**Assignment2**

Name:Dew

ID:18406040245

[1. Tutorial 1](#_Toc345306217)

[Breast Cancer Data Exploration: 1](#_Toc1060774925)

[CIFAR - 10 Data Exploration 2](#_Toc34927081)

[Visualizing the Breast Cancer data 3](#_Toc756414736)

[Visualizing the CIFAR - 10 data 5](#_Toc2106761359)

[2. Homework 8](#_Toc627788977)

# Tutorial

**The code is running in jupyter**

## Breast Cancer Data Exploration:

from sklearn.datasets import load\_breast\_cancer

breast = load\_breast\_cancer()

breast\_data = breast.data

breast\_data.shape

breast\_labels = breast.target

breast\_labels.shape

import numpy as np

labels = np.reshape(breast\_labels,(569,1))

final\_breast\_data = np.concatenate([breast\_data,labels],axis=1)

final\_breast\_data.shape

import pandas as pd

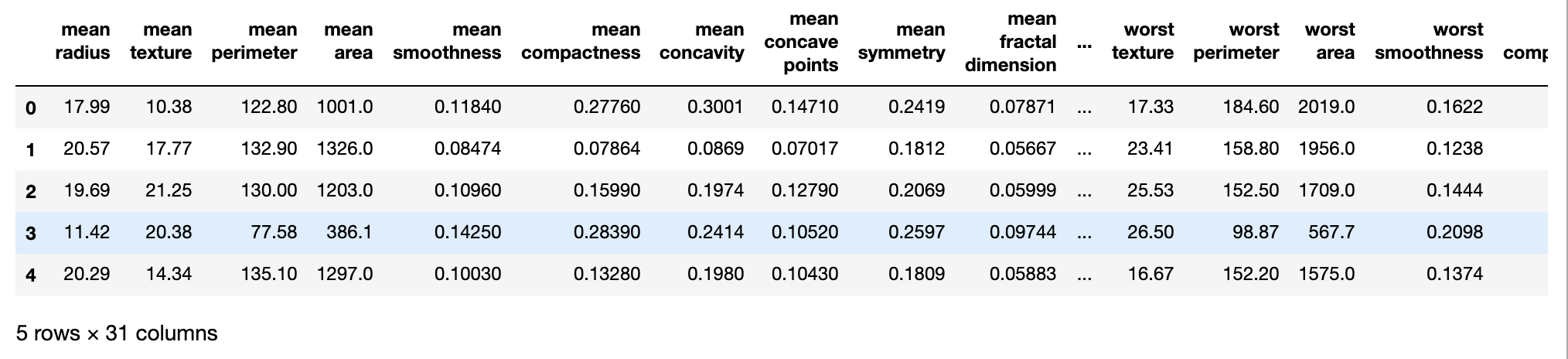
breast\_dataset = pd.DataFrame(final\_breast\_data)

features = breast.feature\_names

Features

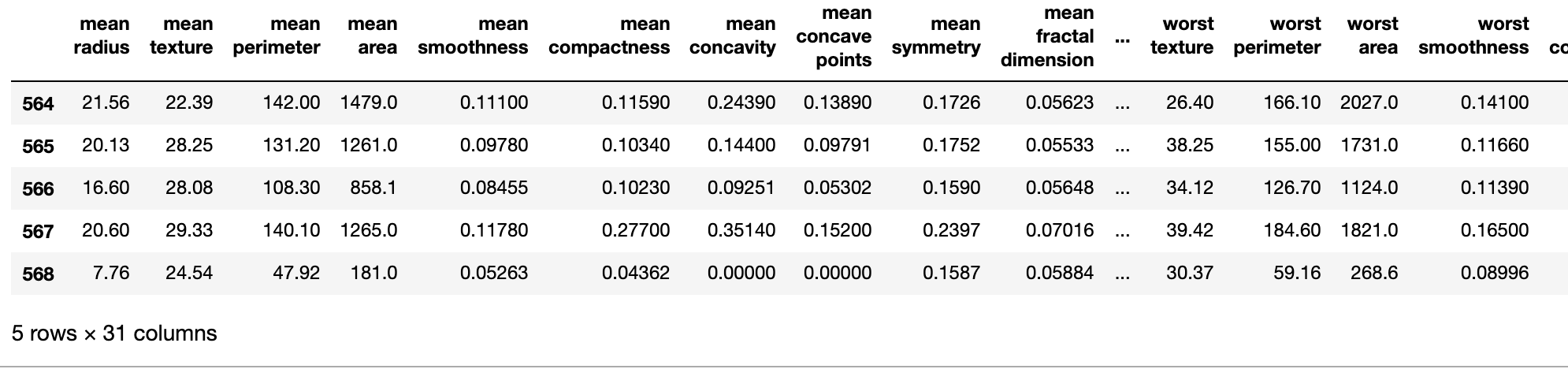
features\_labels = np.append(features,'label')

breast\_dataset.columns = features\_labels

breast\_dataset.head()

breast\_dataset['label'].replace(0, 'Benign',inplace=True)

breast\_dataset['label'].replace(1, 'Malignant',inplace=True)

breast\_dataset.tail()

## CIFAR - 10 Data Exploration

conda install Keras

conda install TensorFlow

from keras.datasets import cifar10

(x\_train, y\_train), (x\_test, y\_test) = cifar10.load\_data()

print('Traning data shape:', x\_train.shape)

print('Testing data shape:', x\_test.shape)

y\_train.shape,y\_test.shape

# Find the unique numbers from the train labels

classes = np.unique(y\_train)

nClasses = len(classes)

print('Total number of outputs : ', nClasses)

print('Output classes : ', classes)

import matplotlib.pyplot as plt

%matplotlib inline

label\_dict = {

0: 'airplane',

1: 'automobile',

2: 'bird',

3: 'cat',

4: 'deer',

5: 'dog',

6: 'frog',

7: 'horse',

8: 'ship',

9: 'truck',

}

plt.figure(figsize=[5,5])

# Display the first image in training data

plt.subplot(121)

curr\_img = np.reshape(x\_train[0], (32,32,3))

plt.imshow(curr\_img)

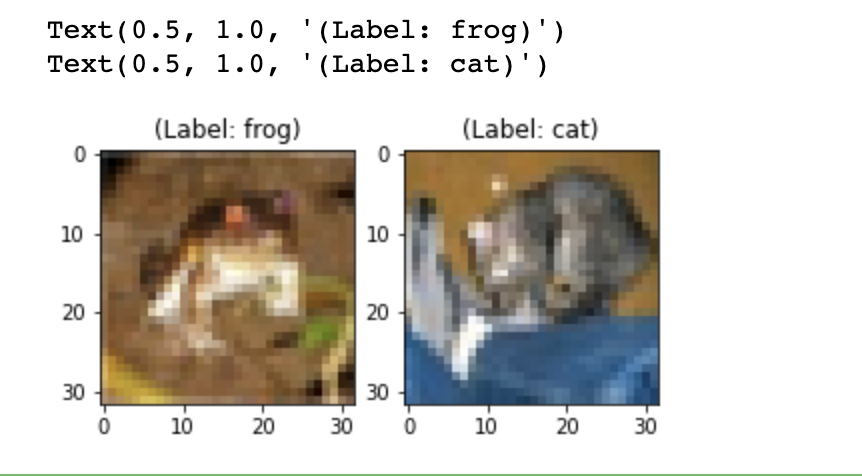
print(plt.title("(Label: " + str(label\_dict[y\_train[0][0]]) + ")"))

# Display the first image in testing data

plt.subplot(122)

curr\_img = np.reshape(x\_test[0],(32,32,3))

plt.imshow(curr\_img)

print(plt.title("(Label: " + str(label\_dict[y\_test[0][0]]) + ")"))

## Visualizing the Breast Cancer data

from sklearn.preprocessing import StandardScaler

x = breast\_dataset.loc[:, features].values

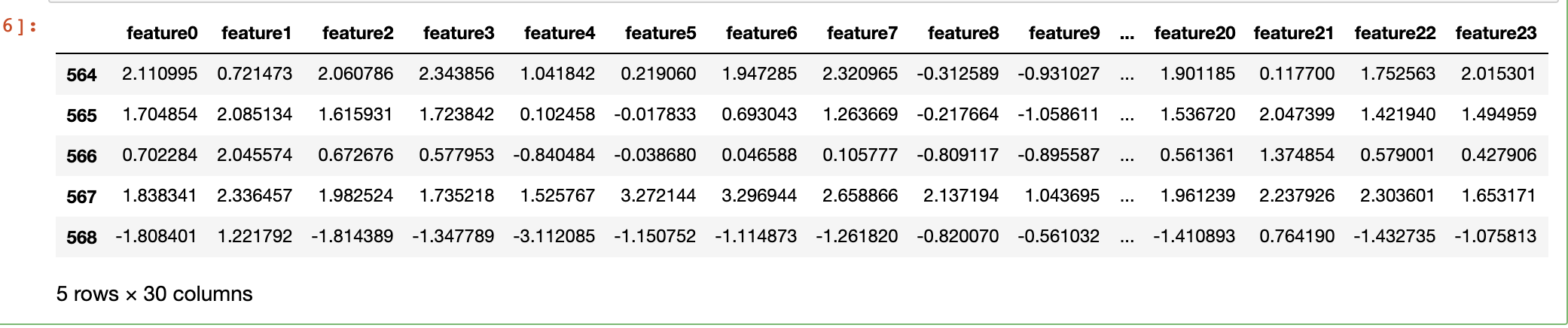
x = StandardScaler().fit\_transform(x) # normalizing the features

x.shape

np.mean(x),np.std(x)

feat\_cols = ['feature'+str(i) for i in range(x.shape[1])]

normalised\_breast = pd.DataFrame(x,columns=feat\_cols)

normalised\_breast.tail()

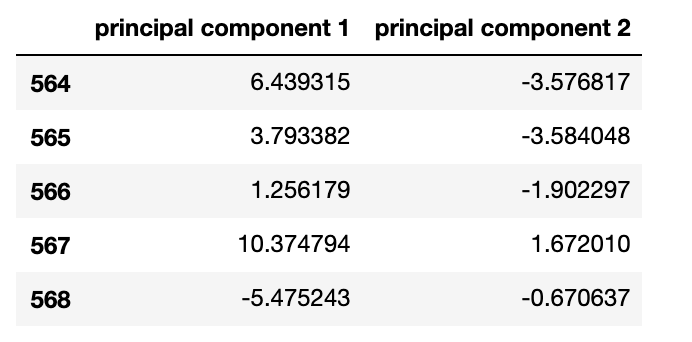
from sklearn.decomposition import PCA

pca\_breast = PCA(n\_components=2)

principalComponents\_breast = pca\_breast.fit\_transform(x)

principal\_breast\_Df = pd.DataFrame(data = principalComponents\_breast

, columns = ['principal component 1', 'principal component 2'])

principal\_breast\_Df.tail()

print('Explained variation per principal component: {}'.format(pca\_breast.explained\_variance\_ratio\_))

plt.figure()

plt.figure(figsize=(10,10))

plt.xticks(fontsize=12)

plt.yticks(fontsize=14)

plt.xlabel('Principal Component - 1',fontsize=20)

plt.ylabel('Principal Component - 2',fontsize=20)

plt.title("Principal Component Analysis of Breast Cancer Dataset",fontsize=20)

targets = ['Benign', 'Malignant']

colors = ['r', 'g']

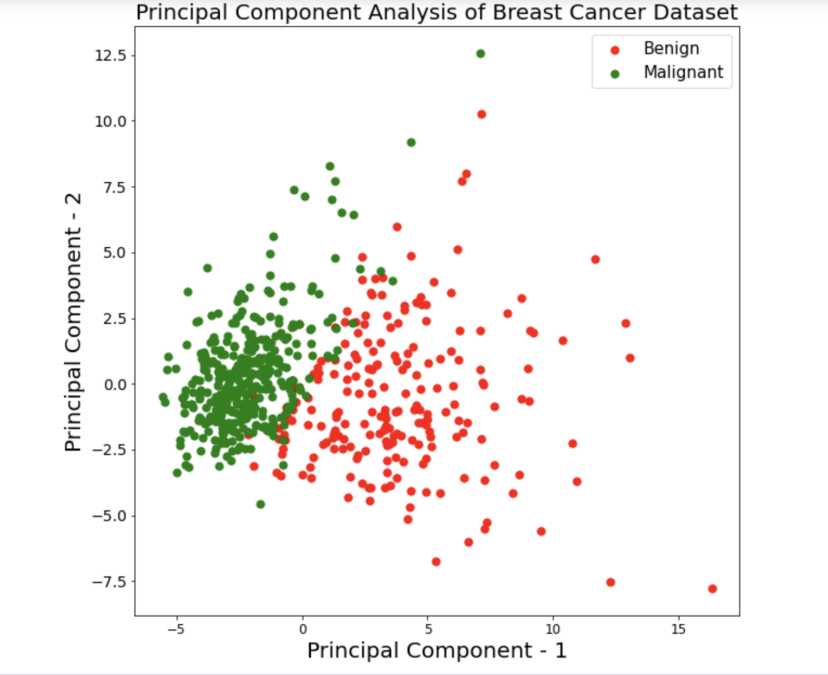
for target, color in zip(targets,colors):

indicesToKeep = breast\_dataset['label'] == target

plt.scatter(principal\_breast\_Df.loc[indicesToKeep, 'principal component 1']

, principal\_breast\_Df.loc[indicesToKeep, 'principal component 2'], c = color, s = 50)

plt.legend(targets,prop={'size': 15})



## Visualizing the CIFAR - 10 data

np.min(x\_train),np.max(x\_train)

x\_train = x\_train/255.0

np.min(x\_train),np.max(x\_train)

x\_train.shape

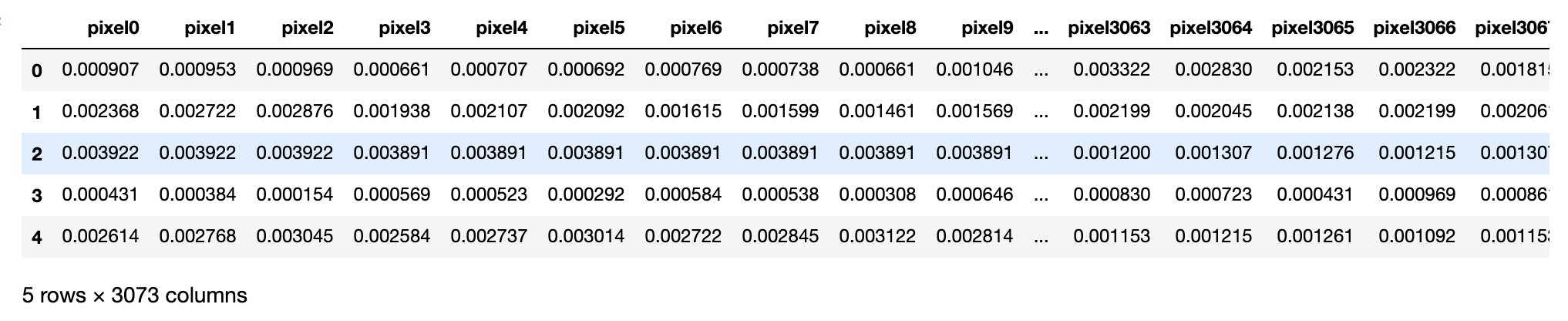
x\_train\_flat = x\_train.reshape(-1,3072)

feat\_cols = ['pixel'+str(i) for i in range(x\_train\_flat.shape[1])]

df\_cifar = pd.DataFrame(x\_train\_flat,columns=feat\_cols)

df\_cifar['label'] = y\_train

print('Size of the dataframe: {}'.format(df\_cifar.shape))

df\_cifar.head()

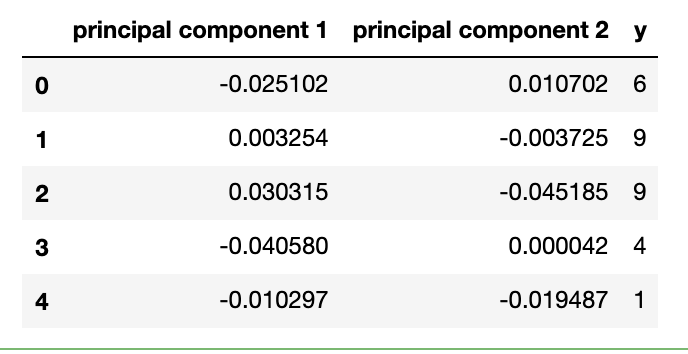
pca\_cifar = PCA(n\_components=2)

principalComponents\_cifar = pca\_cifar.fit\_transform(df\_cifar.iloc[:,:-1])

principal\_cifar\_Df = pd.DataFrame(data = principalComponents\_cifar

, columns = ['principal component 1', 'principal component 2'])

principal\_cifar\_Df['y'] = y\_train

principal\_cifar\_Df.head()

print('Explained variation per principal component: {}'.format(pca\_cifar.explained\_variance\_ratio\_))

import seaborn as sns

plt.figure(figsize=(16,10))

sns.scatterplot(

x="principal component 1", y="principal component 2",

hue="y",

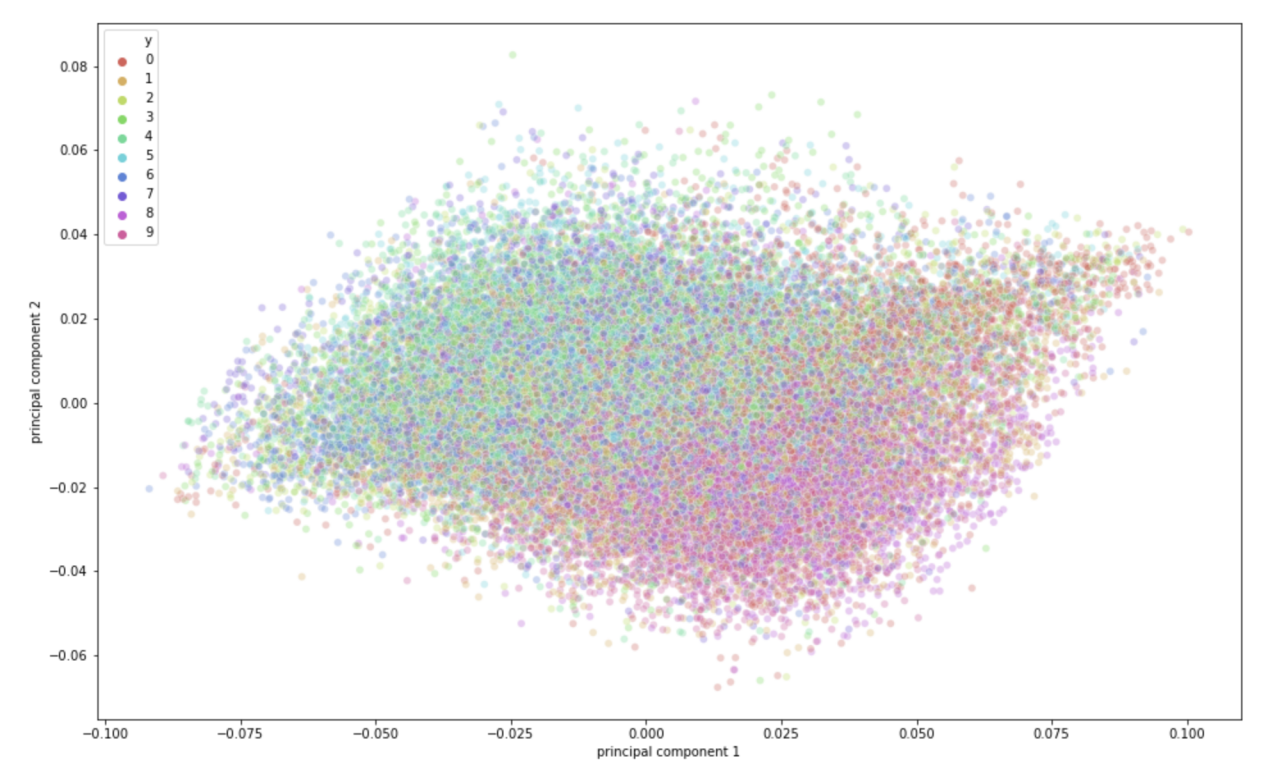
palette=sns.color\_palette("hls", 10),

data=principal\_cifar\_Df,

legend="full",

alpha=0.3

)



Speed Up Deep Learning Training using PCA with CIFAR - 10 Dataset

x\_test = x\_test/255.0

x\_test = x\_test.reshape(-1,32,32,3)

x\_test\_flat = x\_test.reshape(-1,3072)

pca = PCA(0.9)

pca.fit(x\_train\_flat)

pca.n\_components\_

train\_img\_pca = pca.transform(x\_train\_flat)

test\_img\_pca = pca.transform(x\_test\_flat)

from keras.models import Sequential

from keras.layers import Dense

from keras.utils import np\_utils

from keras.optimizers import RMSprop

y\_train = np\_utils.to\_categorical(y\_train)

y\_test = np\_utils.to\_categorical(y\_test)

batch\_size = 128

num\_classes = 10

epochs = 20

model = Sequential()

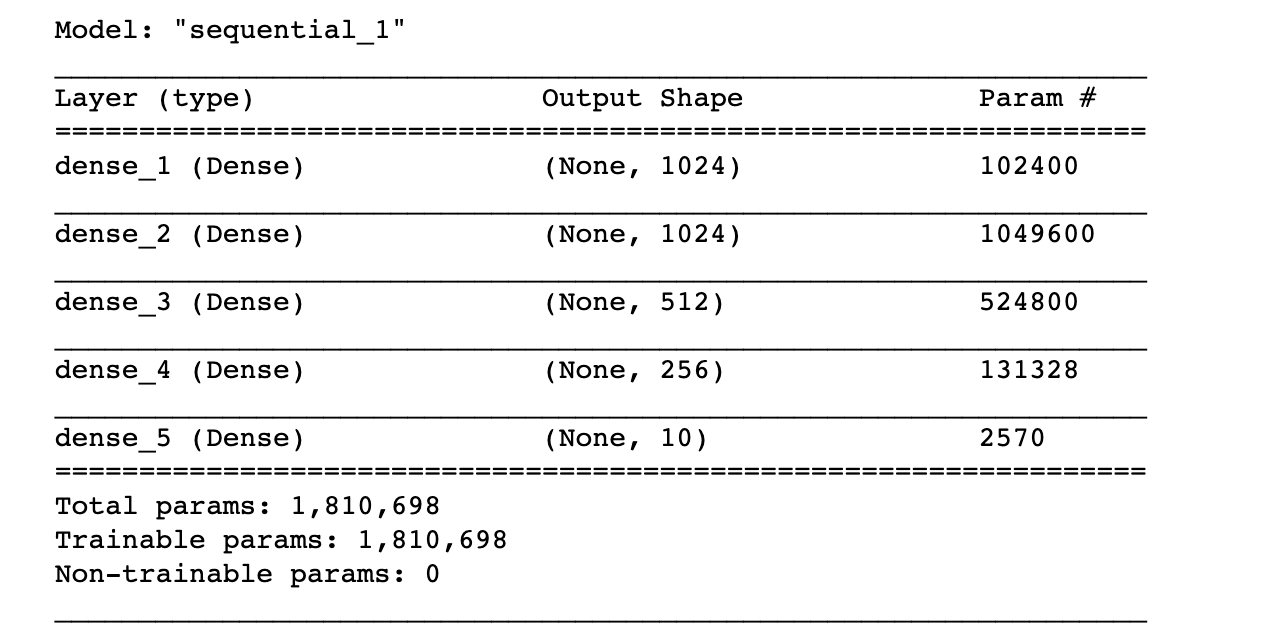
model.add(Dense(1024, activation='relu', input\_shape=(99,)))

model.add(Dense(1024, activation='relu'))

model.add(Dense(512, activation='relu'))

model.add(Dense(256, activation='relu'))

model.add(Dense(num\_classes, activation='softmax'))

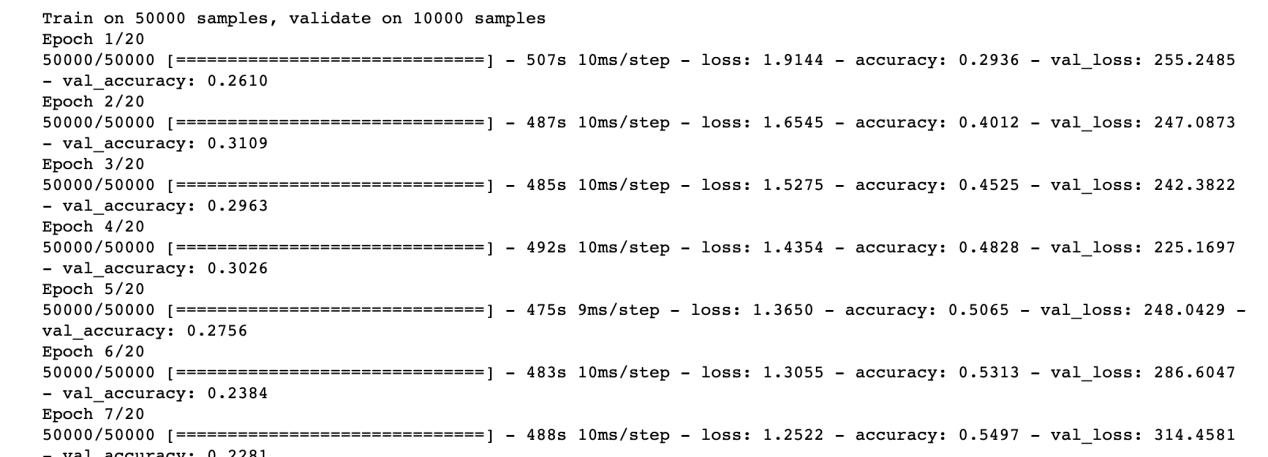
model.summary()

model.compile(loss='categorical\_crossentropy',

optimizer=RMSprop(),

metrics=['accuracy'])

history = model.fit(train\_img\_pca, y\_train,batch\_size=batch\_size,epochs=epochs,verbose=1,

validation\_data=(test\_img\_pca, y\_test))

# Homework

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

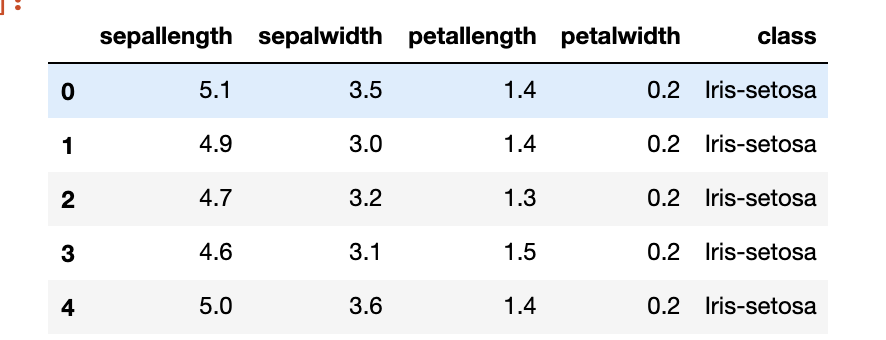
import numpy as np

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

df = pd.read\_csv("/Users/dew/Desktop/iris.csv")

df.head()



df1 = df.loc[:,df.columns[0:4]]

scaler = StandardScaler()

scaler.fit(df1)

scaled\_df = scaler.transform(df1)

pca = PCA(n\_components = 4)

pca.fit(scaled\_df)

x\_pca = pca.transform(scaled\_df)

percent\_variance=pca.explained\_variance\_ratio\_

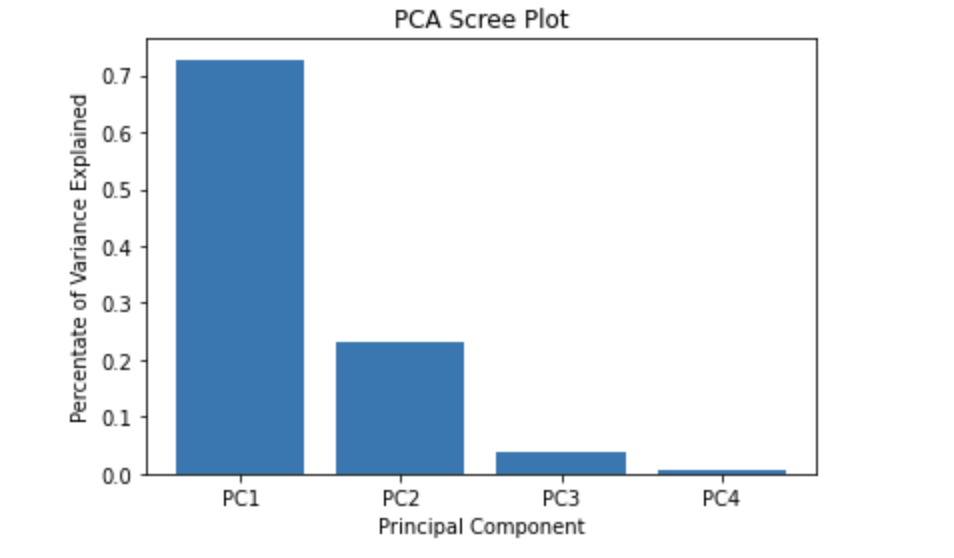
plt.bar(x = range(1,5),height=percent\_variance,tick\_label=["PC" + str(i) for i in range(1,5)])

plt.ylabel('Percentate of Variance Explained')

plt.xlabel('Principal Component')

plt.title('PCA Scree Plot')

plt.savefig("PCA scree Plot.jpg")



df2 = pd.DataFrame(data=x\_pca,columns=["PC"+str(i) for i in range(1,5)])

df['target']=df.loc[:,df.columns[4]]

df3 = pd.concat([df2,df['target']],axis=1)

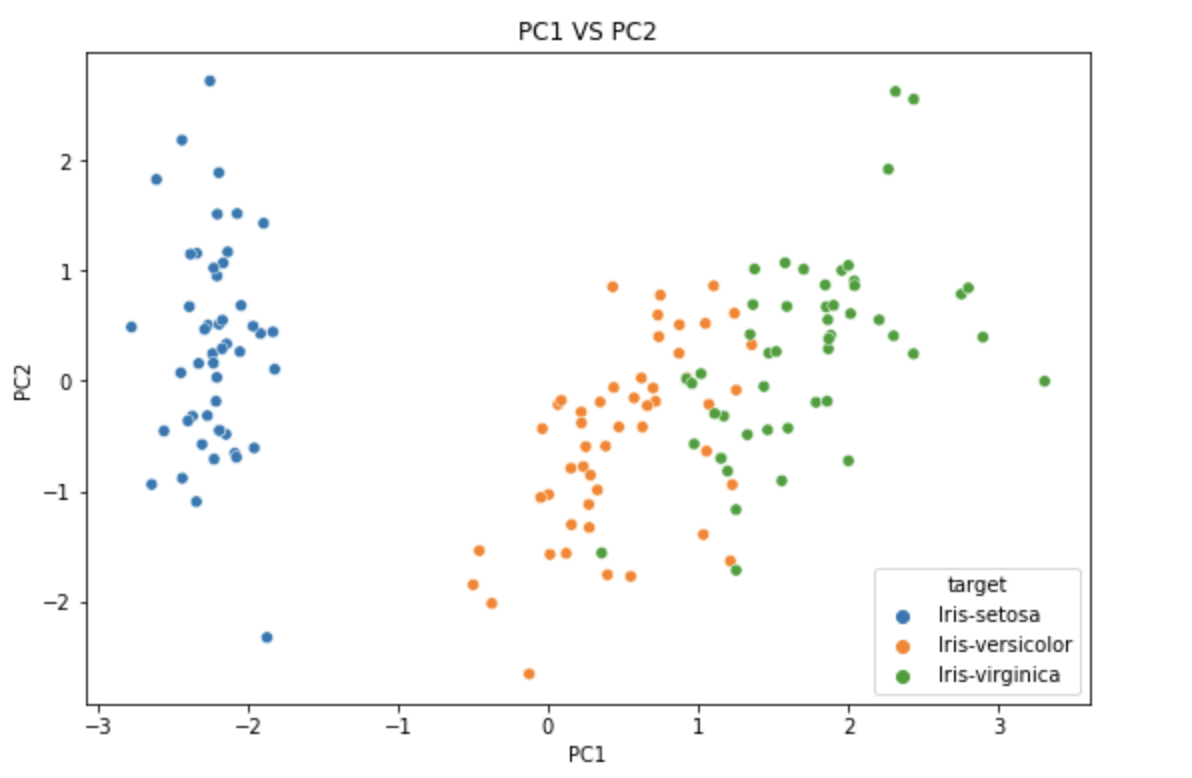
plt.figure(figsize=(9,6))

sns.scatterplot(x="PC1",y="PC2",hue='target',data=df3)

plt.xlabel("PC1")

plt.ylabel("PC2")

plt.title("PC1 VS PC2")

plt.savefig("PC1 and PC2.jpg")

from matplotlib import patches

from scipy.spatial import ConvexHull

def encircle(x,y, ax=None, \*\*kw):

if not ax: ax=plt.gca()

p = np.c\_[x,y]

hull = ConvexHull(p)

poly = plt.Polygon(p[hull.vertices,:], \*\*kw)

ax.add\_patch(poly)

df3\_1 = df3.loc[df3.target == df3.target.unique()[0],:]

df3\_2 = df3.loc[df3.target == df3.target.unique()[1],:]

df3\_3 = df3.loc[df3.target == df3.target.unique()[2],:]

plt.figure(figsize=(9,6))

sns.scatterplot(x="PC1",y="PC2",hue='target',data=df3)

encircle(df3\_1.PC1,df3\_1.PC2,alpha=0.1)

encircle(df3\_2.PC1,df3\_2.PC2,fc = "gold", alpha=0.1)

encircle(df3\_3.PC1,df3\_3.PC2,ec = "firebrick",fc = "gold", alpha=0.1)

plt.savefig("3.jpg")