**Name: Manasi Meshram**

**Batch: ML python batch 1**

**Major Project**

**Problem Statement :**

Design a project from the MNIST dataset to identify digit classification using the SVM algorithm.

**Explaination:**

For this problem, we use the MNIST data which is a large database of handwritten digits. The 'pixel values' of each digit (image) comprise the features, and the actual number between 0-9 is the label.

Since each image is of 28 x 28 pixels, and each pixel forms a feature, there are 784 features. MNIST digit recognition is a well-studied problem in the ML community, and people have trained numerous models (Neural Networks, SVMs, boosted trees etc.) achieving error rates as low as 0.23% (i.e. accuracy = 99.77%, with a convolutional neural network).

Before the popularity of neural networks, though, models such as SVMs and boosted trees were the state-of-the-art in such problems.

We'll first explore the dataset a bit, prepare it (scale etc.) and then experiment with linear and non-linear SVMs with various hyperparameters.

We'll divide the analysis into the following parts:

**Data understanding and cleaning Data preparation for model building Building an SVM model - hyperparameter tuning, model evaluation etc.**

**IDE:** Google Colab

**Packages used:**

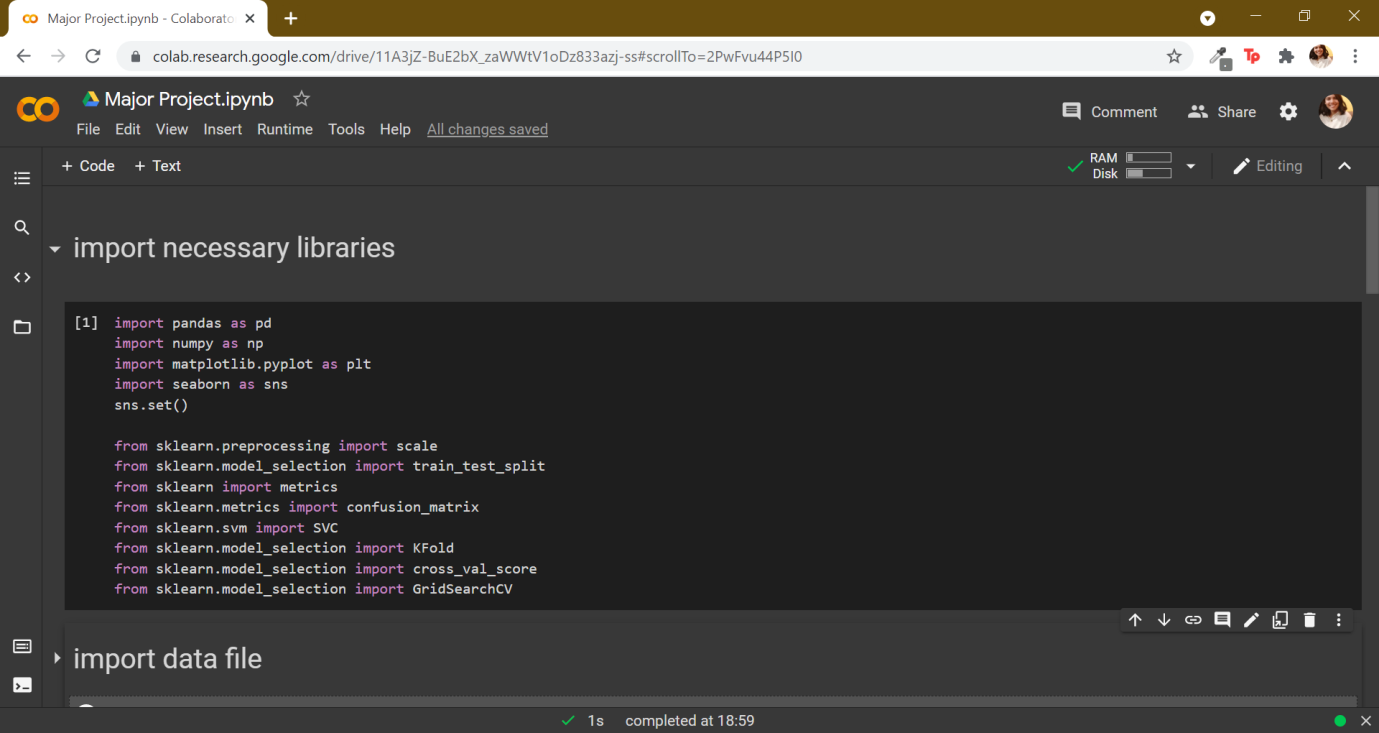
1. Pandas
2. Numpy
3. Scikit-learn
4. Seaborn

**Libraries used:**

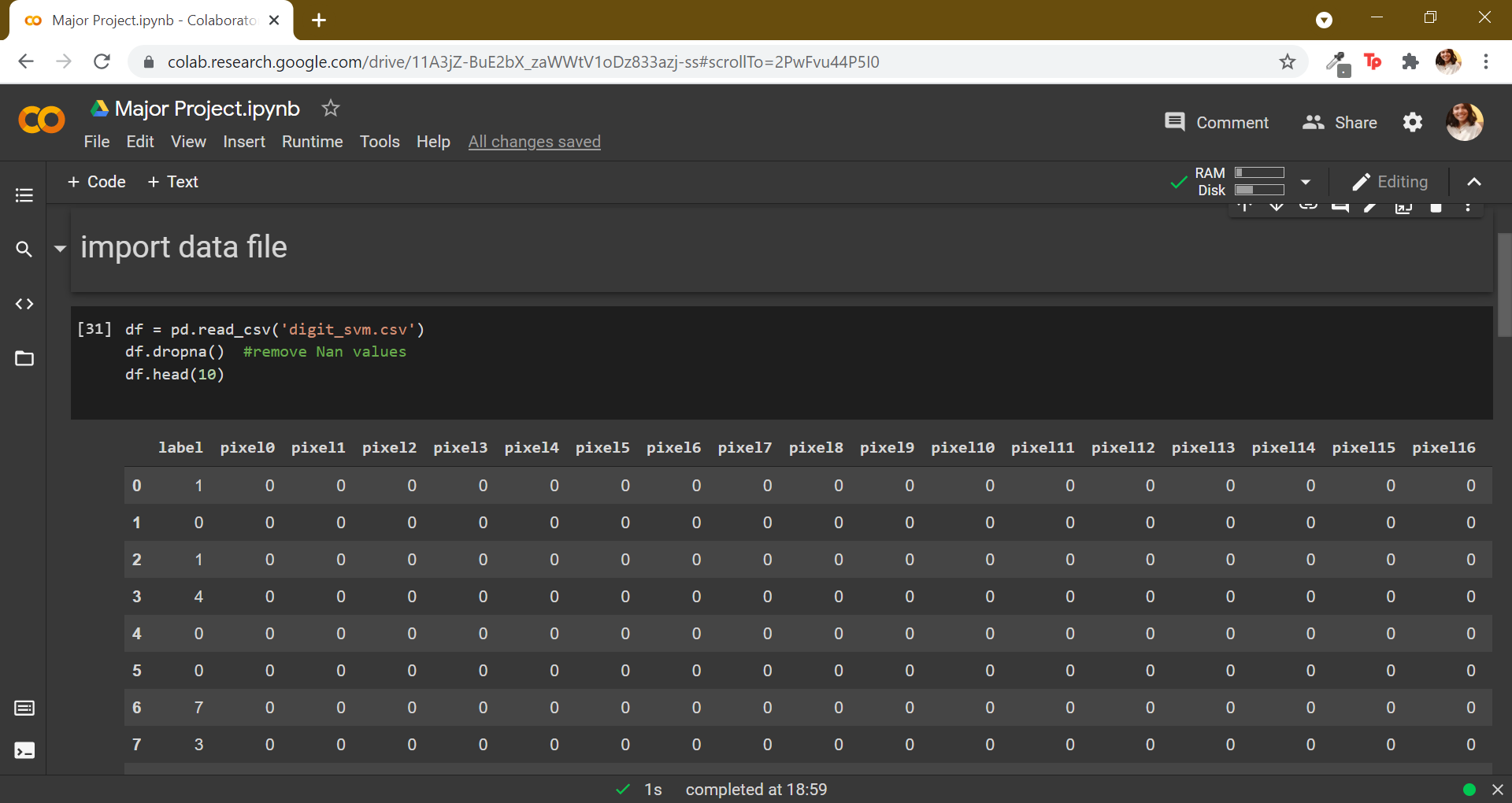
1. From sklearn.model\_selection: train\_test\_split
2. From sklearn.preprocessing: Scale
3. From sklearn: Metrics
4. From sklearn.svm: SVC
5. From sklearn.model\_selection: KFold
6. From sklearn.model\_selection: cross\_val\_score
7. From sklearn.model\_selection: GridSearchCV
8. From sklearn.model\_selection: confusion\_matrix

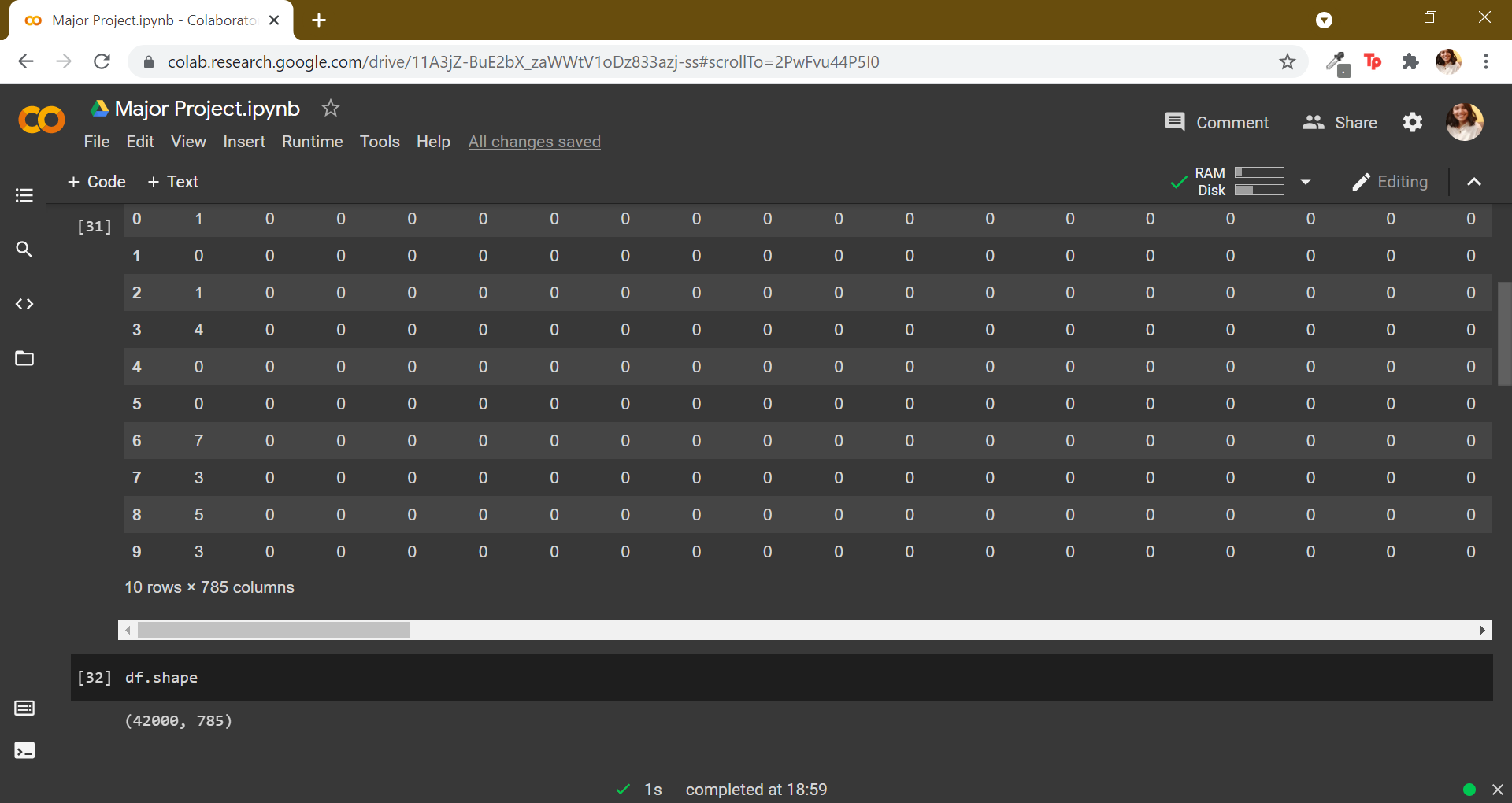
**Solution:**

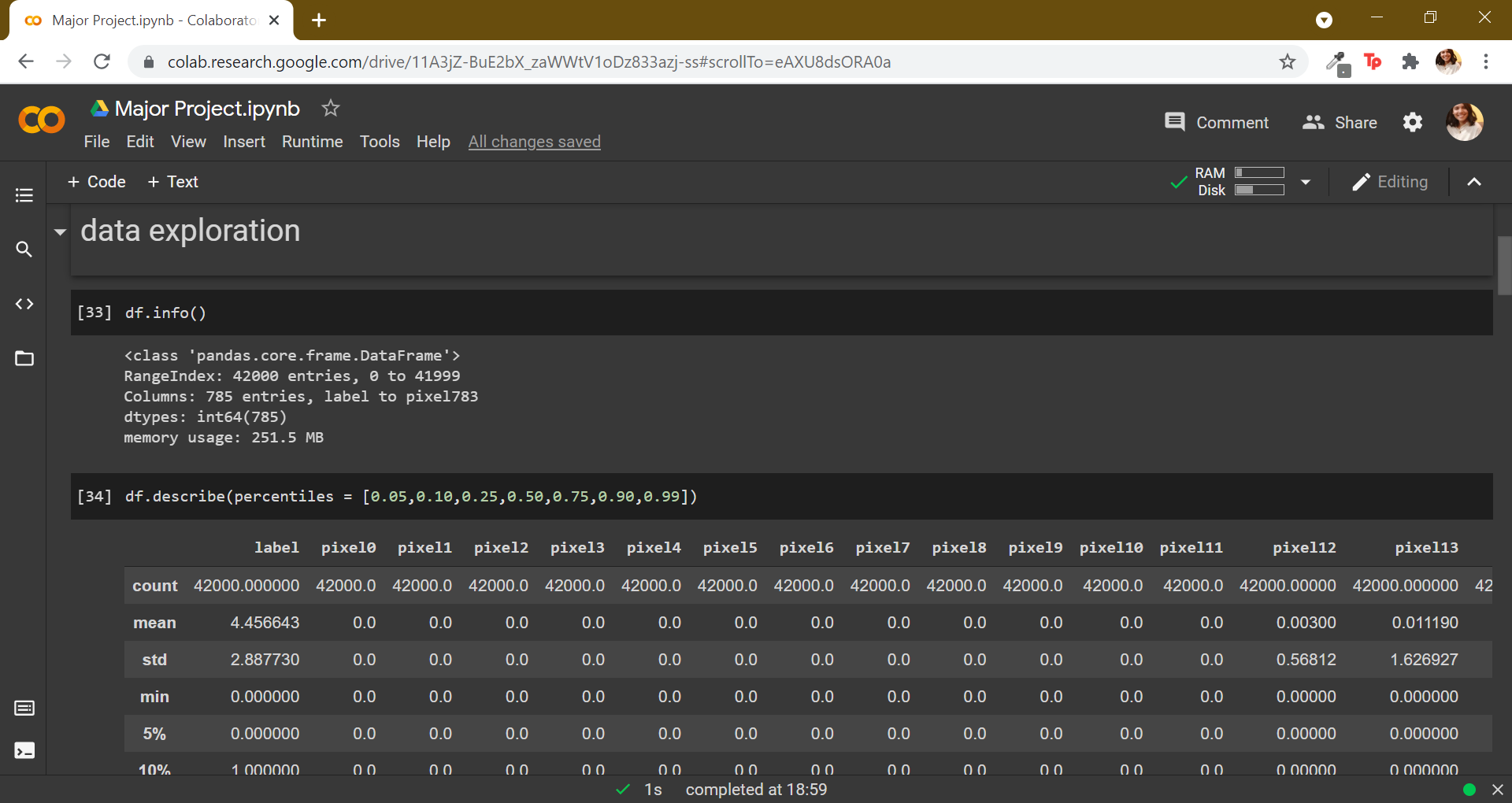
1. We imported basic libraries

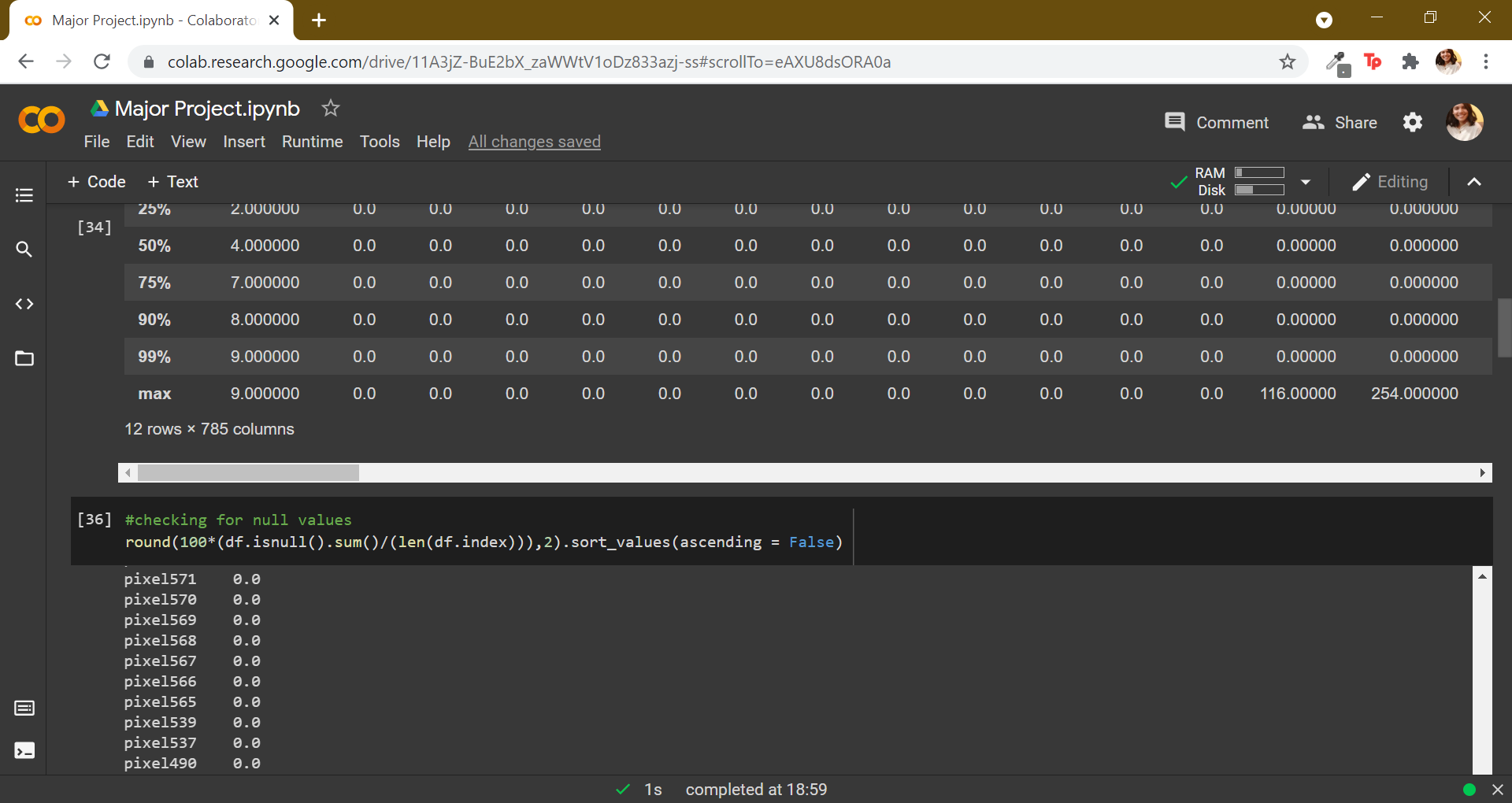


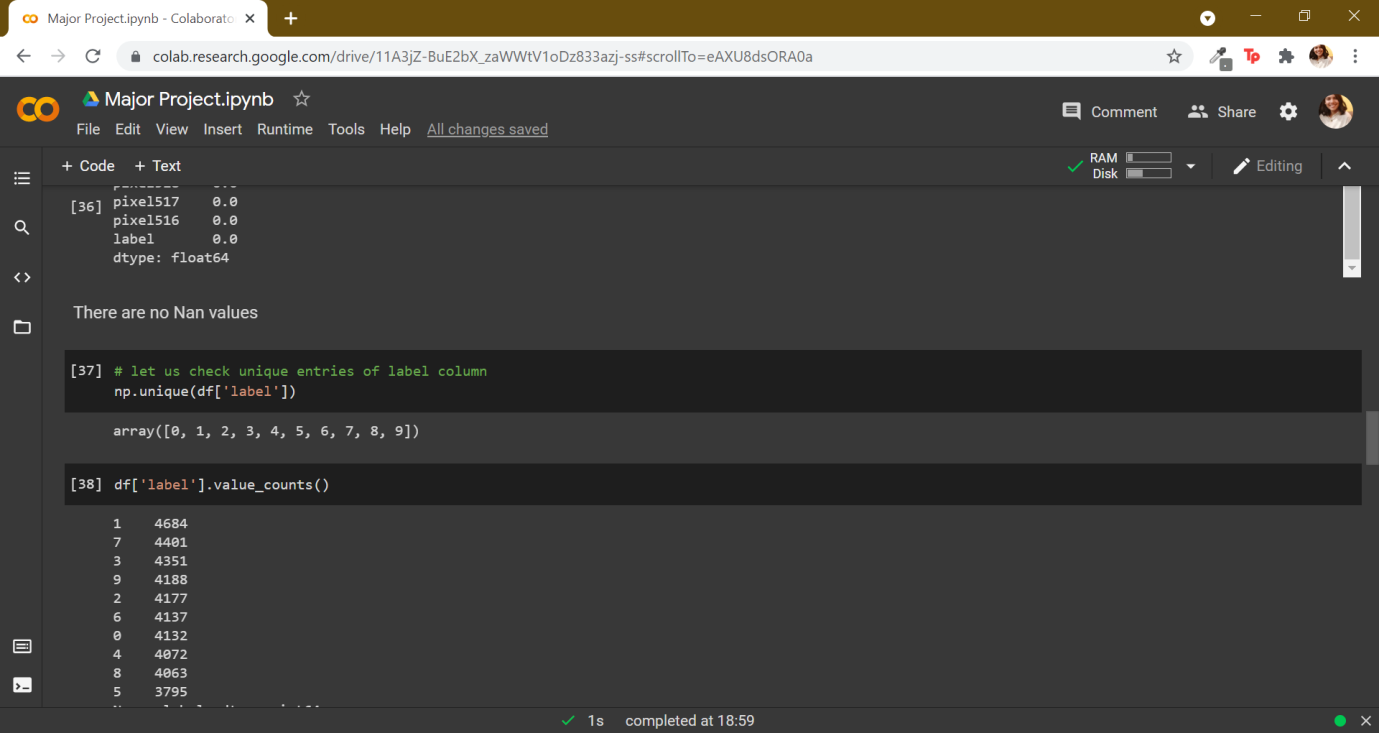
1. Using pandas we opened our dataset and initialised to variable df. Removing the Nan values we printed first 10 values of dataset. We also check the size of the dataframe.

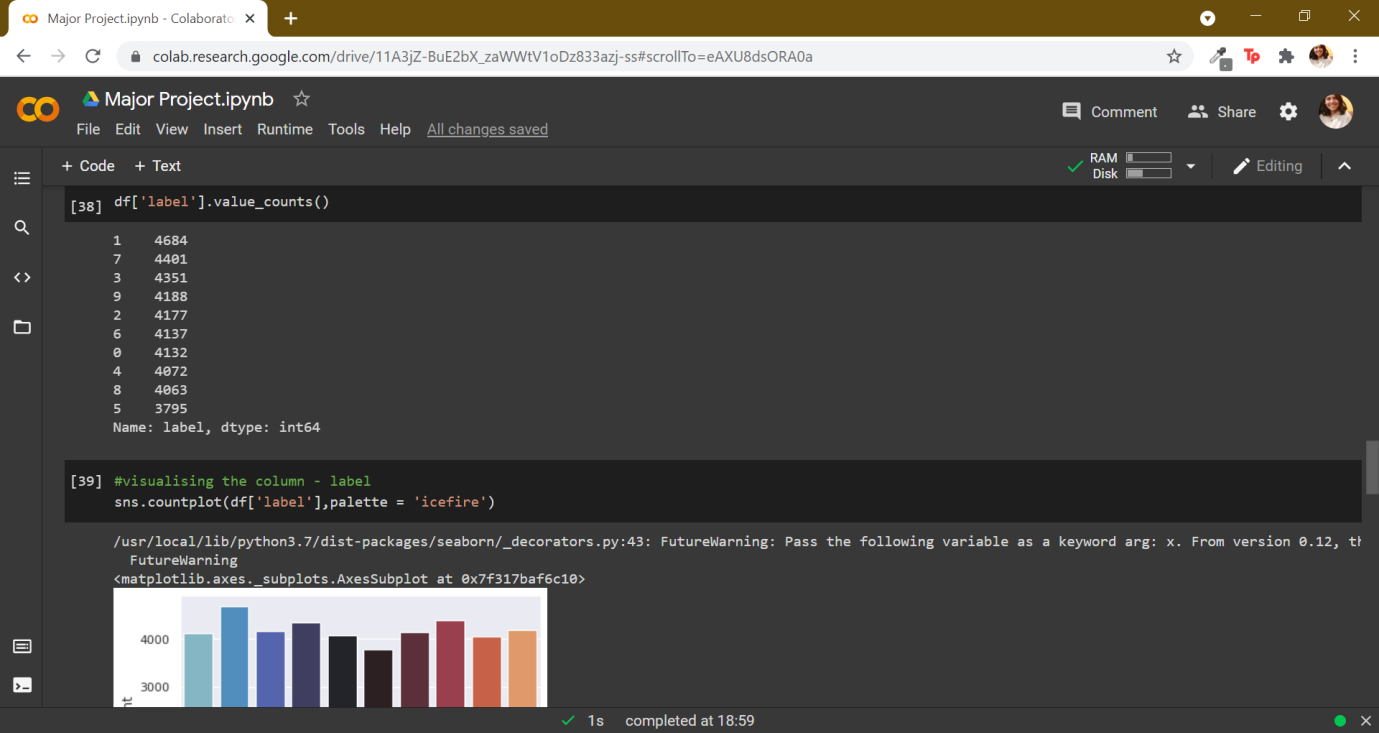


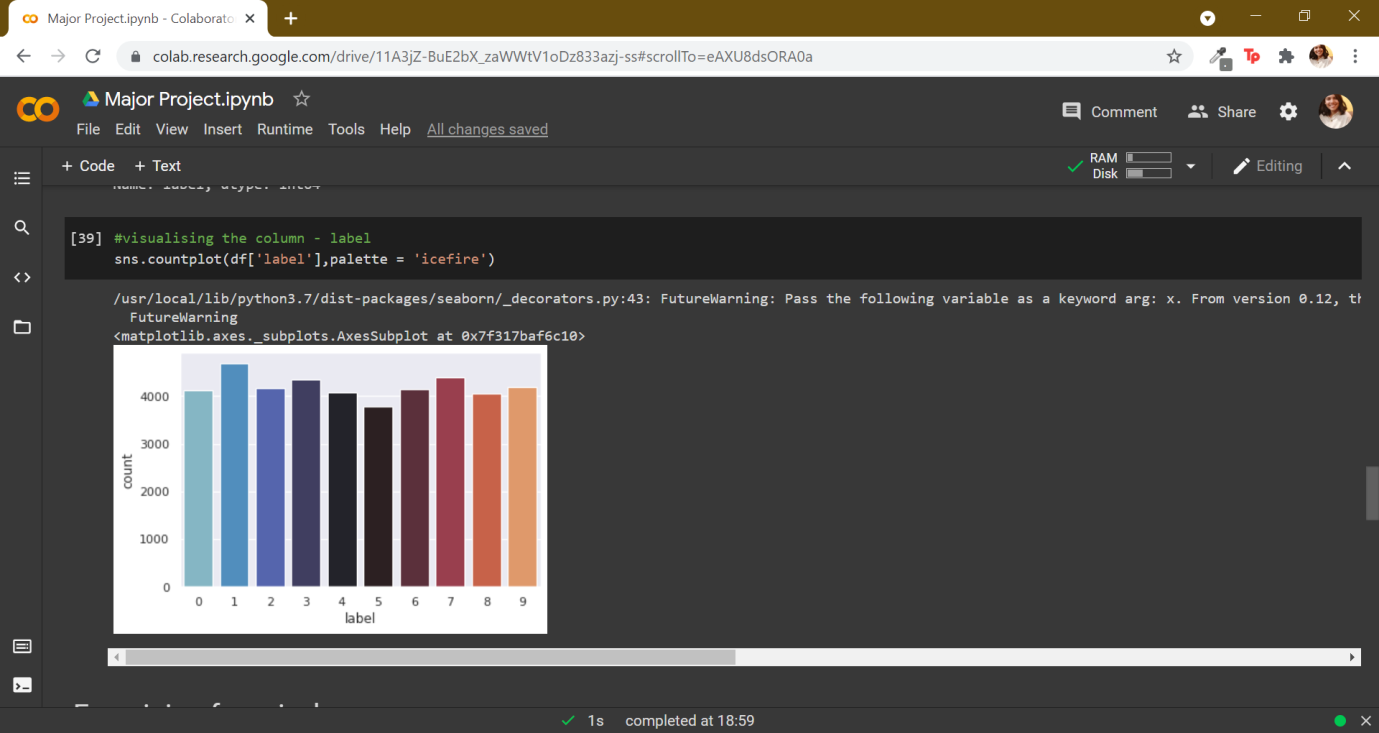


1. While exploring data we checked for Nan values. The label column shows unique 10 entries(0 to 9) which our model has to identify. Using various data visualisation techniques we get ourselves familiarised with the data frame.

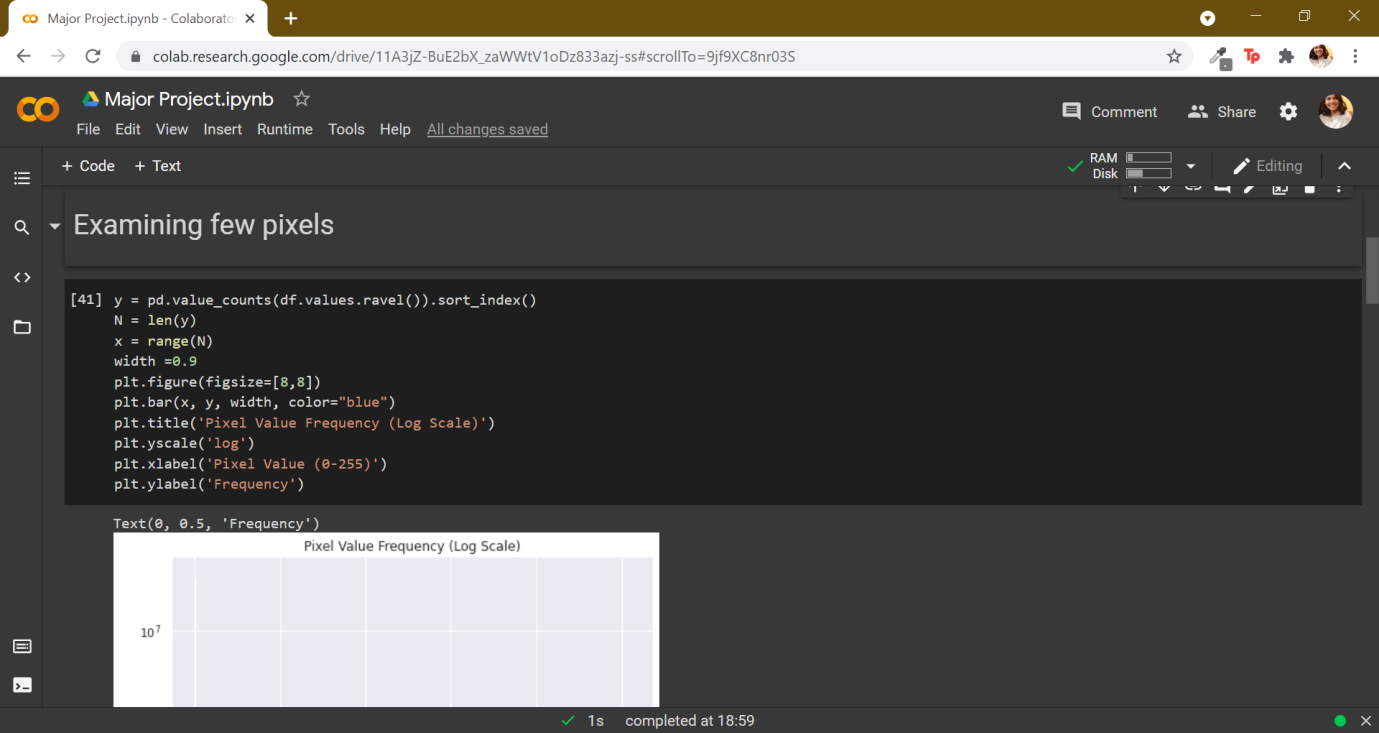


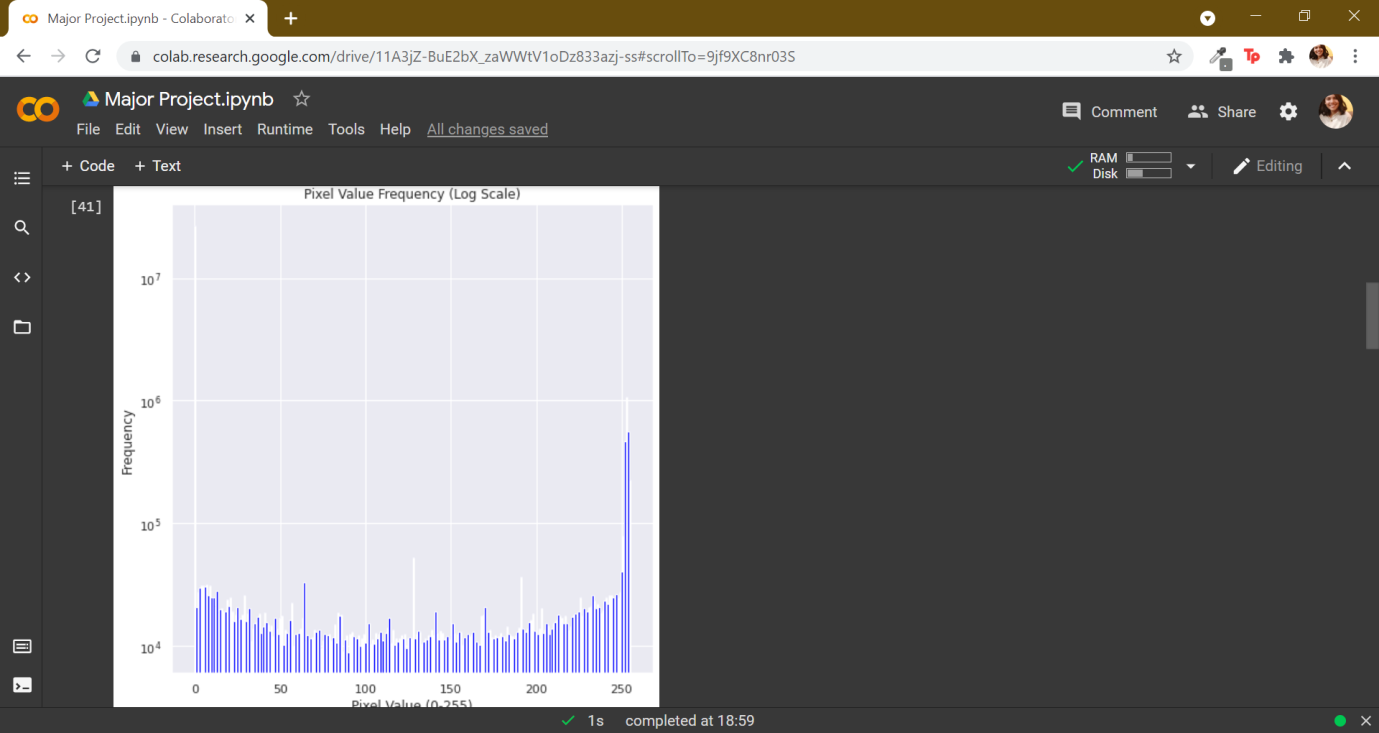


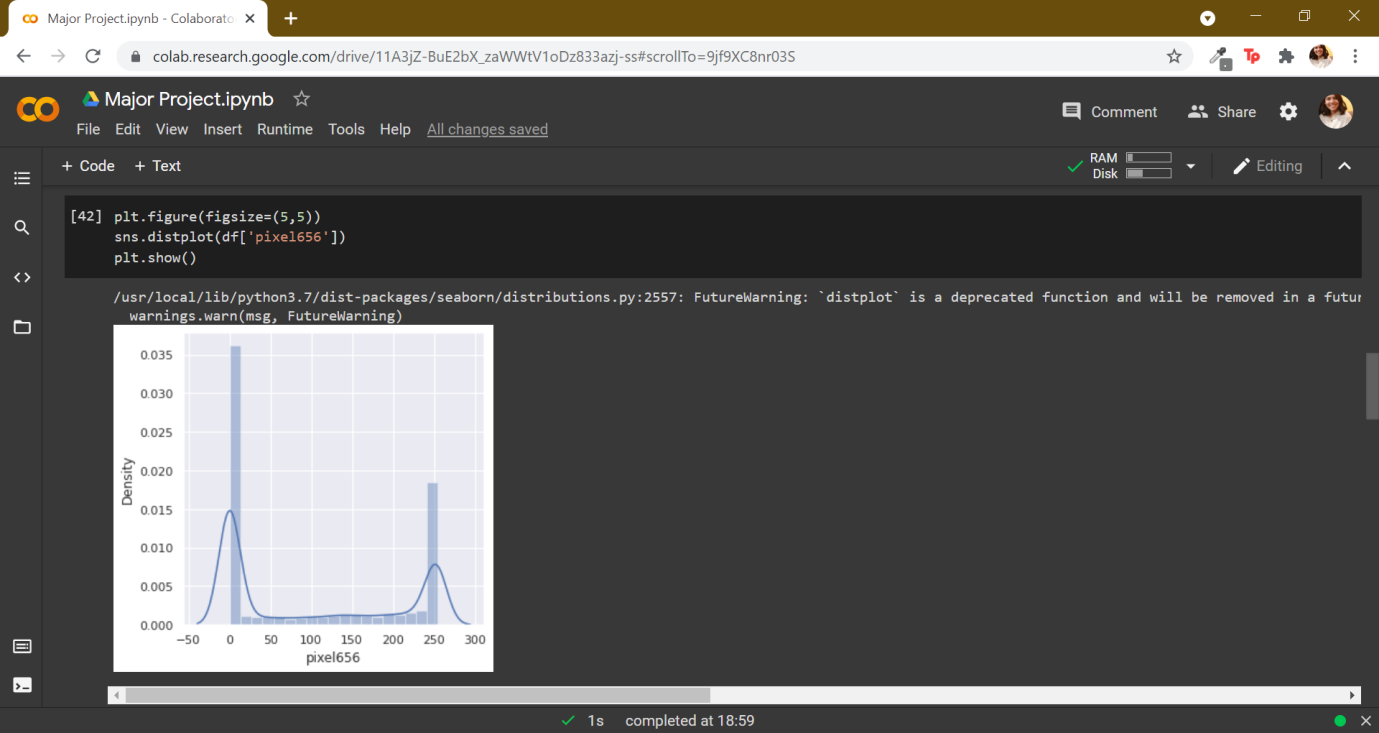


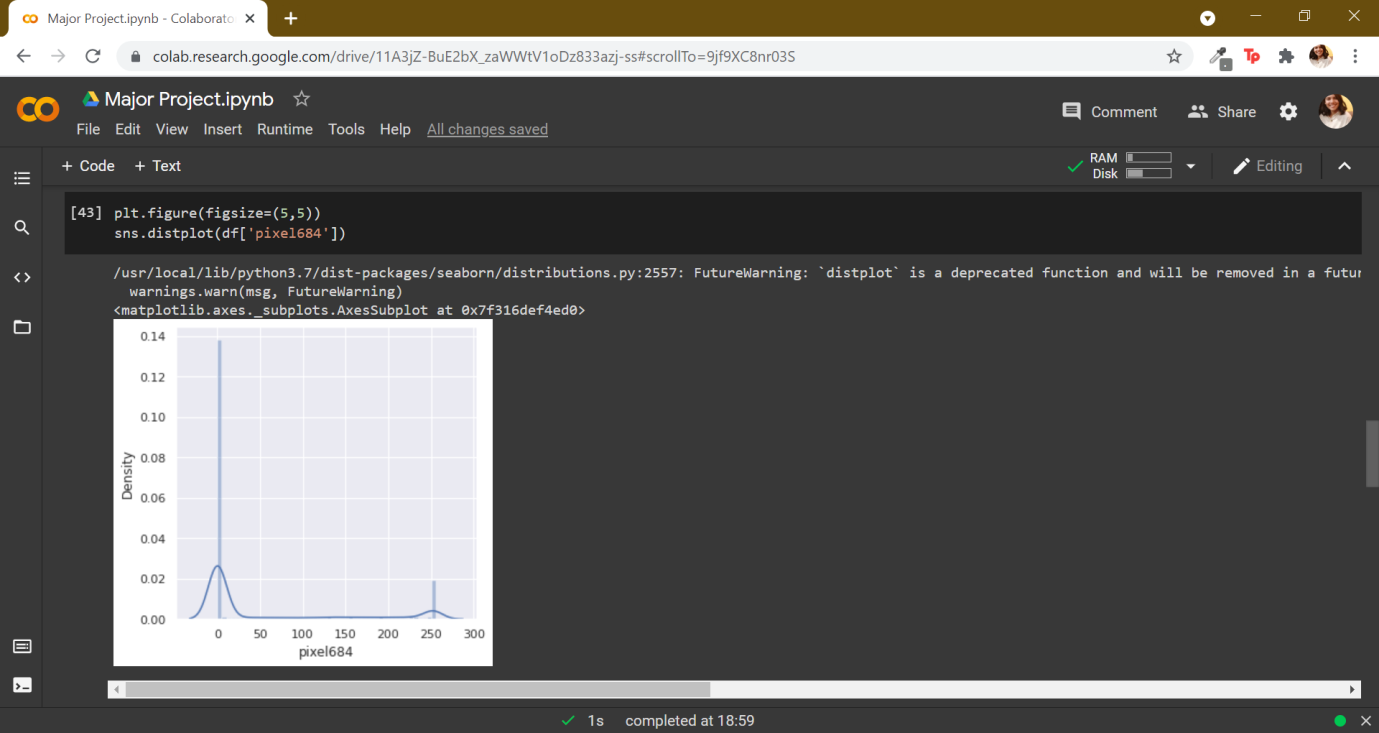


1. In the next section we examine few pixels using data visualisation techniques like learning the the number of times a certain pixel is shown in the dataframe, observing the density of a particular pixel.

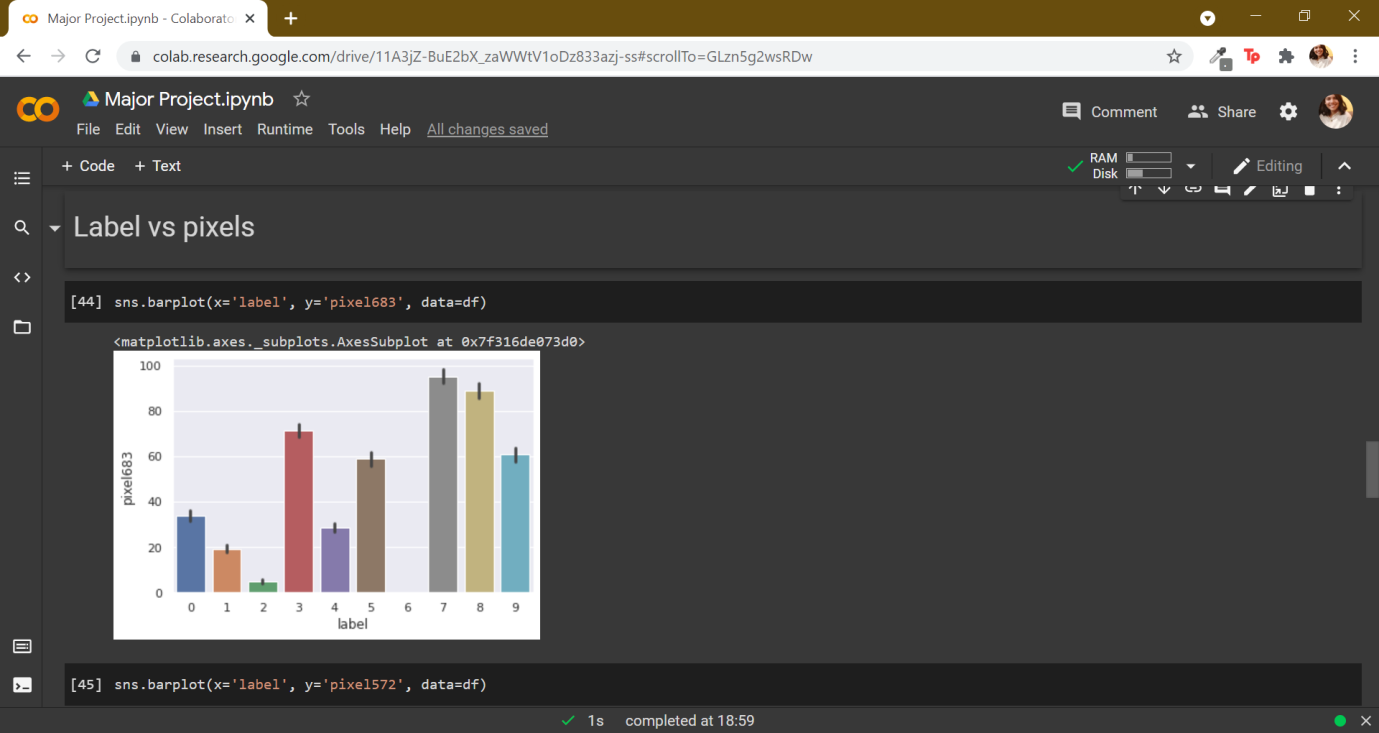


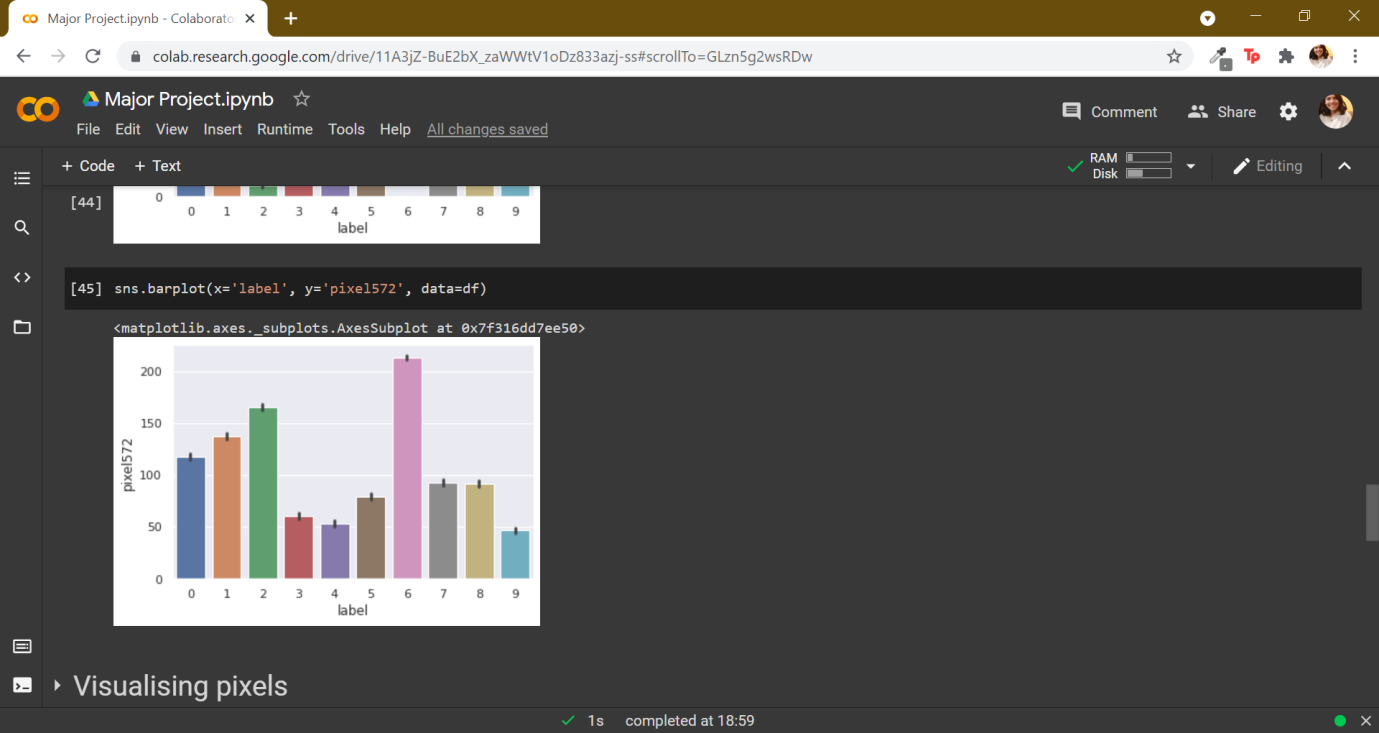




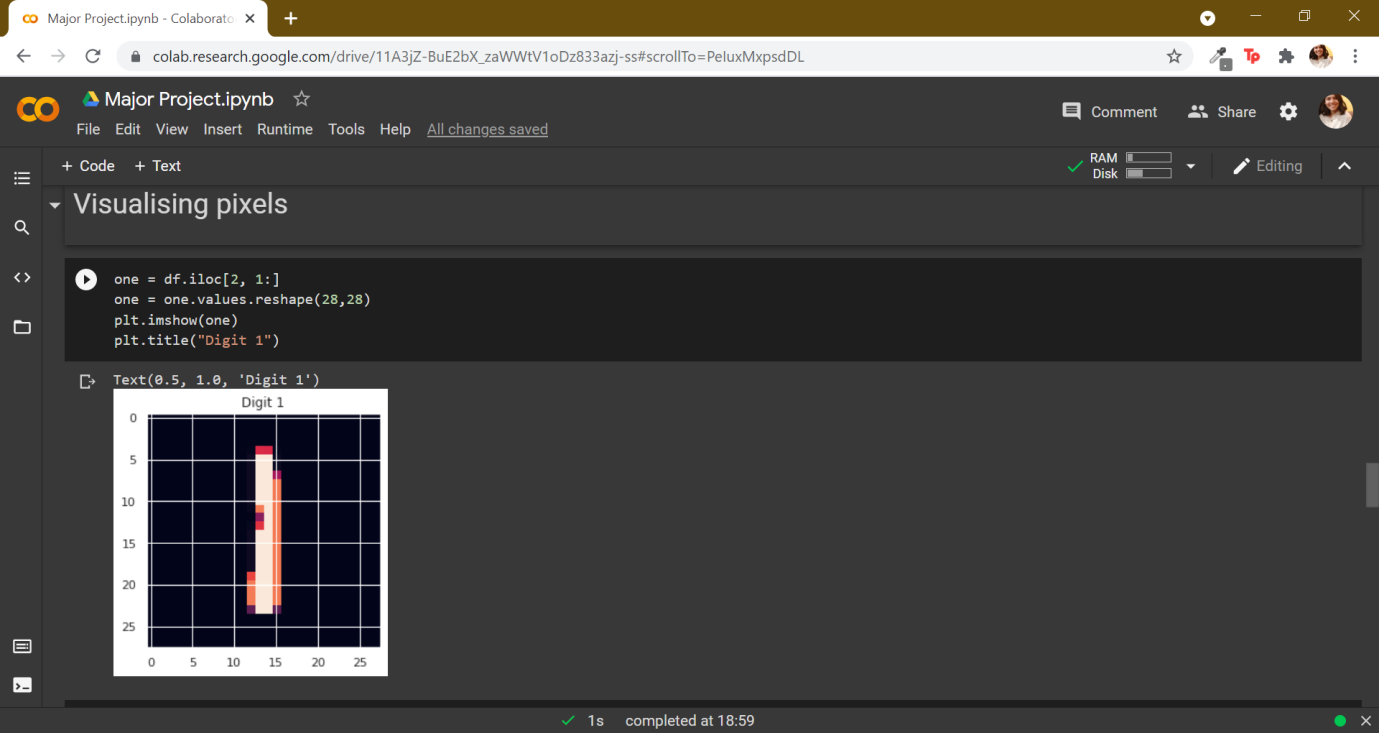


1. In label vs pixels section we use barplots to find relationship with label and their pixels.

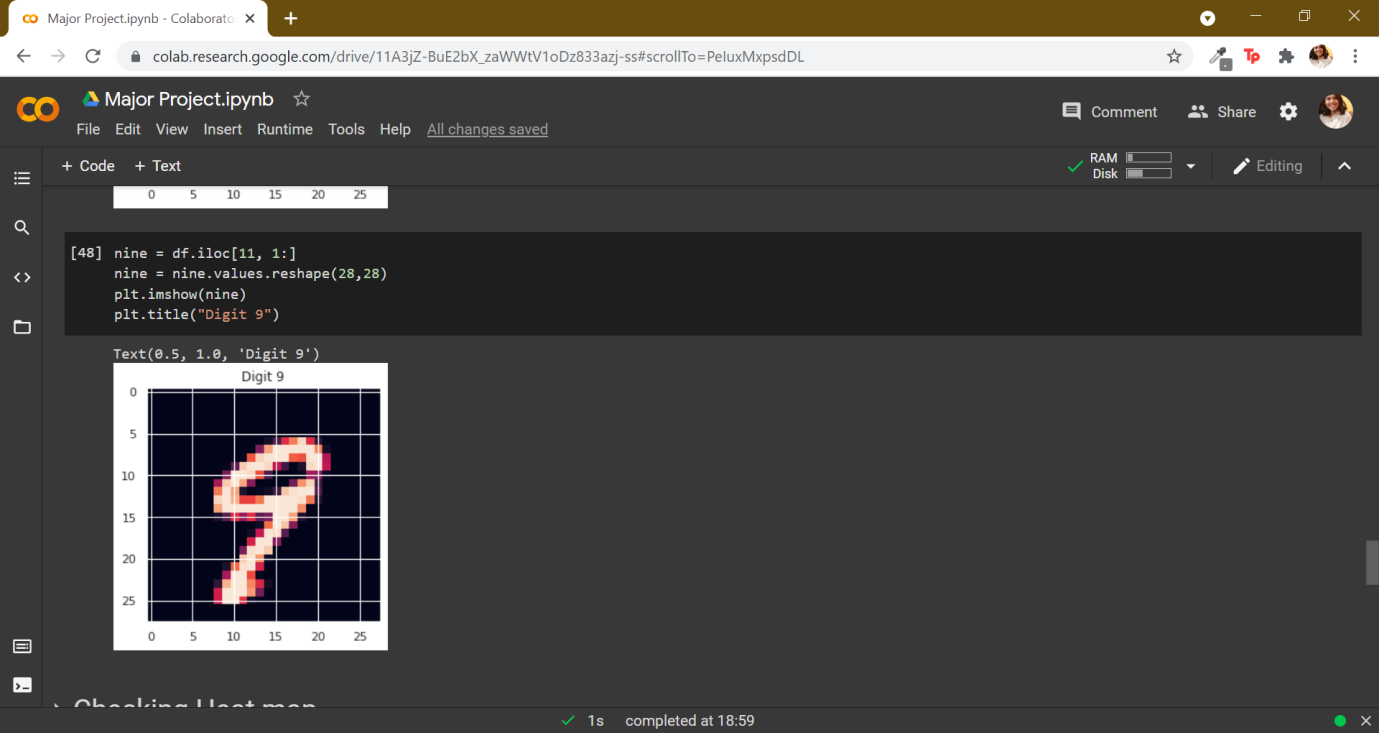




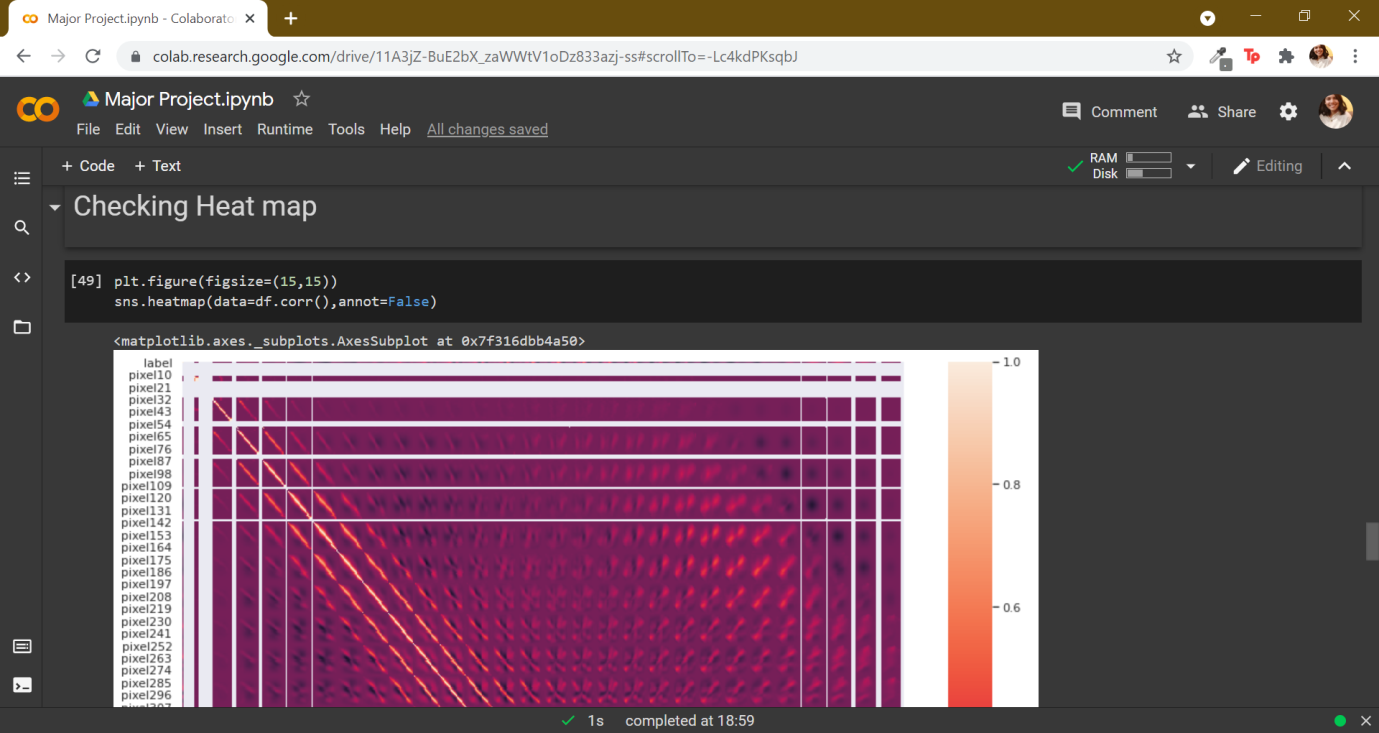
1. Using .imshow() function we can visualise a number between 0 to 9 using pixels.

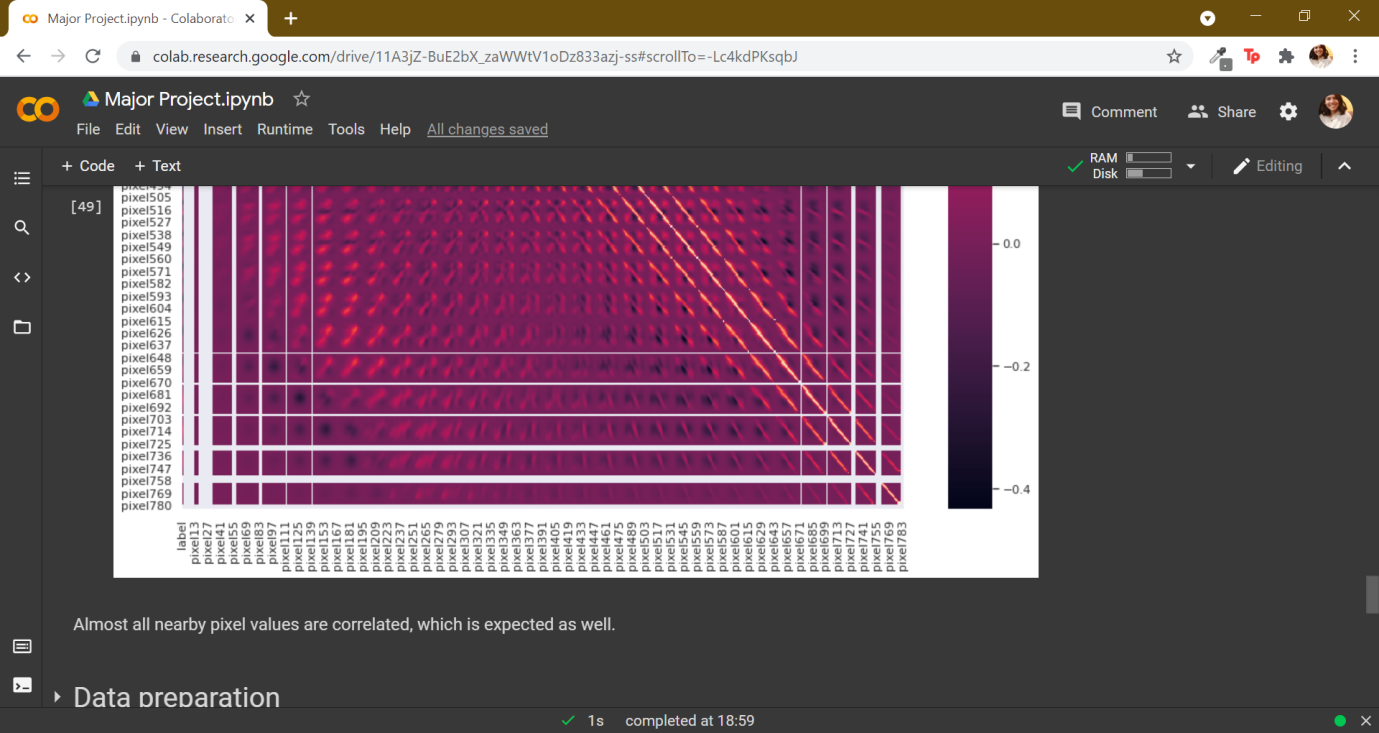




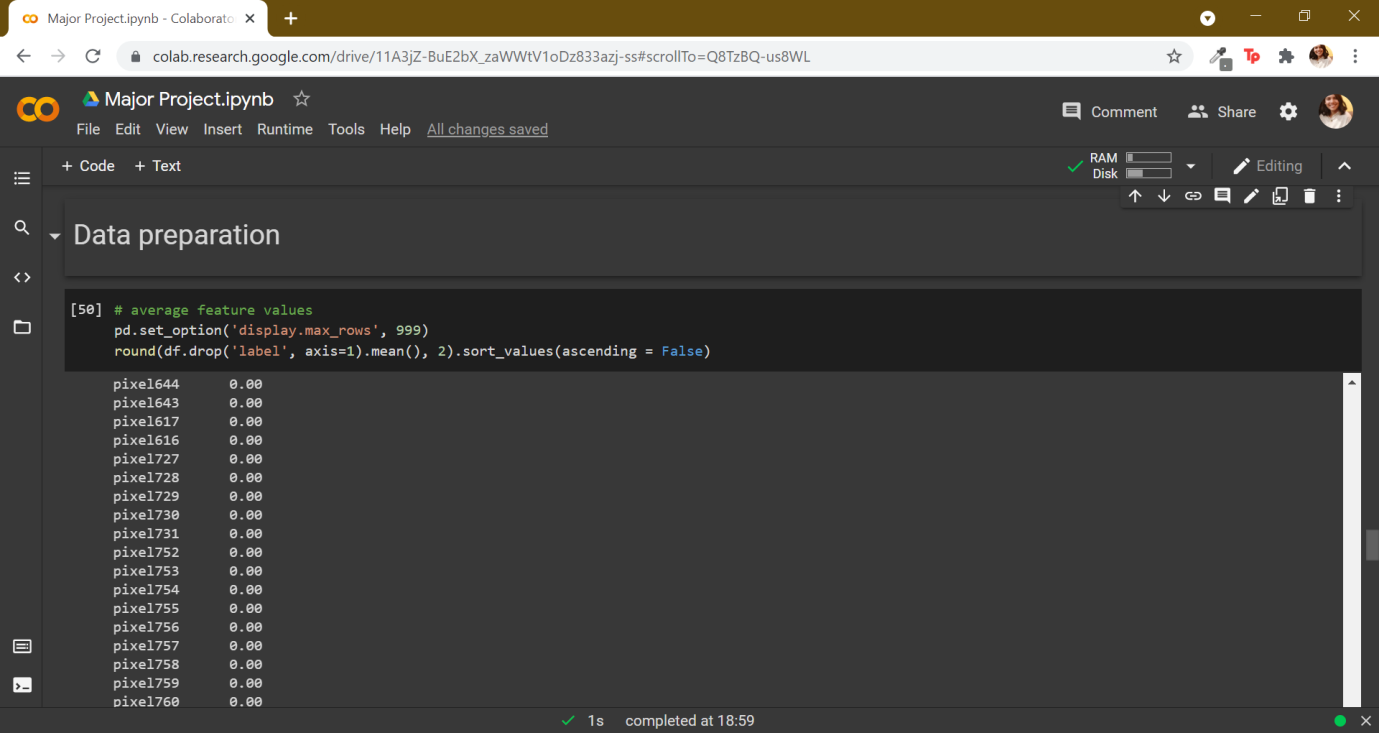


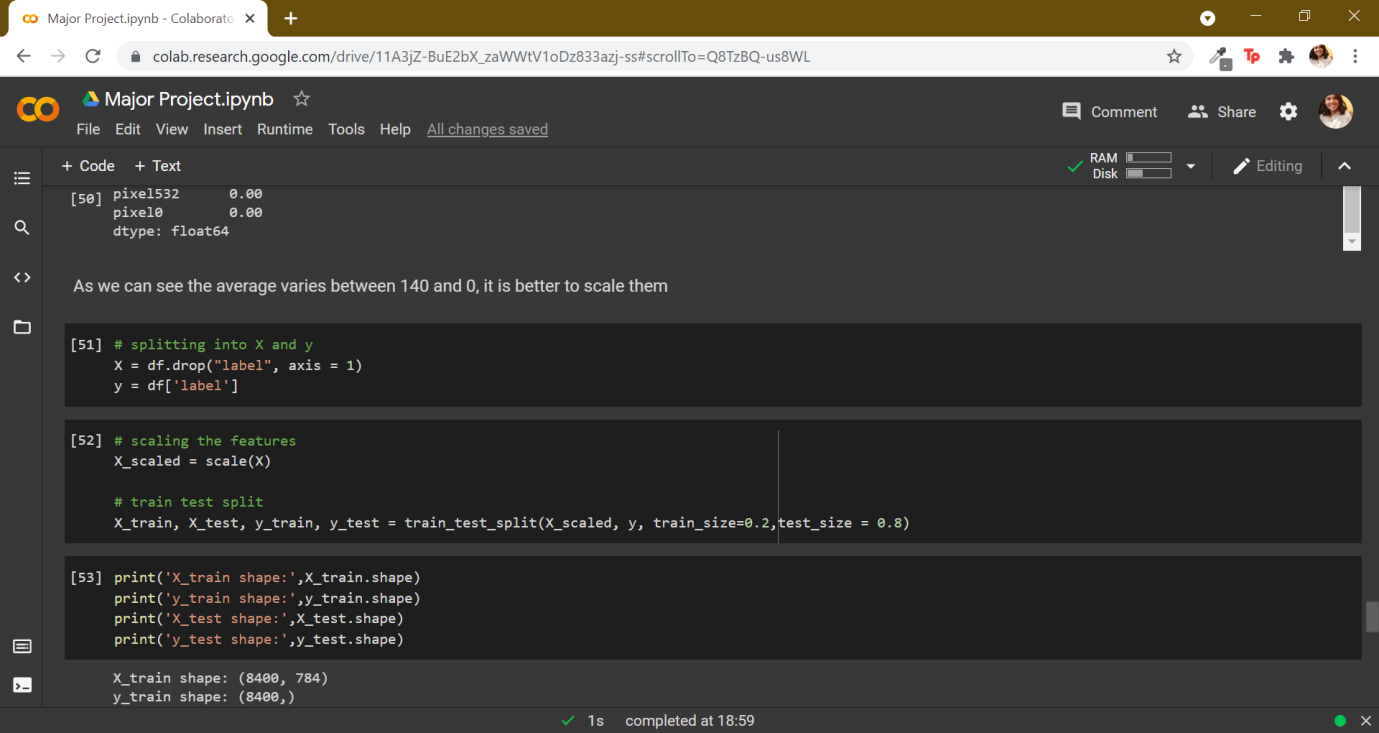
1. We check the heat map which shows all the nearby pixel values are correlated.



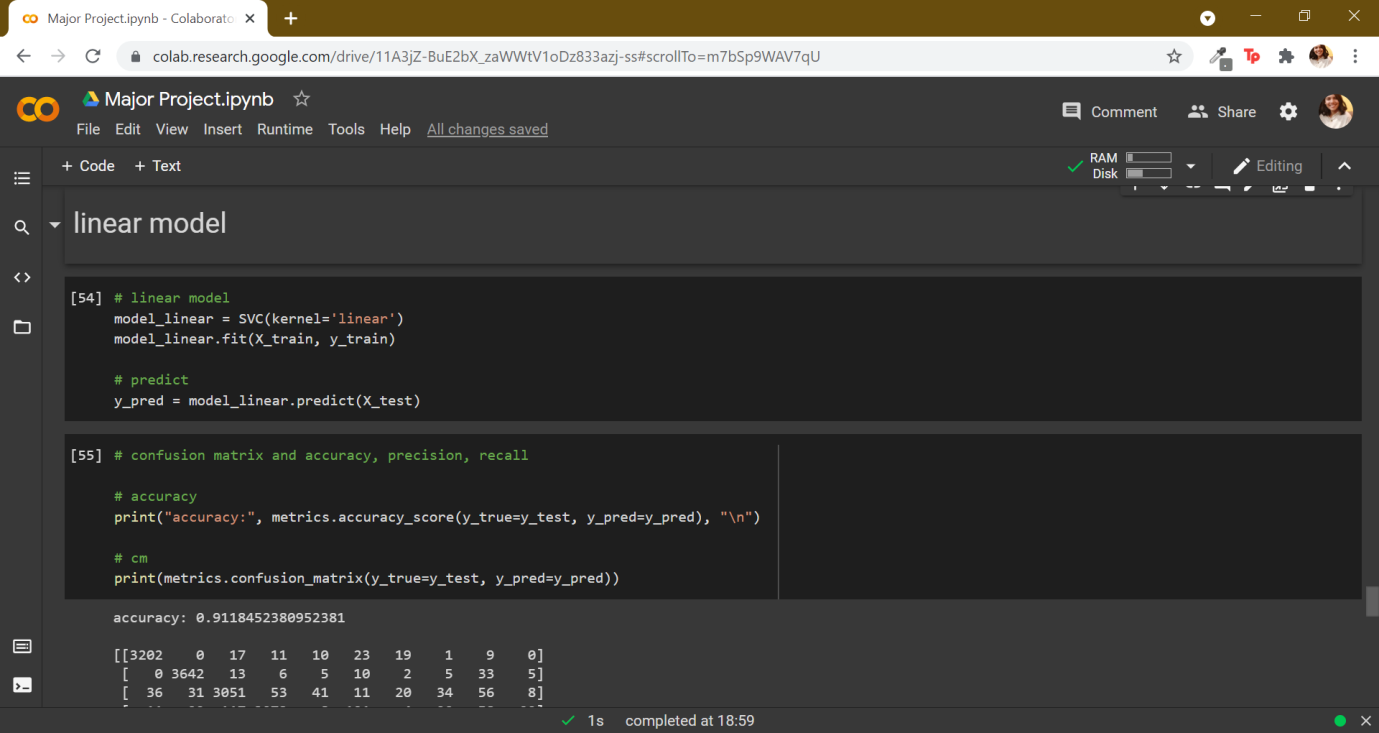


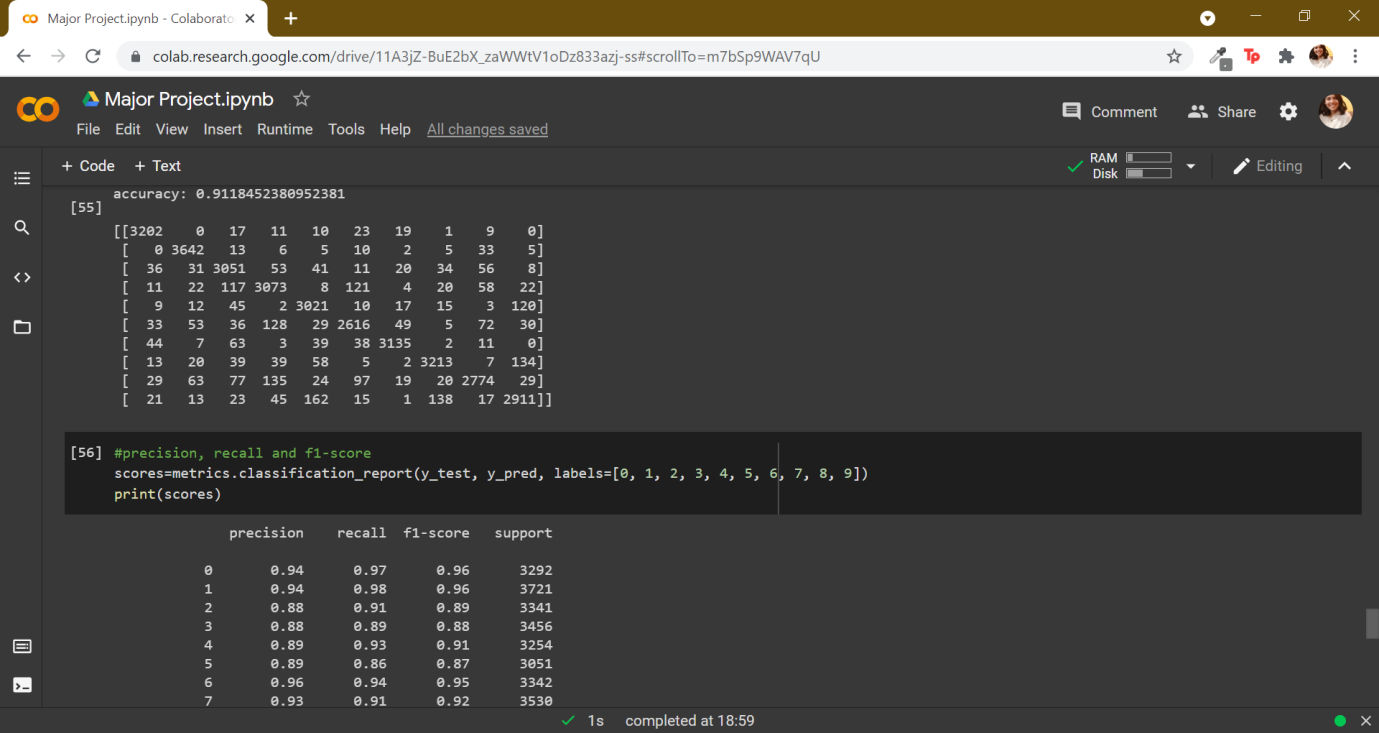
1. In our data preparation section we find that our data is varying between 0 to 140 hence we scale it using function and the split the data into train and test data.

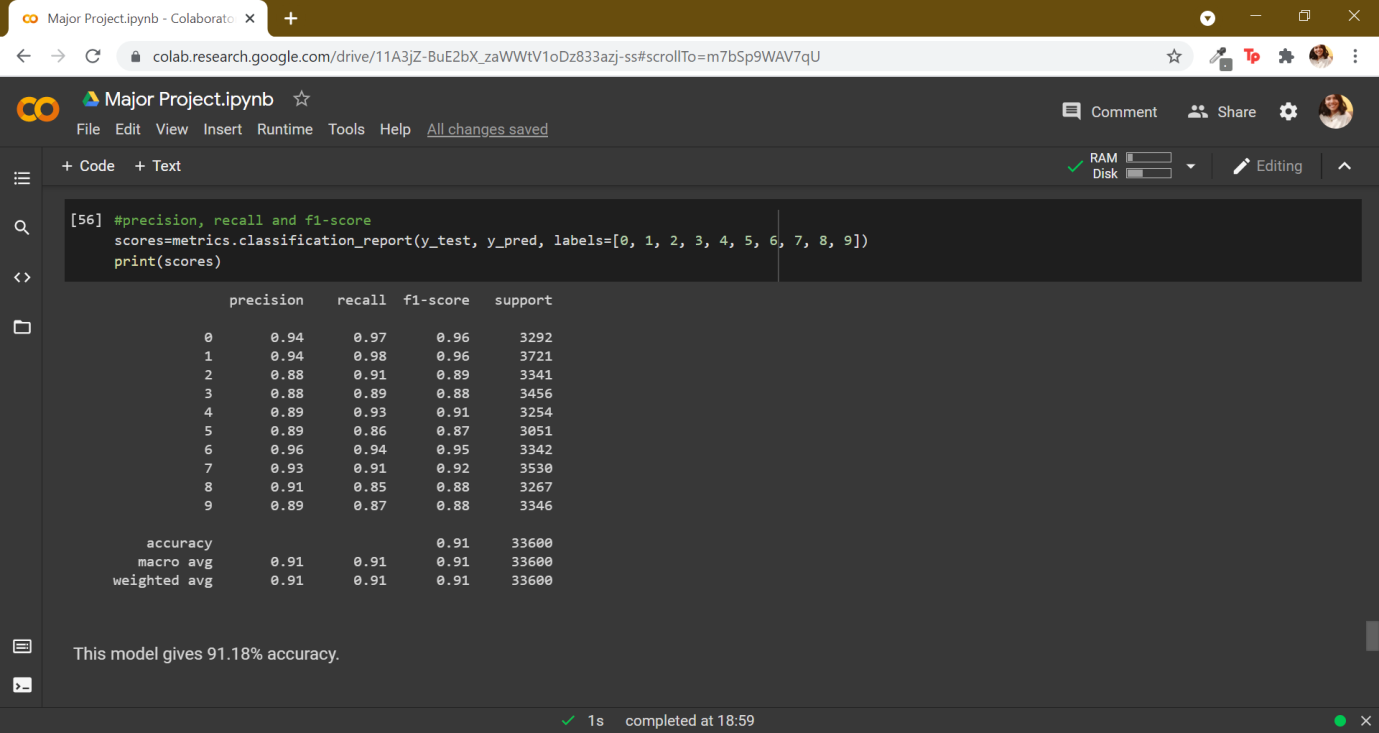




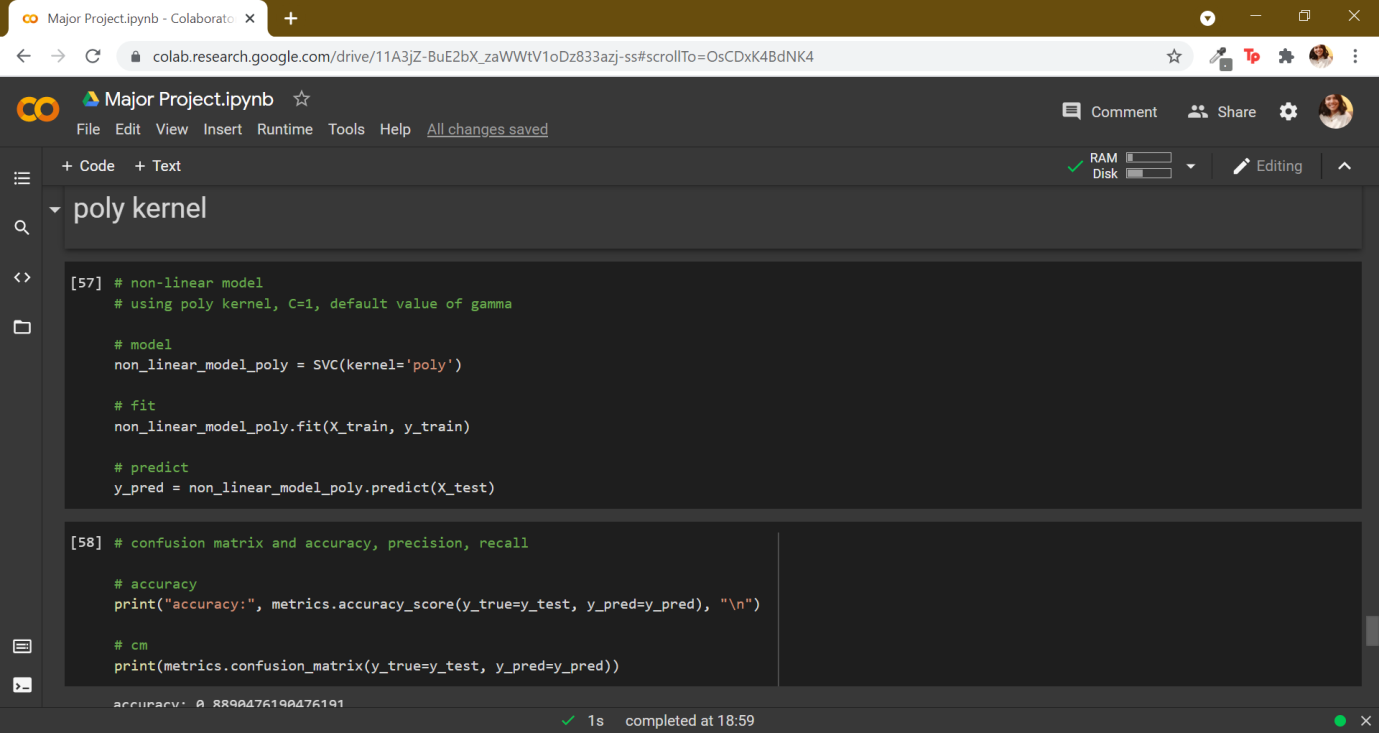
1. We build basic models for our problem statement. We first make linear model. The accuracy of this model was calculated out to be 91.18%.

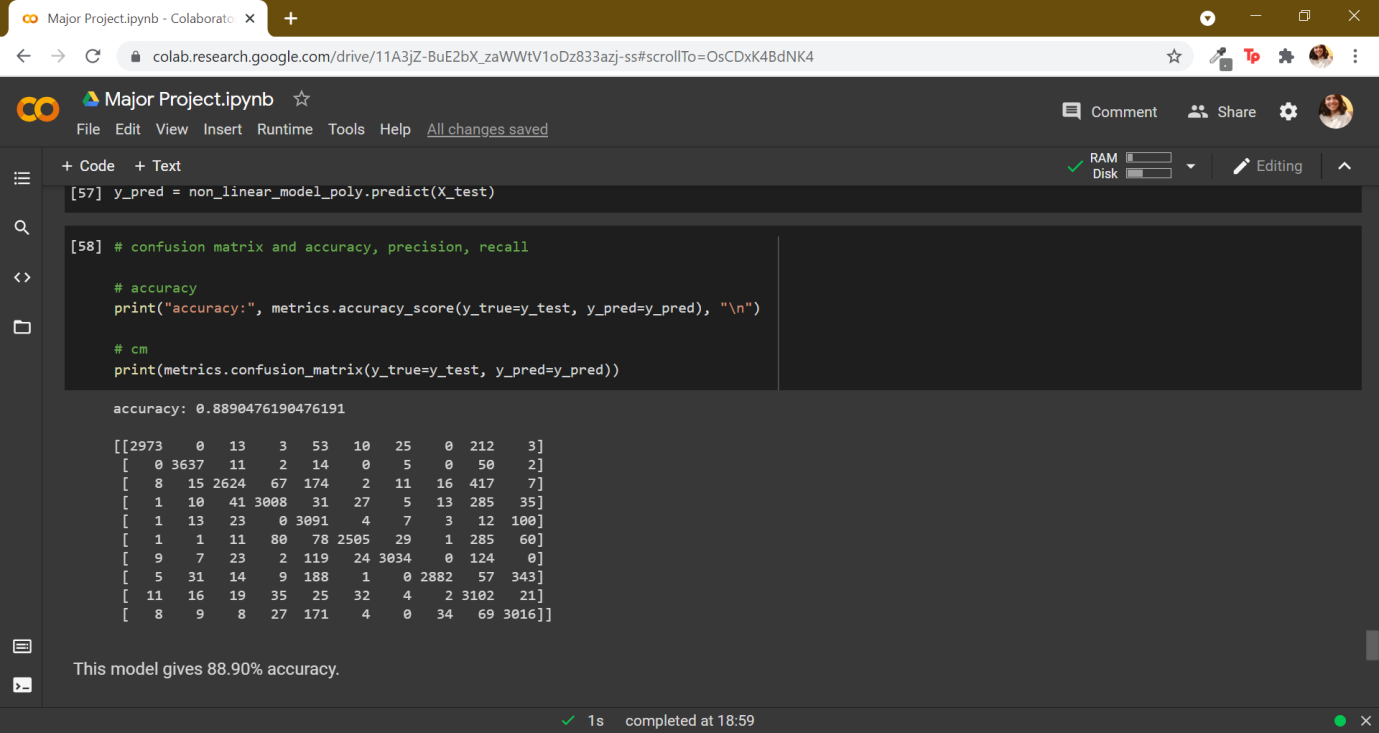




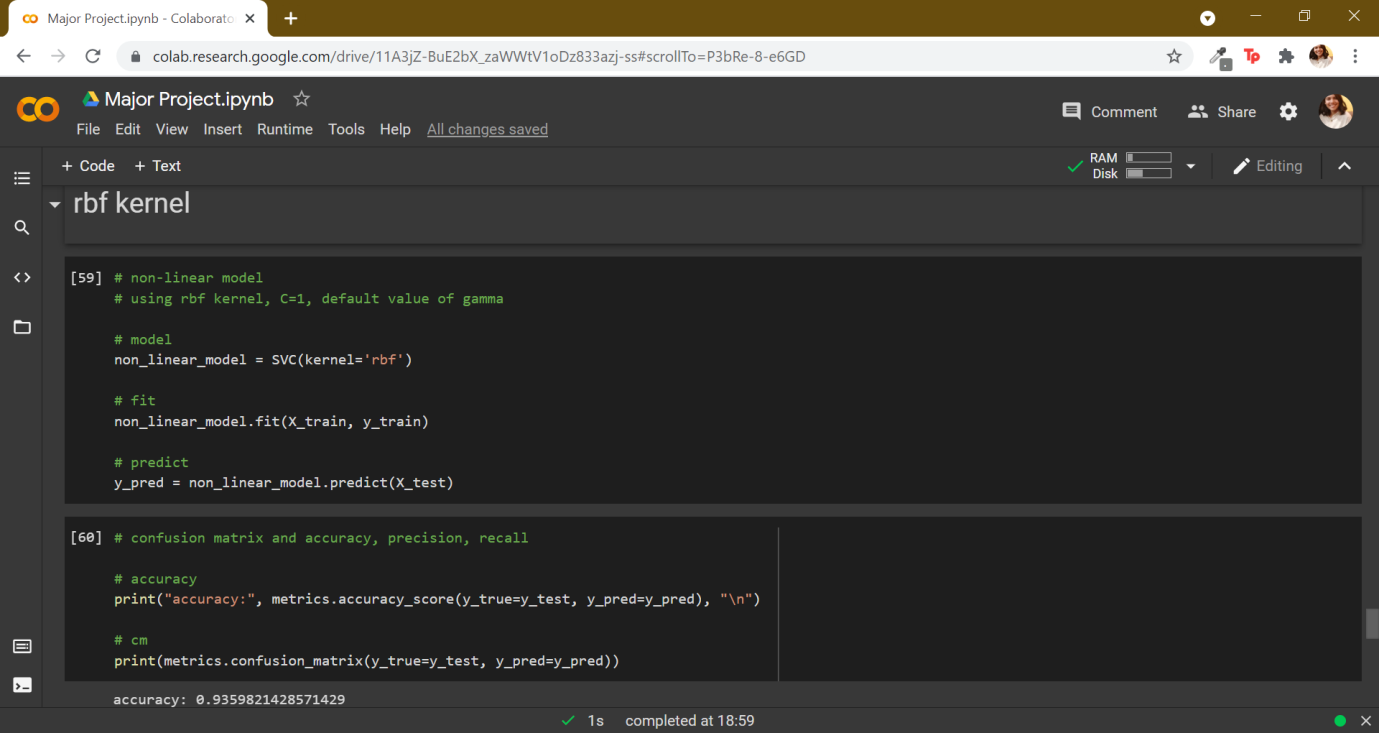


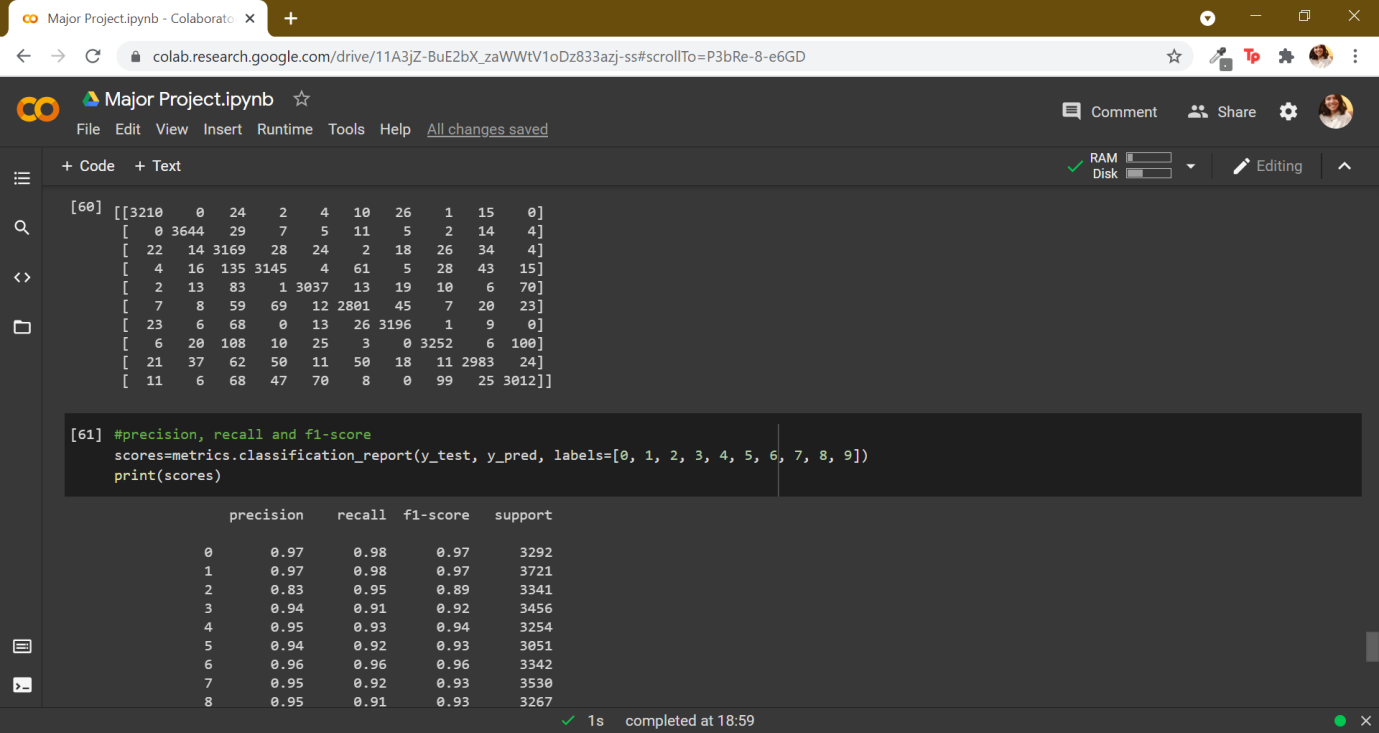
1. We now build non-linear model poly kernel whose accuracy was 88.98%.

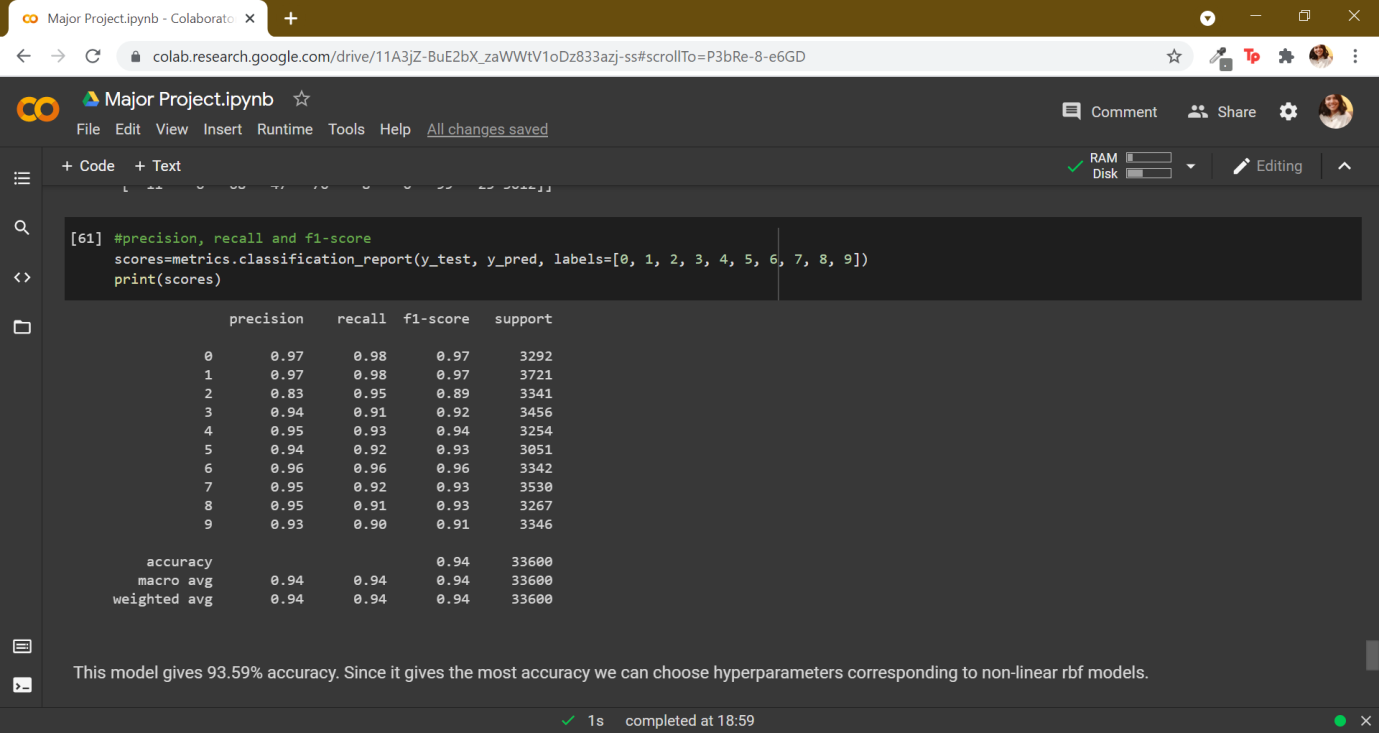




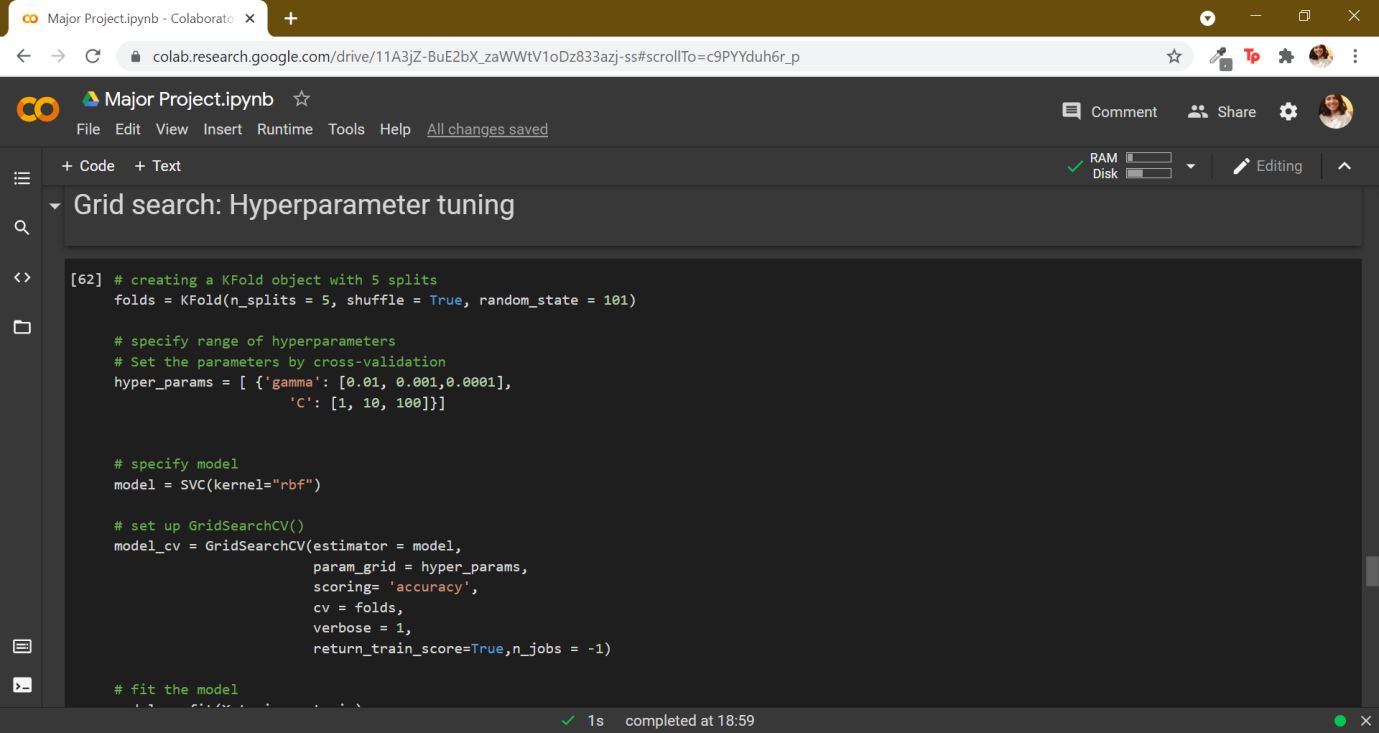
1. Other non-linear model rbf kernel was built and the accuracy was 93.59%. This model has the highest accuracy out of the above models hence we can choose hyperparameters corresponding to nonlinear rbf kernel.

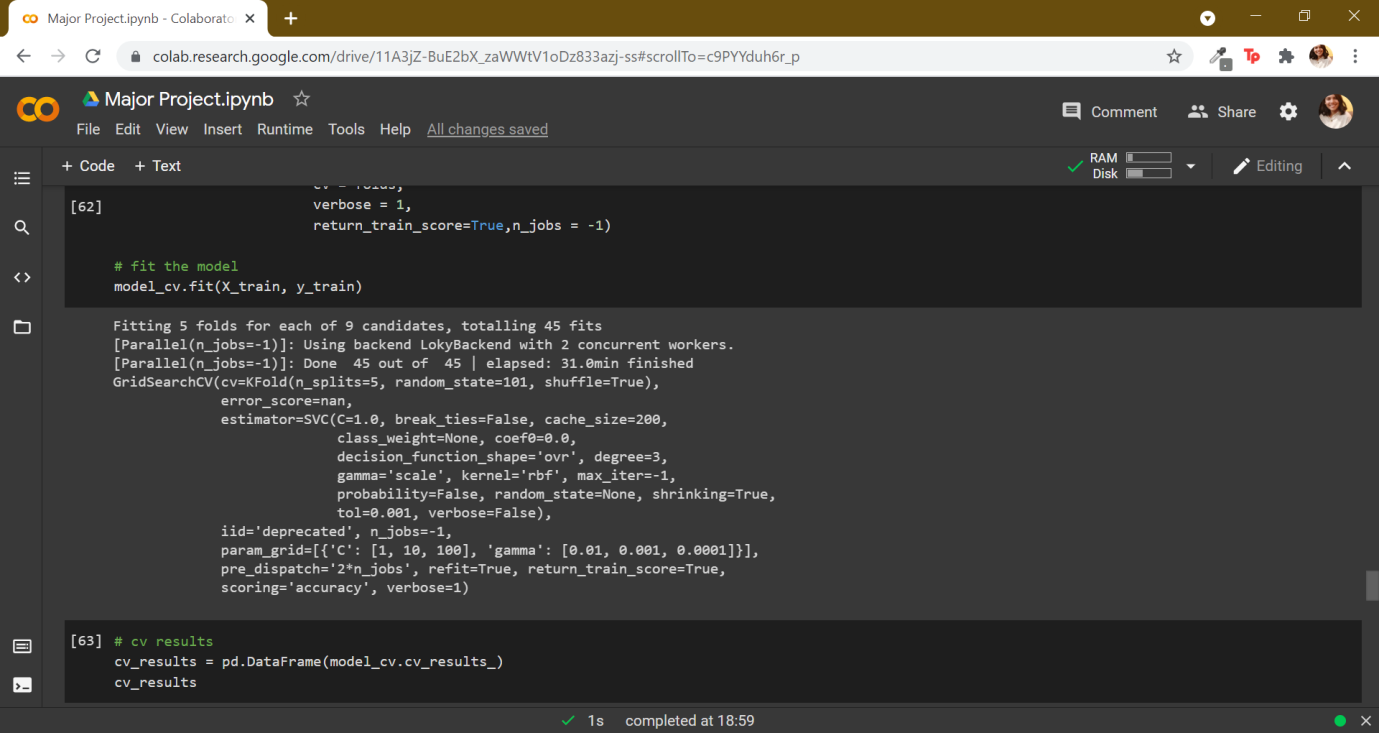


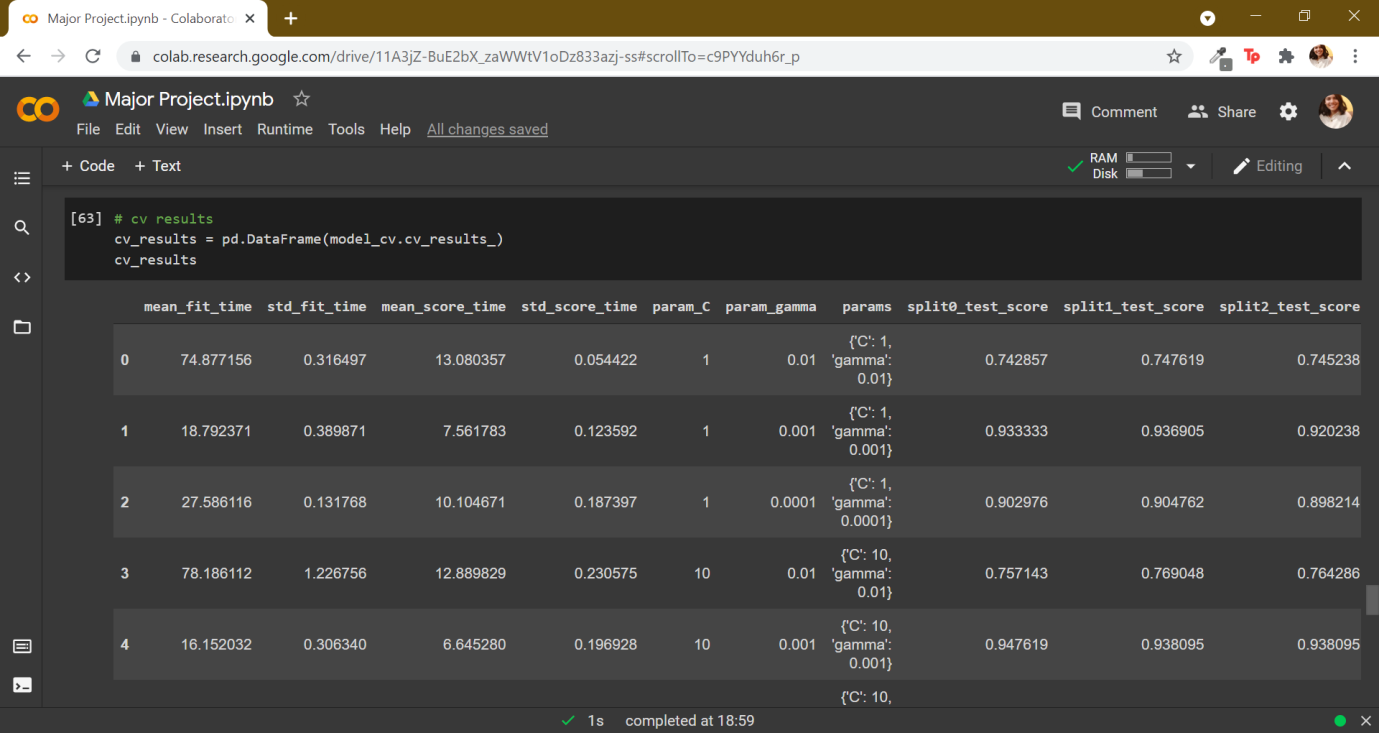


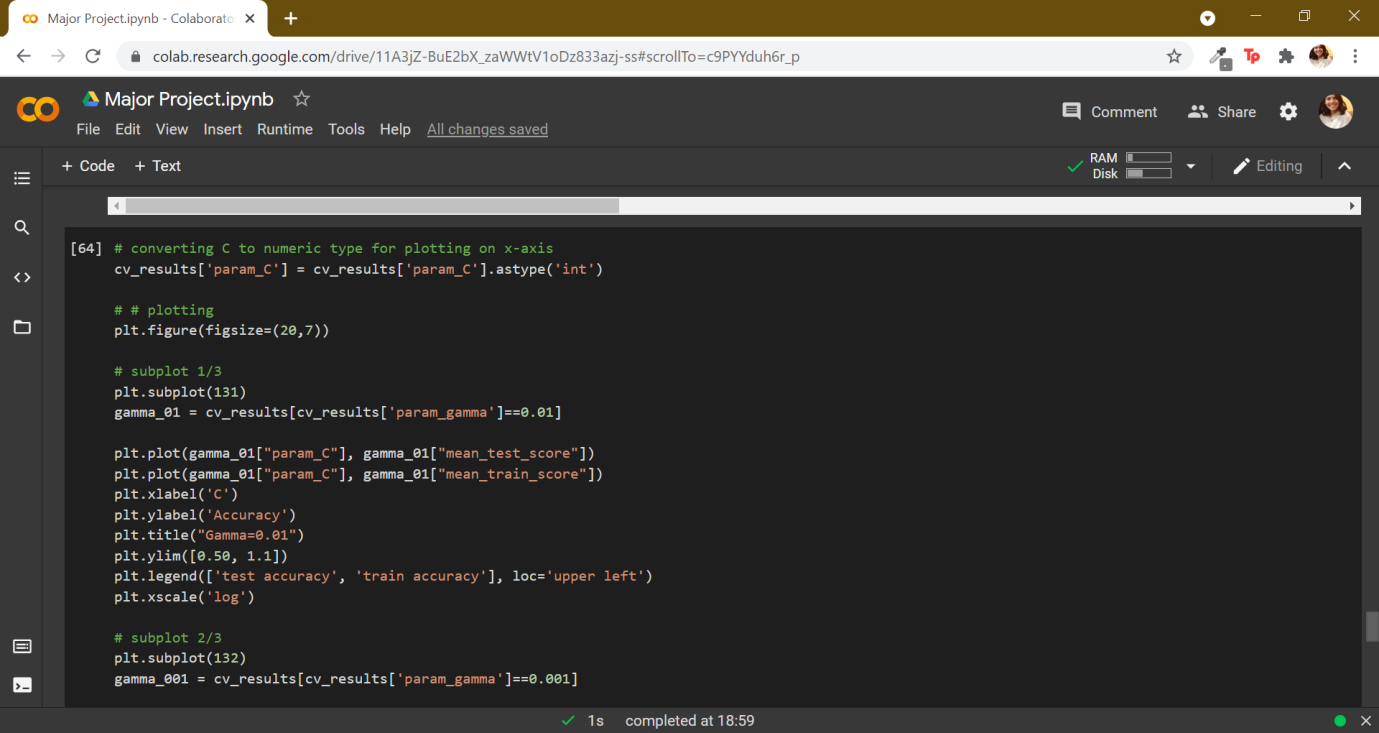


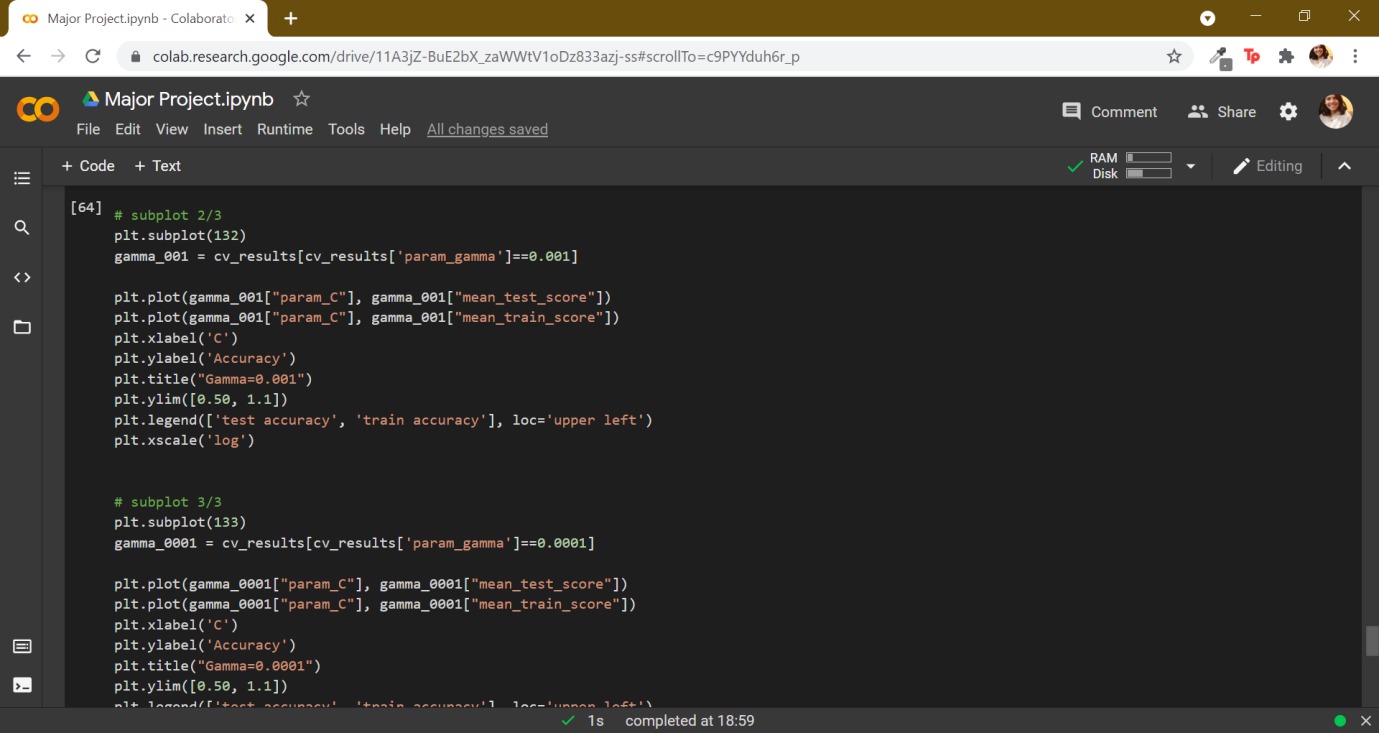
1. We now use SVC and set up GridSearchCV. Now we can fit our data into the model. Finally we find out cv results. Later we plot the values of C and the accuracy. We then calculate and print the optimal accuracy score and hyperparameters. The accuracy score is 94.19% and the hyperparameters are C: 100, gama:0.001.

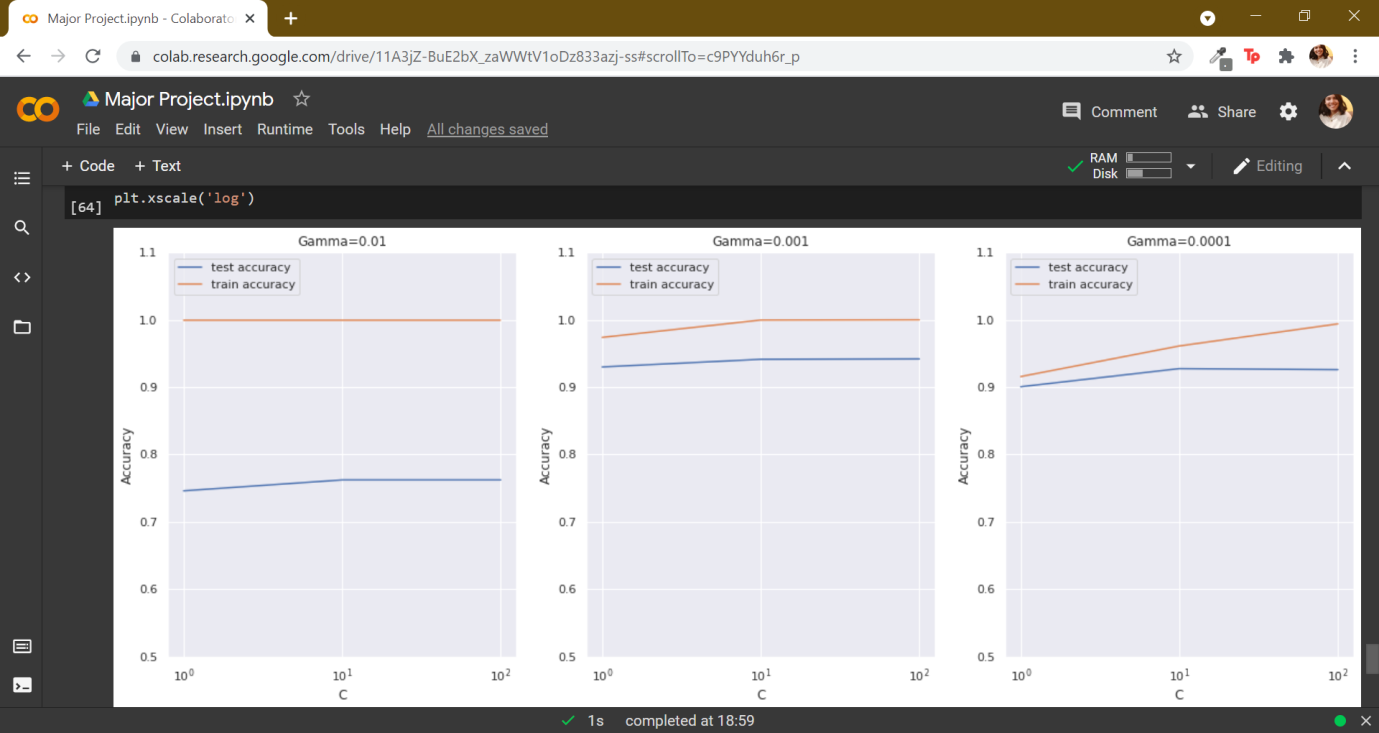


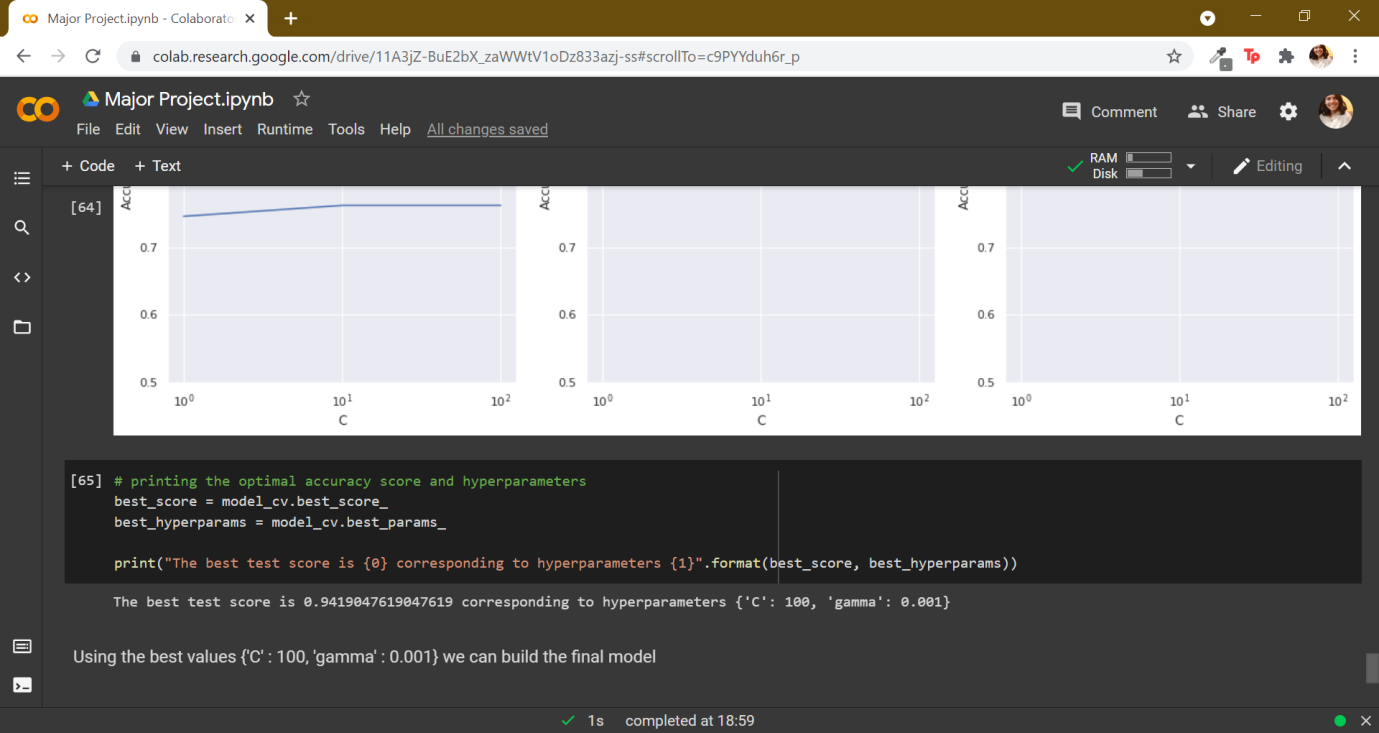




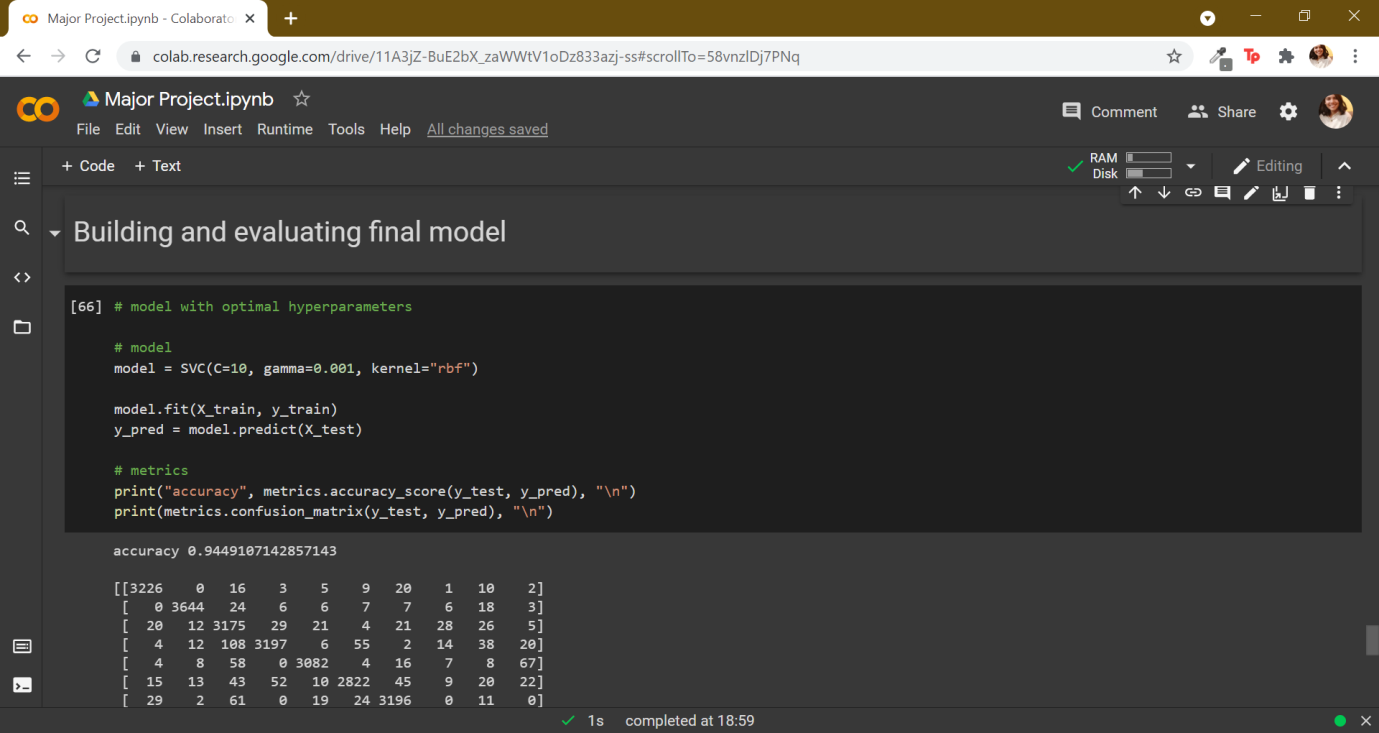


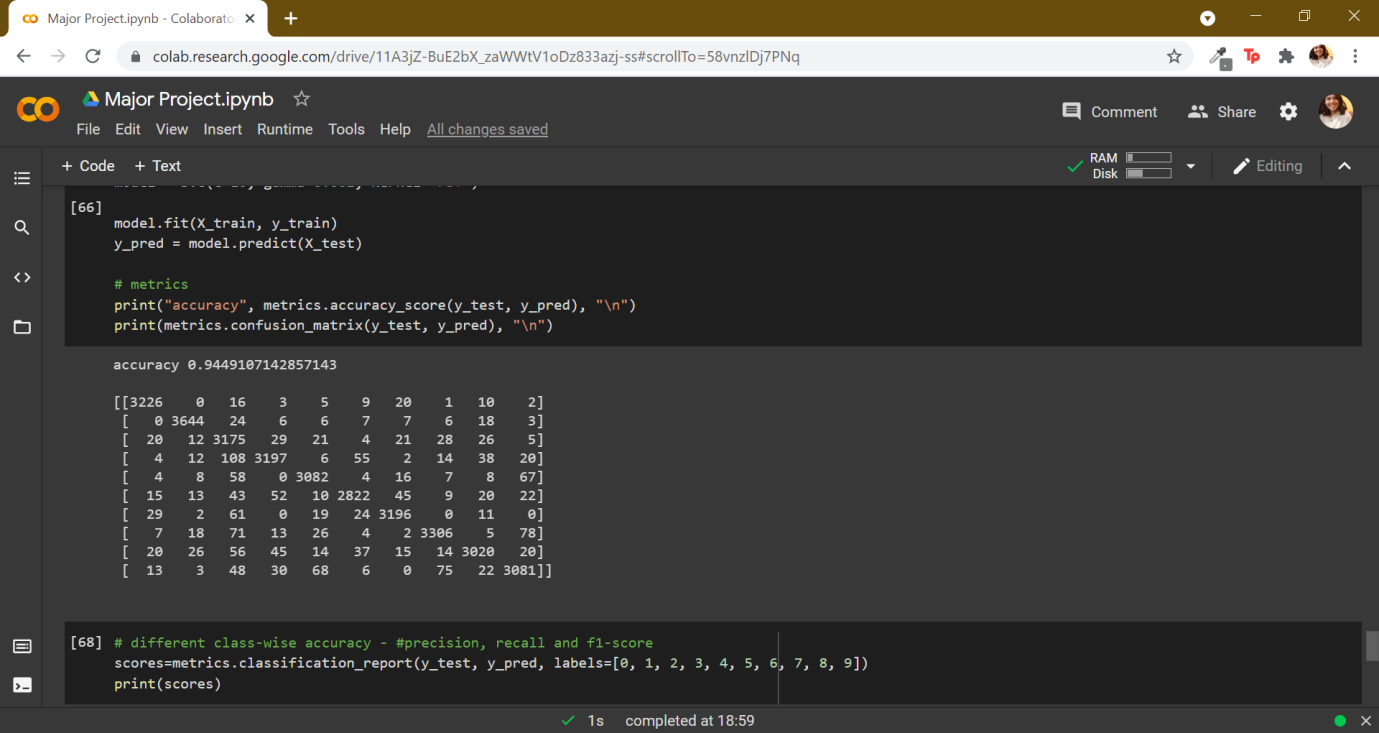


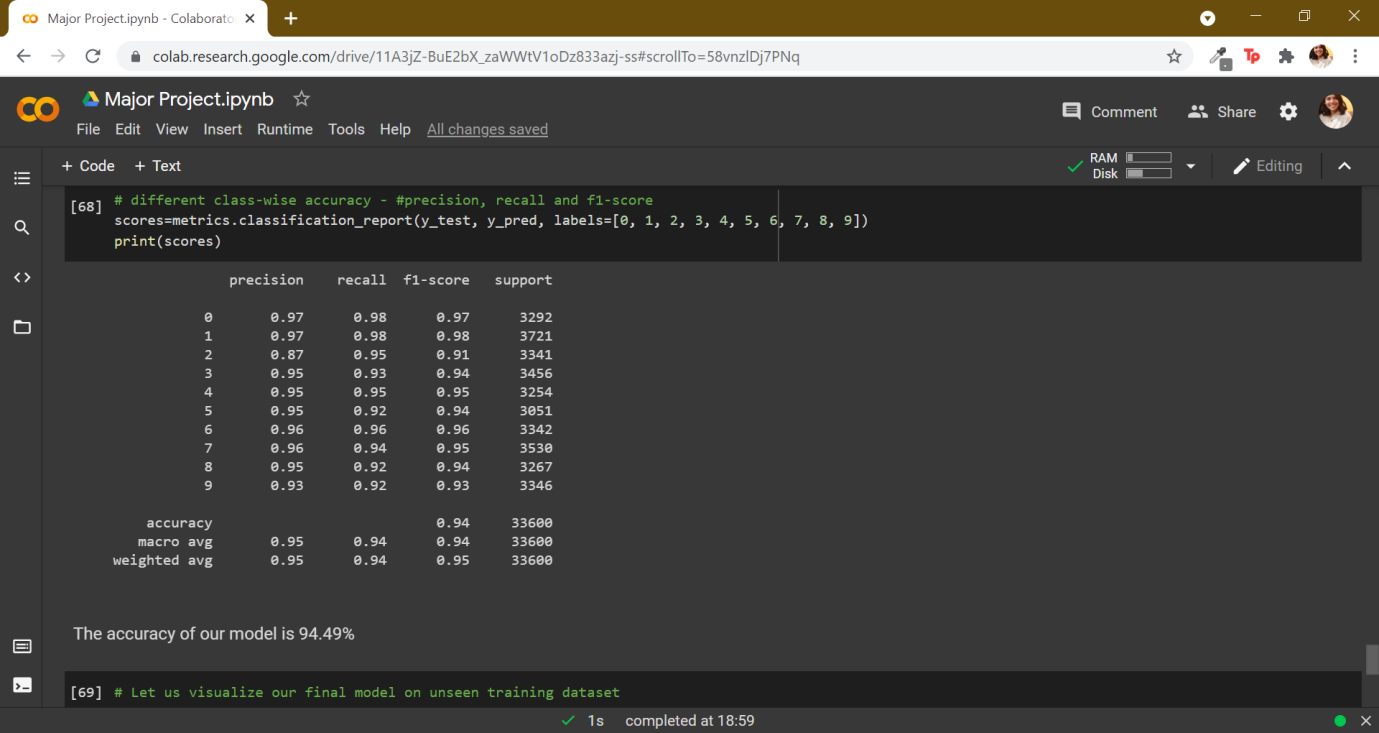


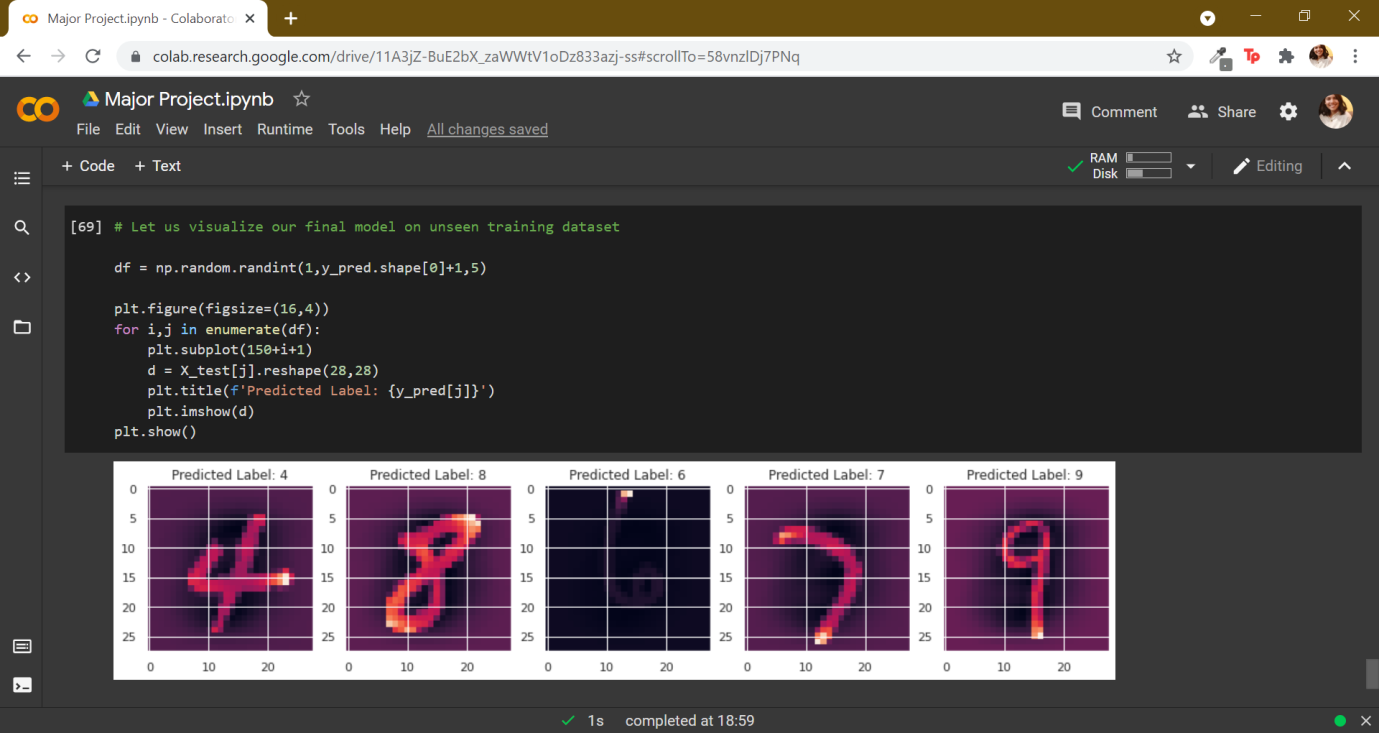


1. Using best values of hyperparameters we build and evaluate our rbf kernel model. The accuracy of this model is 94.49%. We can also visualise the numbers using imshow() function.

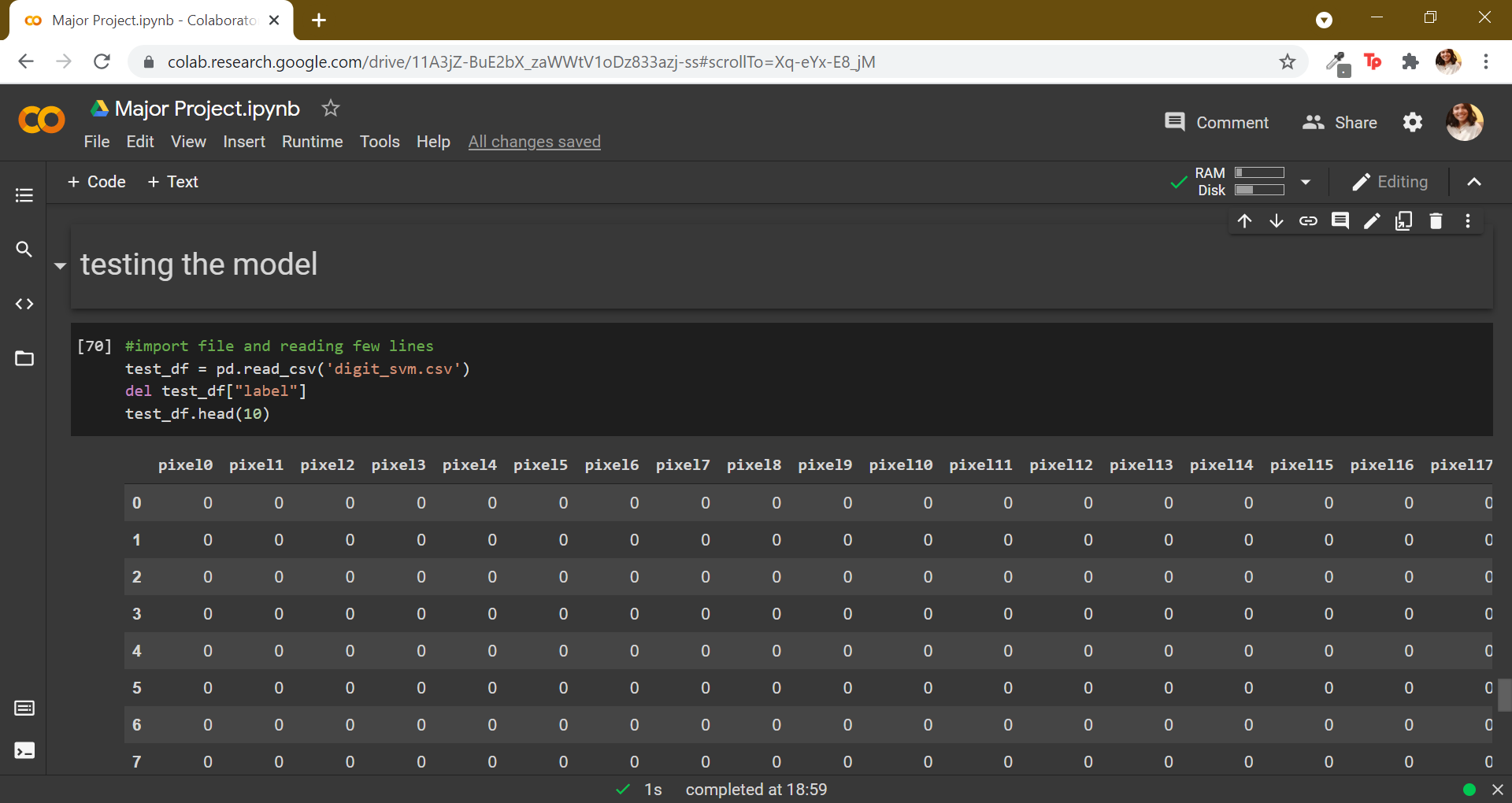


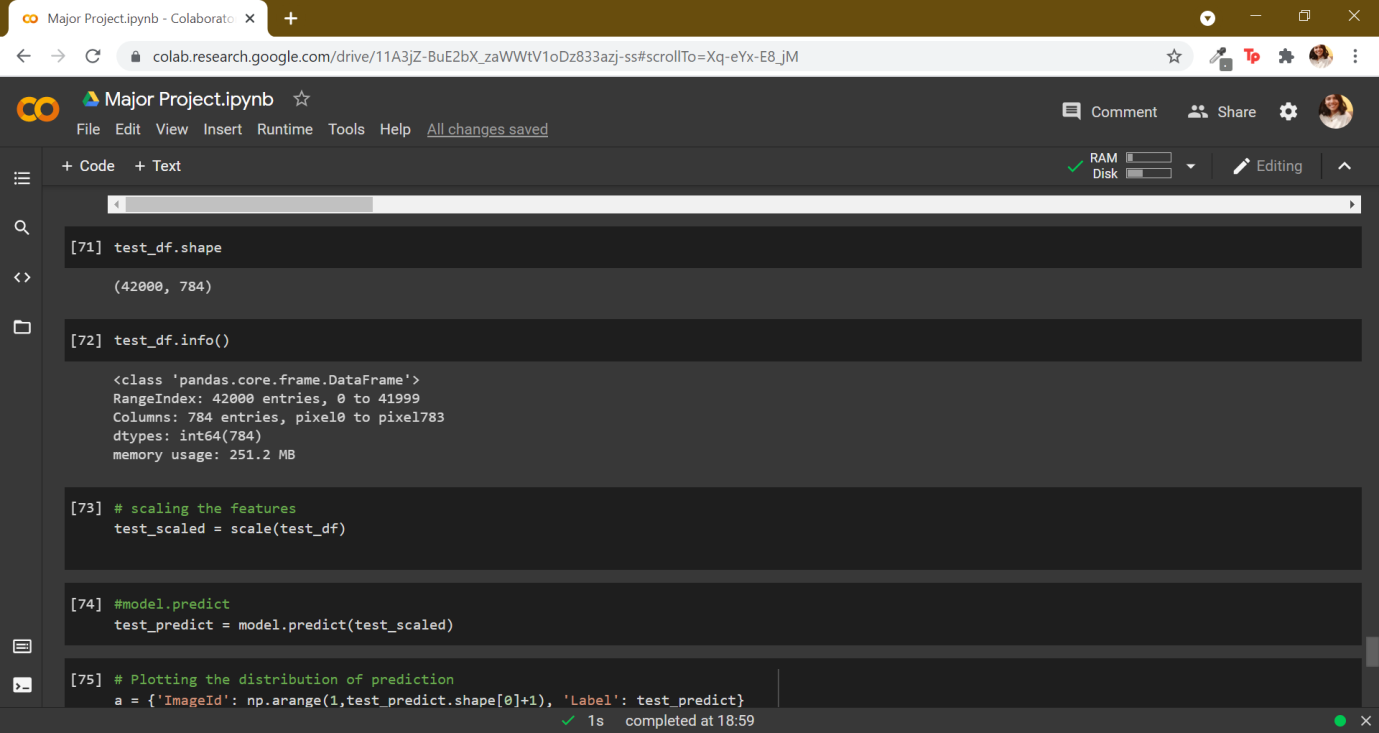


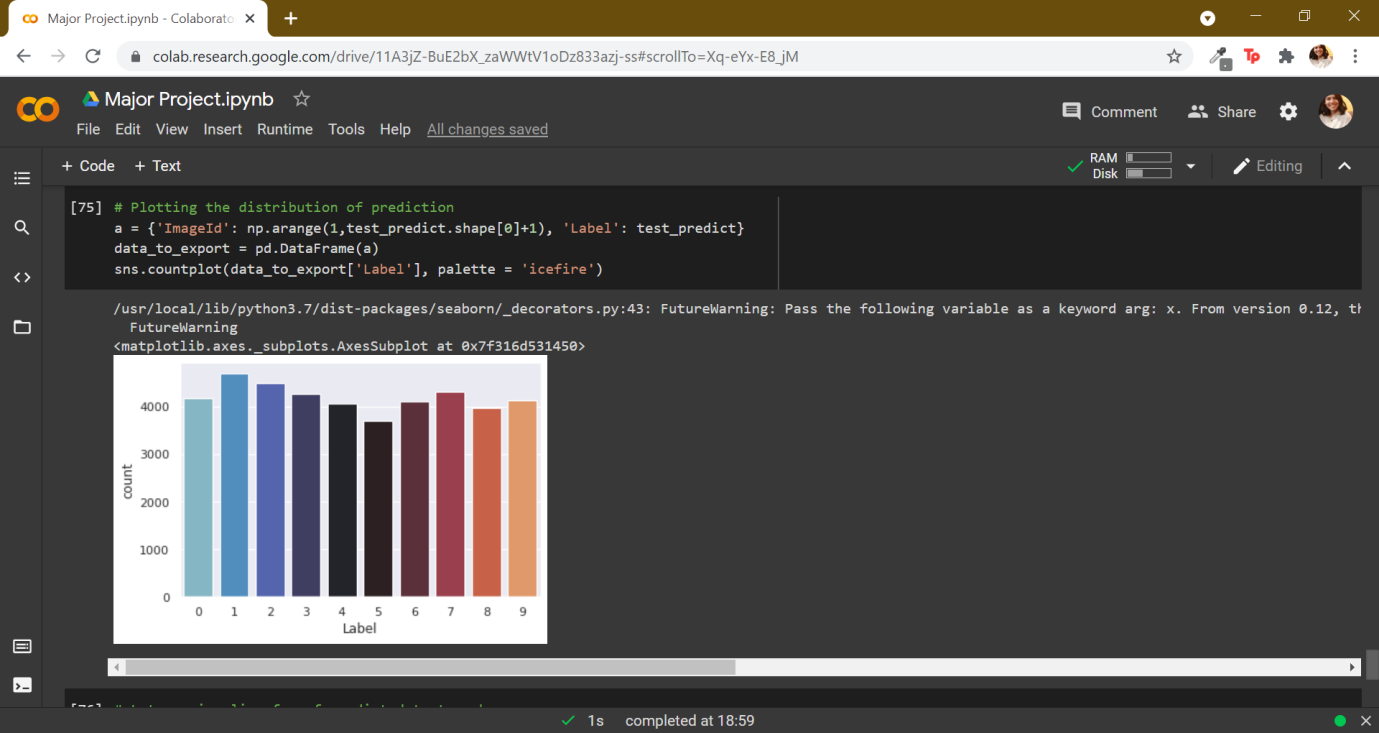


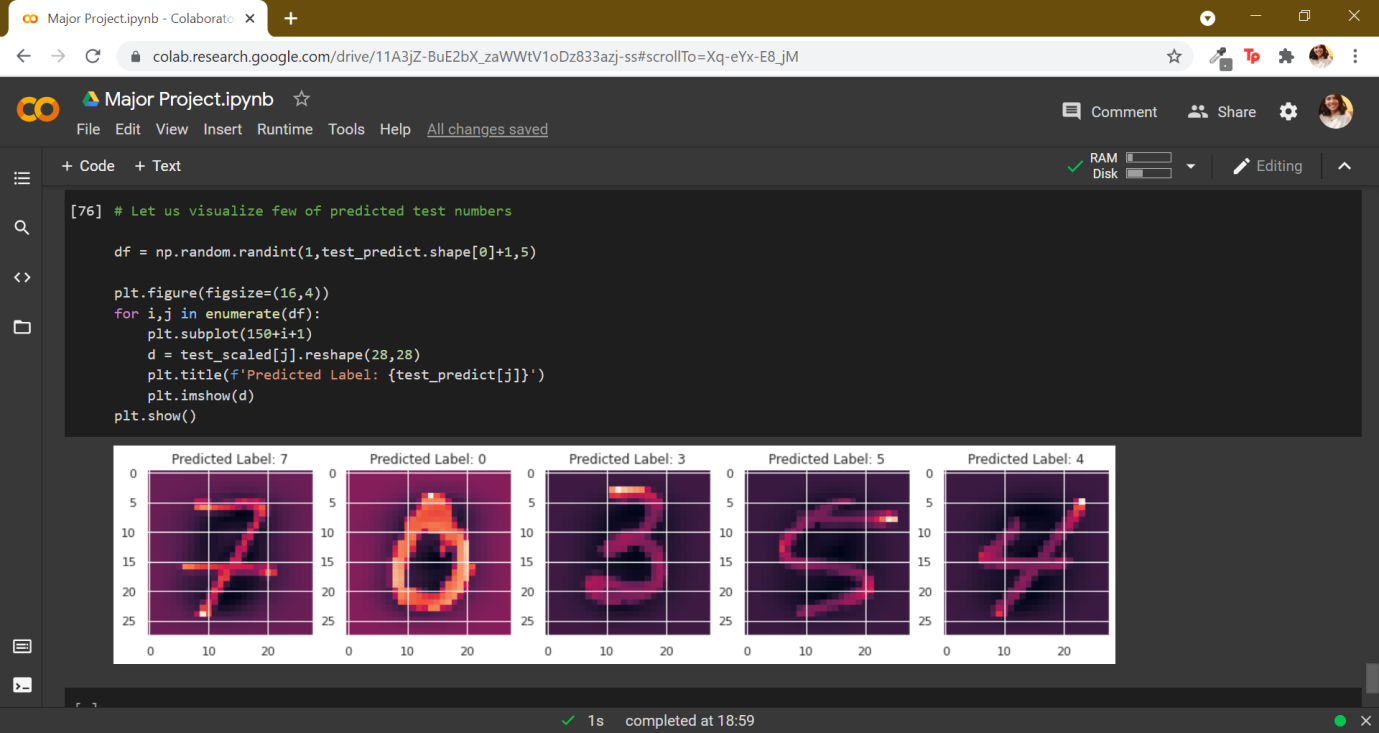


1. We can now test our model and visualise the data. For clearer information about the dataset we also plotted a bar graph to see the distribution of prediction throughout the dataset.









**Conclusion:**

After carefully exploring the data we found out that non-linear rbf kernel model is the best suited with highest accuracy**(93.59%)** for this problem statement. We calculated the optimal accuracy score and the best hyperparameters of this model which were **94.19%** and **C: 100, gama:0.001.** Using these hyperparameter values we built our final model whose accuracy was **94.49%.**