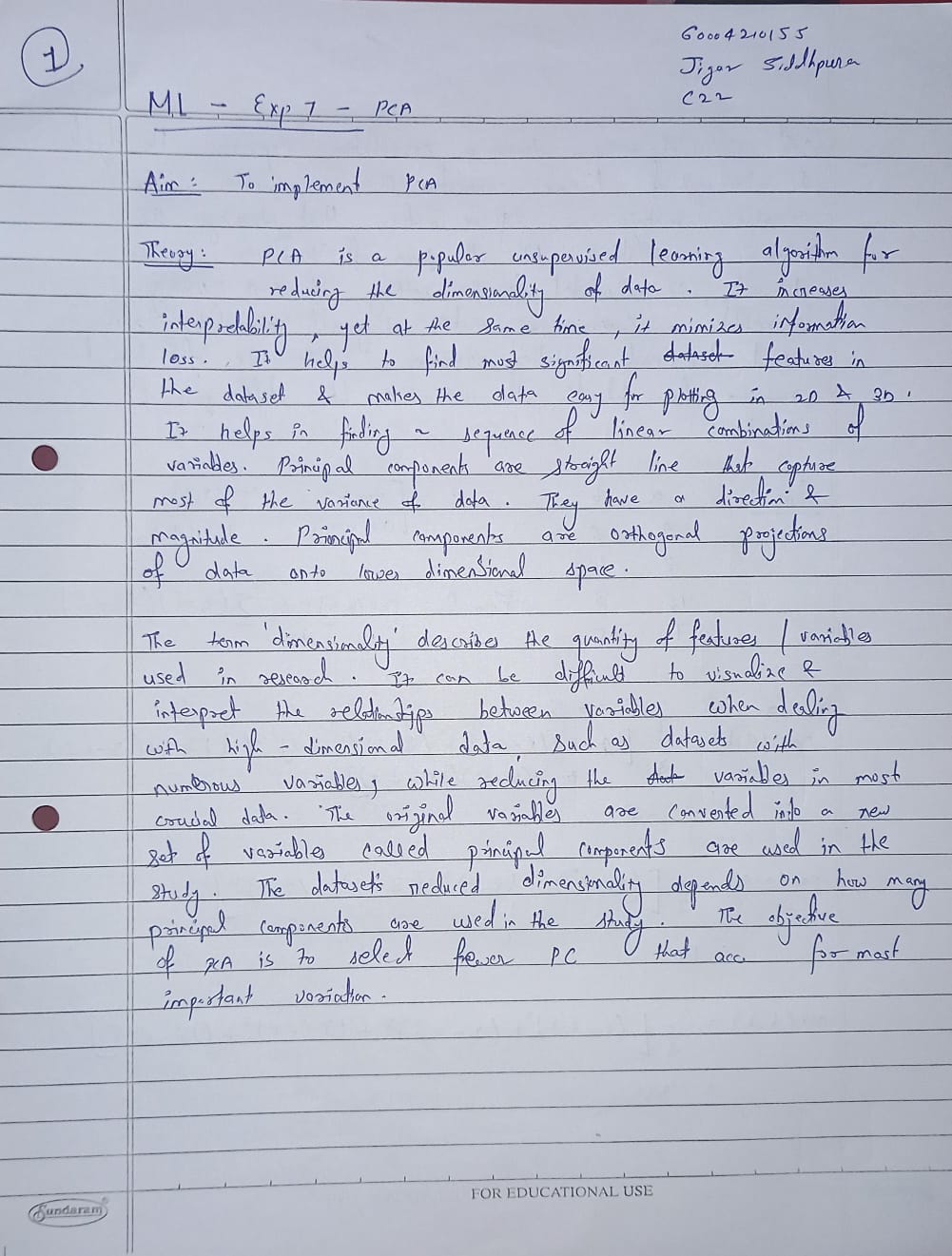
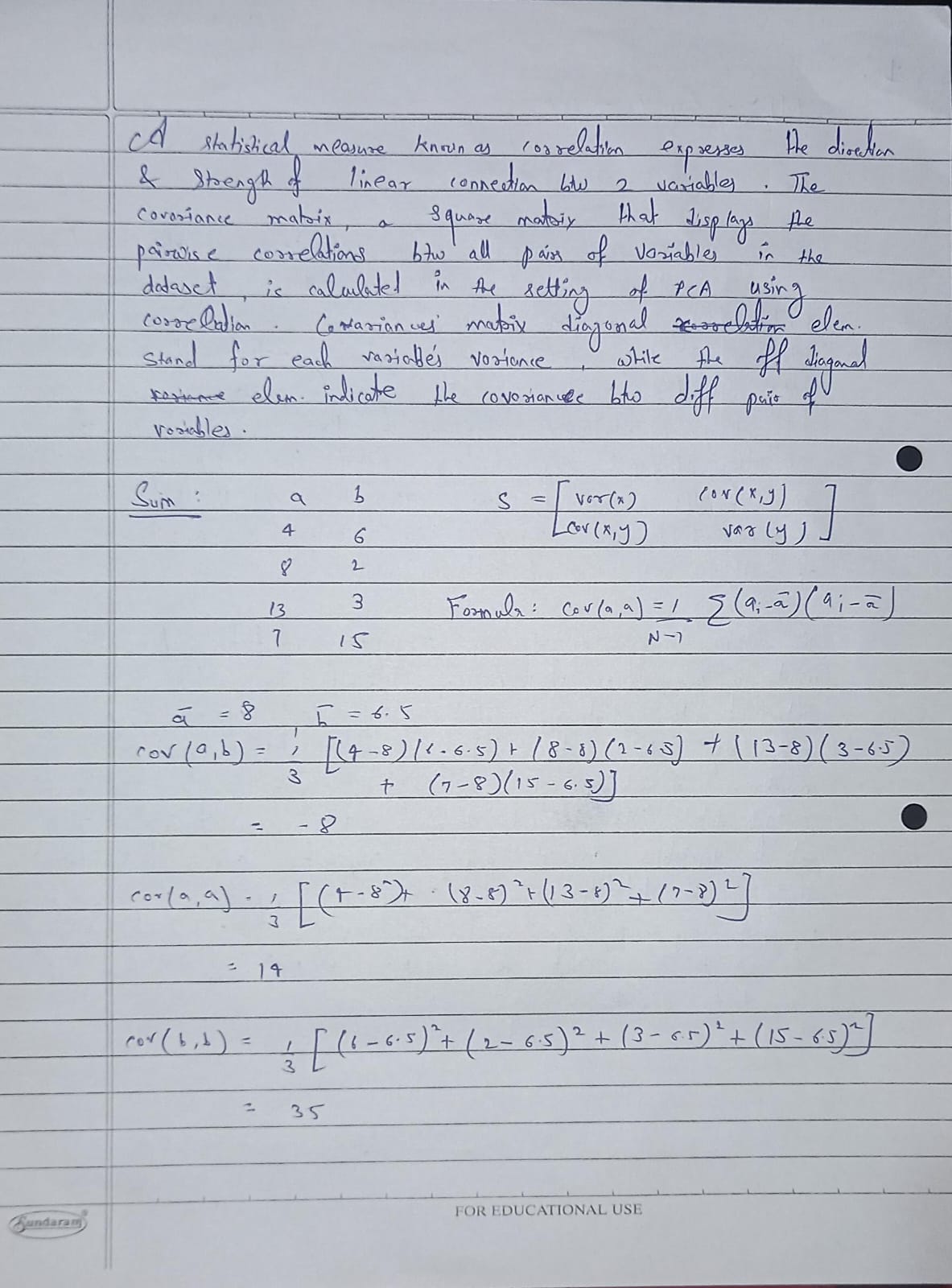
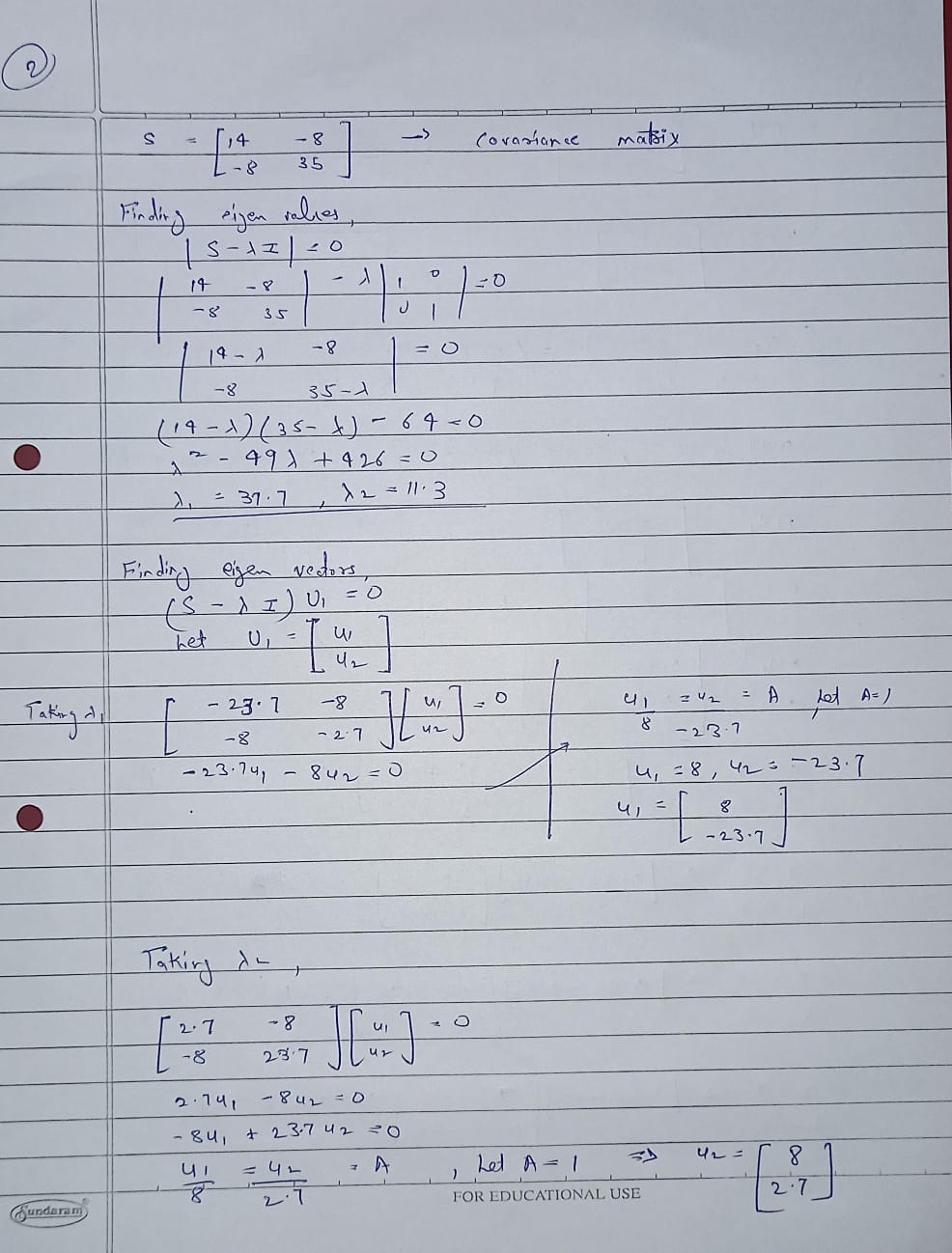
**Name: Jigar Siddhpura SAPID:** 60004200155

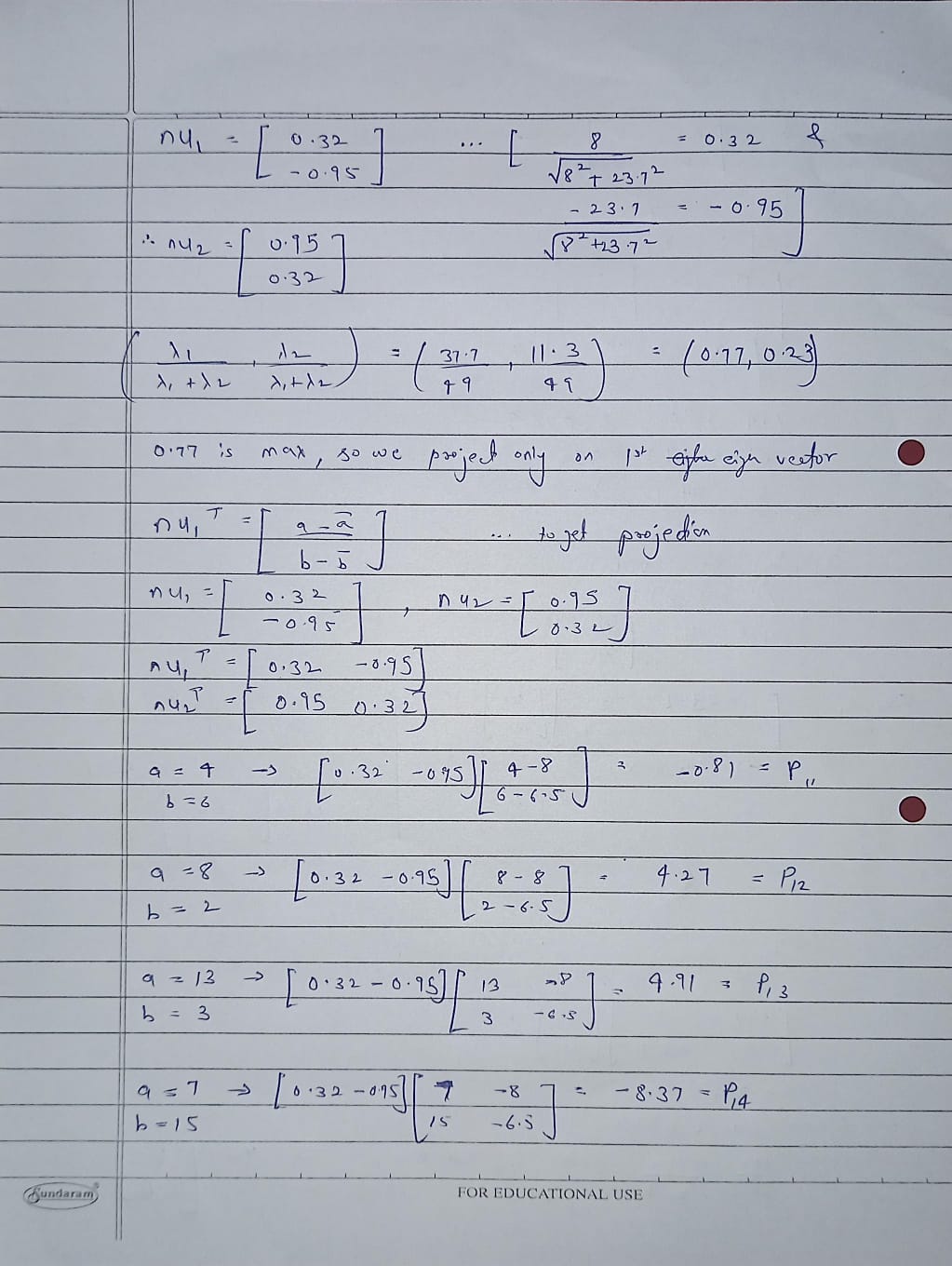
**DIV: C/C2 Branch:** Computer Engineering

ML - Experiment 7 - PCA









import pandas as pd

import numpy as np

from numpy.linalg import eig

**DATASET 1 - CODE**

data = np.array([[4, 6], [8, 2], [13, 3], [7, 15]])

def PCA(df):

  centered\_data = df - df.mean()

  cov\_matrix = np.cov(centered\_data, rowvar=False)

  eigenvalues, eigenvectors = np.linalg.eig(cov\_matrix)

  sorted\_indices = np.argsort(eigenvalues)[::-1]

  eigenvalues = eigenvalues[sorted\_indices]

  eigenvectors = eigenvectors[:, sorted\_indices]

  new\_values = np.dot(centered\_data, eigenvectors)[:,0]

  print("Centered Data:")

  print(centered\_data)

  print("\nCovariance Matrix:")

  print(cov\_matrix)

  print("\nEigenvalues:")

  print(eigenvalues)

  print("\nEigenvectors:")

  print(eigenvectors)

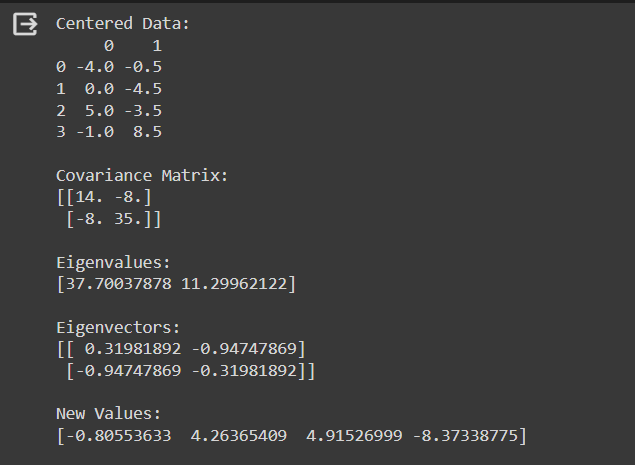
  print("\nNew Values:")

  print(new\_values)

df2 = pd.DataFrame(data)

PCA(df2)

**OUTPUT**



**DATASET 2 - CODE**

df = pd.read\_csv('/content/gdrive/MyDrive/ML/salary\_data.csv')



import pandas as pd

from sklearn.decomposition import PCA

def apply\_pca(data, n\_components):

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

    principalComponents = pca.fit\_transform(data)

    principalDf = pd.DataFrame(data=principalComponents, columns=[f'New Values' for i in range(n\_components)])

    # Calculate additional PCA components

    centered\_data = data - data.mean()

    cov\_matrix = pca.get\_covariance()

    eigenvalues = pca.explained\_variance\_

    eigenvectors = pca.components\_

    print("Centered Data:")

    print(centered\_data)

    print("\nCovariance Matrix:")

    print(cov\_matrix)

    print("\nEigenvalues:")

    print(eigenvalues)

    print("\nEigenvectors:")

    print(eigenvectors)

    print("\n")

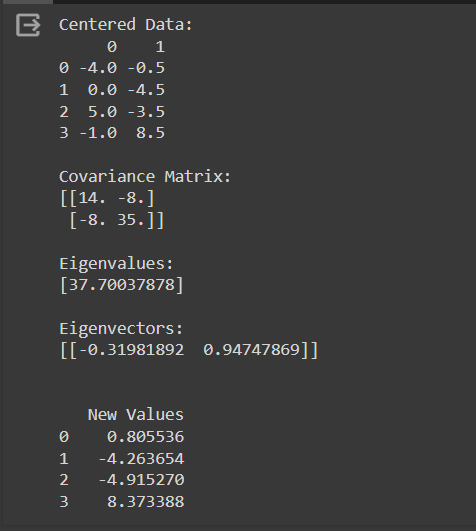
    return principalDf

n\_components = 1

result = apply\_pca(df2, n\_components)

print(result)

**OUTPUT**



**Conclusion :** PCA, a powerful dimensionality reduction technique, simplifies large datasets by transforming variables into a smaller set while retaining essential information. It can be used for for summarizing complex datasets, uncovering relationships between variables, and simplifying data analysis processes effectively.