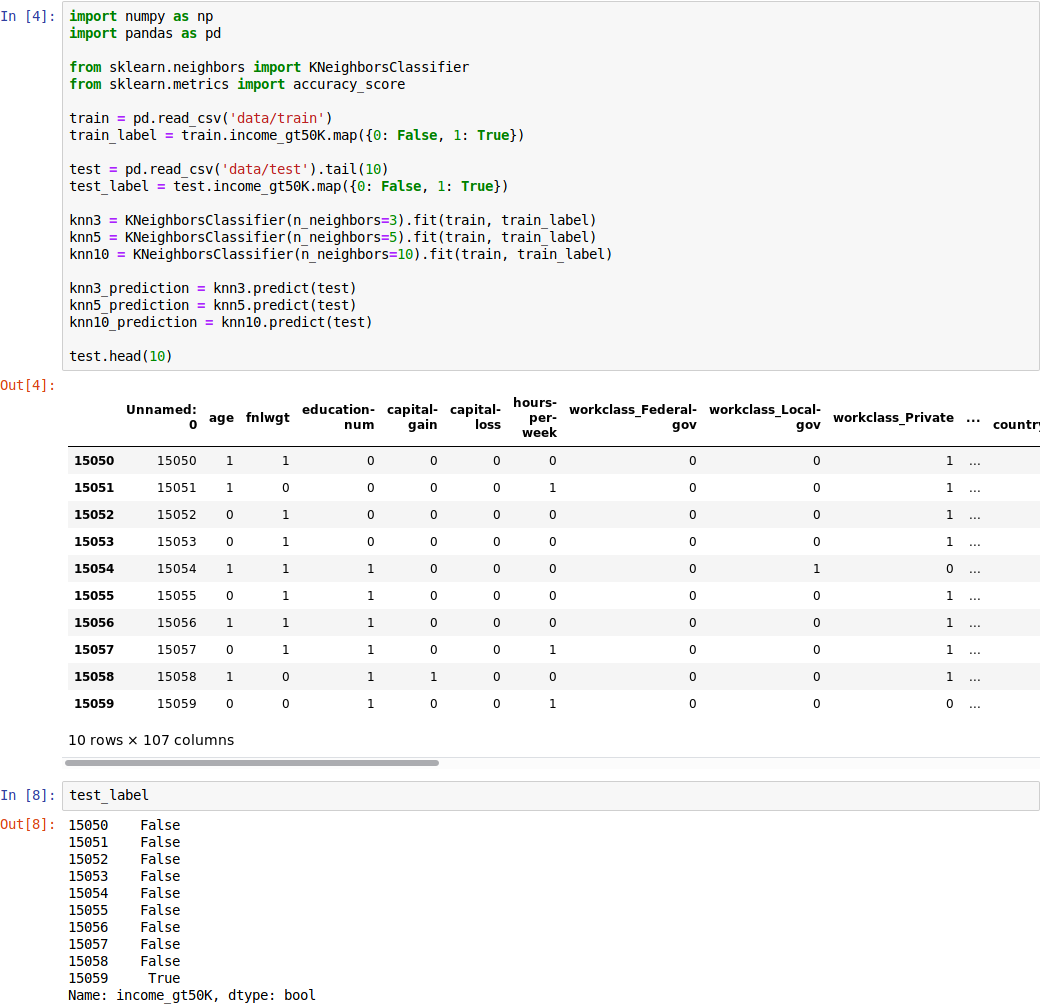
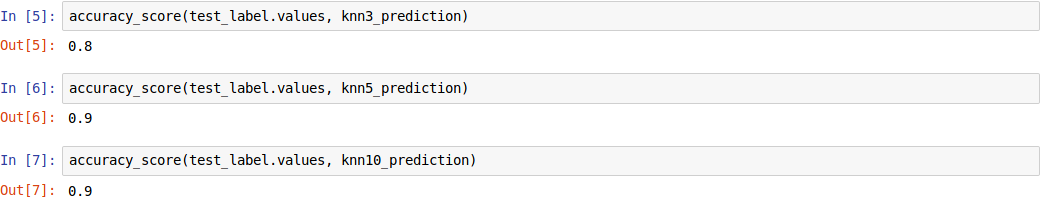
1. The decision tree was built in Python using pandas and scikit learn based on this guide: https://www.ritchieng.com/machine-learning-evaluate-classification-model/. There were two tricky parts in this step:
   1. Scikit learn doesn’t accept categorical attributes, so the pandas function get\_dummies was used as follows: 
   2. The test labels were missing an attributes value that existed on the training labels. SO, the missing values were found using set difference and then populated with 0 for the values. A for loop was used to keep the code more general. 
2. Completing this part was simple after completing part A. It used all the same data frames and labels, but it constructed a Gaussian Naïve-Bayes classifier instead. Then, printing the metrics was only 2 additional lines of code. Both of these steps resulted in the the following metrics:

|  |  |  |
| --- | --- | --- |
|  | **Tree** | **Gaussian Naïve-Bayes** |
| Accuracy | 0.807 | 0.748 |
| Recall / TP Rate | 0.528 | 0.035 |
| FP Rate | 0.102 | 0.980 |
| Precision | 0.627 | 0.365 |
| F1 Score | 0.573 | 0.064 |

It is notable that the Naïve-Bayes classifier apparently has a very low true positive rate and a very high false positive rate. After running it multiple times, this pattern continued.

**Part 2**

Both parts of this problem were done using scikit learn, numpy, and pandas. First, preprocessing was done using regex to remove records with unknown values. Then, pandas.get\_dummies was used to perform one-hot encoding on categorical attributes, and numpy.where was used to binarize continuous attributes. An extra problem that needed solving is that the adult.test dataset contained no rows with Holand-Netherlands as the country.

1. For this part, we needed to construct clusters over the train data using k-means with k-values of 3, 5, and 10. This involved first reading the train data through pandas. The first few rows of the data can be seen printed above as well as the number of attributes. Then, the only thing left was to run k-means using the provided k values. The results of k-means with k-values of 3 and 5 can be seen in the screenshot above. It would be very difficult to determine if all the numbers are correct, but you can see that the number of cluster centers matches the k-value and that the number of attributes matches up with the number of attributes shown in the initial train dataset. Finally, the results of k-means using the k-value of k = 10 can be seen in the below screenshot.
2. On this part, we were to use k nearest neighbors with the adult.data set as train to test the accuracy in predicting the last 10 records of the adult.test set. First, the relevant data sets and labels were created, and then the algorithm was run with k values of 3, 5, and 10. Next, the kNN classifiers with these varied k values were used to predict based on the test set. These results were as follows. After the data was loaded and the predictions were run, the last step was to print out the accuracy of the predictions. The accuracy results can be seen in the following screenshot.