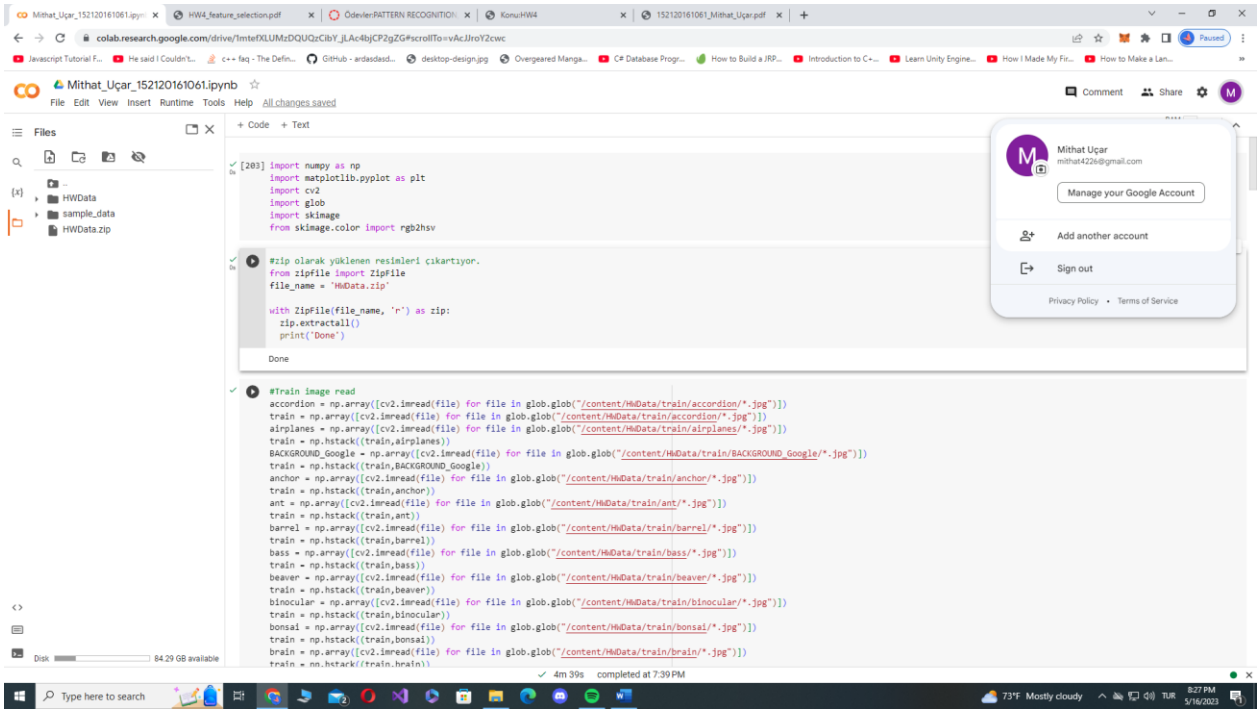
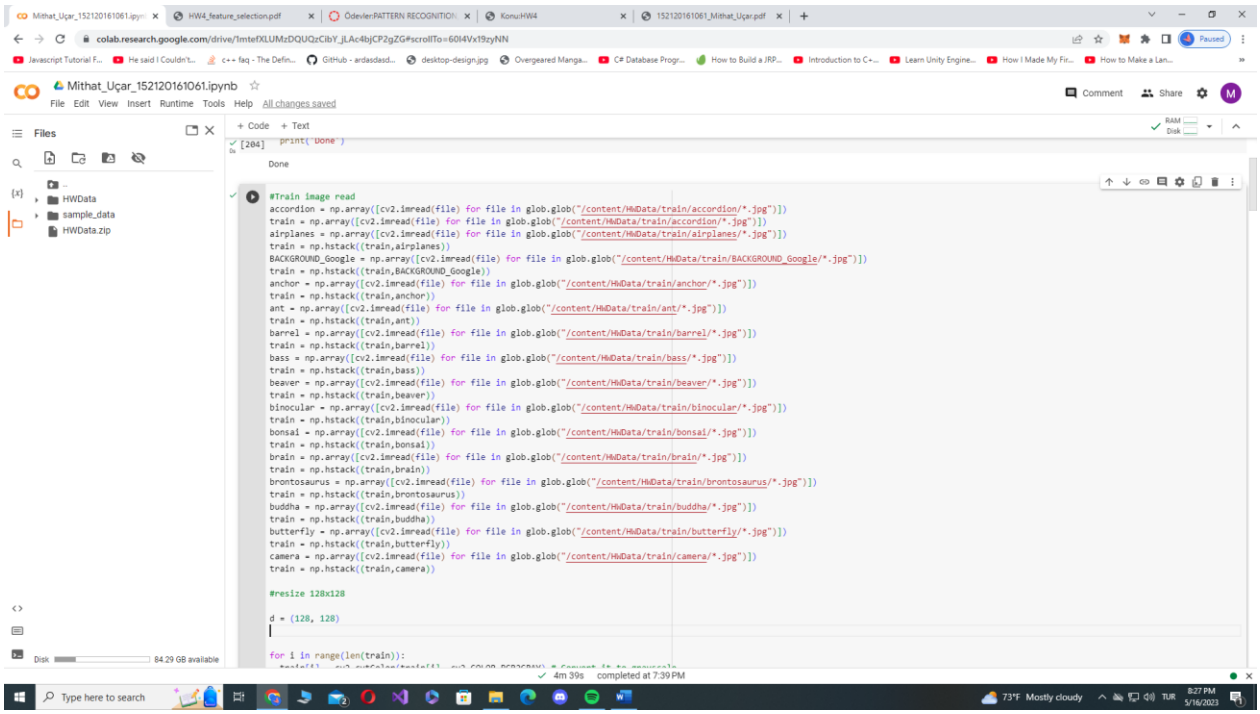
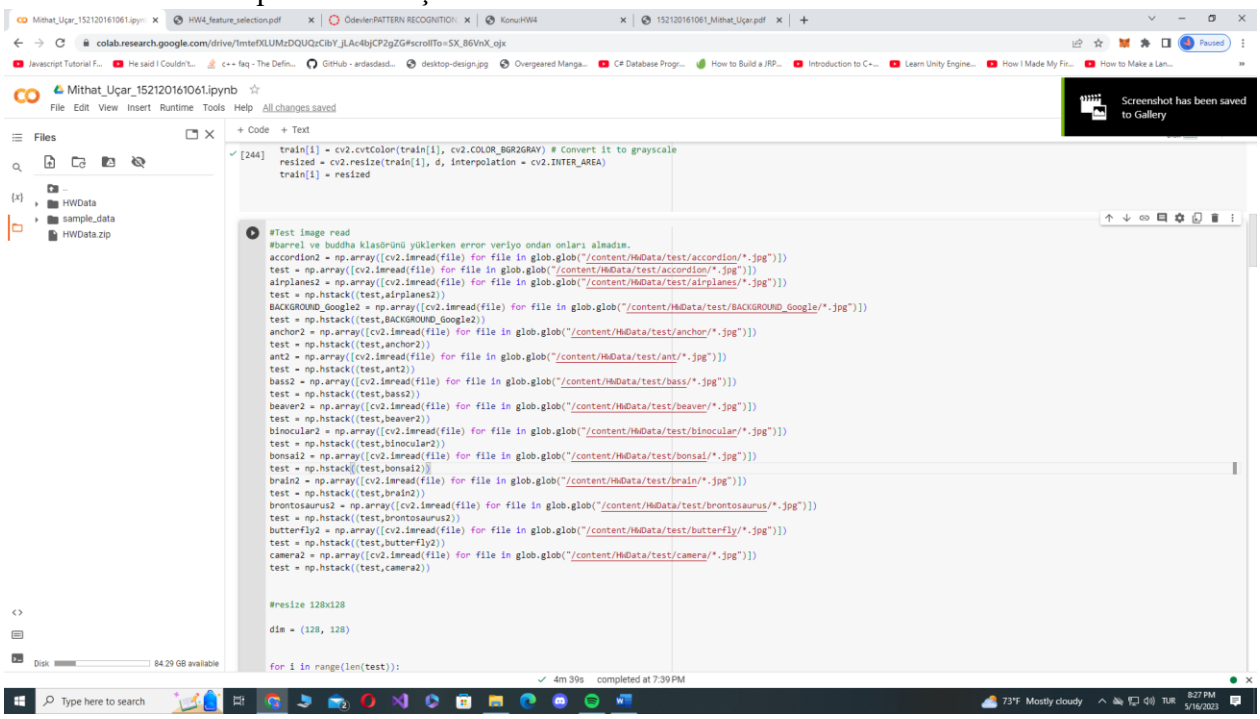
Pattern Recognition HW4



Resimleri colab'a zip olarak yükledim, zip dosyasını kod ile çıkartarak colab'da kullandım.



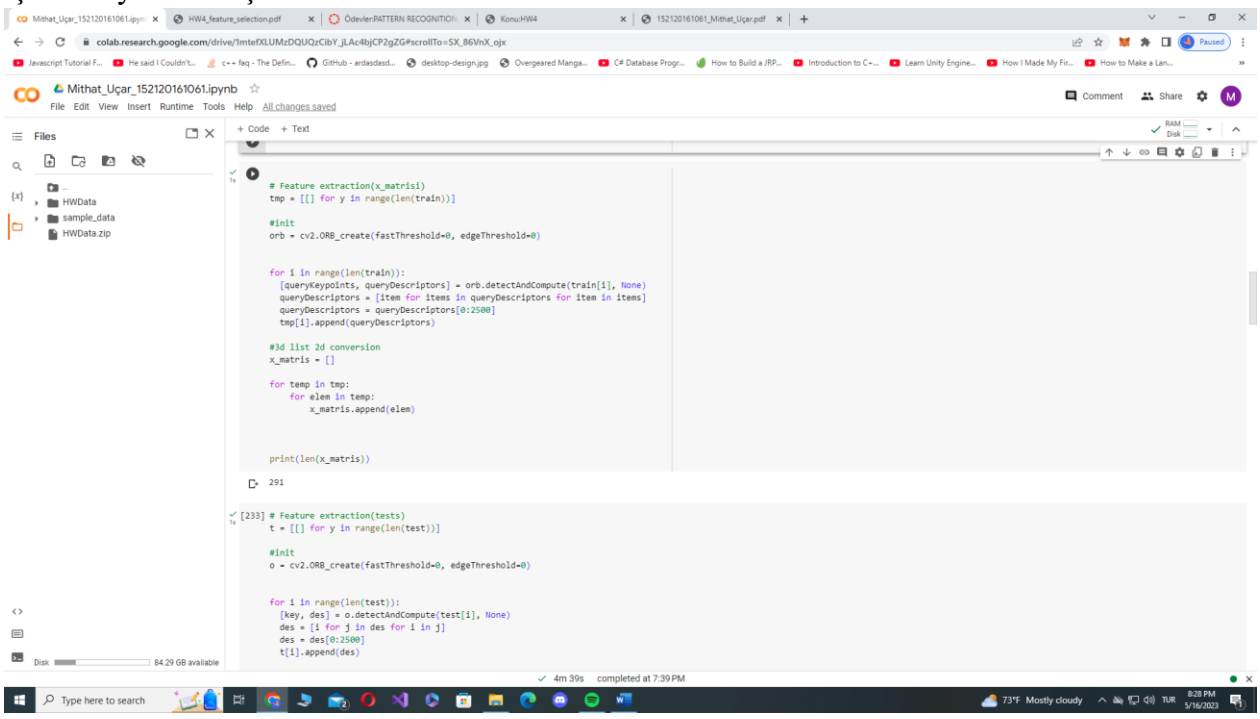
Resimleri 128x128 piksele dönüştürdüm.



```
train[i] = cv2.cvtColor(train[i], cv2.COLOR_BGR2GRAY) # Convert it to grayscale
resized = cv2.resize(train[i], d, interpolation = cv2.INTER_AREA)
train[i] = resized
```

```
#Test image read
#barrel ve buddha klasörünü yüklerken error veriyo ondan onları almadım.
accordion2 = np.array([cv2.imread(file) for file in glob.glob("/content/HWData/test/accordion/*.jpg")])
test = np.array([cv2.imread(file) for file in glob.glob("/content/HWData/test/accordion/*.jpg")])
airplanes2 = np.array([cv2.imread(file) for file in glob.glob("/content/HWData/test/airplanes/*.jpg")])
test = np.hstack((test,airplanes2))
BACKGROUND_Google2 = np.array([cv2.imread(file) for file in glob.glob("/content/HWData/test/BACKGROUND_Google/*.jpg")])
test = np.hstack((test,BACKGROUND_Google2))
anchor2 = np.array([cv2.imread(file) for file in glob.glob("/content/HWData/test/anchor/*.jpg")])
test = np.hstack((test,anchor2))
ant2 = np.array([cv2.imread(file) for file in glob.glob("/content/HWData/test/ant/*.jpg")])
test = np.hstack((test,ant2))
bass2 = np.array([cv2.imread(file) for file in glob.glob("/content/HWData/test/bass/*.jpg")])
test = np.hstack((test,bass2))
beaver2 = np.array([cv2.imread(file) for file in glob.glob("/content/HWData/test/beaver/*.jpg")])
test = np.hstack((test,beaver2))
binocular2 = np.array([cv2.imread(file) for file in glob.glob("/content/HWData/test/binocular/*.jpg")])
test = np.hstack((test,binocular2))
bonsai2 = np.array([cv2.imread(file) for file in glob.glob("/content/HWData/test/bonsai/*.jpg")])
test = np.hstack((test,bonsai2))
brain2 = np.array([cv2.imread(file) for file in glob.glob("/content/HWData/test/brain/*.jpg")])
test = np.hstack((test,brain2))
brontosaurus2 = np.array([cv2.imread(file) for file in glob.glob("/content/HWData/test/brontosaurus/*.jpg")])
test = np.hstack((test,brontosaurus2))
butterfly2 = np.array([cv2.imread(file) for file in glob.glob("/content/HWData/test/butterfly/*.jpg")])
test = np.hstack((test,butterfly2))
camera2 = np.array([cv2.imread(file) for file in glob.glob("/content/HWData/test/camera/*.jpg")])
test = np.hstack((test,camera2))


#resize 128x128

dim = (128, 128)


for i in range(len(test)):
```

Feature extraction işlemini yapmak için orb yöntemini kullandım.Feature matrisini kullanabilmek için 2 boyuta dönüştürdüm.

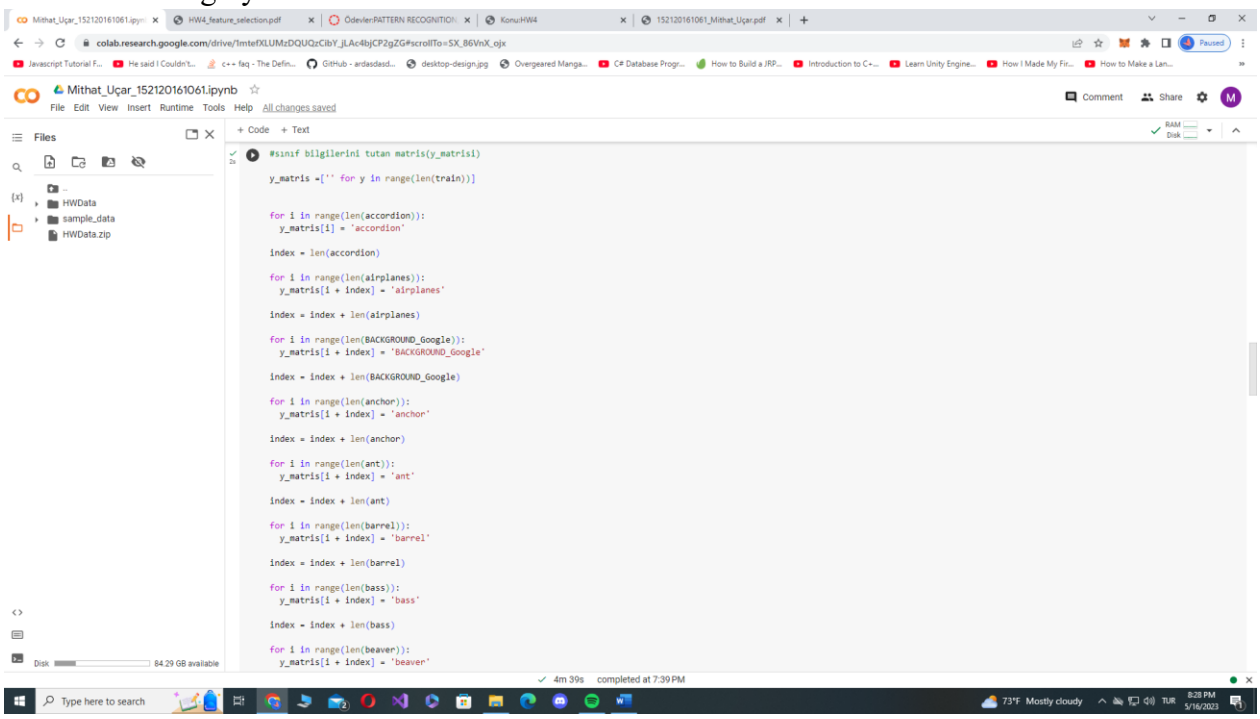


```
# Feature extraction(x_matrisi)
tmp = [[] for y in range(len(train))]

#init
orb = cv2.ORB_create(fastThreshold=0, edgeThreshold=0)


for i in range(len(train)):
  [queryKeypoints, queryDescriptors] = orb.detectAndCompute(train[i], None)
  queryDescriptors = [item for items in queryDescriptors for item in items]
  queryDescriptors = queryDescriptors[0:2500]
  tmp[i].append(queryDescriptors)

#3d list 2d conversion
x_matris = []

for temp in tmp:
    for elem in temp:
        x_matris.append(elem)


print(len(x_matris))
```

```
291
```

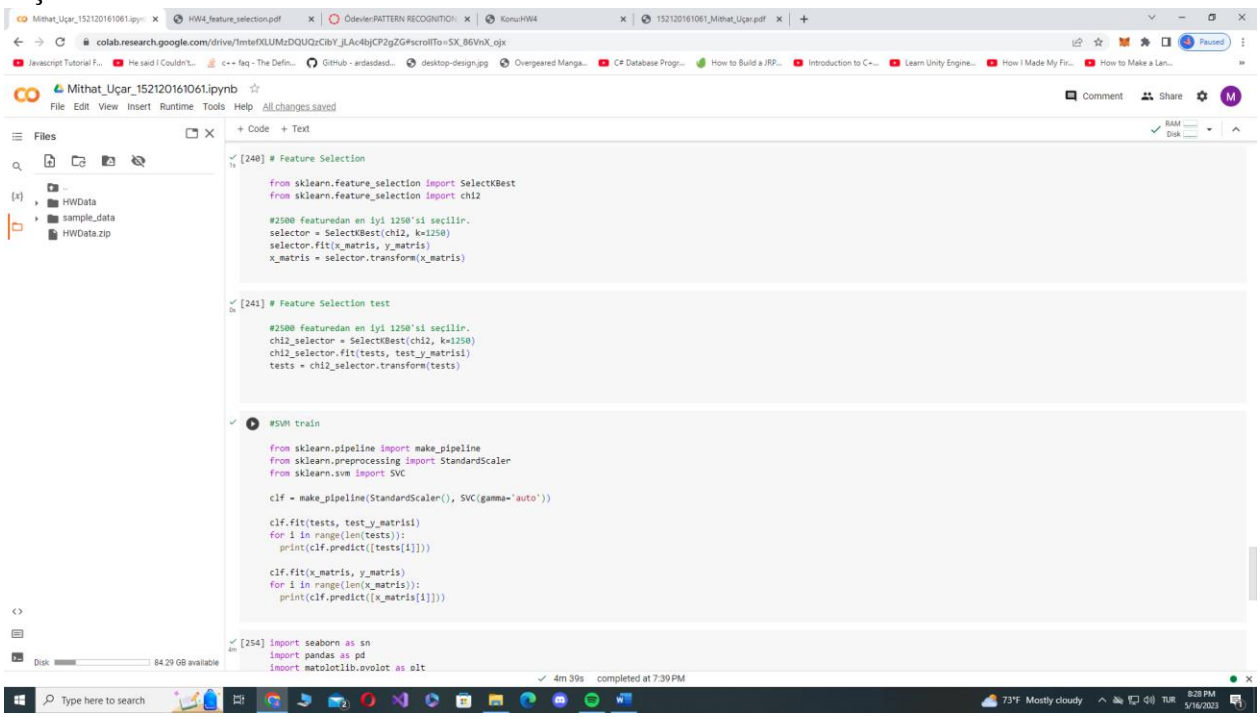
```
# Feature extraction(tests)
t = [[] for y in range(len(test))]

#init
o = cv2.ORB_create(fastThreshold=0, edgeThreshold=0)


for i in range(len(test)):
  [key, des] = o.detectAndCompute(test[i], None)
  des = [i for j in des for i in j]
  des = des[0:2500]
  t[i].append(des)
```

Labelların olduğu y matrisi



```
#sinif bilgilerini tutan matris(y_matrisi)

y_matris =['' for y in range(len(train))]


for i in range(len(accordion)):
  y_matris[i] = 'accordion'

index = len(accordion)

for i in range(len(airplanes)):
  y_matris[i + index] = 'airplanes'

index = index + len(airplanes)

for i in range(len(BACKGROUND_Google)):
  y_matris[i + index] = 'BACKGROUND_Google'

index = index + len(BACKGROUND_Google)

for i in range(len(anchor)):
  y_matris[i + index] = 'anchor'

index = index + len(anchor)

for i in range(len(ant)):
  y_matris[i + index] = 'ant'

index = index + len(ant)

for i in range(len(barrel)):
  y_matris[i + index] = 'barrel'

index = index + len(barrel)

for i in range(len(bass)):
  y_matris[i + index] = 'bass'

index = index + len(bass)

for i in range(len(beaver)):
  y_matris[i + index] = 'beaver'
```

Feature reduction yaparak 2500 featuredan 1250 feature seçilir.Bunun amacı en zengin değerleri seçmektir.



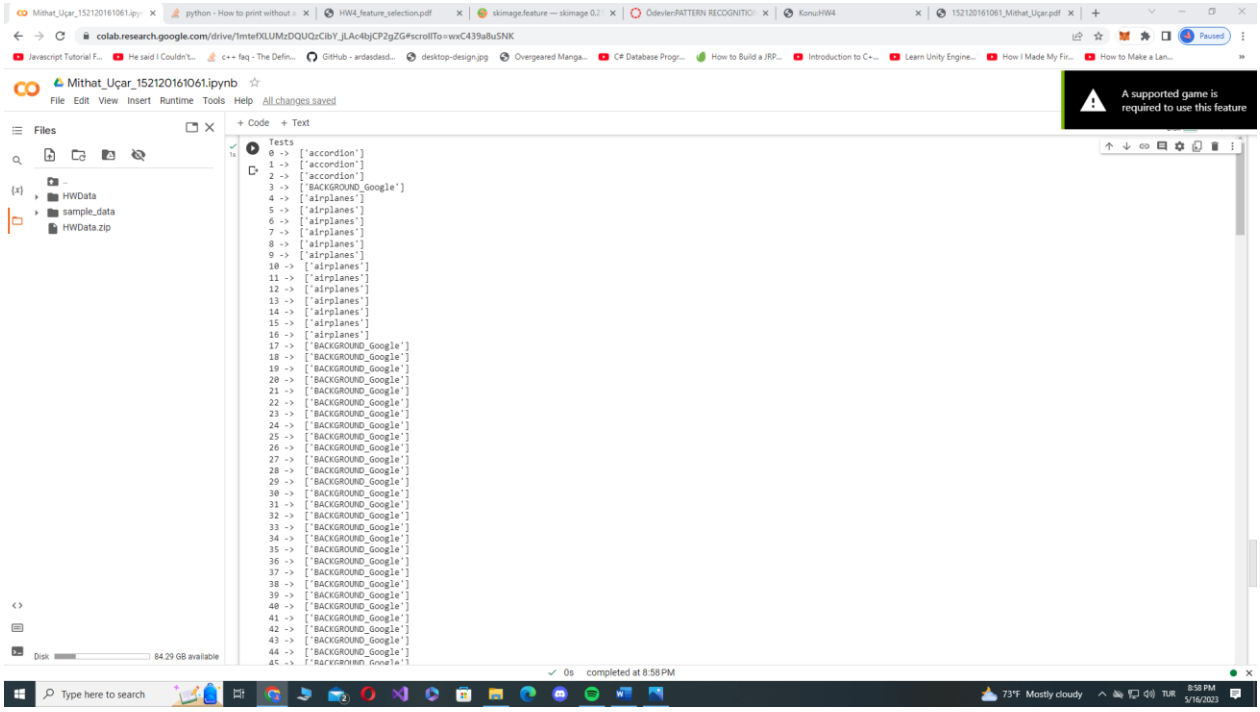
```
[240] # Feature Selection

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

#2500 featuredan en iyi 1250'si seçilir.
selector = SelectKBest(chi2, k=1250)
selector.fit(x_matris, y_matris)
x_matris = selector.transform(x_matris)
```

```
[241] # Feature Selection test

#2500 featuredan en iyi 1250'si seçilir.
chi2_selector = SelectKBest(chi2, k=1250)
chi2_selector.fit(tests, test_y_matrisi)
tests = chi2_selector.transform(tests)
```

```
#SVM train

from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC

clf = make_pipeline(StandardScaler(), SVC(gamma='auto'))

clf.fit(tests, test_y_matrisi)
for i in range(len(tests)):
  print(clf.predict([tests[i]]))

clf.fit(x_matris, y_matris)
for i in range(len(x_matris)):
  print(clf.predict([x_matris[i]]))
```

```
[254] import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt
```

Printler



Confusion Matrix: