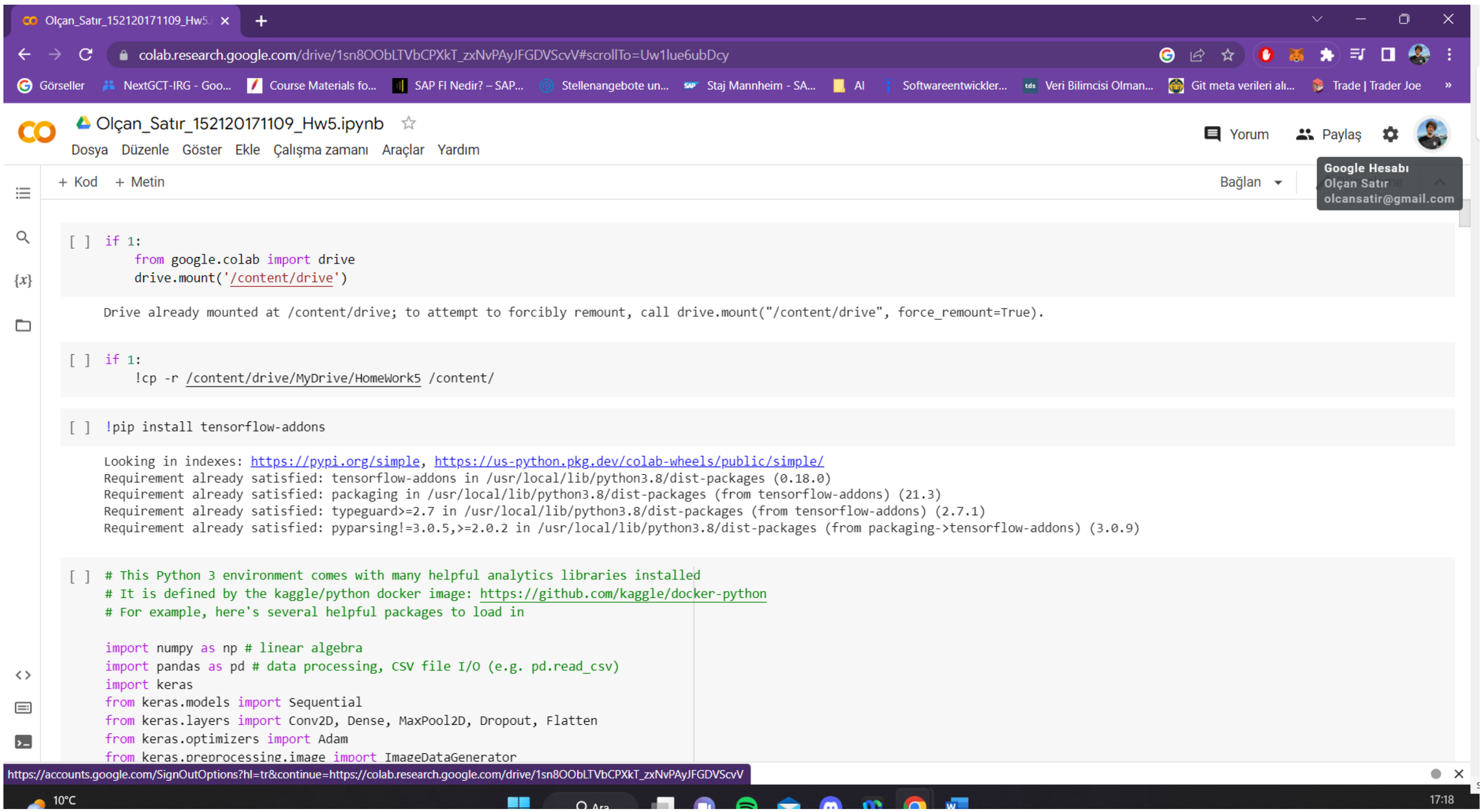
OLÇAN SATIR

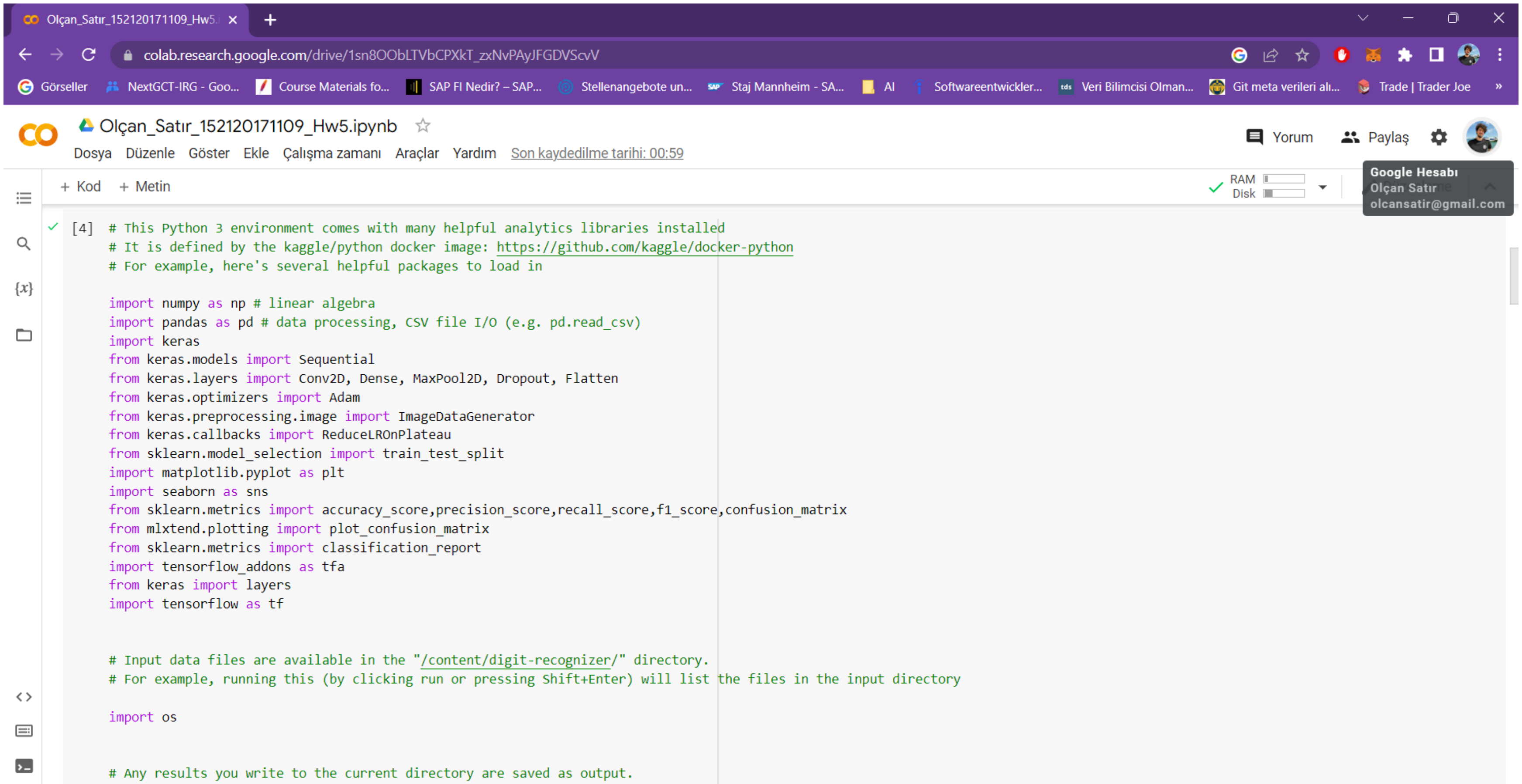
152120171109

Deep Learning HW-5

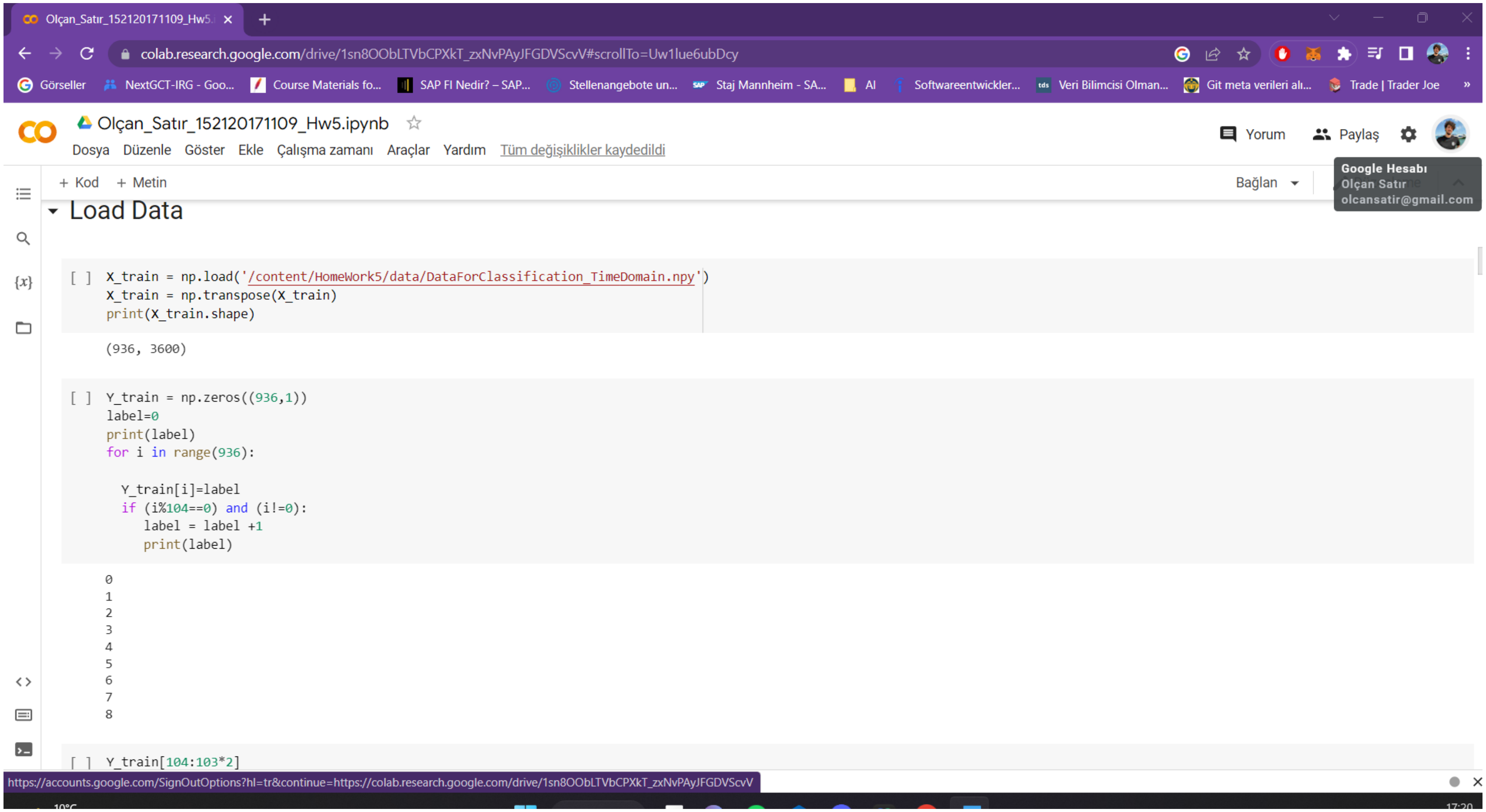
We are establishing our colab connection.



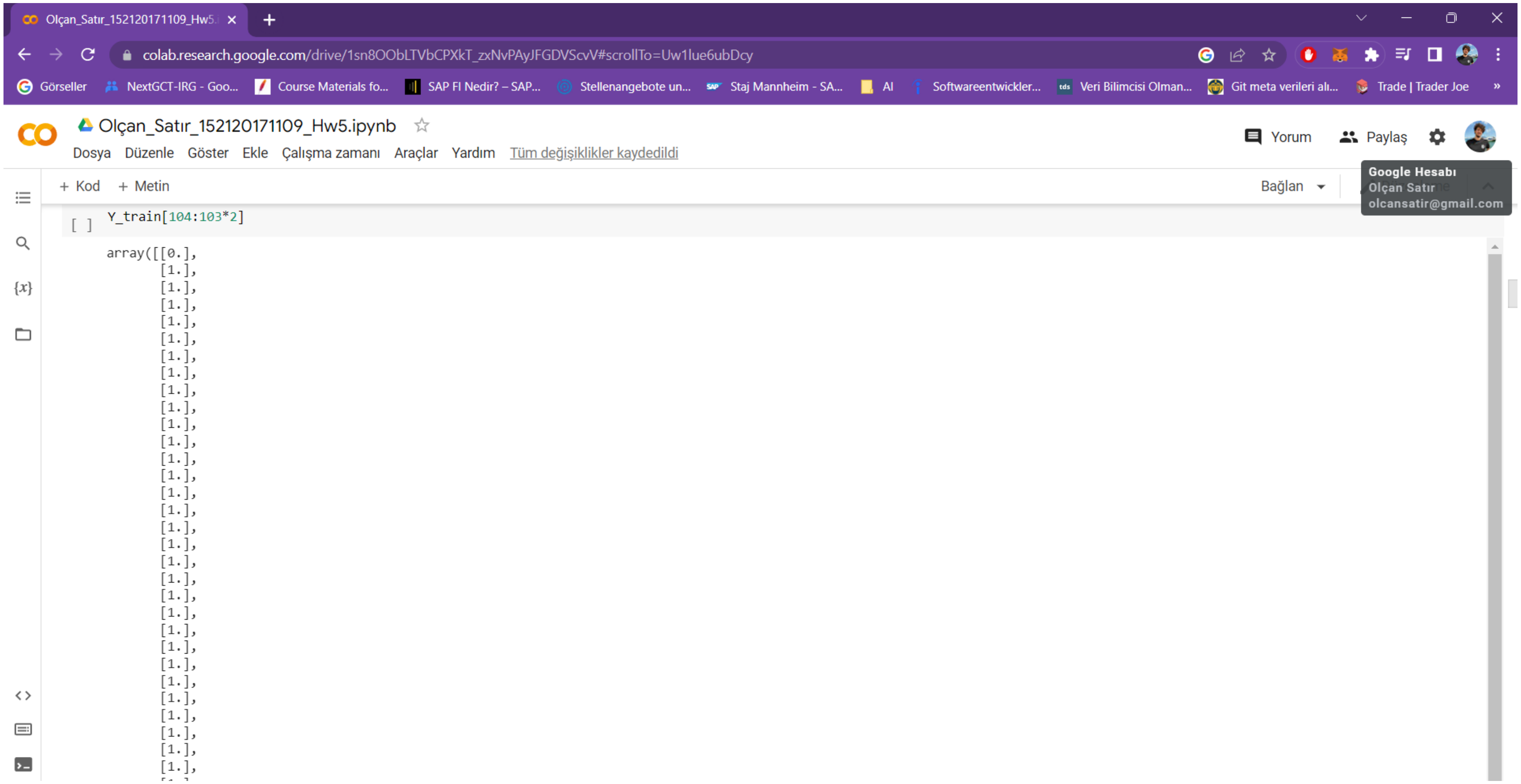
We define our libraries.



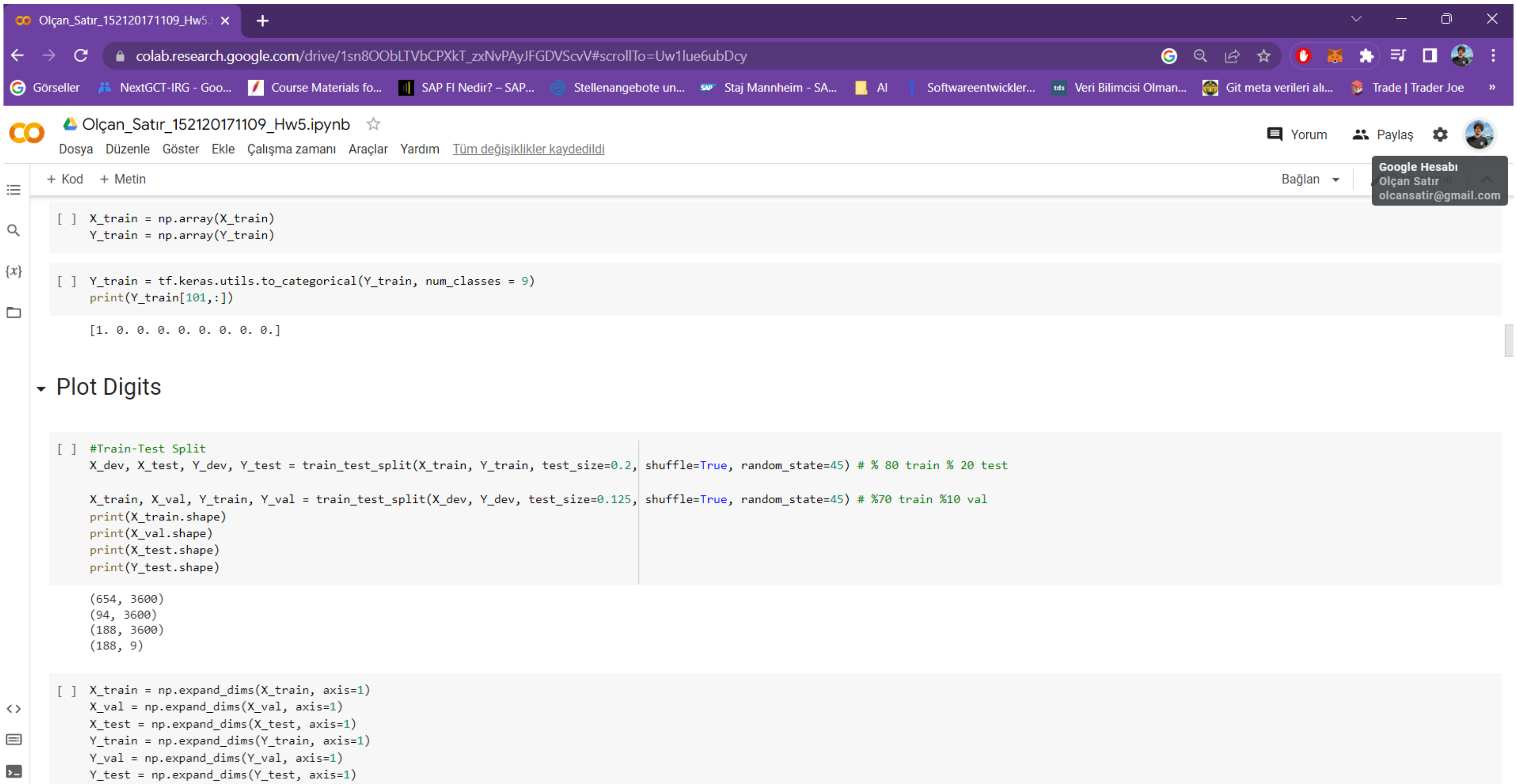
With the help of the for loop, we fill the Y_train and print our labels. For this, our i value must be divisible by 104 and equal to 0. When this condition is met, we increase our label value by 1 and suppress it.



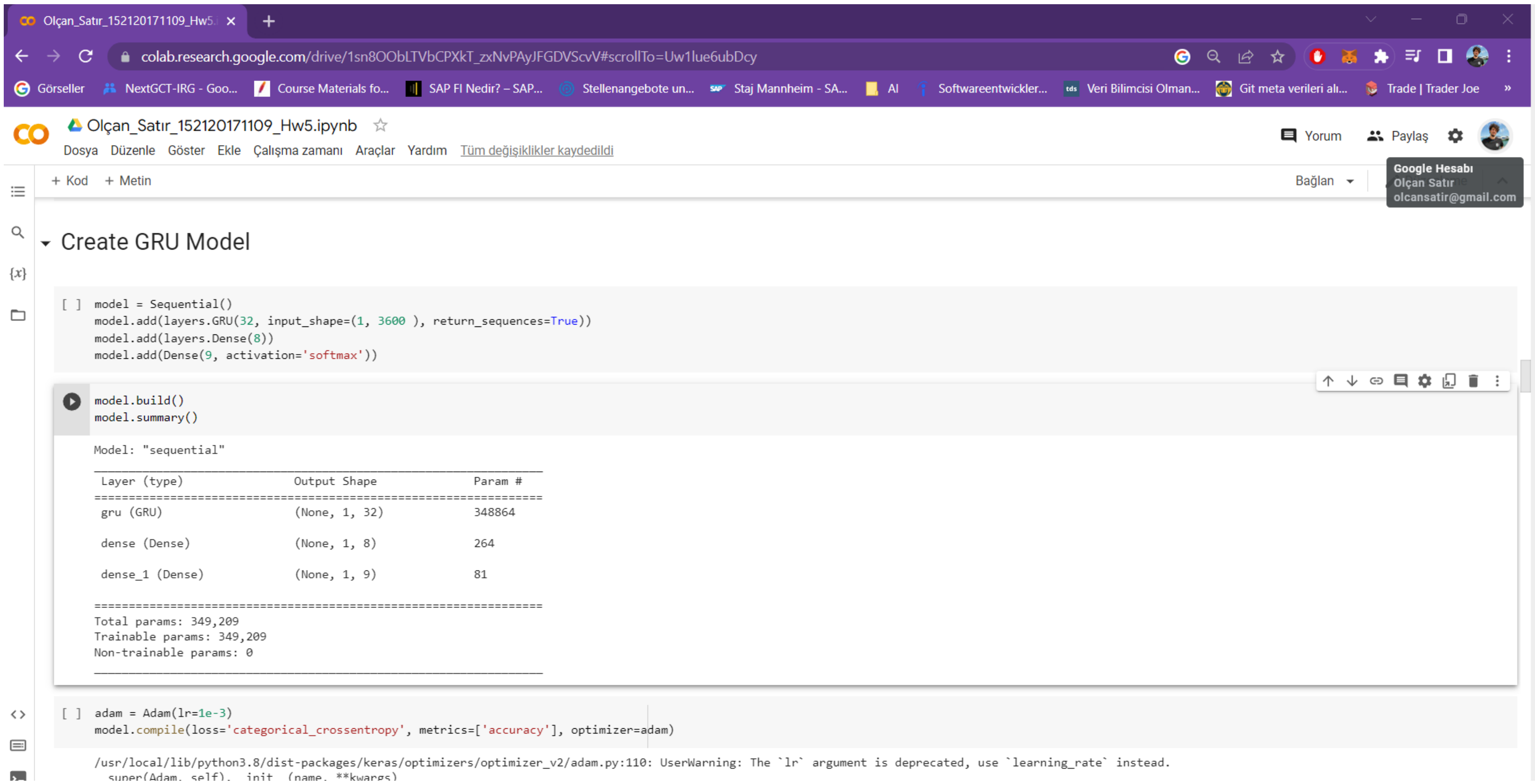
Here we print the values from index 104 to index 206 of the Y_train data. (not including 206)



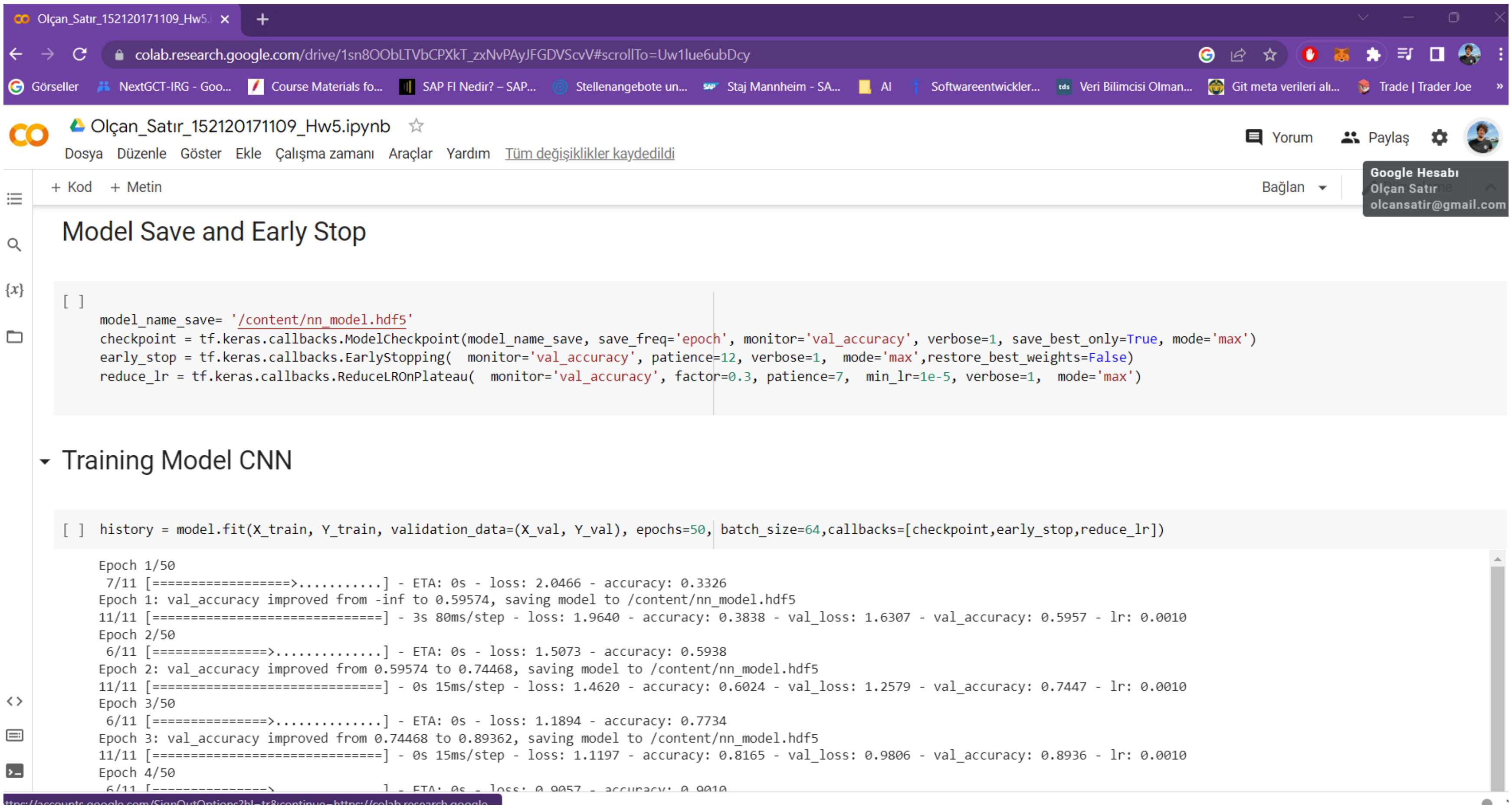
Here we convert X_train and Y_train to array type. Then we make our Y_train data with labels categorically as 9 classes. Then we print our Y_train data 101 and beyond. Afterwards, we first separate our data into 80% train and 20% test. Afterwards, we allocate 12.5% of this 80% train data as our validation data. In the last case, 70% of our data is reserved as train, 20% as test and 10% as validation.



Here we create our GRU model. Then we add Dense. Since we have 9 classes, we give the value 9. Then we compile our model, give the loss function of our model as categorical_crosssentropy and we used Adam as the optimization algorithm. Our learning rate: 1e-3

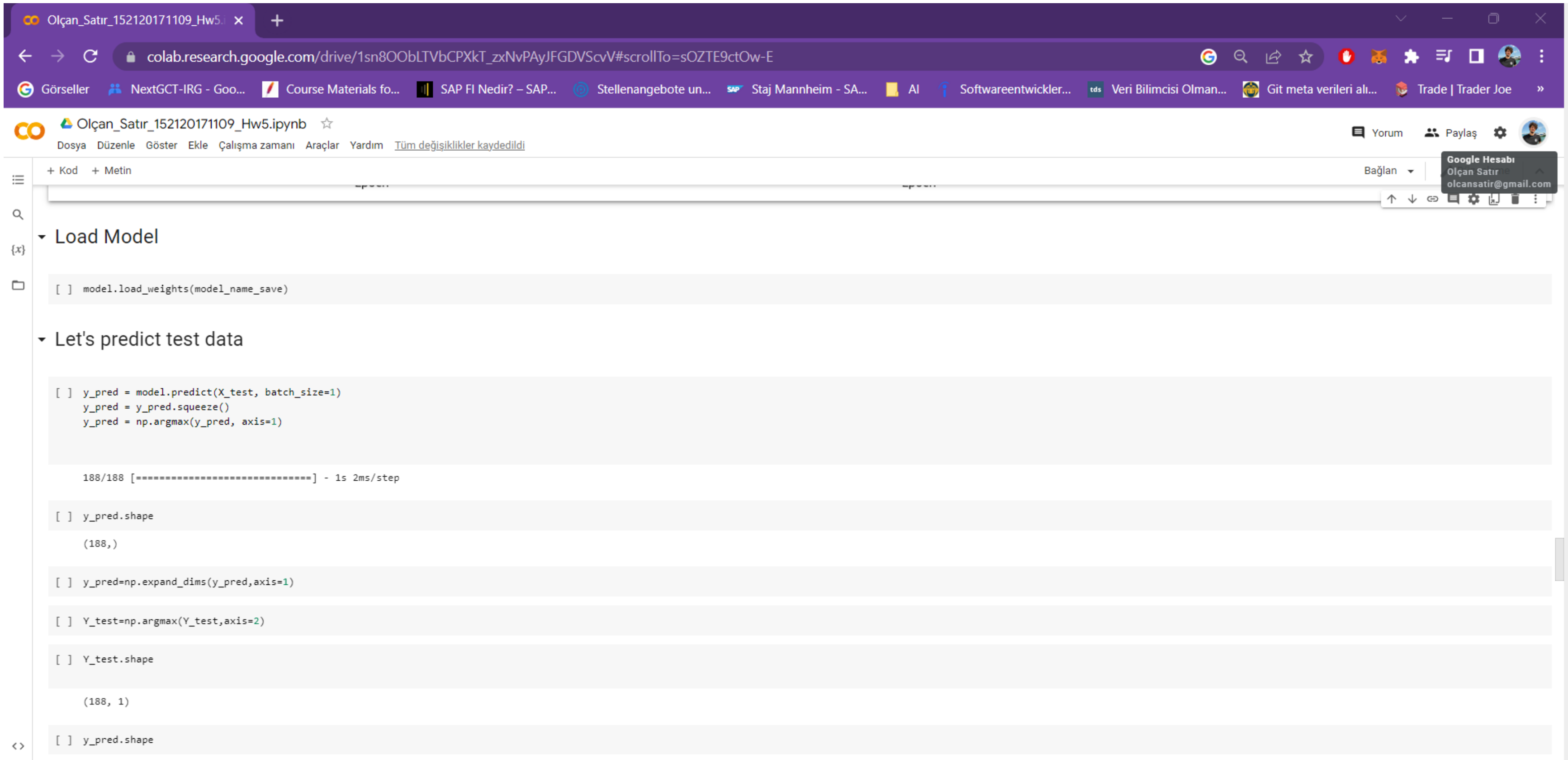


We used early stop here. If our model does not improve accuracy 12 times in a row, it stops and saves the model. Here, we work with an early stop in the 21st training for our model, which we run with 50 epochs, and save the model.

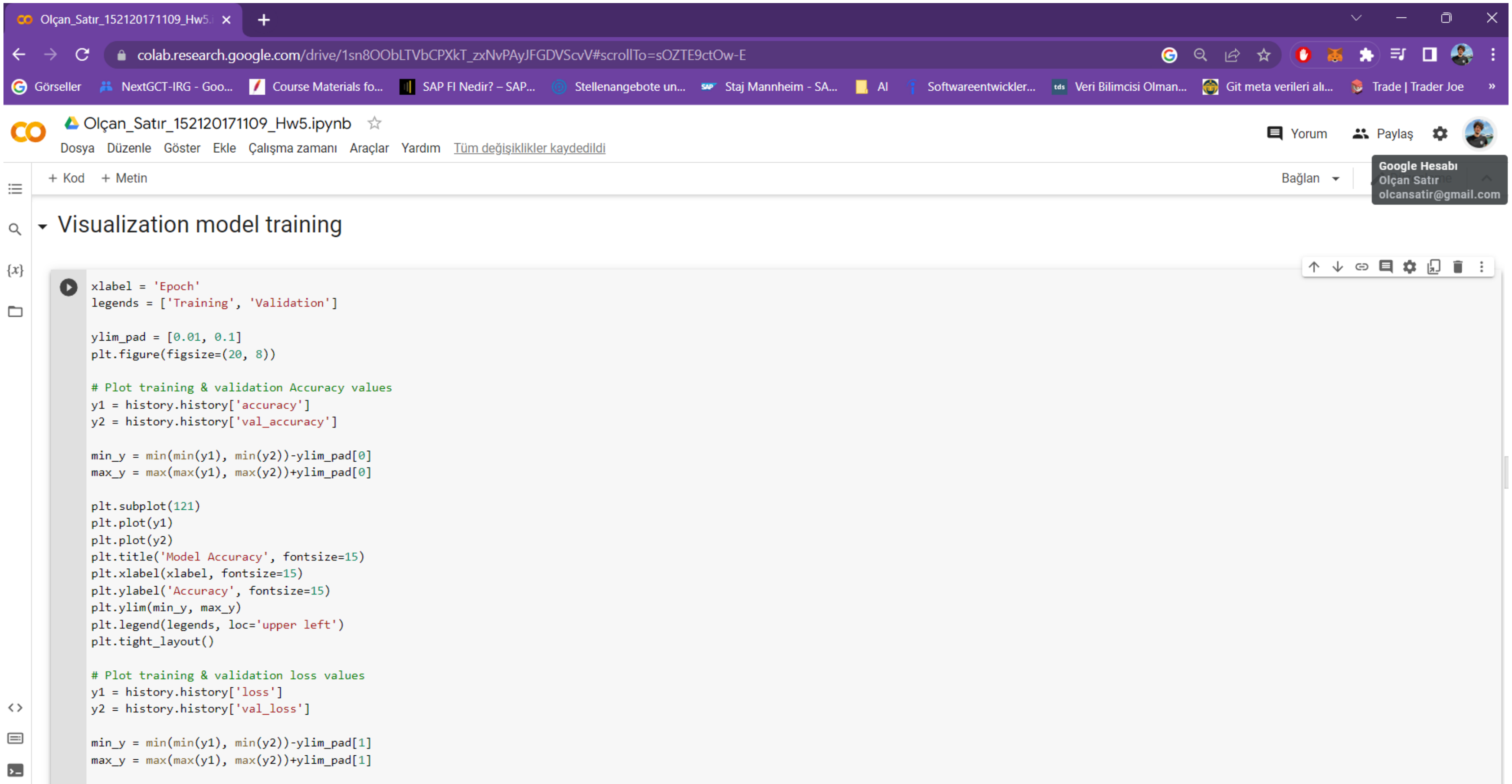


```
11/11 [==============================] - 0s 12ms/step - loss: 0.0614 - accuracy: 0.9939 - val_loss: 0.1412 - val_accuracy: 0.9681 - lr: 3.0000e-04
Epoch 19/50
 7/11 [==================>...........] - ETA: 0s - loss: 0.0519 - accuracy: 0.9978
Epoch 19: val_accuracy did not improve from 0.96809
11/11 [==============================] - 0s 12ms/step - loss: 0.0590 - accuracy: 0.9939 - val_loss: 0.1391 - val_accuracy: 0.9681 - lr: 3.0000e-04
Epoch 20/50
 7/11 [==================>...........] - ETA: 0s - loss: 0.0474 - accuracy: 0.9978
Epoch 20: val_accuracy did not improve from 0.96809
11/11 [==============================] - 0s 13ms/step - loss: 0.0568 - accuracy: 0.9939 - val_loss: 0.1378 - val_accuracy: 0.9681 - lr: 3.0000e-04
Epoch 21/50
 7/11 [==================>...........] - ETA: 0s - loss: 0.0558 - accuracy: 0.9933
Epoch 21: val_accuracy did not improve from 0.96809
11/11 [==============================] - 0s 12ms/step - loss: 0.0547 - accuracy: 0.9939 - val_loss: 0.1362 - val_accuracy: 0.9681 - lr: 3.0000e-04
Epoch 21: early stopping
```

Here we load our model that we recorded using early stopping and have it predicted. Then we make our model fit by adding a dimension because there is a mismatch in the dimensions.



Here we plot the accuracy and loss values of our model's train and validation data.



Olçan_Satır_152120171109_Hw5.

colab.research.google.com/drive/1sn8OObLTVbCPXkT_zxNvPAyJFGDVScvV#scrollTo=sOZTE9ctOw-E

Görseller | NextGCT-IRG - Goo... | Course Materials fo... | SAP FI Nedir? – SAP... | Stellenangebote un... | Staj Mannheim - SA... | AI | Softwareentwickler... | Veri Bilimcisi Olman... | Git meta verileri alı... | Trade | Trader Joe

Olçan_Satır_152120171109_Hw5.ipynb

Dosya Düzenle Göster Ekle Çalışma zamanı Araçlar Yardım Tüm değişiklikler kaydedildi

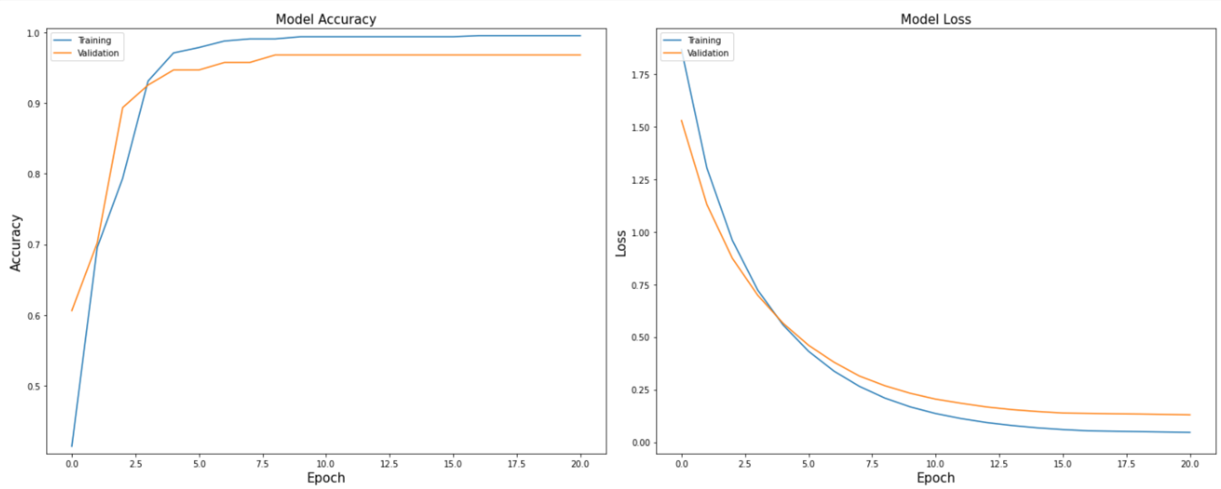
Yorum Paylaş

Google Hesabı
Olçan Satır
olcansatir@gmail.com

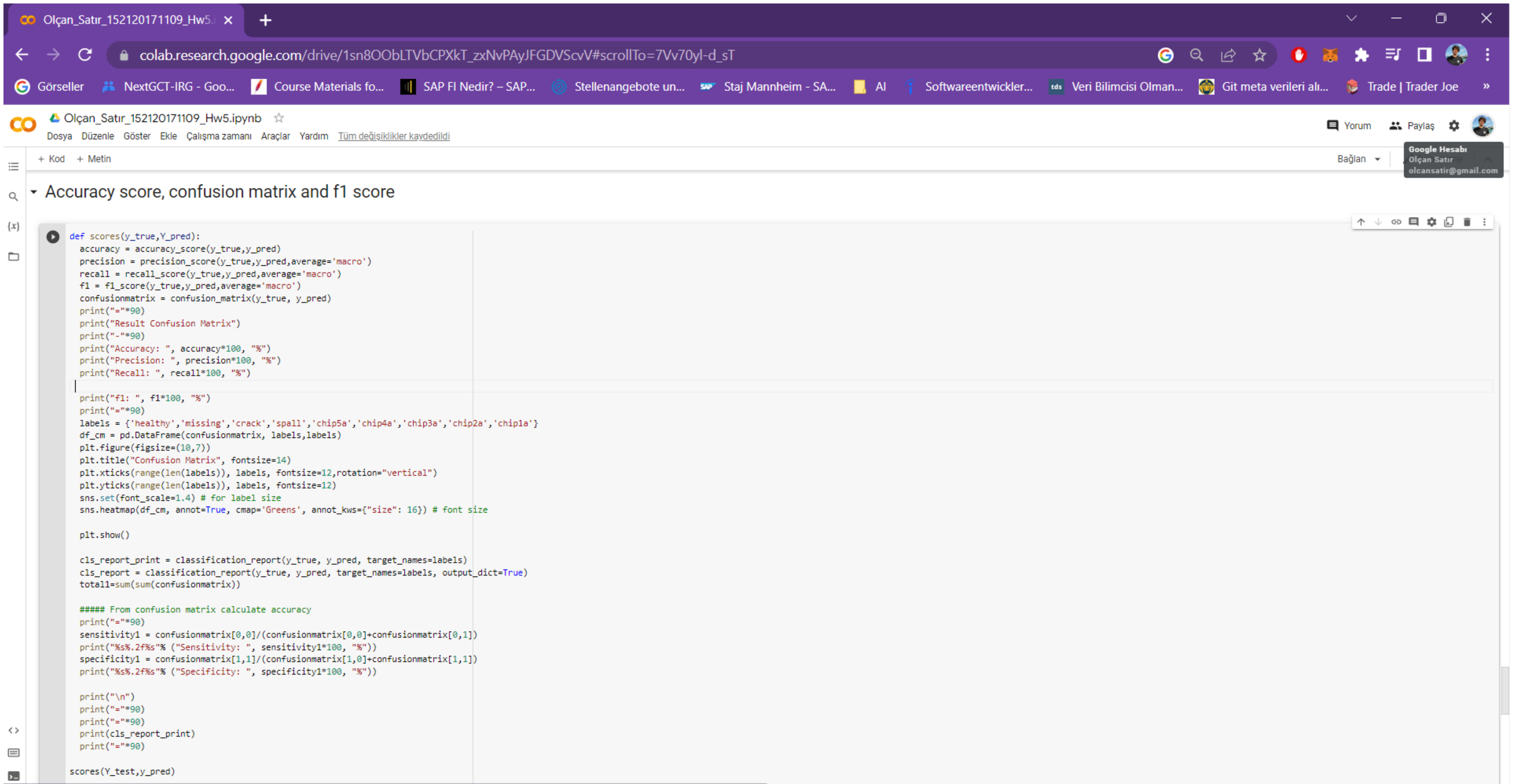
+ Kod + Metin

RAM
Disk

```python
plt.subplot(122)
plt.plot(y1)
plt.plot(y2)
plt.title('Model Loss', fontsize=15)
plt.xlabel(xlabel, fontsize=15)
plt.ylabel('Loss', fontsize=15)
plt.ylim(min_y, max_y)
plt.legend(legends, loc='upper left')

plt.tight_layout()
plt.show()
```

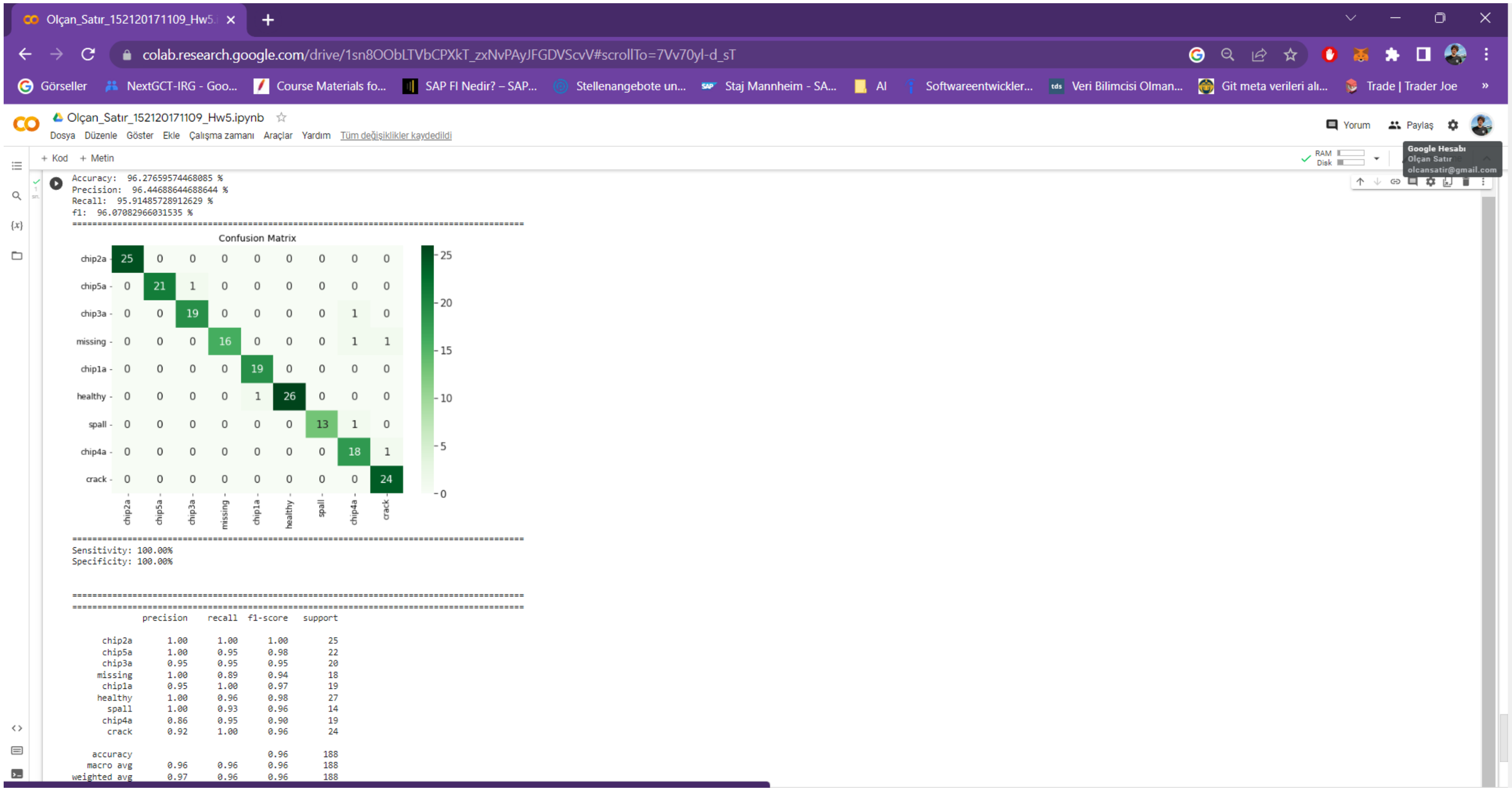


Here we suppress our accuracy, precision, recall and f1 scores within the def scores function. Then we create our confusion matrix and give our labels. We use the seaborn library as requested.



Accuracy score, confusion matrix and f1 score

```
def scores(y_true,Y_pred):
  accuracy = accuracy_score(y_true,y_pred)
  precision = precision_score(y_true,y_pred,average='macro')
  recall = recall_score(y_true,y_pred,average='macro')
  f1 = f1_score(y_true,y_pred,average='macro')
  confusionmatrix = confusion_matrix(y_true, y_pred)
  print("="*90)
  print("Result Confusion Matrix")
  print("-"*90)
  print("Accuracy: ", accuracy*100, "%")
  print("Precision: ", precision*100, "%")
  print("Recall: ", recall*100, "%")

  print("f1: ", f1*100, "%")
  print("="*90)
  labels = {'healthy','missing','crack','spall','chip5a','chip4a','chip3a','chip2a','chip1a'}
  df_cm = pd.DataFrame(confusionmatrix, labels,labels)
  plt.figure(figsize=(10,7))
  plt.title("Confusion Matrix", fontsize=14)
  plt.xticks(range(len(labels)), labels, fontsize=12,rotation="vertical")
  plt.yticks(range(len(labels)), labels, fontsize=12)
  sns.set(font_scale=1.4) # for label size
  sns.heatmap(df_cm, annot=True, cmap='Greens', annot_kws={"size": 16}) # font size

  plt.show()

  cls_report_print = classification_report(y_true, y_pred, target_names=labels)
  cls_report = classification_report(y_true, y_pred, target_names=labels, output_dict=True)
  total1=sum(sum(confusionmatrix))

  ##### From confusion matrix calculate accuracy
  print("="*90)
  sensitivity1 = confusionmatrix[0,0]/(confusionmatrix[0,0]+confusionmatrix[0,1])
  print("%s%.2f%s"% ("Sensitivity: ", sensitivity1*100, "%"))
  specificity1 = confusionmatrix[1,1]/(confusionmatrix[1,0]+confusionmatrix[1,1])
  print("%s%.2f%s"% ("Specificity: ", specificity1*100, "%"))

  print("\n")
  print("="*90)
  print("="*90)
  print(cls_report_print)
  print("="*90)

scores(Y_test,y_pred)
```



When we look at our confusion matrix, our model classified 25 of 25 test data belonging to the chip2a class correctly. It correctly classified 21 of the 22 test data belonging to the Chip5a class. It correctly classified 19 of the 20 test data belonging to the Chip3a class. He correctly classified 16 of the 18 test data belonging to the Missing class. It correctly classified 19 of our 19 test data belonging to Chip1a class. He classified 26 of our 27 test data belonging to the Healthy class correctly. He classified 13 of our 14 data belonging to the Spall class correctly. It correctly classified 18 of 19 test data belonging to Chip4a class. It correctly classified 24 of the 24 test data belonging to the Crack class. Our model has a 96% accuracy.