SEP 786 – Artificial Intelligence and Machine Learning Fundamentals

Assignment 2 – Feature Selection

Submitted by-

Neelanjan Goswami

400414867

M.Eng Electrical and Computer Engineering

Faculty Advisor-

Jeff Fortuna

W Booth School of Engineering Practice and Technology

Question 1

```
import numpy as np

# Generating random 10 values and their mean
from random import randint
mean_1=[]
mean_2=[]

for i in range(1,11):
  value_1= randint(0,10)
  value_2= randint(0,10)
  mean_1.append(value_1)
  mean_2.append(value_2)
print(mean_1)
print(mean_2)
```

```
[6, 10, 8, 4, 3, 6, 3, 8, 6, 10]
[10, 10, 7, 2, 4, 7, 1, 1, 10, 9]
```

```
# Creating 10x10 diagonal matrices
import random
diag1=np.diag(random.sample(range(1,11),10))
diag2=np.diag(random.sample(range(1,11),10))
m=0


for m in range(5):
  diag1[random.randrange(10)][random.randrange(10)] = random.randrange(10)
  diag2[random.randrange(10)][random.randrange(10)] = random.randrange(10)
  m=-1
cov1=diag1
cov2=diag2

print(cov1)
print(cov1.shape)
print(cov2)
print(cov2.shape)
```

```
[[ 2  0  0  0  0  0  0  0  0  7]
 [ 0 10  0  0  0  0  0  0  0  0]
 [ 0  0  3  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0  0]
 [ 0  0  0  0  9  0  4  0  0  0]
 [ 0  0  0  0  0  7  0  0  0  0]
 [ 0  0  0  0  0  0  2  2  0  0]
 [ 0  0  0  0  0  0  0  6  0  0]
 [ 0  0  0  0  0  0  0  0  8  0]
 [ 0  0  0  0  0  0  0  0  0  4]]
(10, 10)
[[ 2  0  0  0  0  0  0  0  0  0]
 [ 0  7  0  7  0  0  0  0  0  0]
 [ 0  0  9  0  0  0  0  0  0  0]
 [ 0  0  0  4  0  0  0  0  0  0]
 [ 0  0  0  0  6  0  0  0  0  0]
 [ 0  0  0  0  0  1  0  0  0  0]
 [ 0  0  0  0  0  0  3  0  0  0]
 [ 0  0  0  0  0  0  6  8  0  0]
 [ 0  0  0  0  0  0  0  0 10  7]
 [ 0  0  0  0  0  0  0  0  0  5]]
(10, 10)
```

```
a, b, c, d, e, f, g, h, i, j = np.random.multivariate_normal(mean_1,cov1,10
00).T
A1 = np.vstack((a, b, c, d, e, f, g, h, i, j)).T
print(A1)
```

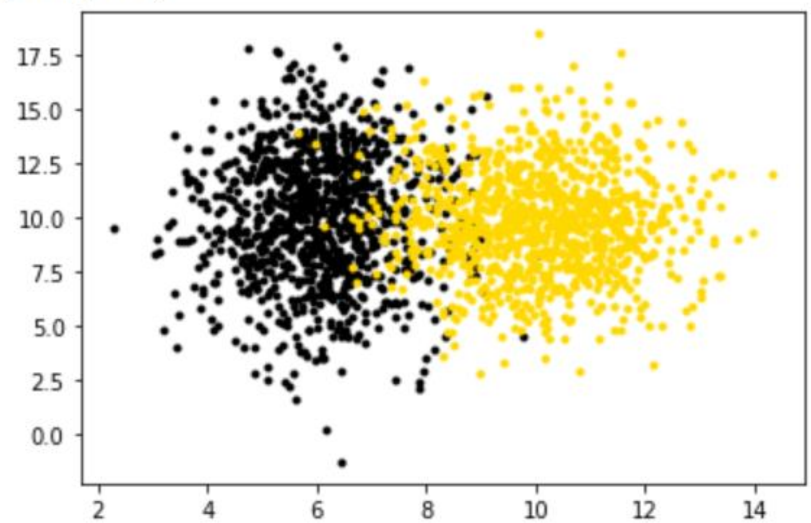
```
[[ 5.31158737  5.15147719  9.77967549 ...  8.80700886  7.95708908
  13.72535202]
 [ 6.48432782  4.82873433  4.98094872 ... 13.6524264   4.77382417
  11.61317204]
 [ 6.50786962  9.88679922  8.67363008 ...  7.80520884  9.17021511
   5.05118702]
 ...
 [ 6.5568581   4.50193167  7.12562788 ...  5.15936336  0.33642114
   6.08421411]
 [ 5.5469796   6.24844118  8.29678359 ...  4.83103248  7.00188091
   3.52276692]
 [ 6.32221963  7.35650786  8.85184681 ...  7.73578166  7.51383536
  16.65987277]]
```

```
k, l, m, n, o, p, q, r, s, t = np.random.multivariate_normal(mean_2,cov2,10
00).T
B1 = np.vstack((k, l, m, n, o, p, q, r, s, t)).T
print(B1)
```

```
[[ 9.28713771 11.57434864  1.91049679 ... -0.90926212  9.404487
   8.02090585]
 [ 9.61532929 11.15673549  9.78115384 ...  0.85661163  6.48629307
   4.43351632]
 [11.43422053  4.83879142 12.72942502 ...  2.24798472 16.65972923
   9.25192499]
 ...
 [ 9.03169528  5.82147569  3.31847786 ...  1.2879837  12.62979665
  17.60874117]
 [10.34469156 10.89908396 14.44002813 ...  1.1912543  13.02521783
   7.45860793]
 [12.47166205 13.41707981  7.50933557 ... -0.24136152 11.96410091
  13.30485188]]
```

```
import matplotlib.pyplot as plt
plt.scatter(A1[:,0],A1[:,1], c='black', marker='.')
plt.scatter(B1[:,0],B1[:,1], c='gold' , marker='.')
ans=np.concatenate((A1,B1),axis=0)
print(ans.shape)
```

```
(2000, 10)
```



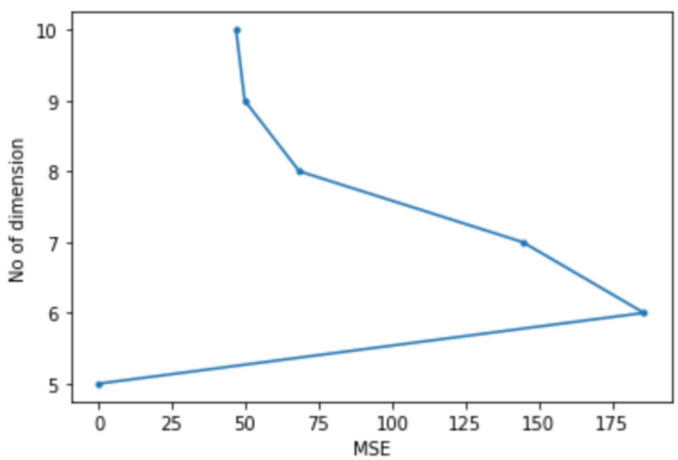
Question 2
(2A)

```
center_mean = ans - np.mean(ans, axis=0)
cov= np.cov(center_mean, rowvar = False)
eigen_vector,eigen_value= np.linalg.eigh(cov)
sorted_index = np.argsort(eigen_vector)[::-1]
new_ev = eigen_vector[sorted_index]
new_eig_vec = eigen_value[:, sorted_index]
reduced_Dataset = np.dot(ans, new_eig_vec)
```

```
MSE_VAL=[]
for k in range(9,4,-1):
  Post_pca = reduced_Dataset[:,0:k]
  Post_pca = np.dot(Post_pca, new_eig_vec[:,0:k].T)
  MSE= sum(sum((Post_pca - ans)**2))/len(ans)
  MSE_VAL.append(MSE)
MSE_VAL.append(0)
MSE_VAL= [x for x in reversed (MSE_VAL)]
print("MSE values = ",MSE_VAL)
x=MSE_VAL
```

```
MSE values =  [0, 185.76459109244732, 144.46877491224802, 68.41439224415188, 49.826613546197386, 46.896926927690224]
```

```
x=MSE_VAL

yplot = [5,6,7,8,9,10]
plt.plot(x,yplot,marker='.')
plt.ylabel("No of dimension")
plt.xlabel("MSE")
plt.show()
```



Question 2

(2B)

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.metrics import accuracy_score
scores=[]

Xc=np.zeros(1000)
Y=np.concatenate((Xc,np.ones(1000)))
for i in range(1, 10):
```

```
  X_train= ans[:1500,:-i]
  y_train= Y[:1500]
  X_test= ans[1500:,:-i]
  y_test= Y[1500:]
  model = LDA()
  model.fit(X_train , y_train)
  y_preds=model.predict(X_test)
  scores=np.append(scores,accuracy_score(y_test,y_preds))
  print(accuracy_score(y_test,y_preds))
```

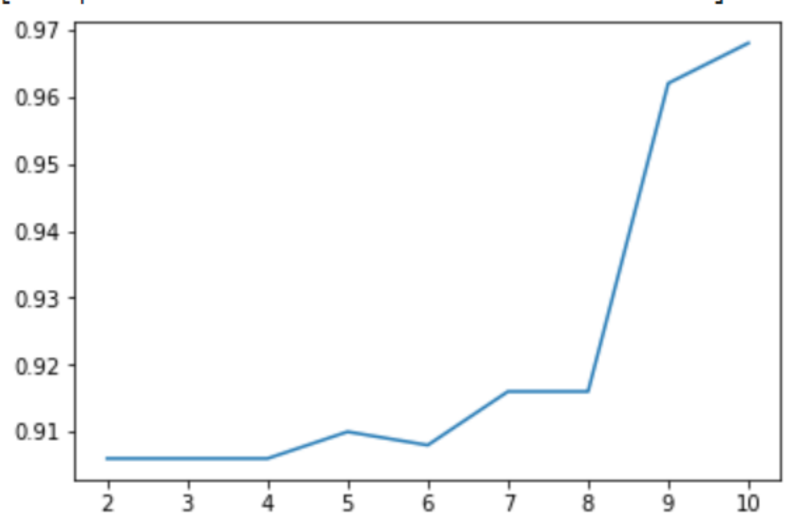
```
0.968
0.962
0.916
0.916
0.908
0.91
0.906
0.906
0.906
```

```
print(scores)
```

```
[0.968 0.962 0.916 0.916 0.908 0.91  0.906 0.906 0.906]
```

```
plt.plot([10,9,8,7,6,5,4,3,2],scores)
```

```
[<matplotlib.lines.Line2D at 0x7fca75eaf850>]
```



Question 3

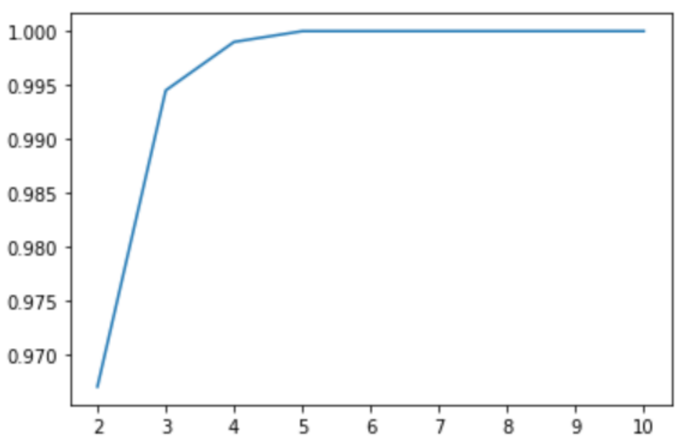
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import accuracy_score
LDA=LinearDiscriminantAnalysis()
```

```python
Xc=np.zeros(1000)
Y=np.concatenate((Xc,np.ones(1000)))
X_mod=ans.copy()
scores=np.array([])
scr_plt=np.array([])
for i in range(0,9):
  for j in range(0,X_mod.shape[1]):
    LDA.fit(np.delete(X_mod,j,1),Y.ravel())
    y_preds=LDA.predict(np.delete(X_mod,j,1))
    scores=np.append(scores,accuracy_score(Y.ravel(),y_preds))
  print("Number of of attributes removed:", i+1)
  print("Accuracies:", scores)
  drop=np.argsort(scores)[::-1]
  X_mod=np.delete(X_mod,drop[0],1)
  print("Accuracy after removing: ",scores[drop[0]],"\n")
  scr_plt=np.append(scr_plt,scores[drop[0]])
  scores=np.array([])
```

```
Number of of attributes removed: 1
Accuracies: [0.9515 0.9895 0.9885 0.989  0.9895 0.989  0.9895 0.9745 0.9875 0.987 ]
Accuracy after removing:  0.9895

Number of of attributes removed: 2
Accuracies: [0.9515 0.9895 0.9885 0.989  0.9895 0.989  0.9715 0.9875 0.987 ]
Accuracy after removing:  0.9895

Number of of attributes removed: 3
Accuracies: [0.9475 0.9895 0.989  0.988  0.989  0.97   0.9875 0.988 ]
Accuracy after removing:  0.9895

Number of of attributes removed: 4
Accuracies: [0.9445 0.989  0.988  0.989  0.9695 0.987  0.9865]
Accuracy after removing:  0.989

Number of of attributes removed: 5
Accuracies: [0.943 0.989 0.987 0.971 0.986 0.985]
Accuracy after removing:  0.989

Number of of attributes removed: 6
Accuracies: [0.9445 0.987  0.9705 0.9855 0.9855]
Accuracy after removing:  0.987

Number of of attributes removed: 7
Accuracies: [0.934  0.965  0.984  0.9855]
Accuracy after removing:  0.9855

Number of of attributes removed: 8
Accuracies: [0.927  0.9525 0.9805]
Accuracy after removing:  0.9805

Number of of attributes removed: 9
Accuracies: [0.9085 0.9335]
Accuracy after removing:  0.9335
```

```
plt.plot([10,9,8,7,6,5,4,3,2],scr_plt)
```

```
[<matplotlib.lines.Line2D at 0x7fca757de410>]
```



Question 4

The results show that when applied to data, Backward Feature selection provides more accuracy than PCA. The dimensions are decreased using PCA without taking the target variable into account. The greedy method of backward selection identifies the best characteristics.