**BITS F464 MACHINE LEARNING**

**Assignment-2**

Group details:

Dhairya Agrawal (2020A7PS0130H)

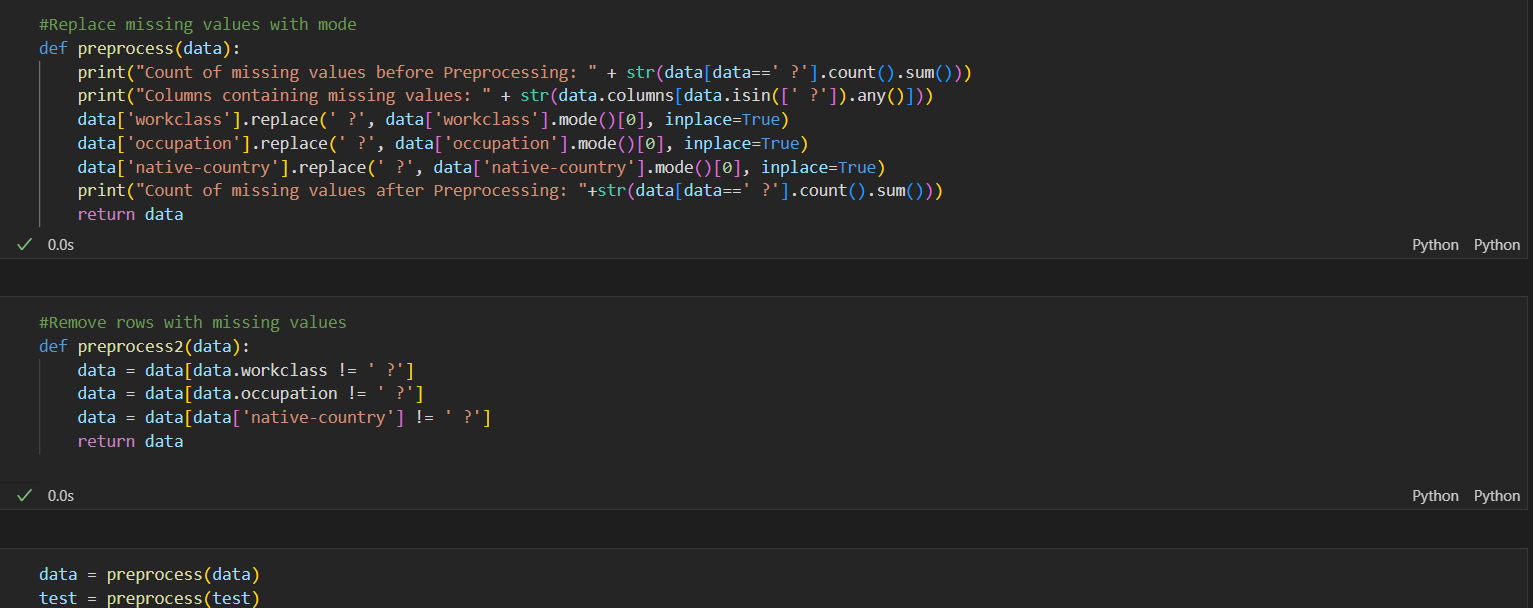
Pulkit Agrawal (2020A7PS2072H)

Mohit Agarwal (2020A7PS0189H)

**PART A-Naive Bayes Classifier to predict income**

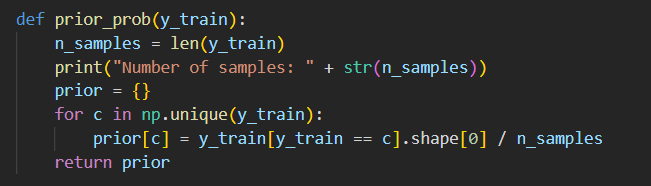
Task 1: Data Preprocessing

The given task required us to import the dataset and handle the missing values and split the data into training and testing data, however, since the original data was already divided into the testing and training datasets so we used the already split dataset for our implementation. On inspection, we found that the missing values were only categorical and were present in only the work class, occupation, and native-country features/columns. So we replaced the missing values in them with the mode (the most frequently occurring value).

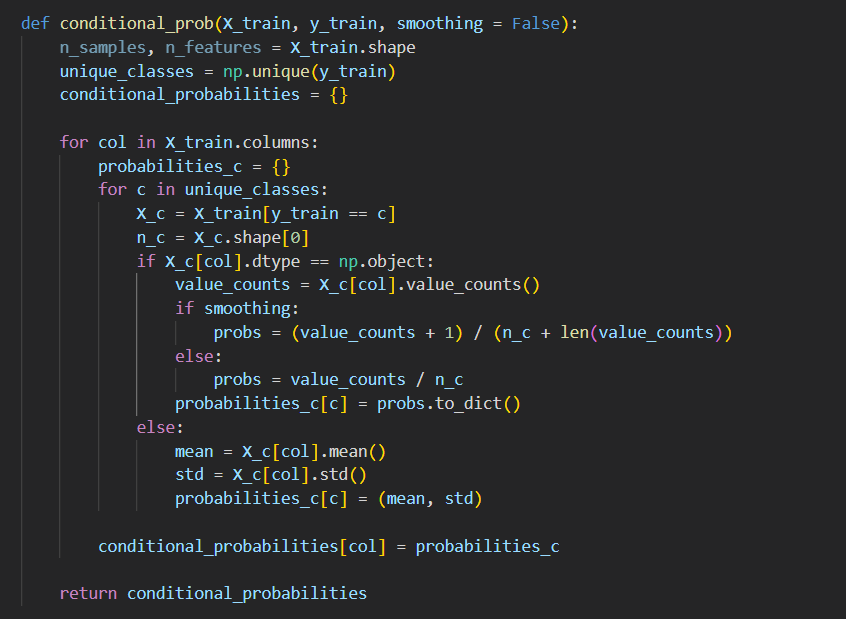


Task-2:Naive Bayes Classifier Implementation

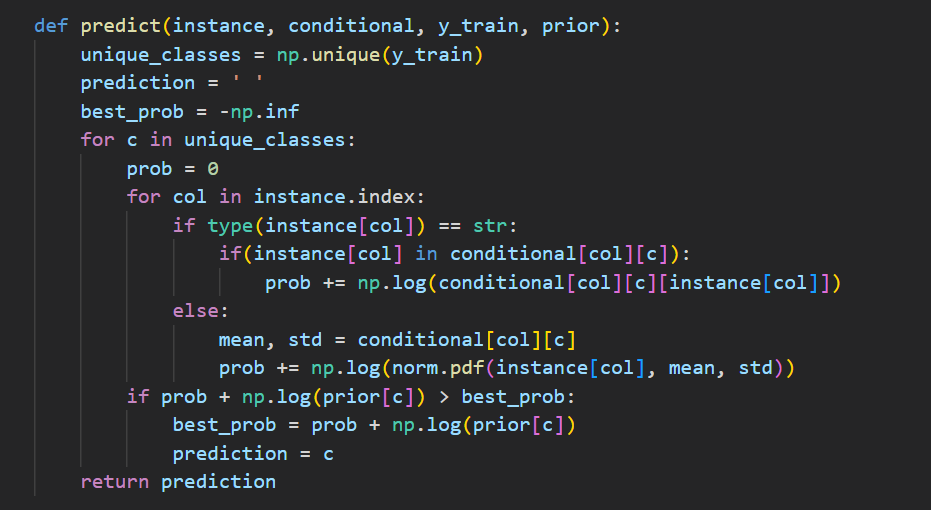
The first given sub-task required us to implement a function to calculate the prior probability of each class. Our dataset has only two classes, ‘ <=50K’ and ‘ >50K’. So for the respective classes, we calculated the number of occurrences of that class divided by the total number of examples.



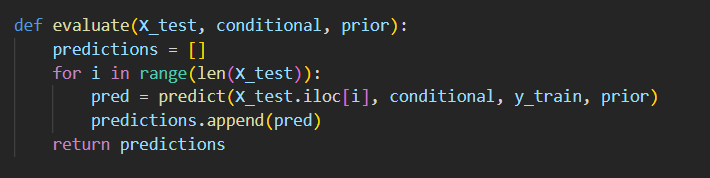
The given sub-task required us to implement a function to calculate the conditional probability of each feature in the training set. For this purpose, for each feature, we must calculate the conditional probability for each feature given a particular class. The dataset has two types of features: continuous features and categorical ones. For categorical features, we found the conditional probability for each unique feature's unique value. For continuous features, we found the mean and standard deviation of the values to create a normal distribution and finally predict a probability given a value.



The given sub-task required us to implement a function to predict the class of a given instance using the Naive Bayes algorithm. For this sub-task, we multiplied each of the values of the conditional probability of the instance for categorical. We found out the probability using the normal distribution for the continuous features of the instance.

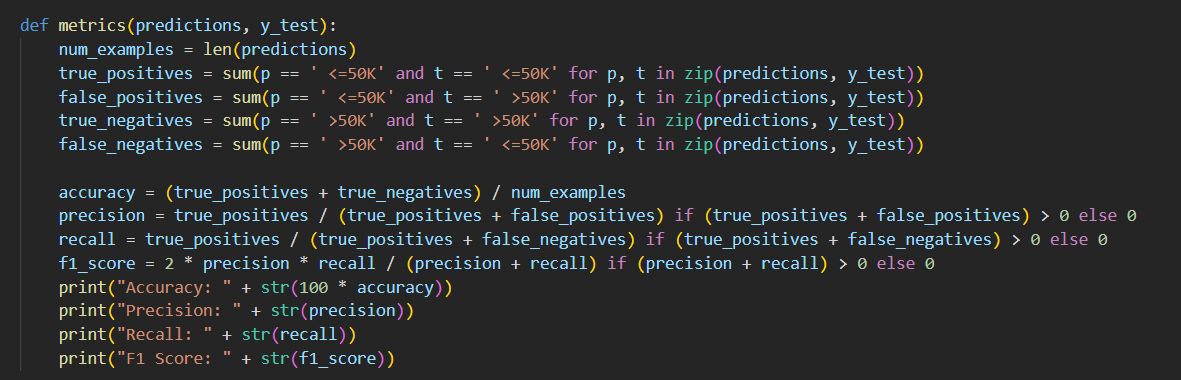


The given sub-task required us to Implement a function to calculate the accuracy of the Naive Bayes classifier on the testing set. In this sub-task, we called the prediction function for each testing example in the dataset. The predictions are stored in the predictions list and will be later used to calculate the accuracies and other performance metrics.



Task-3: Evaluation and Improvement

The given subtask required us to evaluate the performance of the Naive Bayes classifier using accuracy, precision, recall, and F1-score.



The given subtask required us to experiment with different smoothing techniques to improve the performance of our classifier. The smoothing technique we used was Laplace’s smoothing.



The given subtask required us to compare the performance of the Naive Bayes classifier with logistic regression and k-nearest neighbors. We used sklearn to train and find the performance metrics using logistic regression and k-nearest neighbors.

****

**Part B – Building a Basic Neural Network for Image Classification**

The goal is to build a basic neural network that can classify images of handwritten digits from the MNIST dataset.

We have built 12 distinct artificial neural network classifiers by varying various following parameters -

(i). A number of hidden layers – 2 or 3

(ii) Total number of neurons in the hidden layer is 100 or 150

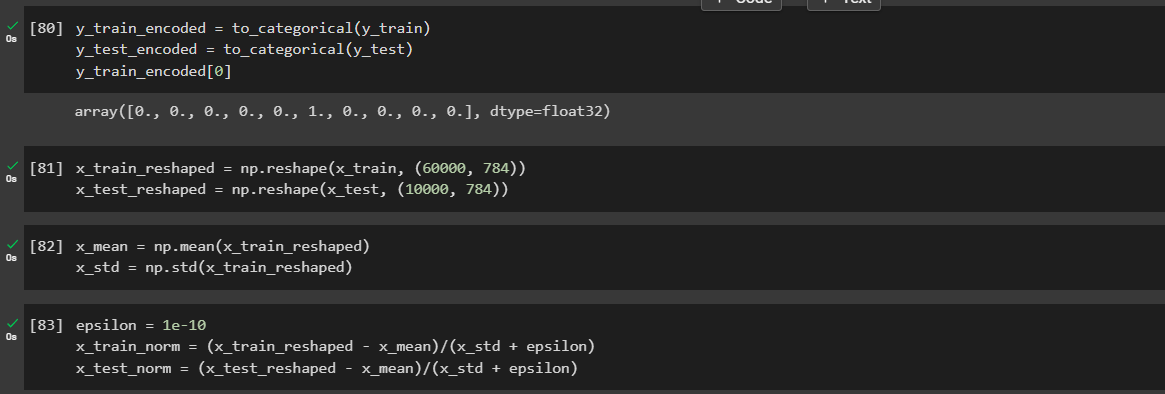
(iii) Activation function is from any of the following functions: tanh, sigmoid, ReLu

We ran the 12 different models on 2 different optimizers that is Adam and SGD.

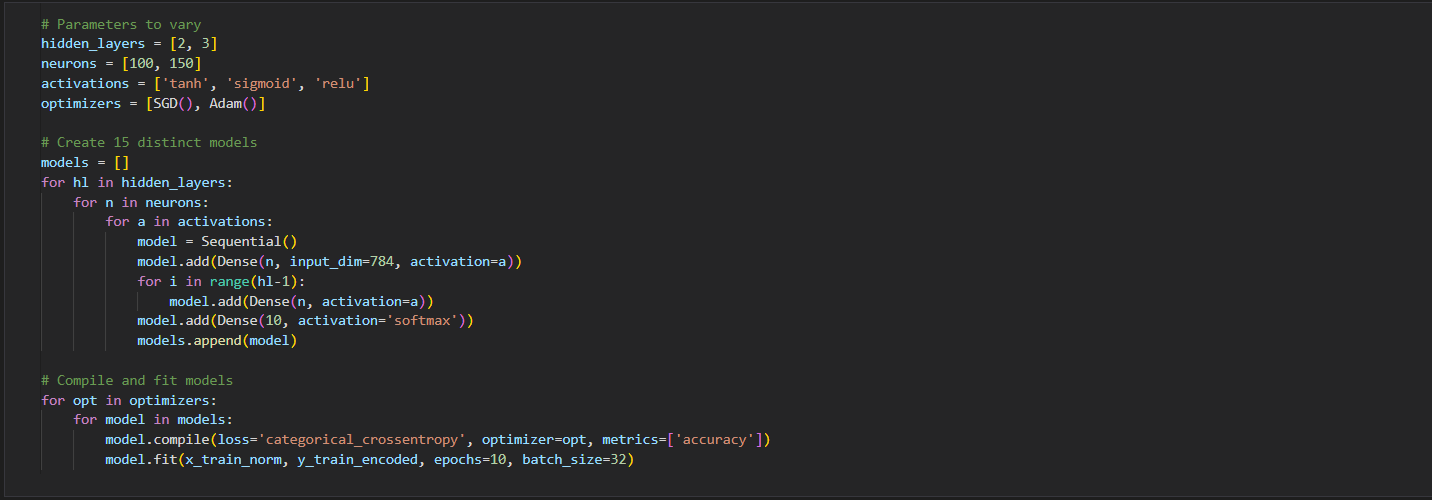
Code Snippets for the model:

**Training and Testing:**

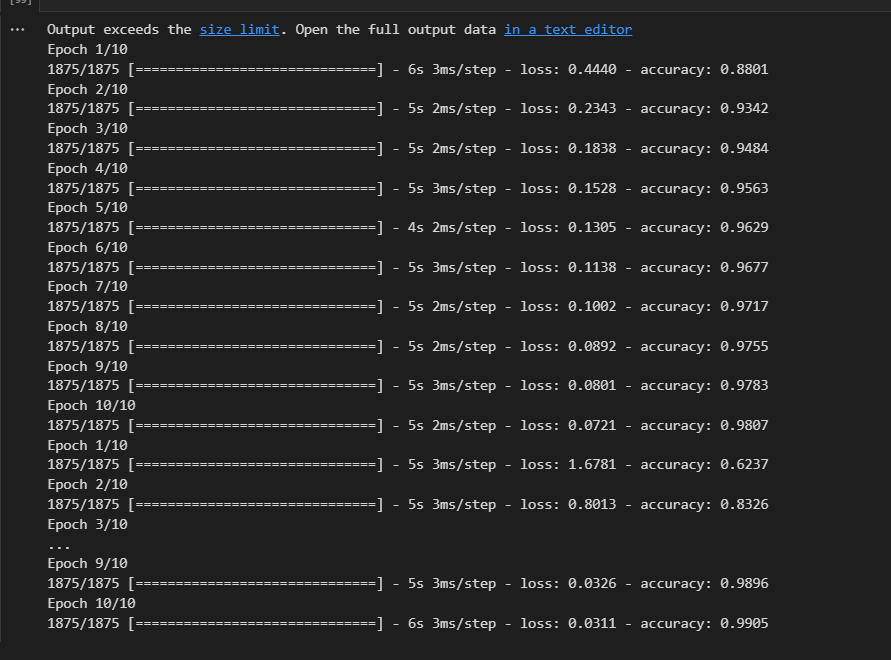
We have trained the neural network on the MNIST dataset and used Adam and SGD as an optimization algorithm. The different accuracy and confusion matrices of the best model are shown below.



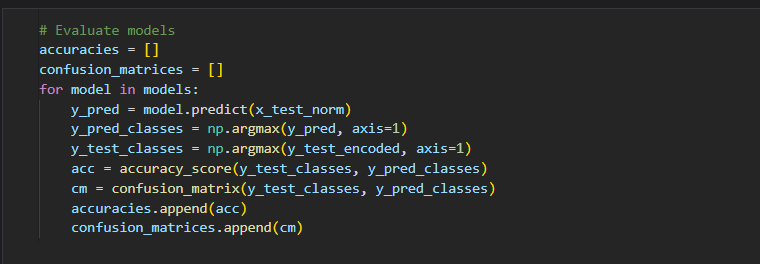
Code for 12 distinct models

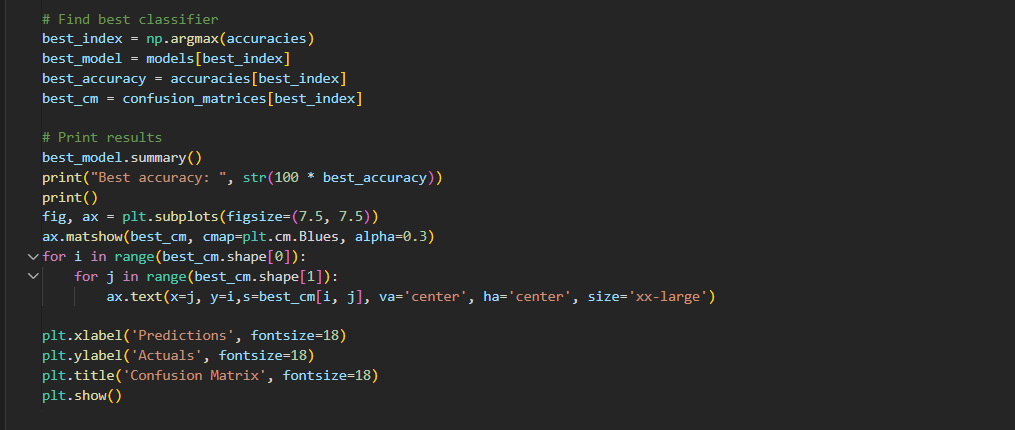


Output accuracies after running the above code.

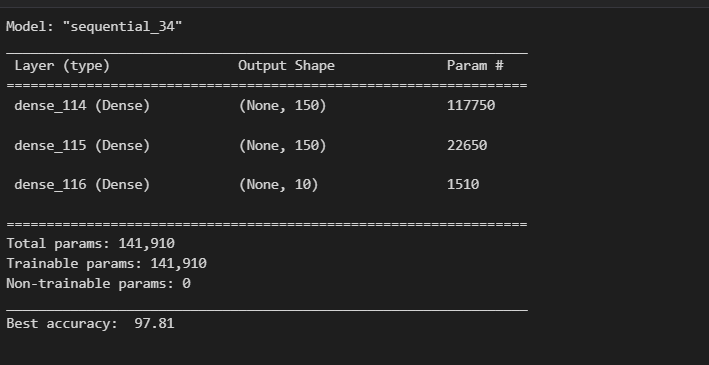


Comparing the distinct models generated by their accuracies

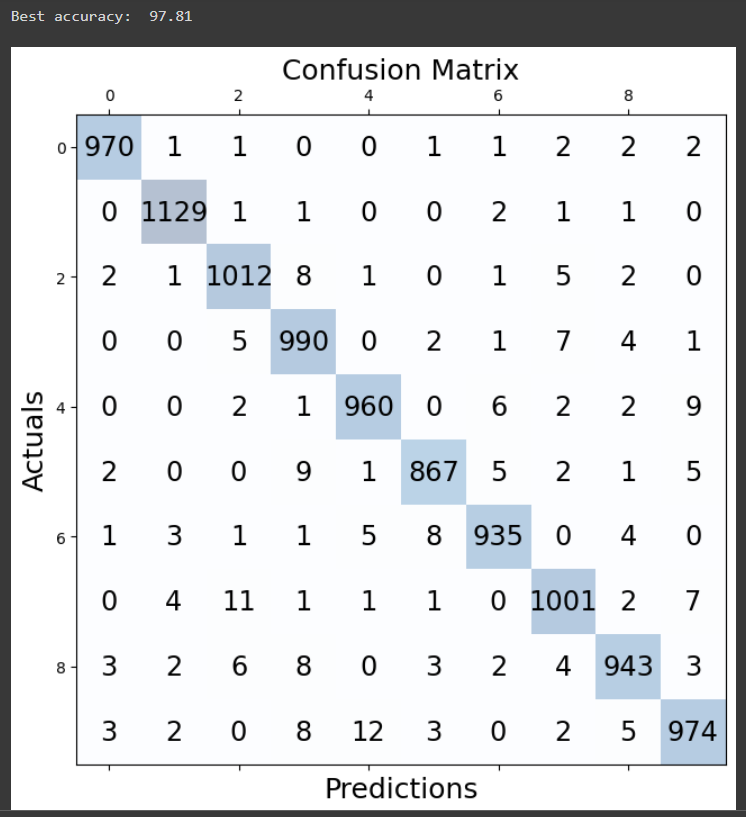




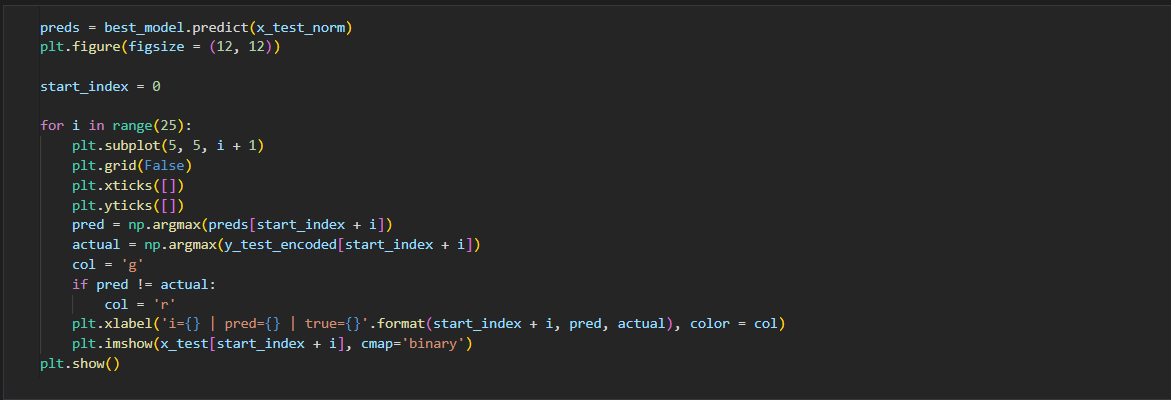
Results after running the above code snippet:-



Best accuracy to be found out to be 97.81% .



Running the test data set to predict the y\_test with the best model



Output:

