Lab -8
PRML
Dimensionality reduction and Feature Selection

Name: Akshaykumar Kanani (B19EE008)

Q1:
ANSWER:

From the given iris data we do some preprocessing and got :

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

```
df_in.describe()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

And we also check if there is any NAN values in the data:

```
df_in.isna().sum()
```

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

And after that we standardize the value and got:

dtype: int64

```
X_std = StandardScaler().fit_transform(X)
X__std = pd.DataFrame(X_std)
X__std
```

	0	1	2	3
0	-0.900681	1.032057	-1.341272	-1.312977
1	-1.143017	-0.124958	-1.341272	-1.312977
2	-1.385353	0.337848	-1.398138	-1.312977
3	-1.506521	0.106445	-1.284407	-1.312977
4	-1.021849	1.263460	-1.341272	-1.312977
...	...	...	...	...
145	1.038005	-0.124958	0.819624	1.447956
146	0.553333	-1.281972	0.705893	0.922064
147	0.795669	-0.124958	0.819624	1.053537
148	0.432165	0.800654	0.933356	1.447956
149	0.068662	-0.124958	0.762759	0.790591

150 rows × 4 columns

And after that we find the covariance matrix and we got:

```
array([[ 1.    , -0.1094,  0.8718,  0.818 ],
       [-0.1094,  1.    , -0.4205, -0.3565],
       [ 0.8718, -0.4205,  1.    ,  0.9628],
       [ 0.818 , -0.3565,  0.9628,  1.    ]])
```

And then we got the eigenvalues and eigenvectors as follows:

```
[2.9304 0.9274 0.1483 0.0207]
```

```
[[ 0.5224 -0.3723 -0.721   0.262 ]
 [-0.2634 -0.9256  0.242  -0.1241]
 [ 0.5813 -0.0211  0.1409 -0.8012]
 [ 0.5656 -0.0654  0.6338  0.5235]]
```

And after that we we fit PCA method and got the below results:

we find the pca1 and pca2 for our data as:

	pca1	pca2
0	-2.684207	0.326607
1	-2.715391	-0.169557
2	-2.889820	-0.137346
3	-2.746437	-0.311124
4	-2.728593	0.333925
...	...	...
145	1.944017	0.187415
146	1.525664	-0.375021
147	1.764046	0.078519
148	1.901629	0.115877
149	1.389666	-0.282887

150 rows × 2 columns

And we also got some minor details as:

```
print(pca.explained_variance_ratio_)
```

```
[0.9246 0.053 ]
```

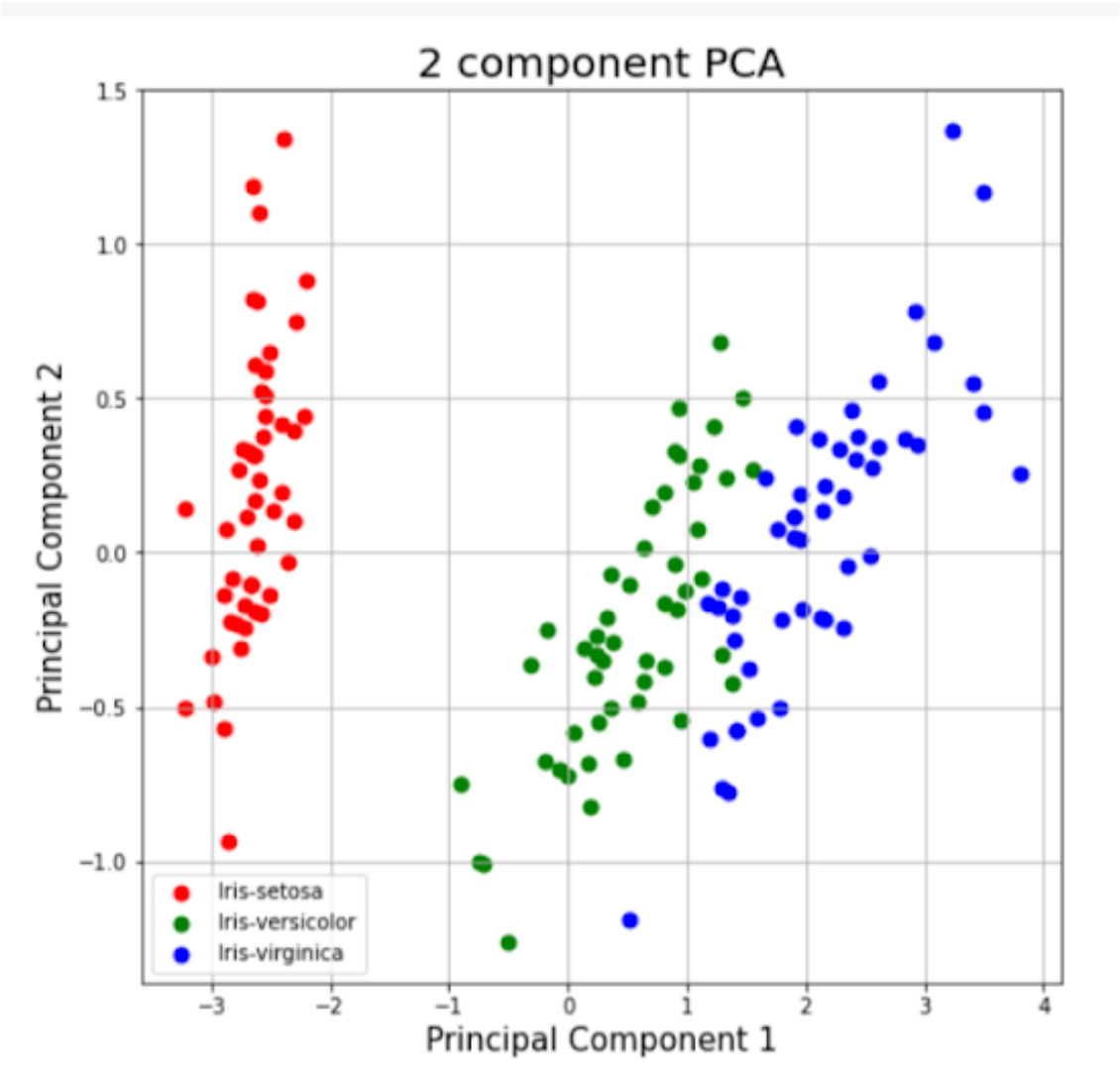
```
[74] print(pca.explained_variance_)
```

```
[4.2248 0.2422]
```

```
[75] print(pca.components_)
```

```
[[ 0.3616 -0.0823  0.8566  0.3588]
 [ 0.6565  0.7297 -0.1758 -0.0747]]
```

And after that we plot the plot as:



And from these all we can clearly see that for 90% data information preserve we only have to select one eigenvector.because it gives around 93%(this can also be varified from the graph in the code file). And our fiest eigen vector is:
[0.5224 -0.3723 -0.721 0.262] in 4D.
And we can clearly see that two direction are positive and two are negative.

So all the part are briefly disbribed above.

Q2:

ANSWER:

We skip the preprocessing part for this as we already did it above. (as same file for colab):
Then we did model fitting and got the LDA1 and LDA2 as:
And we also encode the ‘species’ With class column and got:

	LDA1	LDA2	class
0	8.084953	0.328454	0
1	7.147163	-0.755473	0
2	7.511378	-0.238078	0
3	6.837676	-0.642885	0
4	8.157814	0.540639	0

And after that we also did some data reaserch as:

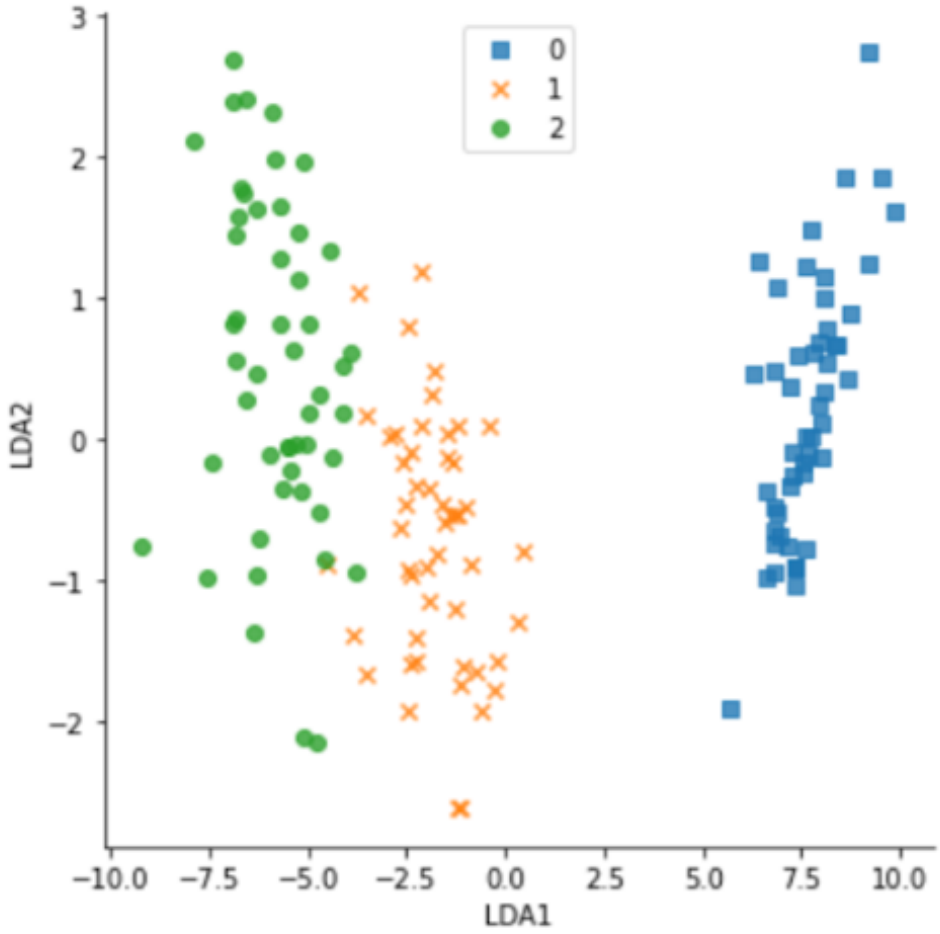
```
print("Training data shape",X_train_lda.shape)
```

```
Training data shape (150, 2)
```

```
[83] #Explaned variance ratio
     lda.explained_variance_ratio_ * 100
```

```
array([99.1472,  0.8528])
```

And we also draw the graph od this two components and got:



And after that, we also split(80,20) the data to use the Bayes method and got:
The accuracy= 0.9583333333333334
The Variance= 0.0013888888888888902
The std deviation= 0.03726779962499651

For part a:
From observing the two graphs we got for LDA and PCA we concluded that
In the case of uniformly distributed data, LDA almost always performs better than PCA. However, if the data is highly skewed then we will use PCA since LDA can be biased towards the majority class.Finally, it is beneficial that PCA can be applied to labeled as well as unlabeled data since it doesn't depend on the output classes. On the other hand, LDA requires output classes for finding linear discriminants and hence requires labeled data.

Q3:

ANSWER:

First, we did preprocess of data and got:

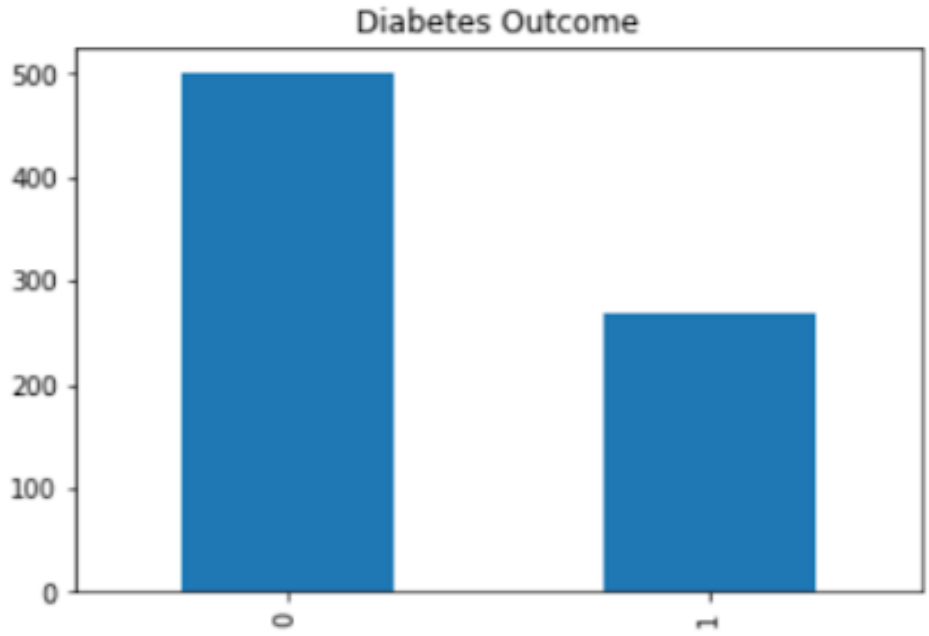
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

And:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

And we also count the count of each class and got:

```
0    500
1    268
Name: Outcome, dtype: int64
Text(0.5, 1.0, 'Diabetes Outcome')
```

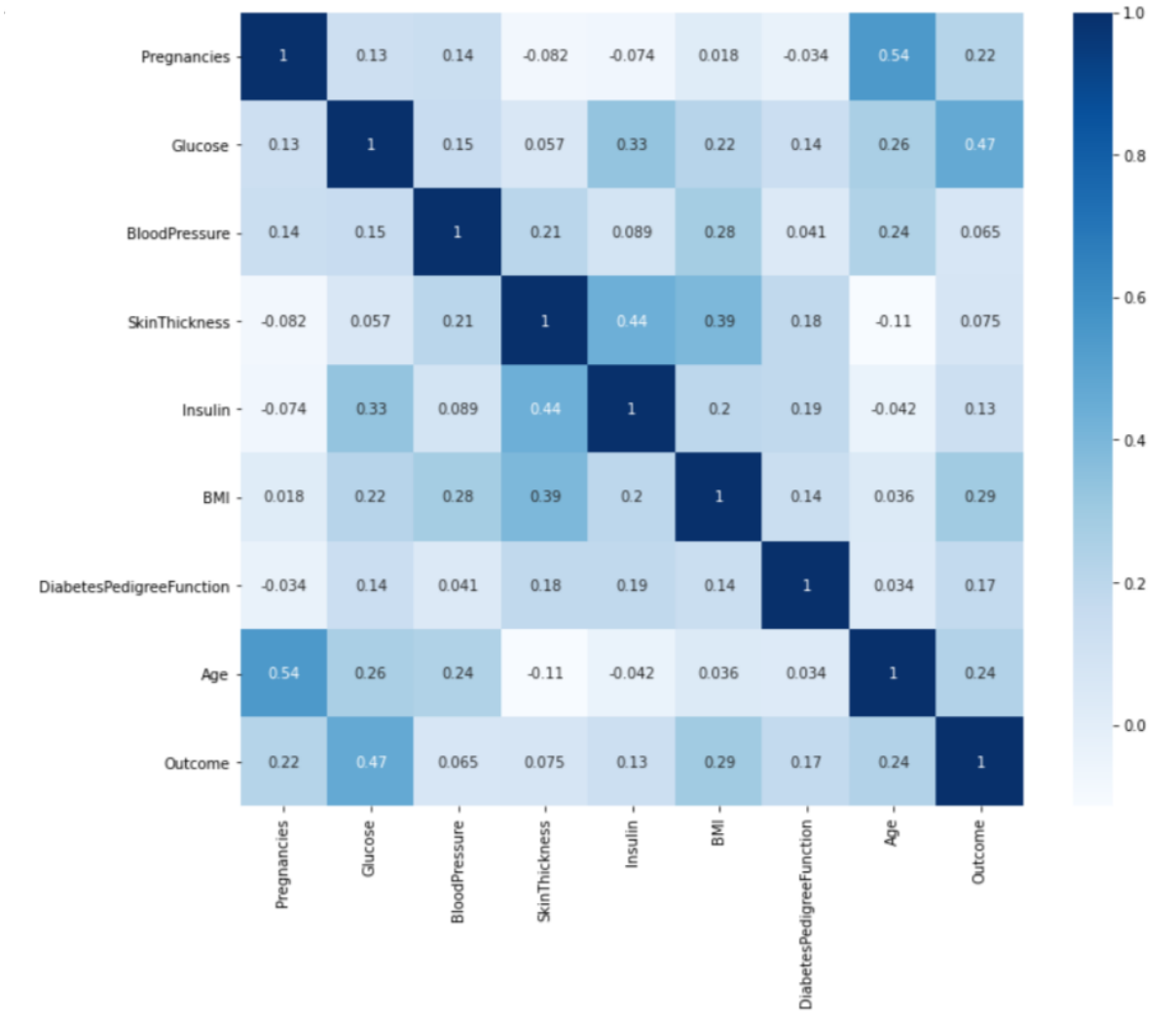


And: we also check if there is any missing value. And we got 0 missing valus as shown in each colomn:

```
df.isnull().sum()
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

After that we get the relation matrix as:



And we use select k best method and RFE method to choose the best 3 feathers and before fitting it to any classifier we also have to standadize it so after std we get:

```
array([[ 0.6399,  0.8483,  0.1496, ...,  0.204 ,  0.4685,  1.426 ],
       [-0.8449, -1.1234, -0.1605, ..., -0.6844, -0.3651, -0.1907],
       [ 1.2339,  1.9437, -0.2639, ..., -1.1033,  0.6044, -0.1056],
       ...,
       [ 0.343 ,  0.0033,  0.1496, ..., -0.7352, -0.6852, -0.2758],
       [-0.8449,  0.1598, -0.4707, ..., -0.2402, -0.3711,  1.1707],
       [-0.8449, -0.873 ,  0.0462, ..., -0.2021, -0.4738, -0.8714]])
```

We fit the selectkbets with chi2 function and got the result with

For randomforest
accuracy=0.7597402597402597
f1=[0.8141 0.6606]

For LogisticRegression
accuracy=0.7532467532467533
f1=[0.8061 0.6607]

And we also got the feacher priority as:
[111.5197 , 1411.887 , 17.6054 , 53.108 , 2175.565 ,127.6693 , 5.3927 ,181.3037]
respectively.

And after that we fit RFE model with the same two classifier and get the result as:

For LogisticRegression
Selected Features: [True True False False False True False False]
Feature Ranking: [1 1 3 6 5 1 2 4]

And for randomforest
Selected Features: [False True False False False True True False]
Feature Ranking: [4 1 3 6 5 1 1 2]

And after that we also see if their is any feacher with more than 70% relation but we didn't get any, as result we got empty array.
We can also variety this fact from the correlation plot above.

Thank you