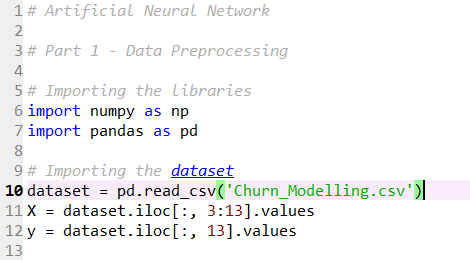
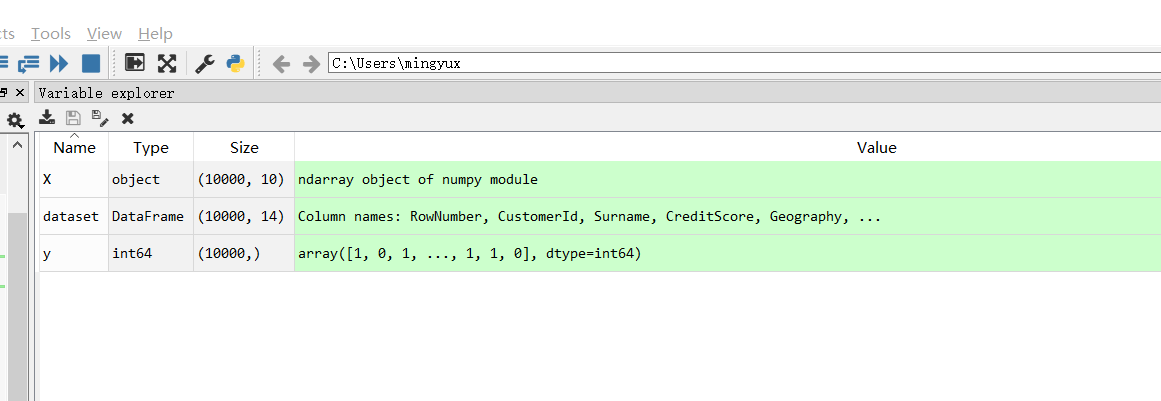
1. Rows 1-13:



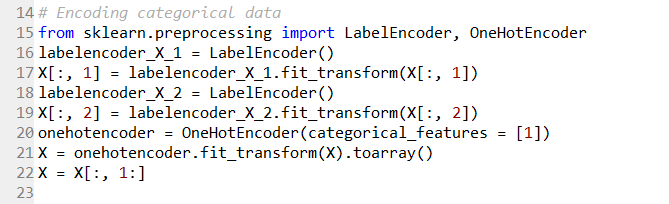
Import package ‘numpy’ and package ‘pandas’ and rename them as ‘np’,’pd’, read the csv file 'Churn\_Modelling.csv', and ‘dataset’ variable used to store this table as a DataFrame object.

Then use iloc() function to select values: all the rows, column from3 to 13 from dataset as X, select all the rows, column 13 in the dataset as Y. iloc() function select elements in dataset according to the index.



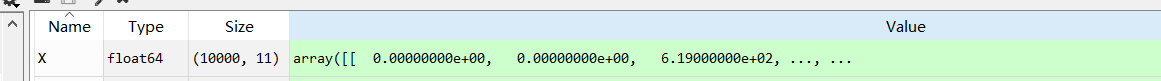
We can see from this screenshot when running the code step by step: X, Y, dataset has been assigned valued

1. Rows 14-23:

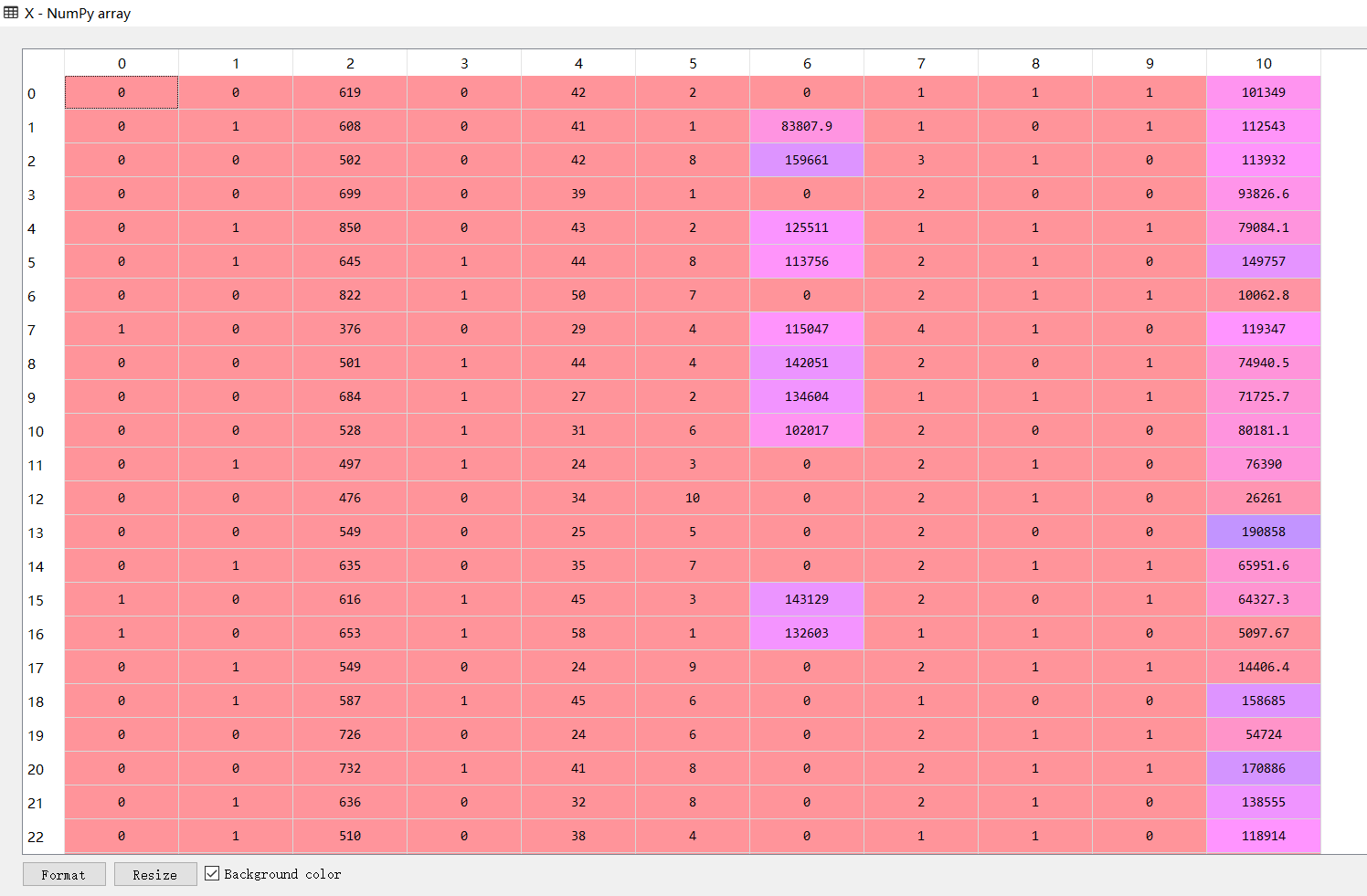


Import classes: *LabelEncoder*, *OneHotEncoder* in package ‘*sklearn.preprocessing*’, generate object ‘labelencoder\_X\_1’ from class ‘LabelEncoder’, use fin\_transform() function to fit label encoder X[:,1]（all rows and column 1）and return encoded labels, then X[:,1] get assigned with those encoded values. Same, labelencoder\_X\_2 is an object generated by LabelEncoder() function, it use fit\_transform() function to transform values in X[:,2] to encoded ones, and assign values to X[:,2]

Then use OneHotEncoder()function to encode categorical integer features using a one-hot aka one-of-K scheme, where array of categorical feature indices is [1], by using onehotencoder.fit\_transform(X), it fit OneHotEncoder to X, then transform X. Use toarray() function to form result to array. X will be all rows, 1 to end columns of X.

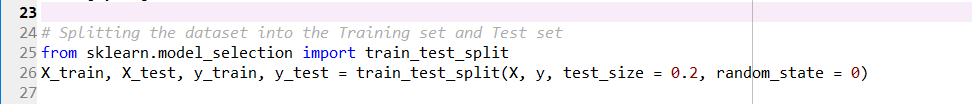


We can find that value of X has changed after rows 14-23, if we open the value of X, we can see:



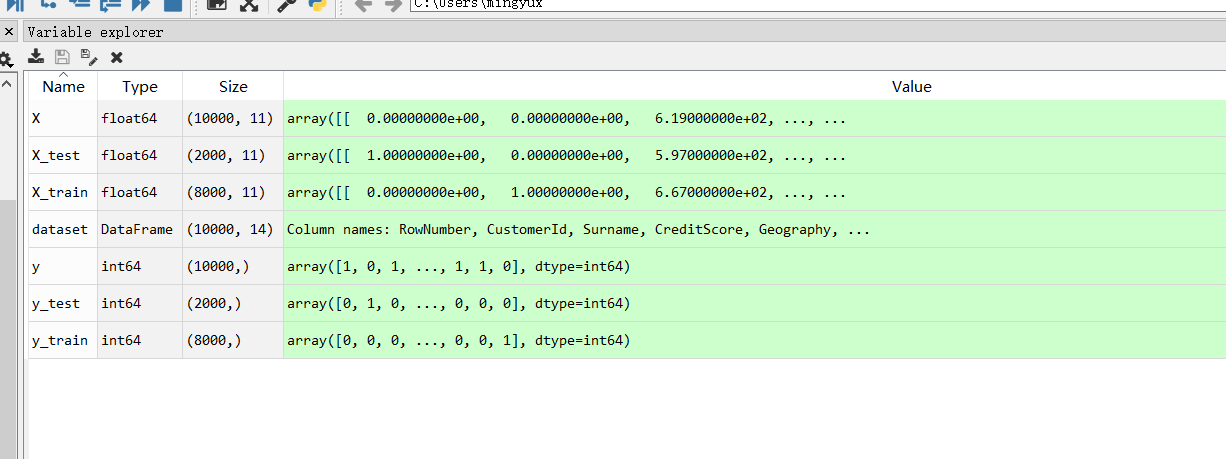
X has 11 columns and 10000 rows

1. Rows 24-27



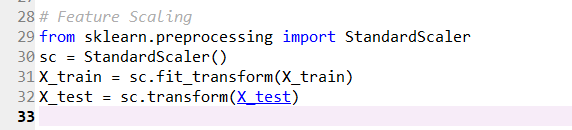
It imports train\_test\_split() function from package ‘sklearn.model\_selection’, and use train\_test\_split() function to split arrays or matrices into random train and test subsets, ‘random\_state’ is the seed used by the random number generator when the value is 0, ‘test\_size’ is representing the proportion of the dataset to include in the test split, when ‘test\_size’ = 0.2, the proportion for test set on all data is 0.2. ‘X\_train’ is train set part split in X set, ‘X\_test’ the test set part in X set, ‘y\_train’ is the train set part in y set, ‘y\_test’ is the test set pat of y set.

After executing step 26, we can get:



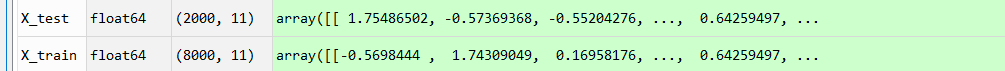
Which contains the train sets and test sets for both X and y. 8000 rows in X train set, 2000 rows in X test set; 8000 rows in y train set, 2000 rows in y test set.

1. Rows 28-33



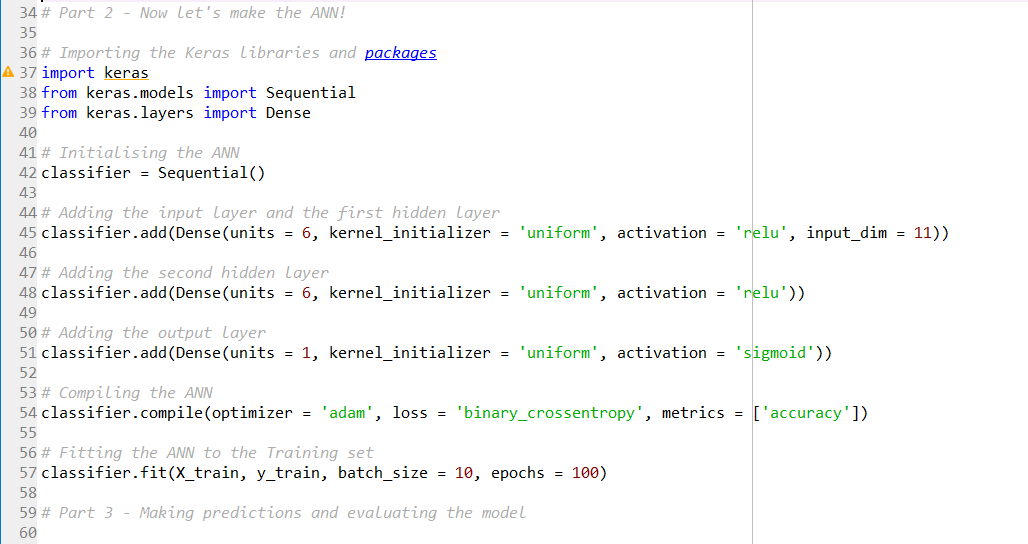
Import class ‘StandardScaler’ from package ‘sklearn.preprocessing’, ‘sc’ is the object generated by StandartScaler() function, standardize features by removing the mean and scaling to unit variance, fit X\_train into this fit\_transform()function, then transform values and assign them to X\_train, perform standardization by centering and scaling to X\_test set, then assign those values back to X\_test.

After applying 28-33 codes, we can get:



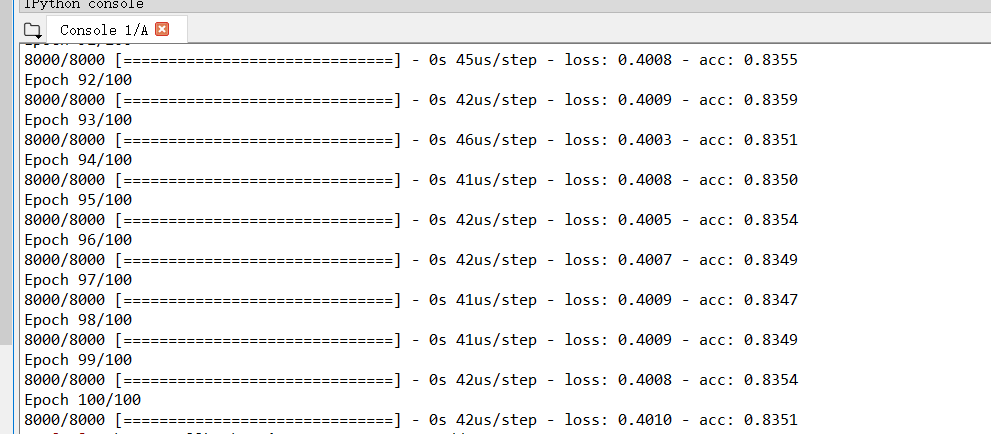
The values in X\_test and X\_train have been changed to a standardization form.

1. Rows 34-60

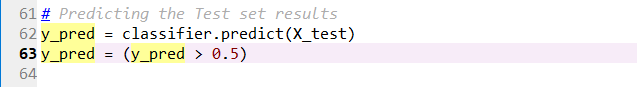


Import package ‘keras’. Import classes: Sequential and Dense from ‘keras’, use Sequential() function to initialize the ANN classifier, use add() function to add the input layer and the first hidden layer, second hidden layer, output layer. Then compiling the ANN classifier with compile() function, after that, fitting the ANN classifier to the training set by fit() function.

After applying rows 34-60, we can find the process running in console



1. Rows 61-64

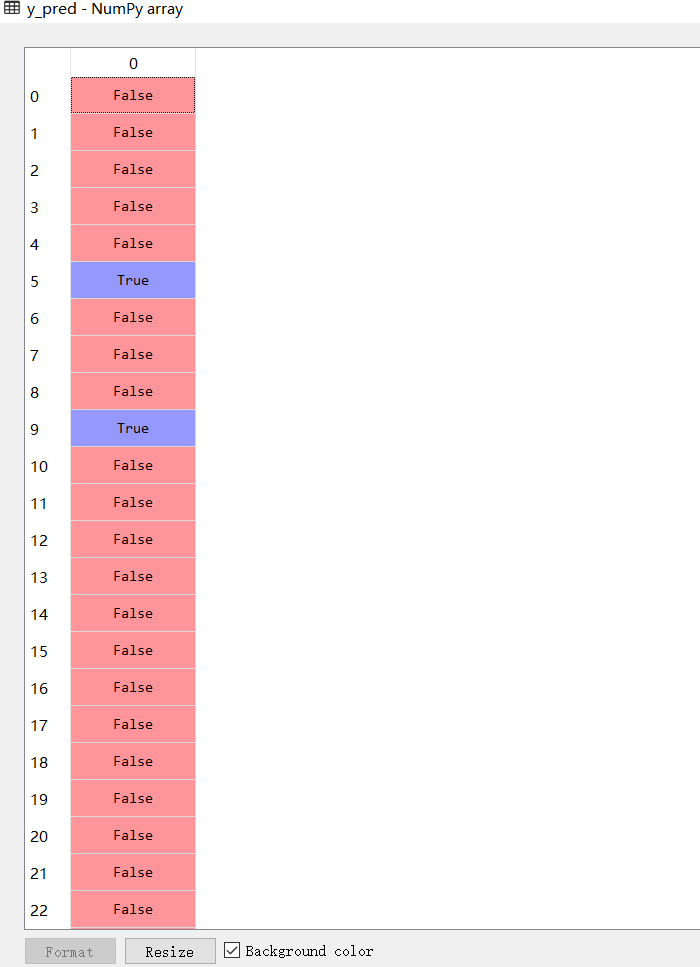


Use predict() function to predict response value of X\_test data set, and assign those predicted values to y\_pred, filter the predicted value, select those who are > 0.5 to be assigned to y\_pred.

After applying rows 61-64, we can find variable ‘y\_pred’:

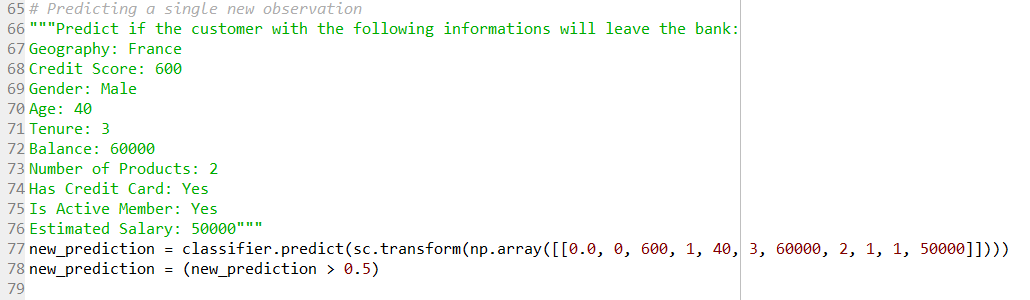


Let’s see the detail of this variable:



So there is some prediction get here.

1. Rows 65-79

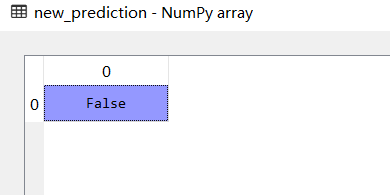


According to the information of this customer, use predict() function to predict if the customer will leave the bank. Then select the values which are > 0.5, and assign back to variable ‘new\_prediction‘.

After executing the rows 77-78, we can get:

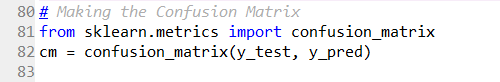


For the detail of this result:



So the prediction of this new customer is ‘False’

1. Rows 80-83

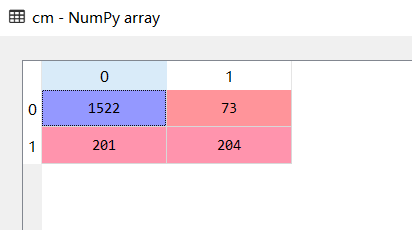


Import confusion\_matrix() function from sklearn.metrics package, use confuxion\_matrix() function to compute confusion matrix to evaluate the accuracy of a classification.

After executing the codes in row 81-82, we can get:



When we see the detail of this confusion matrix:



So the accuracy = (1522+204)/(1522+204+201+73) = 86.3%