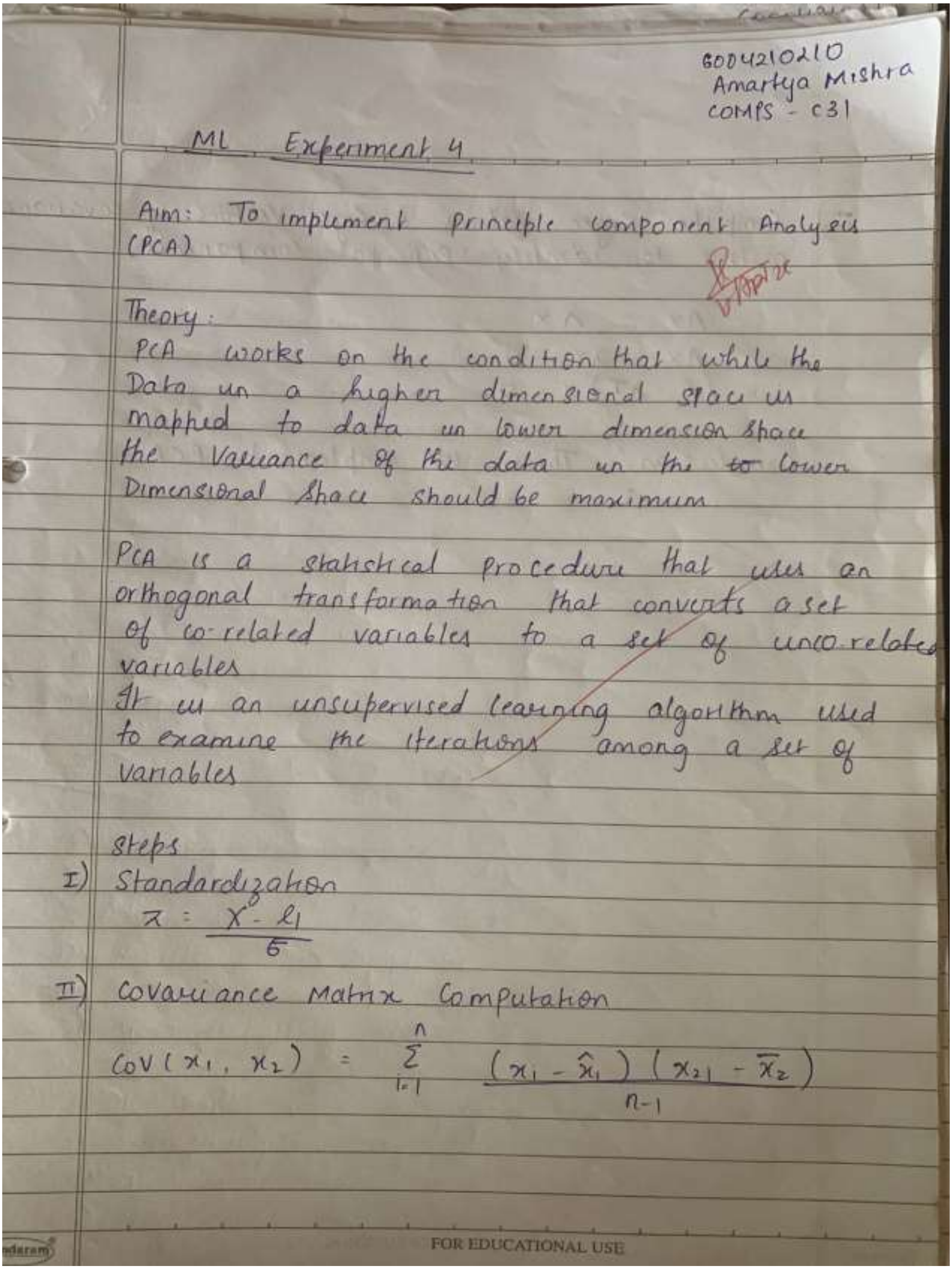
60004210210

Amartya Mishra

COMPS – C31

ML Experiment 4



6004210210
Amartya Mishra
COMPS - C31

ML Experiment 4

Aim: To implement principle component Analysis (PCA)

Theory:

PCA works on the condition that while the Data in a higher dimensional space is mapped to data in lower dimension space the Variance of the data in the ~~to~~ lower Dimensional Space should be maximum

PCA is a statistical procedure that uses an orthogonal transformation that converts a set of co-related variables to a set of unco-related variables
It is an unsupervised learning algorithm used to examine the iterations among a set of variables

Steps

I) Standardization

$$x = \frac{X - \mu}{\sigma}$$

II) Covariance Matrix Computation

$$\mathrm{cov}(x_1, x_2) = \sum_{i=1}^{n} \frac{(x_1 - \hat{x}_1)(x_{21} - \bar{x}_2)}{n-1}$$

III) Compute Eigen values & Eigen Vectors of covariance matrix to identify principal components

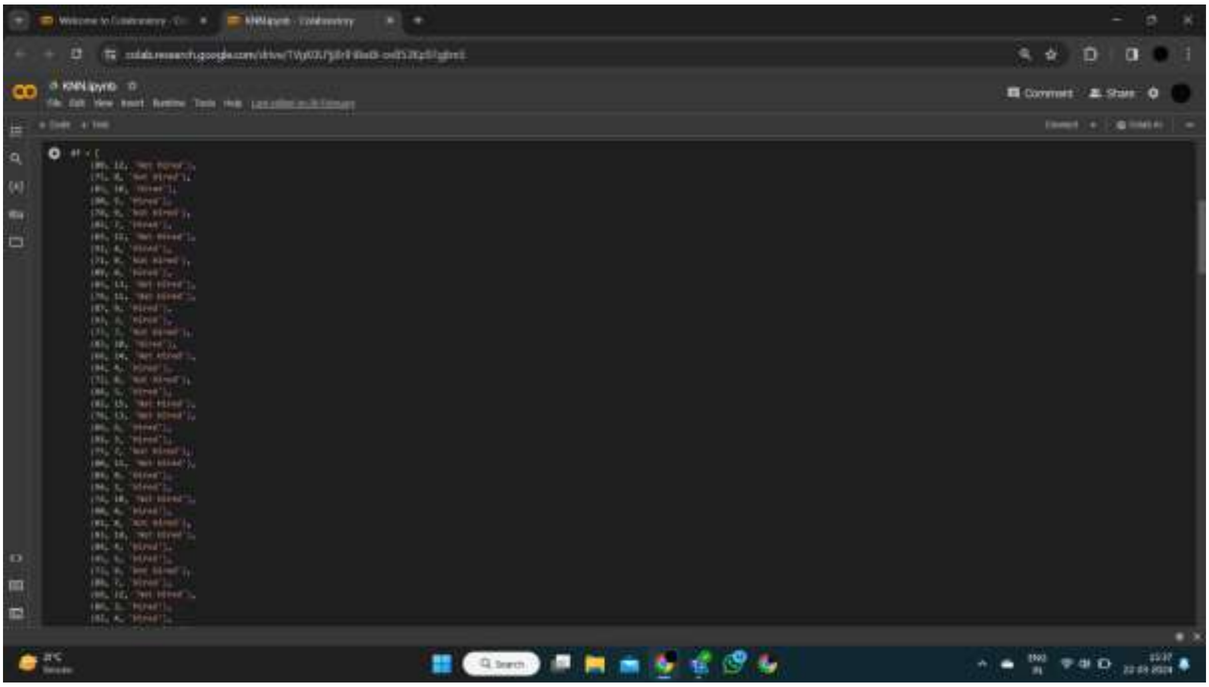
$$Ax = \lambda x$$

$$Ax - \lambda x = 0$$

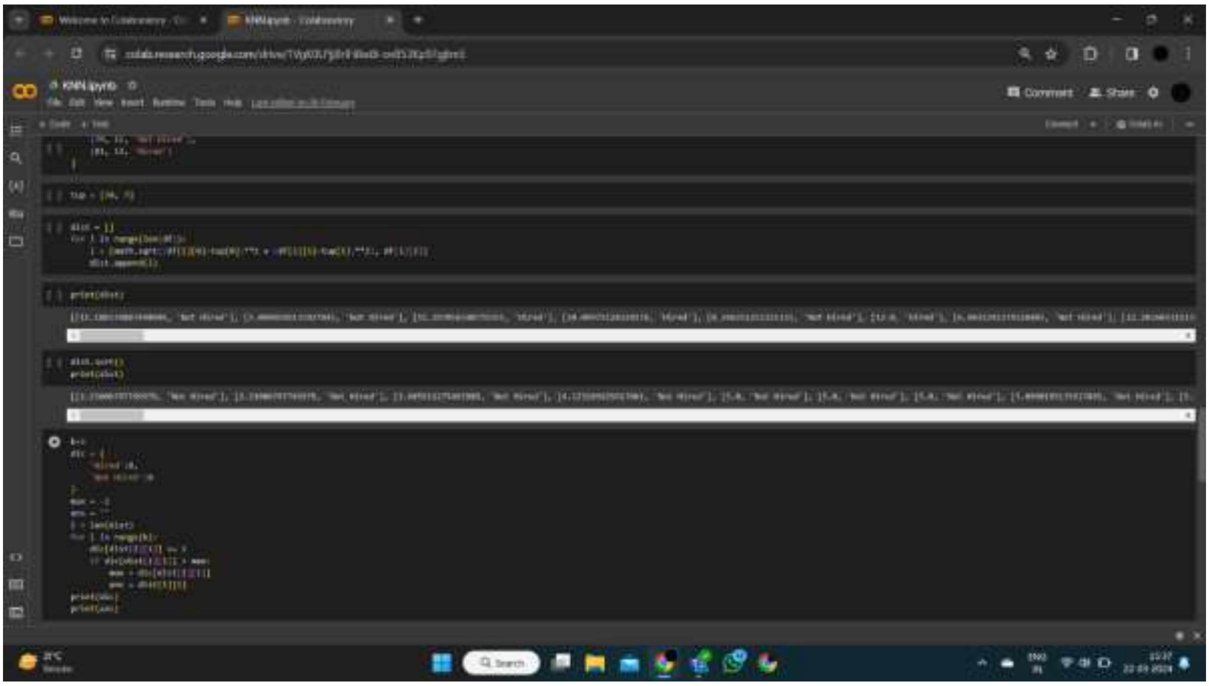
$$(A - \lambda I)x = 0$$

Conclusion : Thus we implement PCA

Implementation:



```
df = [
    [illegible]
```



```
    [illegible]
]
```

```
tup = [illegible]
```

```
dist = []
for i in range(len(df)):
    l = [math.sqrt((df[i][0]-tup[0])**2 + (df[i][1]-tup[1])**2), df[i][2]]
    dist.append(l)
```

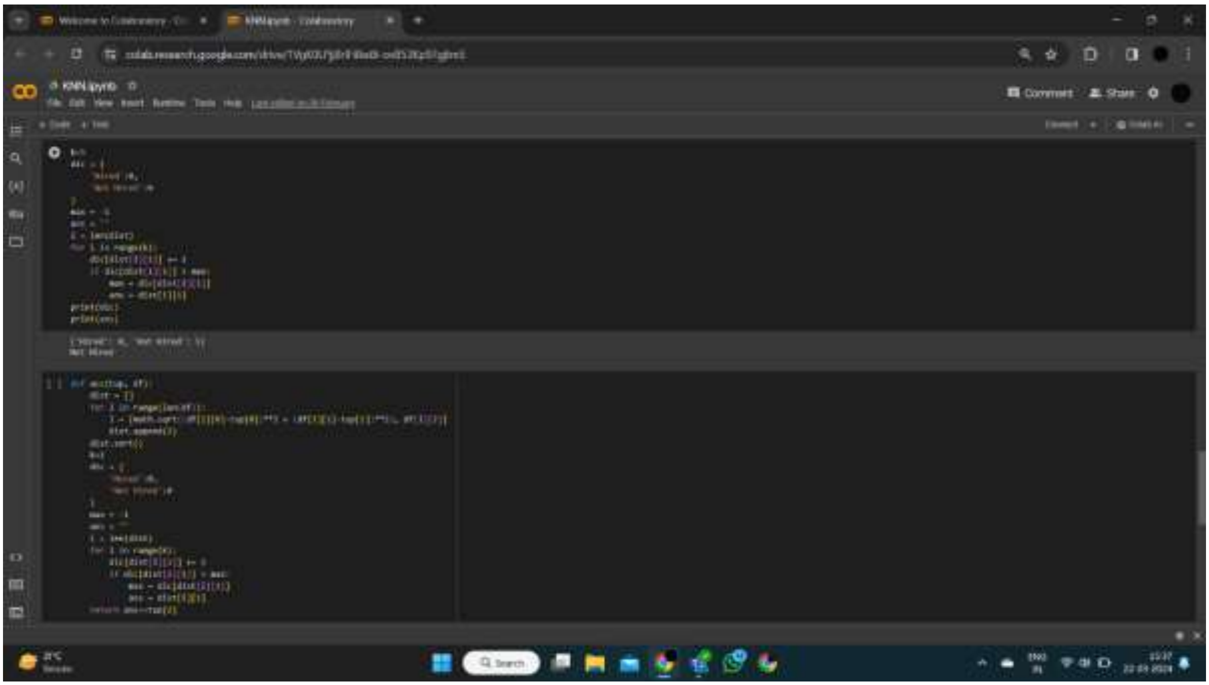
```
print(dist)
```

```
[illegible]
```

```
dist.sort()
print(dist)
```

```
[illegible]
```

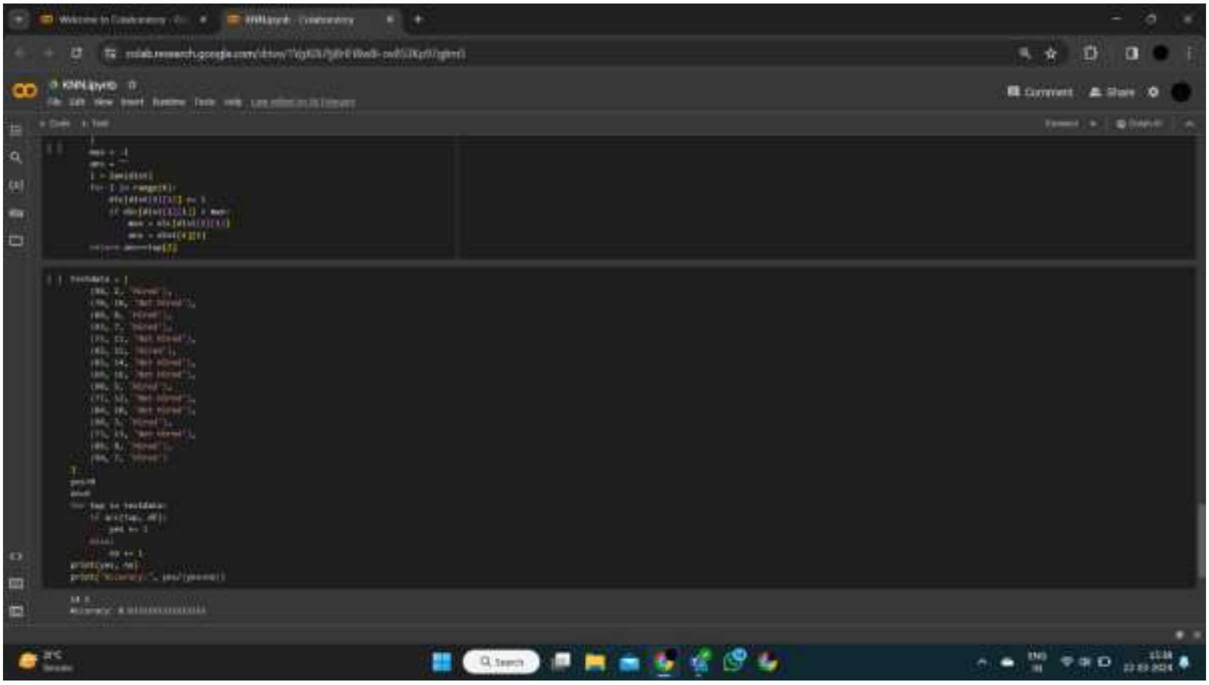
```
k=5
dic = {
    'Hired':0,
    'Not Hired':0
}
max = -1
ans = ""
l = len(dist)
for i in range(k):
    dic[dist[i][1]] += 1
    if dic[dist[i][1]] > max:
        max = dic[dist[i][1]]
        ans = dist[i][1]
print(dic)
print(ans)
```



```
k=5
dic = {
    'Hired' :0,
    'Not Hired' :0
}
max = -1
ans = ''
l = len(dist)
for i in range(k):
    dic[dist[i][1]] += 1
    if dic[dist[i][1]] > max:
        max = dic[dist[i][1]]
        ans = dist[i][1]
print(dic)
print(ans)
```

```
{'Hired': 0, 'Not Hired': 5}
Not Hired
```

```
def acc(tup, df):
    dist = []
    for i in range(len(df)):
        l = [math.sqrt((df[i][0]-tup[0])**2 + (df[i][1]-tup[1])**2), df[i][2]]
        dist.append(l)
    dist.sort()
    k=3
    dic = {
        'Hired' :0,
        'Not Hired' :0
    }
    max = -1
    ans = ''
    l = len(dist)
    for i in range(k):
        dic[dist[i][1]] += 1
        if dic[dist[i][1]] > max:
            max = dic[dist[i][1]]
            ans = dist[i][1]
    return ans==tup[2]
```



```
    }
    max = -1
    ans = ''
    l = len(dist)
    for i in range(k):
        dic[dist[i][1]] += 1
        if dic[dist[i][1]] > max:
            max = dic[dist[i][1]]
            ans = dist[i][1]
    return ans==tup[2]
```

```
testdata = [
    (90, 2, 'Hired'),
    (70, 10, 'Not Hired'),
    (80, 8, 'Hired'),
    (85, 7, 'Hired'),
    (71, 11, 'Not Hired'),
    (82, 11, 'Hired'),
    (65, 14, 'Not Hired'),
    (69, 16, 'Not Hired'),
    (90, 3, 'Hired'),
    (71, 12, 'Not Hired'),
    (64, 10, 'Not Hired'),
    (90, 5, 'Hired'),
    (73, 15, 'Not Hired'),
    (85, 8, 'Hired'),
    (90, 7, 'Hired')
]
yes=0
no=0
for tup in testdata:
    if acc(tup, df):
        yes += 1
    else:
        no += 1
print(yes, no)
print("Accuracy: ", yes/(yes+no))
```

```
14 1
Accuracy: 0.9333333333333333
```