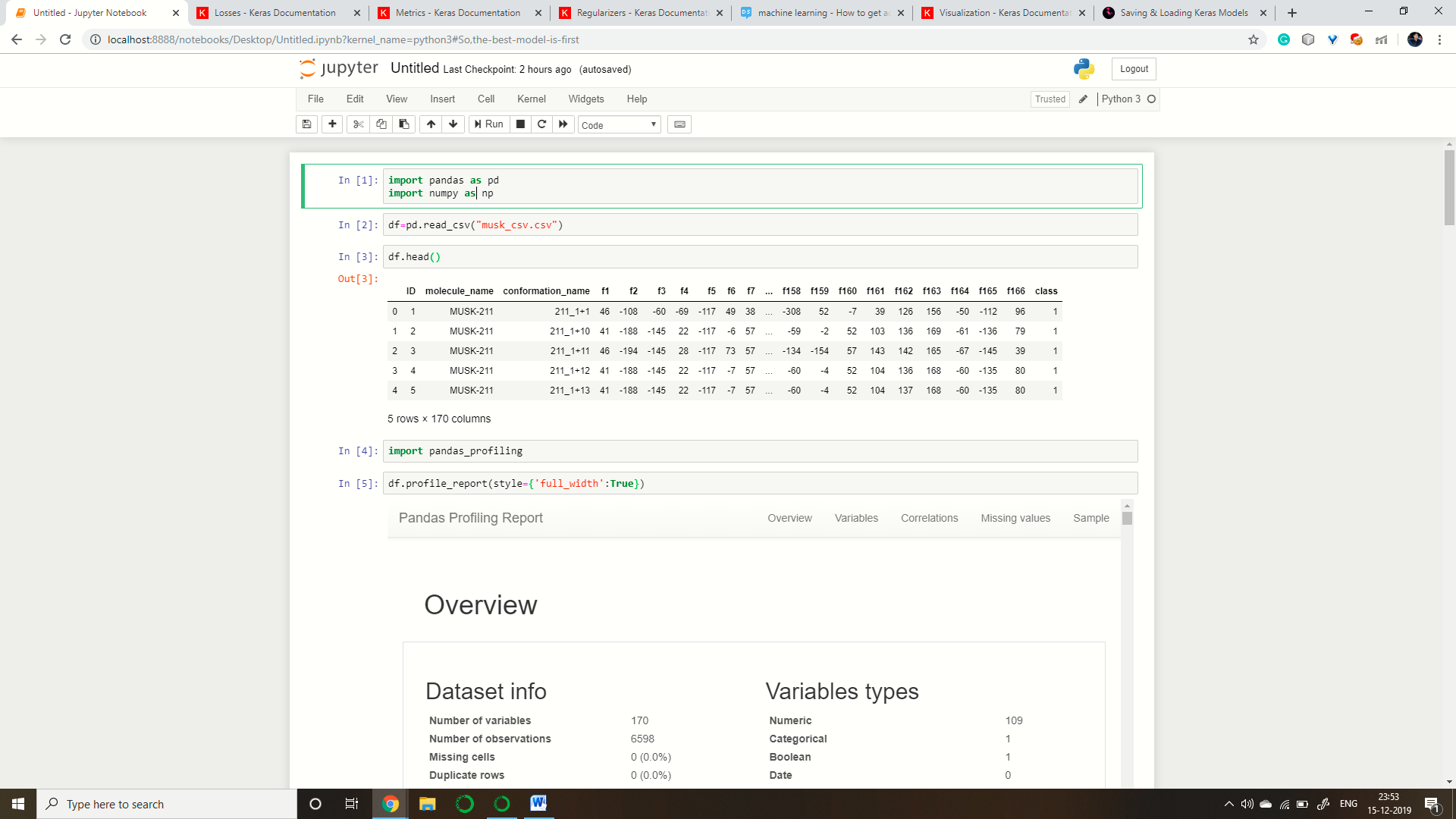
**Report**

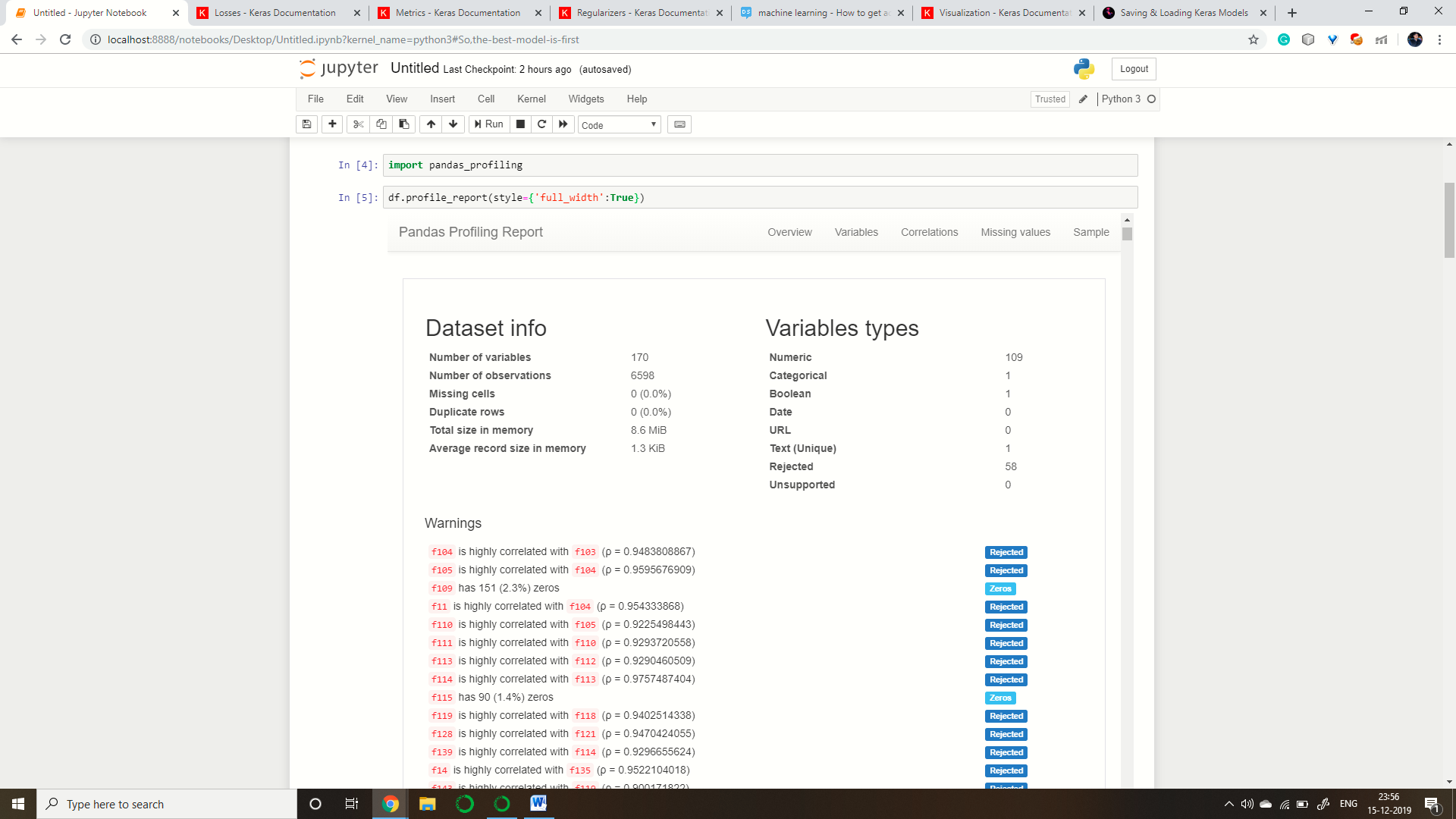
**1. Data input**

I used pandas to import the musk\_csv file



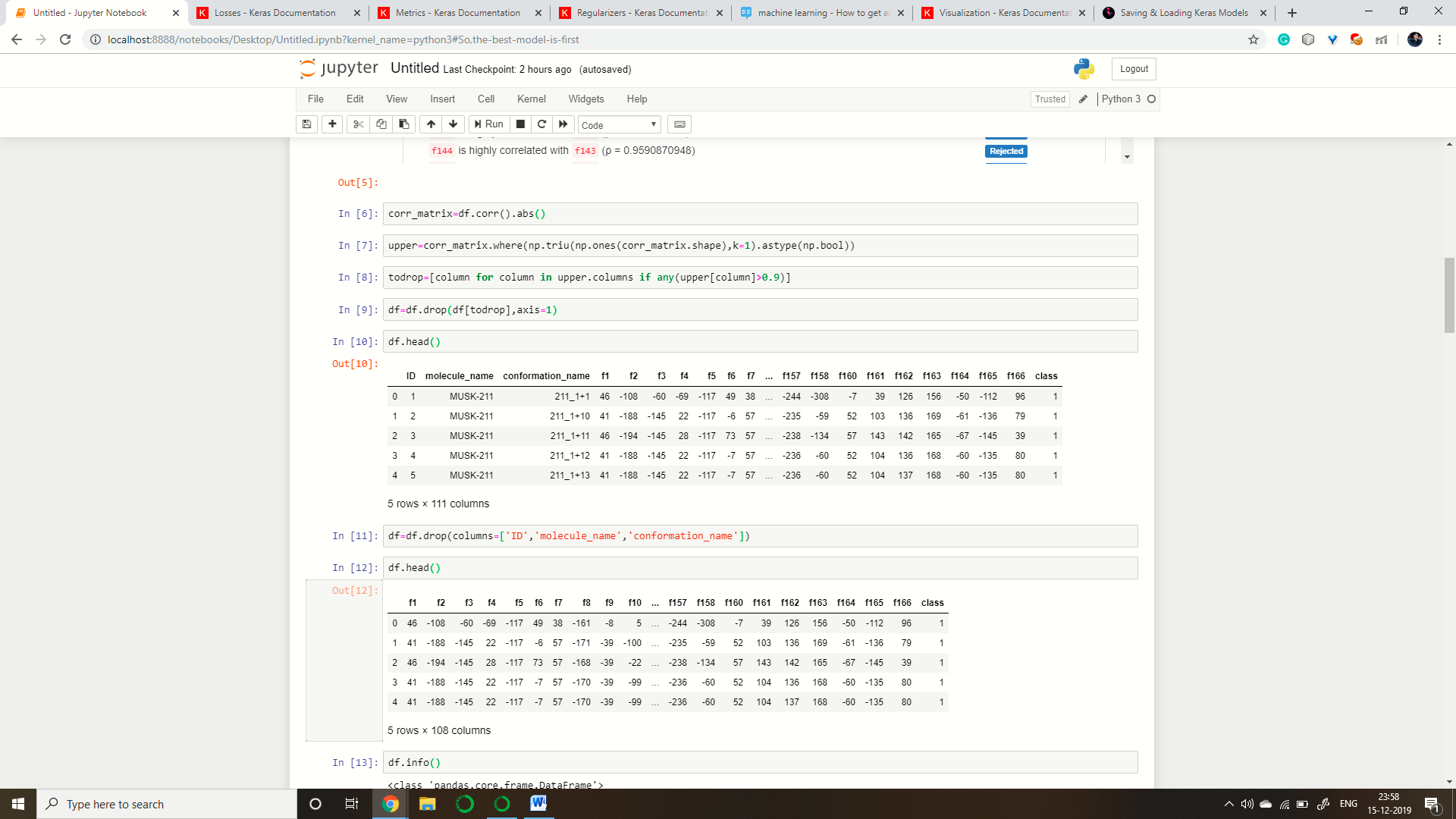
**2. Analysing the dataset**

I used pandas profiling to analyse the dataset because pandas profiling provides a deep analysation of a dataset.

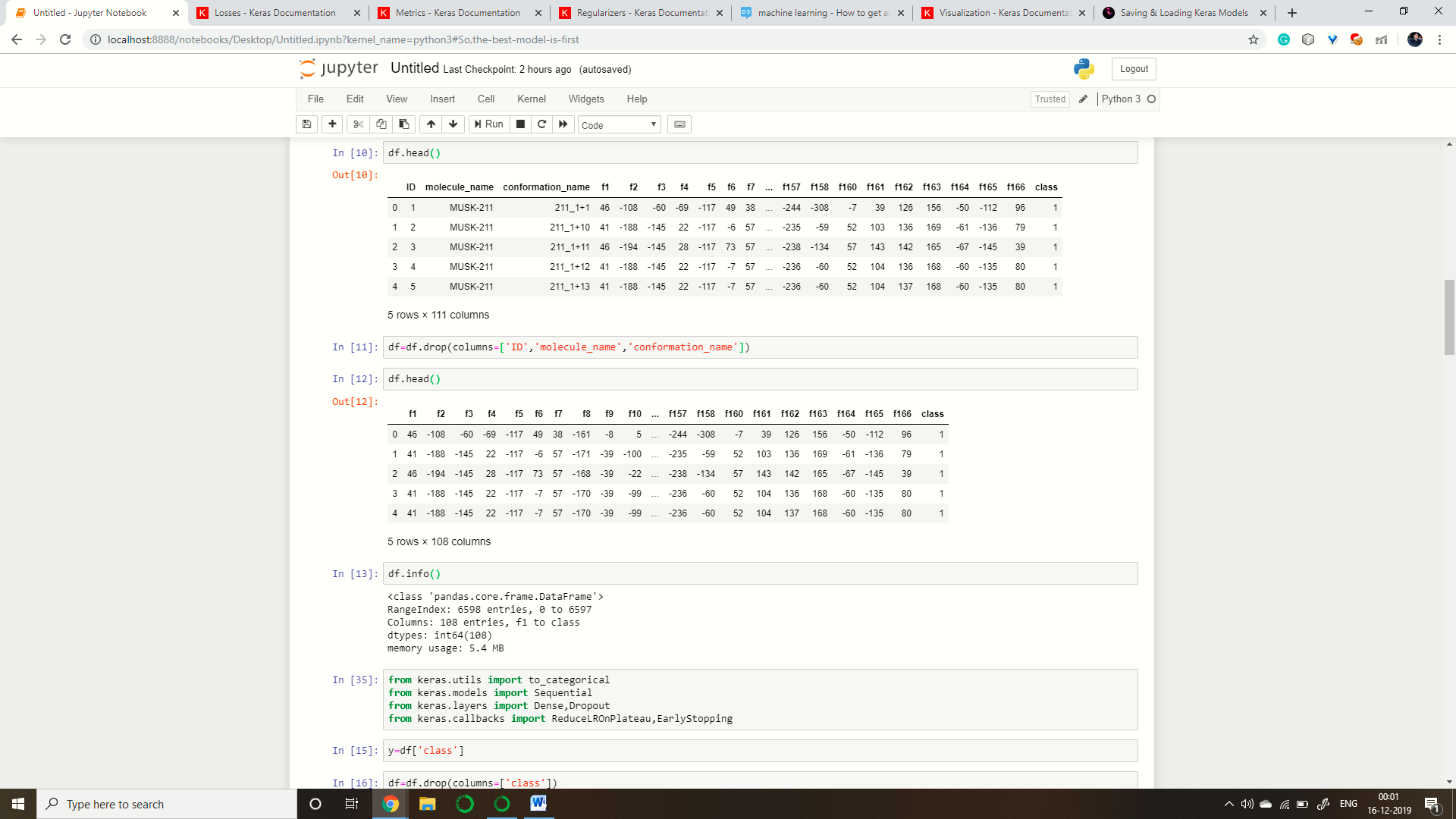


**3. Preprocessing**

In the starting I removed the columns that were highly correlated to each other because features that are highly correlated won’t give a better performance.



Then I removed the columns that weren’t important for the data as they were just names and wouldn’t contribute towards the final goal.

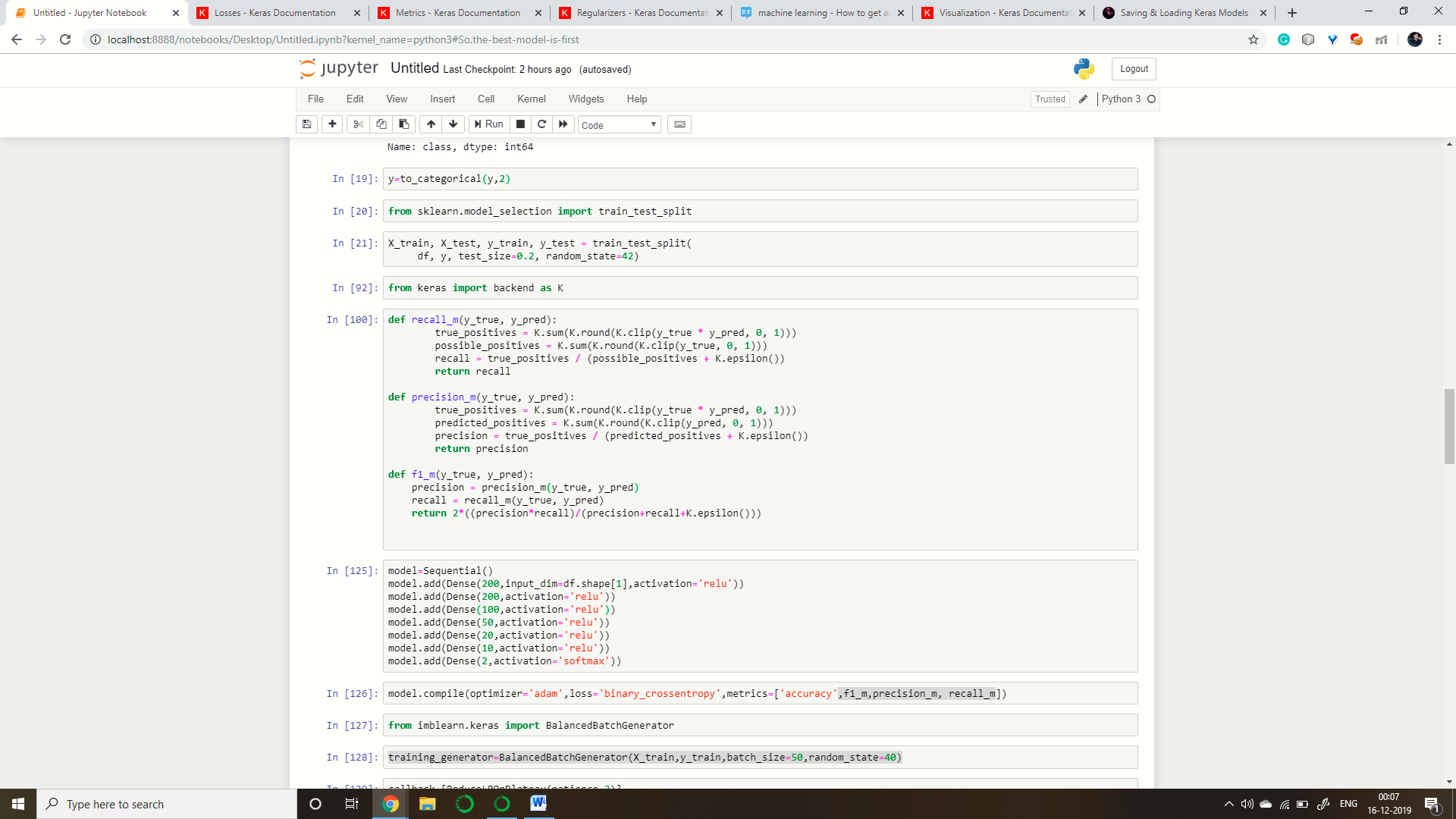


Then I analysed from pandas profiling output that if there is any missing data and it was 0% .So, there was no need to do that.

Then, I noticed that the data is imbalanced and here, we don’t want to use machine learning so we have to use neural nets so we have to resample the data.

4. Creating the Model

Now, I divided the data into train and test as asked and created recall, precision and f1score functions .

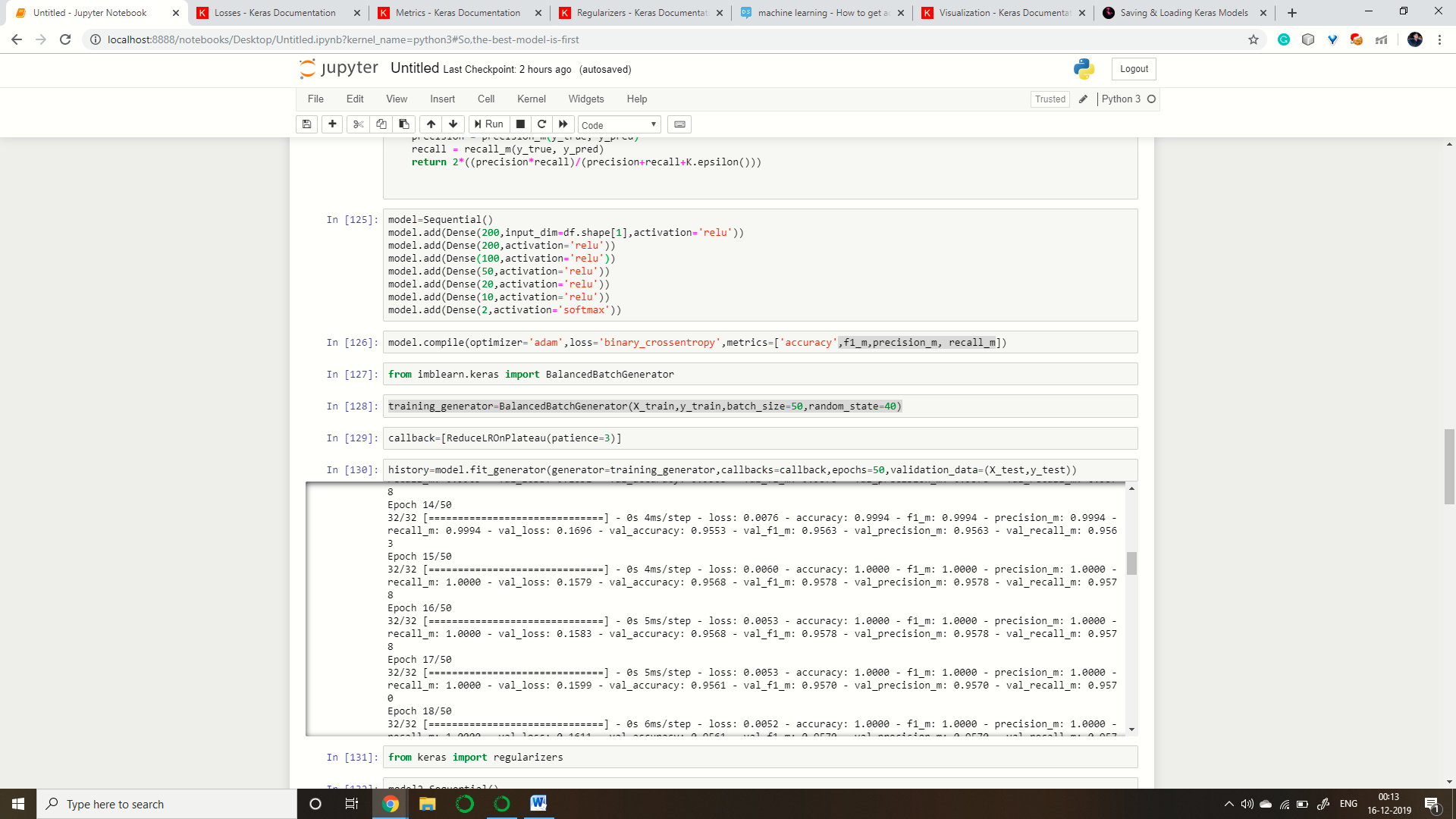


Then I created the first model and trained it. In the model I used 7 layers without any regularization .I used adam boost optimizer as it is the best optimizer these days. I used binary cross entropy for the loss and different matrices to check the model.

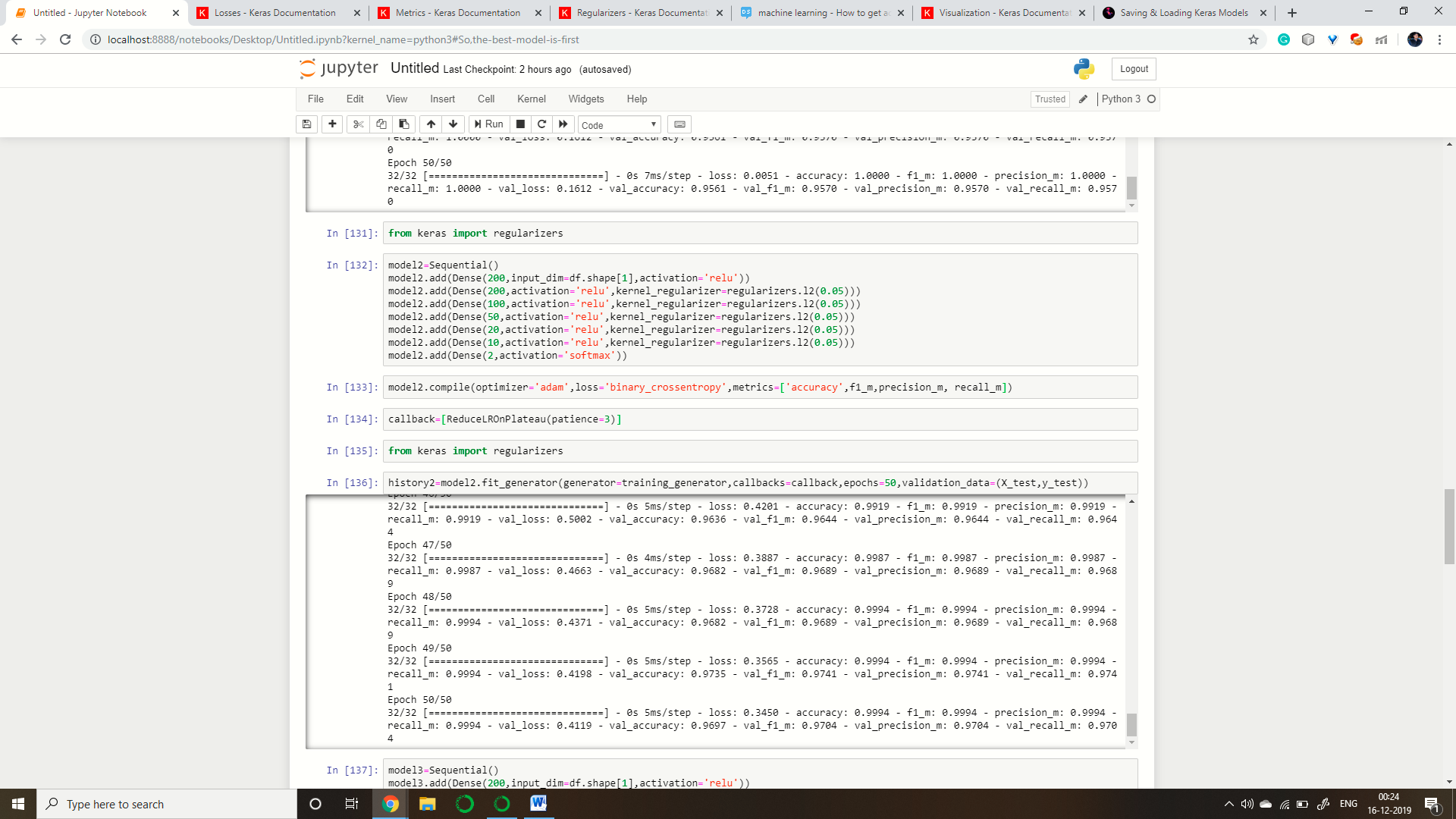
Then I created a data generator using imblearn so as to generate the data using under sampling technique so as to solve the problem of unbalanced data.

Then I created a callback function that is using reduceLROnPlateau function which reduce the learning rate to half if no progress is seen till 3 epochs so increases the performance of the model.

Then I trained the model till 50 epochs.

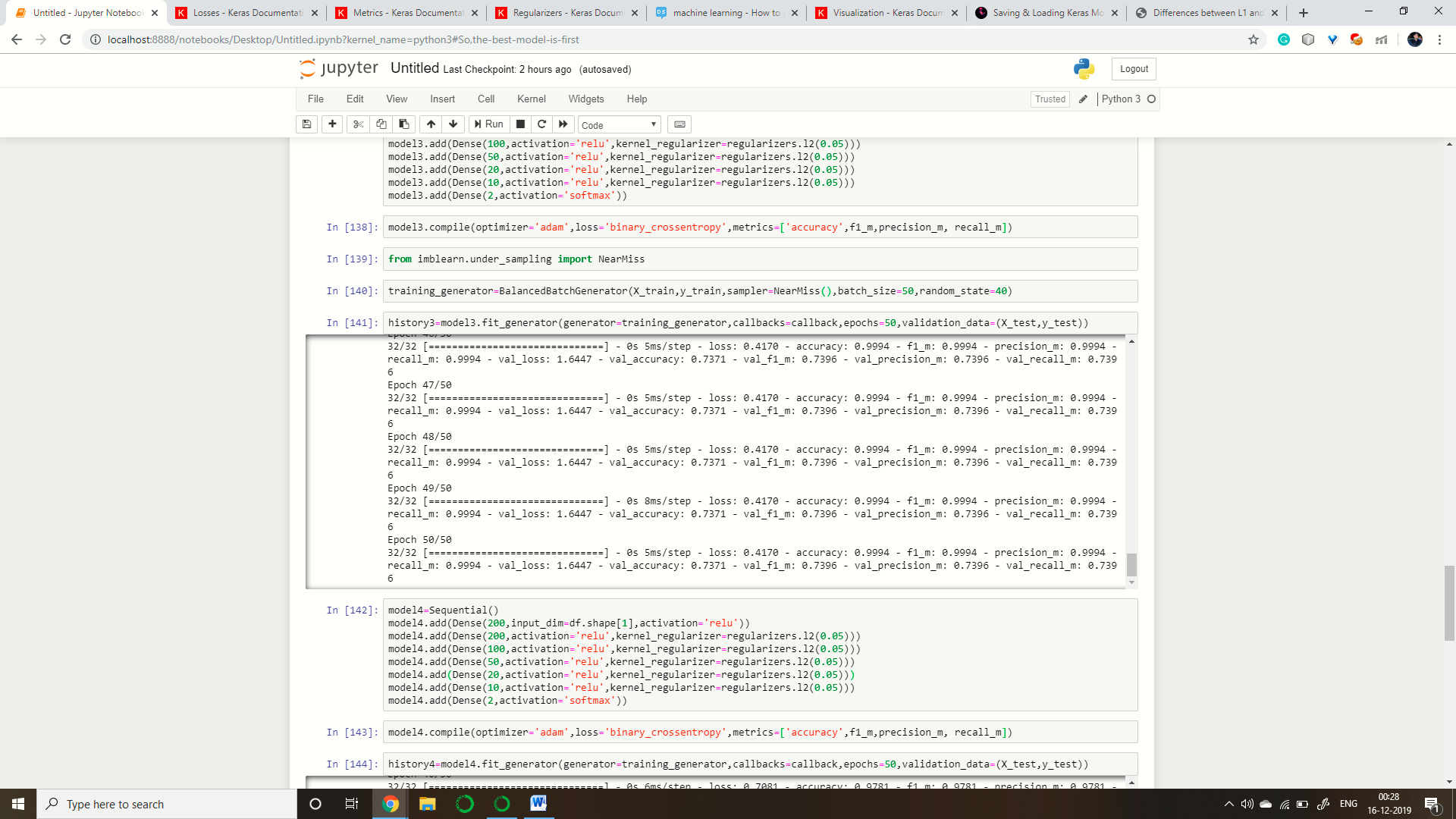


Now, the model was over fit as it was performing much better on training data as compared to test data so I created another Model with regularization.

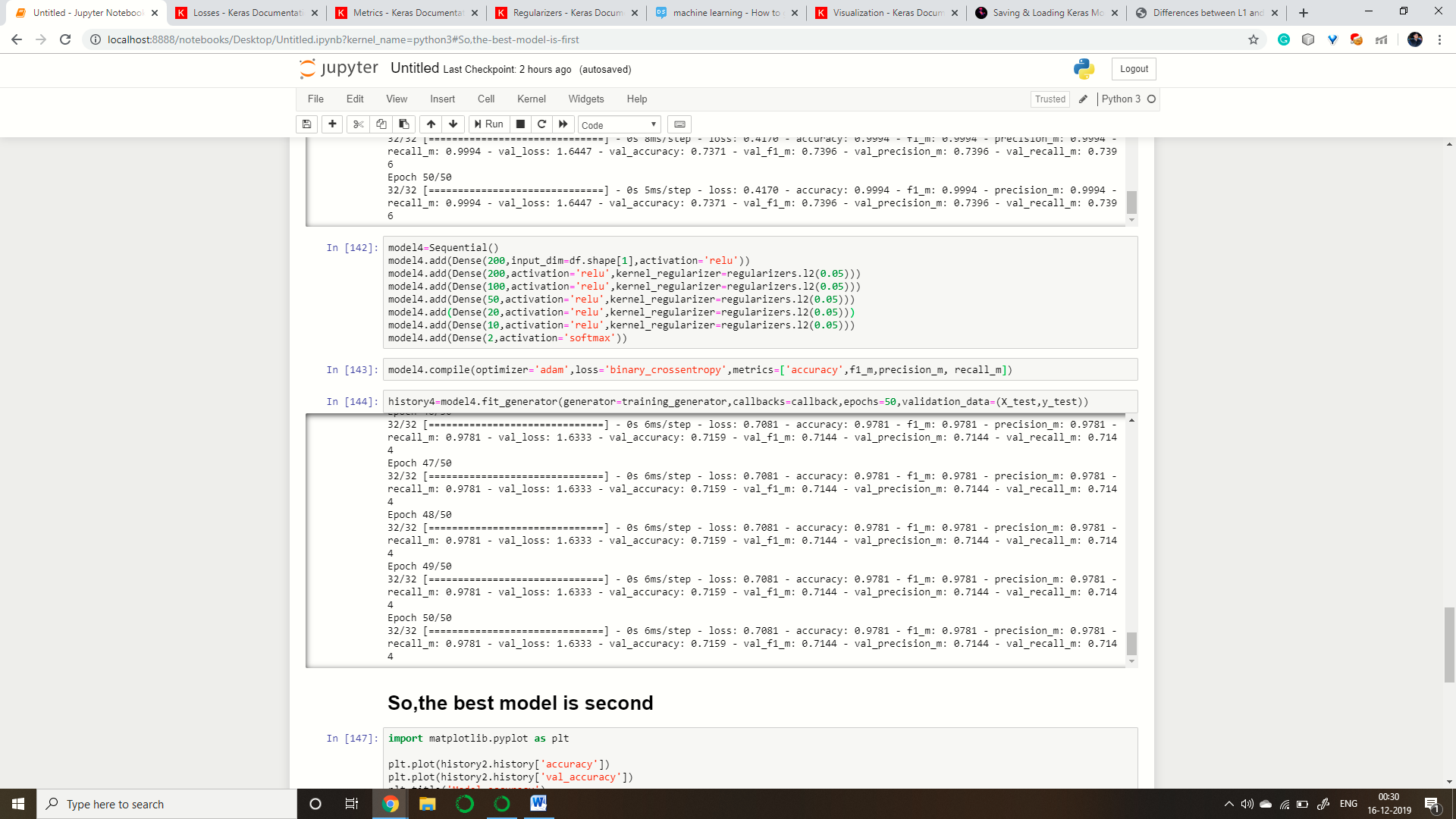


This model outperformed the previous model it had l2 regularization as the outputs of l2 regularization are more stable and l2 is quite good with non-sparse data.

Then I trained the model with different sampling technique called near miss in the hope of better output.



Then I regularized the model and trained again.



**5. Comparing the Models**

Then I compared the 4 Models

1. For First Model:

val\_loss: 0.1612

val\_accuracy: 0.9561

val\_f1\_m: 0.9570

val\_precision\_m: 0.9570

val\_recall\_m: 0.9570

2. For Second Model:

val\_loss: 0.4119

val\_accuracy: 0.9697

val\_f1\_m: 0.9704

val\_precision\_m: 0.9704

val\_recall\_m: 0.9704

3. For Third Model:

val\_loss: 1.6447

val\_accuracy: 0.7371

val\_f1\_m: 0.7396

val\_precision\_m: 0.7396

val\_recall\_m: 0.7396

4. For Fourth Model:

val\_loss: 1.6333

val\_accuracy: 0.7159

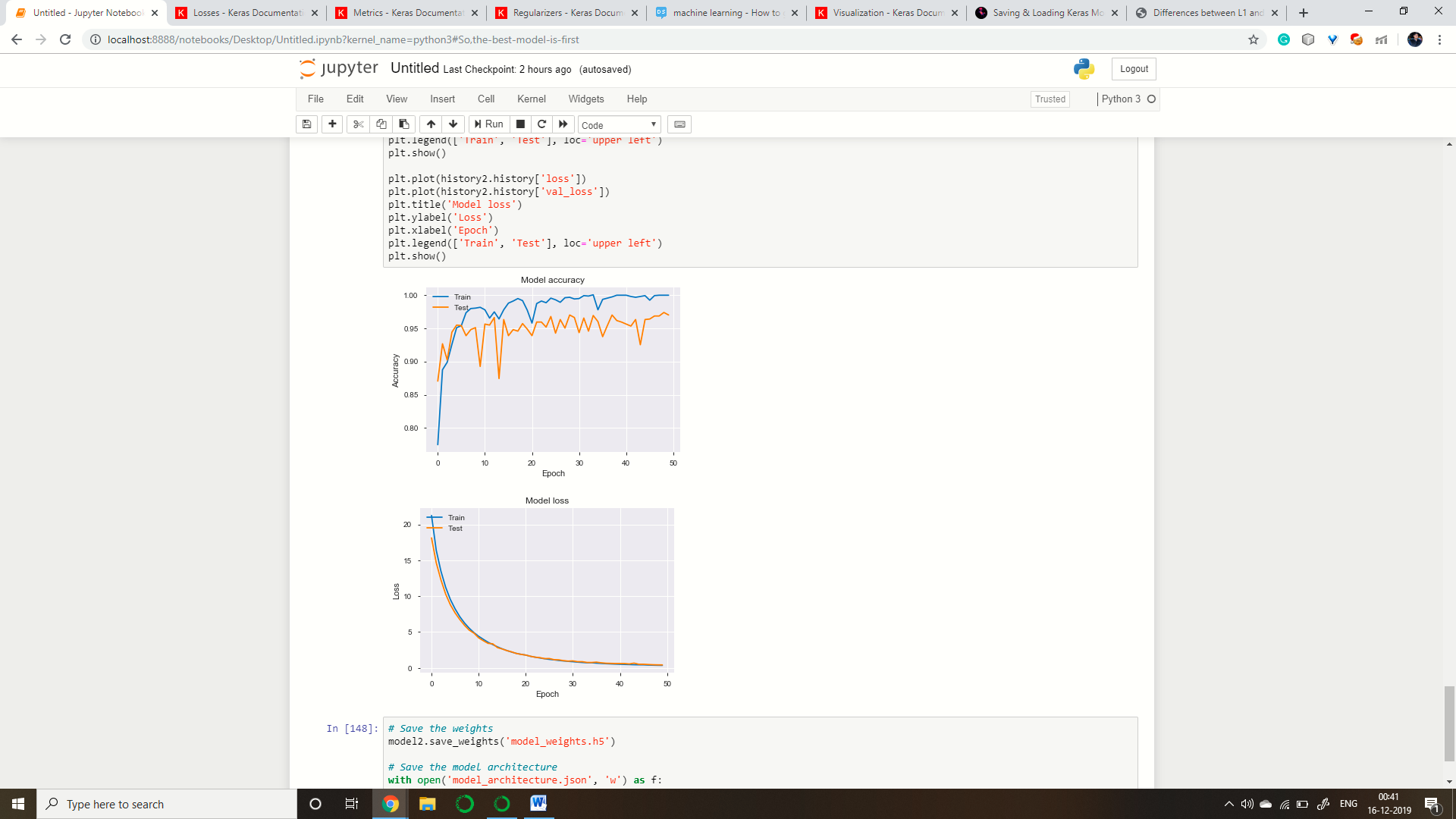
val\_f1\_m: 0.7144

val\_precision\_m: 0.7144

val\_recall\_m: 0.7144

On comparison second model outperformed all the others.

**6. Plotting Loss and Accuracy Graph**



**7. Saving the Model**

Finally, I saved the architecture of the model and the model.

