Lab -8

PRML
AY 2020-21 Trimester - III

Dimensionality reduction and Feature Selection J Jahnavi(B19CSE109)

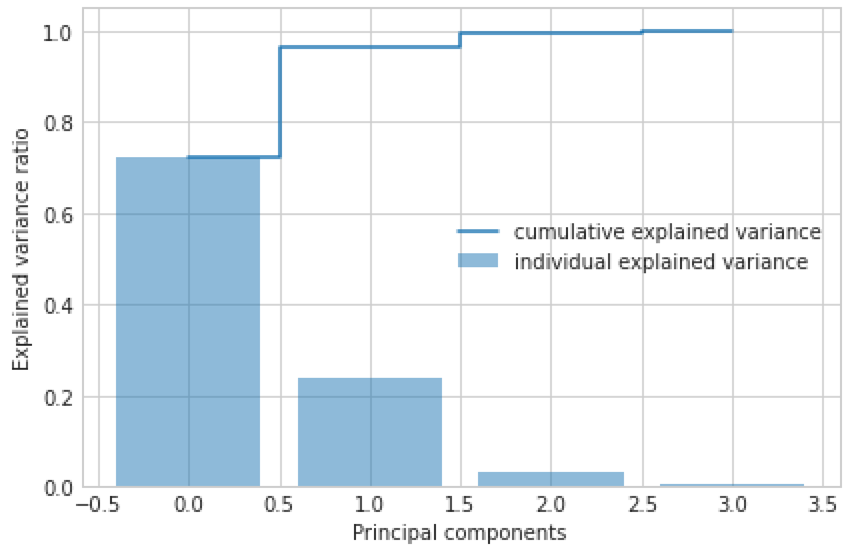
PCA analysis

- Standardization of the data
  Variables measured at different scales do not contribute equally to the model fitting & model learned function and might create a bias. Thus, to deal with this potential problem feature-wise standardized (μ=0, σ=1) is usually used before model fitting.

```
from sklearn.preprocessing import StandardScaler
#object
scaler = StandardScaler()
# Fit on training set only.
scaler.fit(X_train)
# Apply transform to both the training set and the test set.
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
X = scaler.transform(X)
```

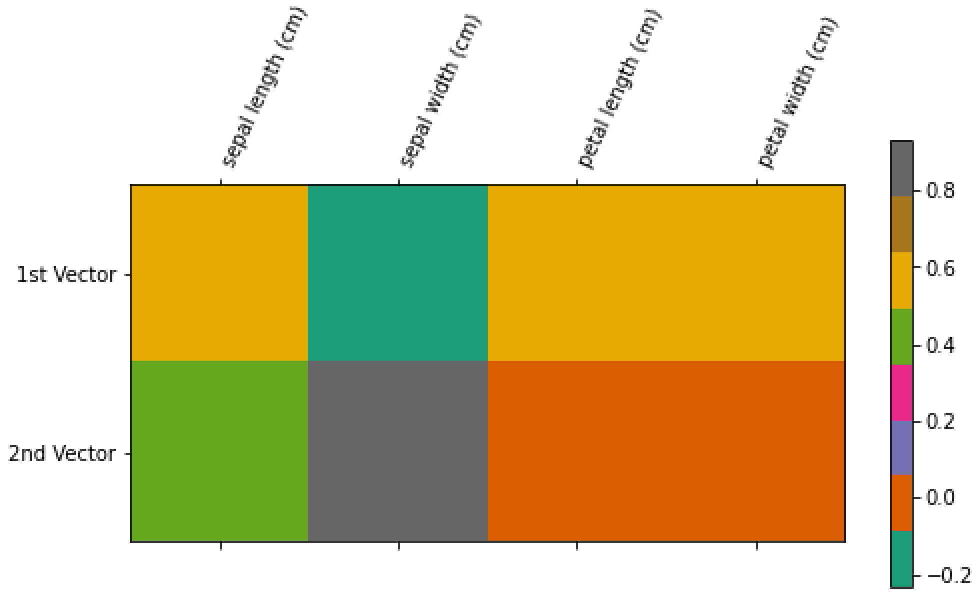
- Principal Component Analysis

Without setting the value of n_componets:

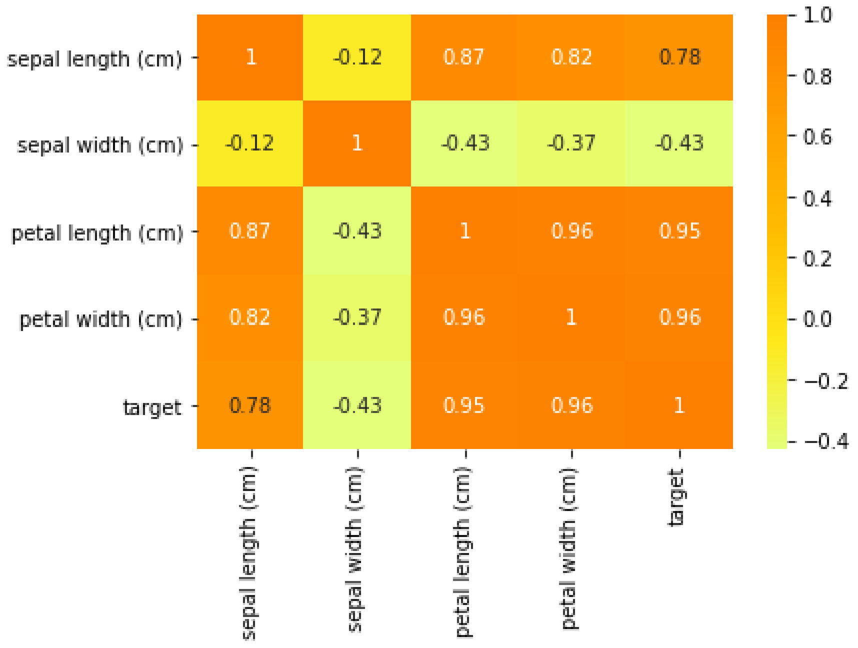


The variance captured by the first 2 features when arranged in descending order contribute to more than 90% of the total variance by the features. Thus, we can eliminate the remaining 2 features as they don't contribute much to the overall variance.

- Dimensions that are primary contributors to the first eigenvector

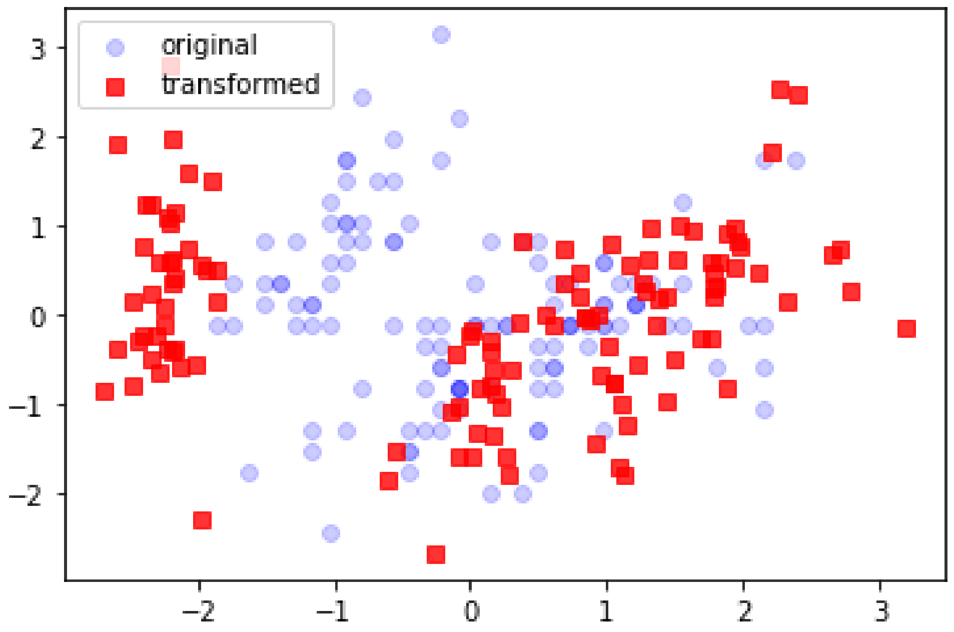


The above correlation plot shows that the primary contributors to the first eigenvector petal length (cm) and sepal length (cm), both are positively correlated to the eigenvector and each other as follows:



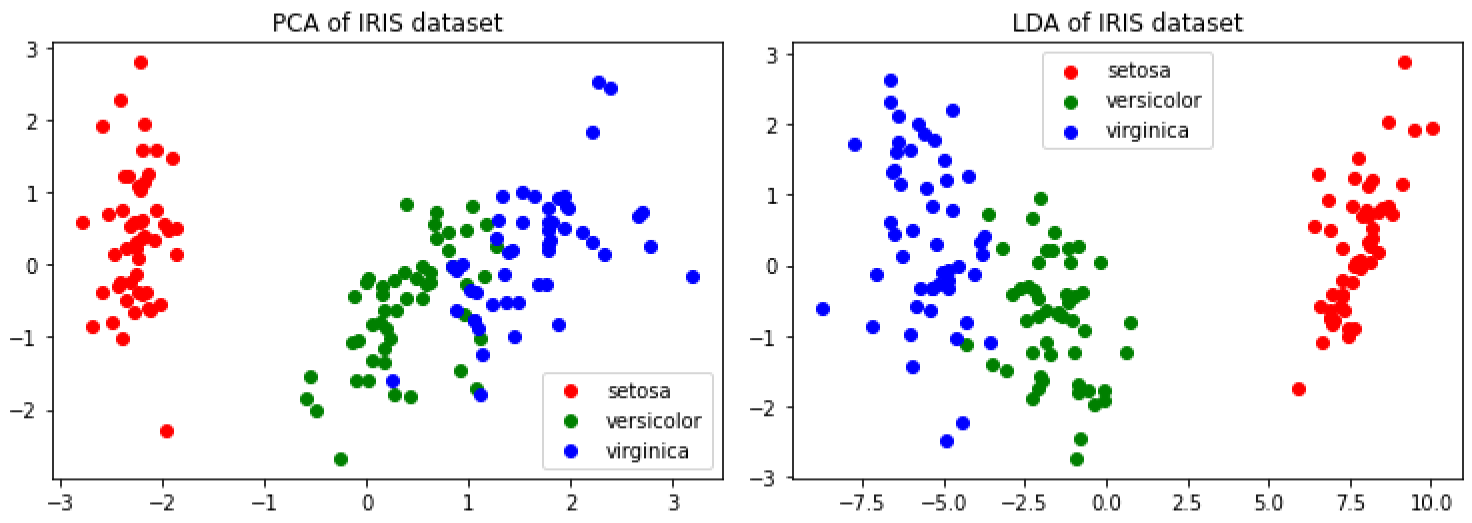
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
sepal length (cm)	1	-0.12	0.87	0.82	0.78
sepal width (cm)	-0.12	1	-0.43	-0.37	-0.43
petal length (cm)	0.87	-0.43	1	0.96	0.95
petal width (cm)	0.82	-0.37	0.96	1	0.96
target	0.78	-0.43	0.95	0.96	1

- The plot of the transformed data using the first two eigenvectors is as follows:

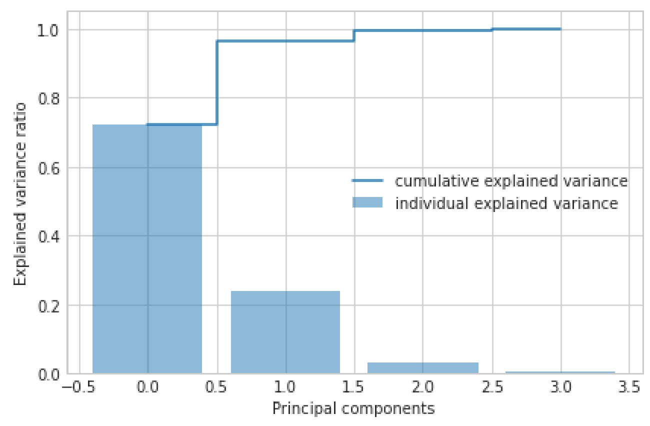
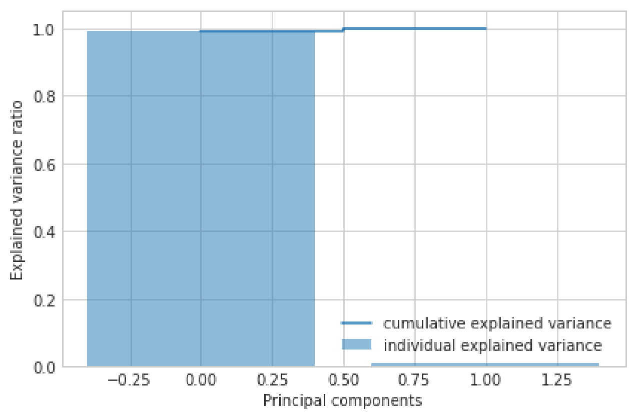


PCA and LDA comparison

- Plots for distribution of samples using the first 2 principal components and the first 2 linear discriminants:



- Comparing PCA and LDA

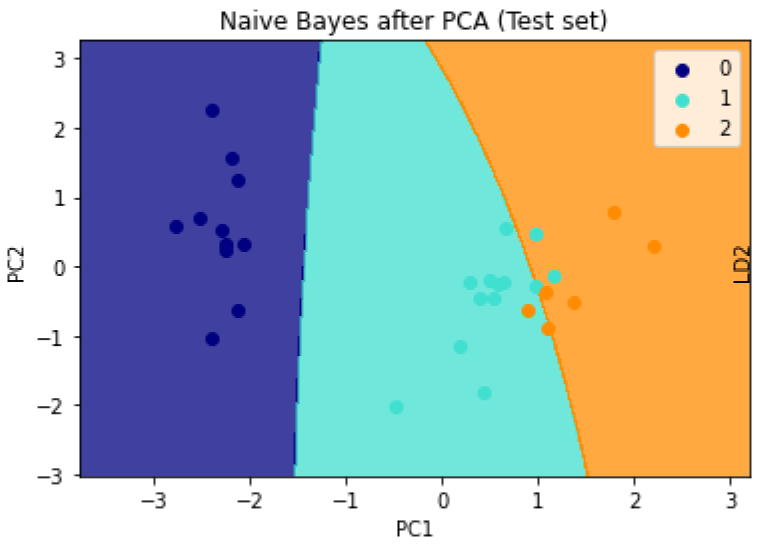
PCA	LDA
	

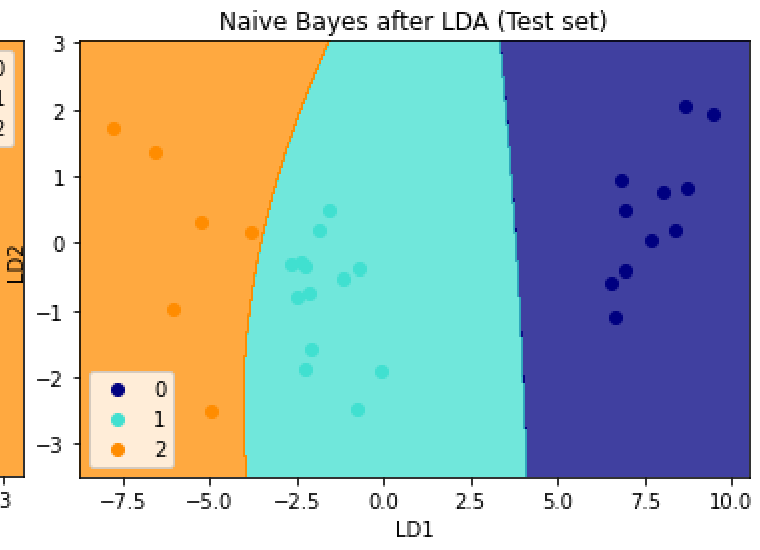
- Learn a Bayes classifier using the original features and compare its performance with the features obtained in part:

	Without Dimensionality reduction	PCA	LDA
score()	0.9666666666666667	0.8666666666666667	1.0

Confusion Matrices	`array([[11,  0,  0],`
	`       [ 0, 13,  0],`
	`       [ 0,  1,  5]])`
		`array([[11,  0,  0],`
		`       [ 0, 11,  2],`
		`       [ 0,  2,  4]])`
			`array([[11,  0,  0],`
			`       [ 0, 13,  0],`
			`       [ 0,  0,  6]])`

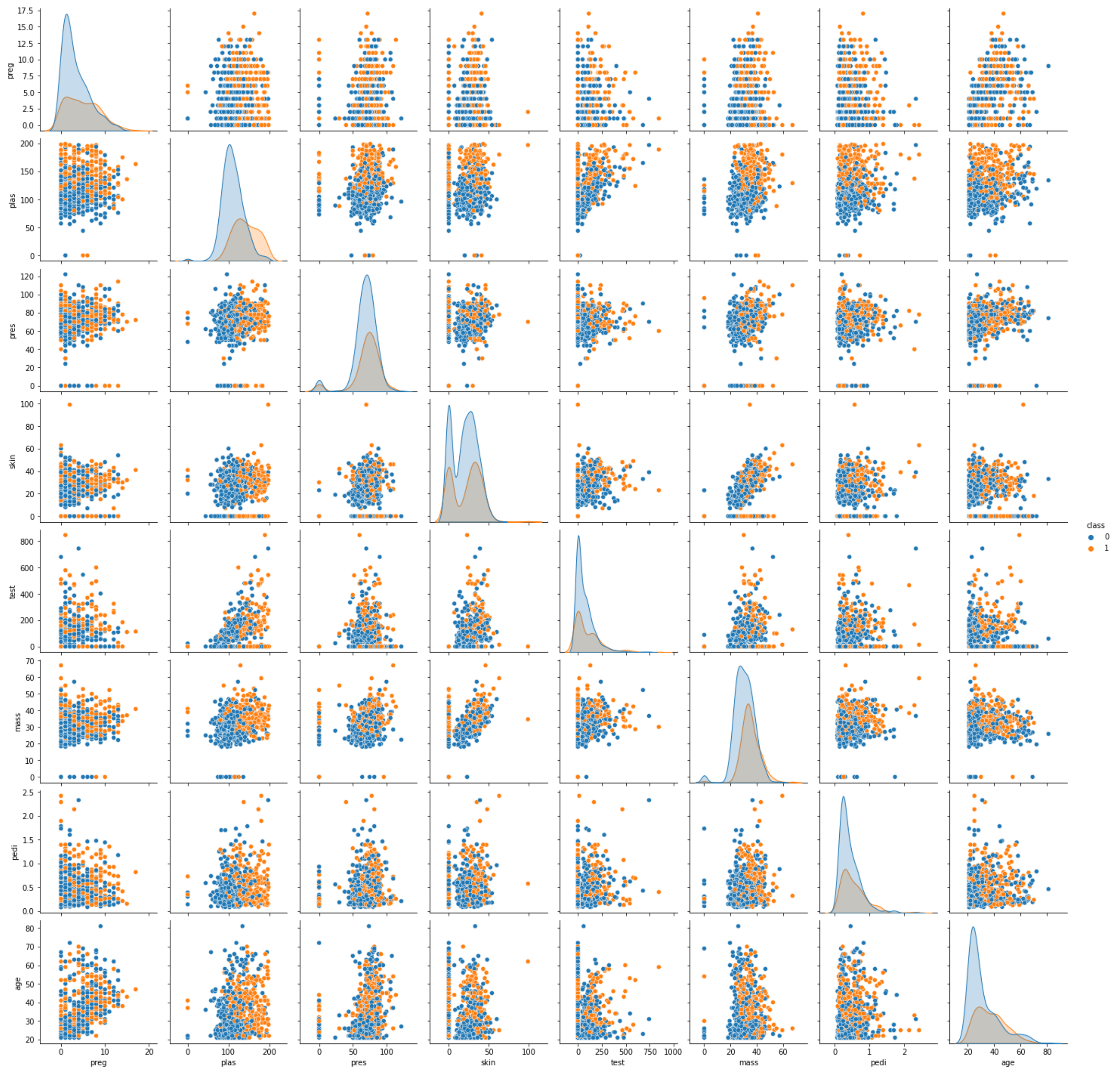
- Visualizing the test results:

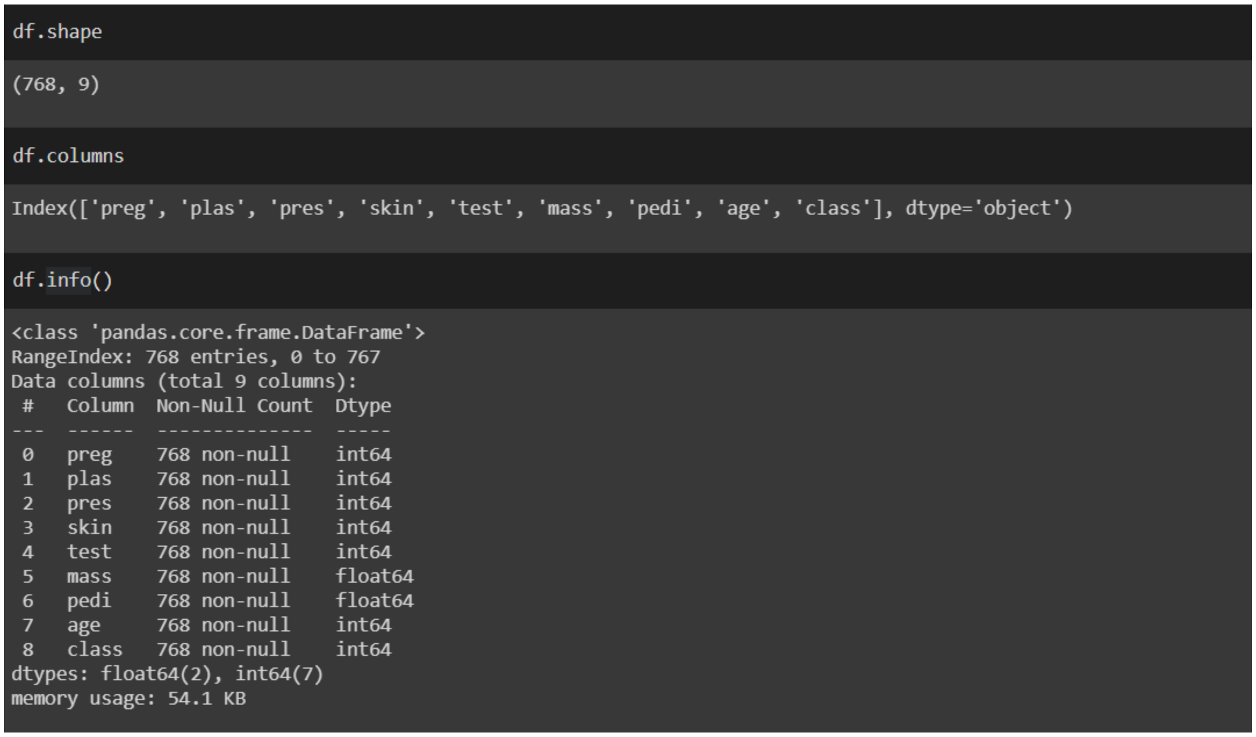




Feature Selection

- Preprocessing and Exploratory Data Analysis:





```
df.shape
```

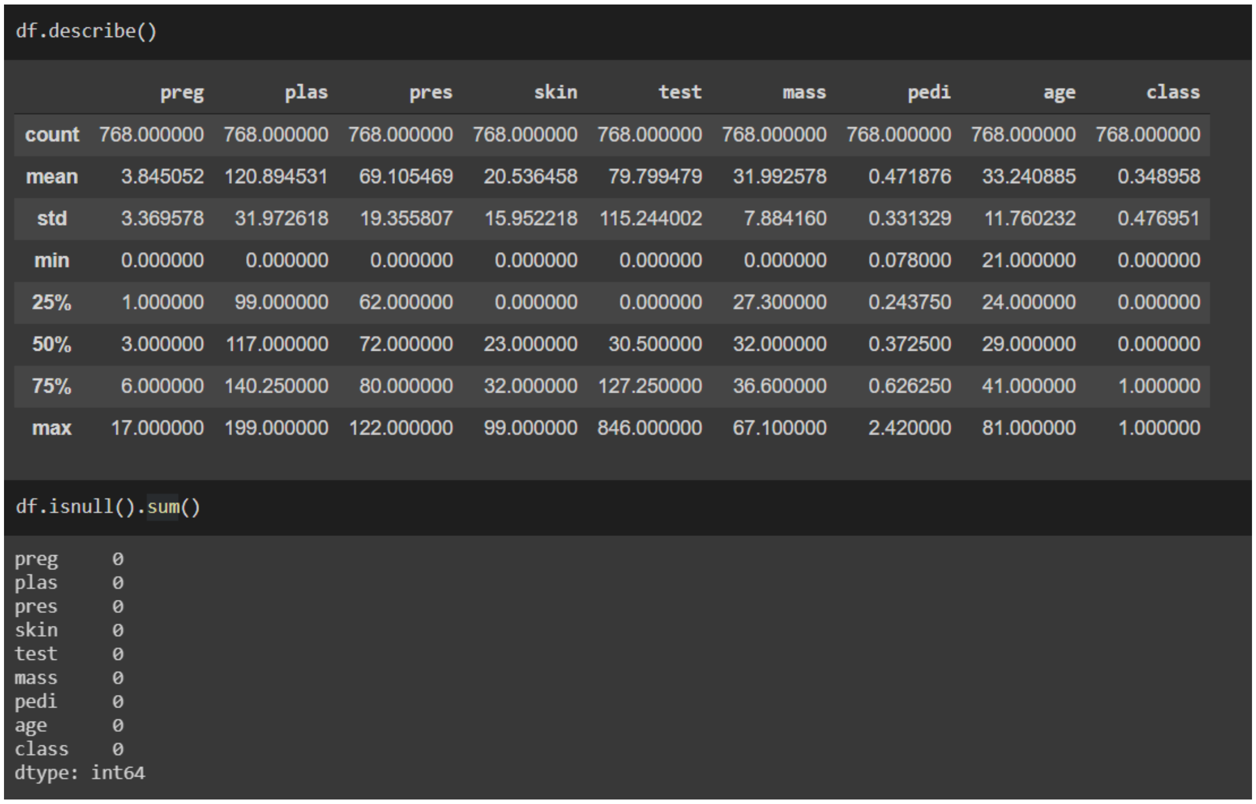
```
(768, 9)
```

```
df.columns
```

```
Index(['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class'], dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   preg    768 non-null    int64  
 1   plas    768 non-null    int64  
 2   pres    768 non-null    int64  
 3   skin    768 non-null    int64  
 4   test    768 non-null    int64  
 5   mass    768 non-null    float64
 6   pedi    768 non-null    float64
 7   age     768 non-null    int64  
 8   class   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```



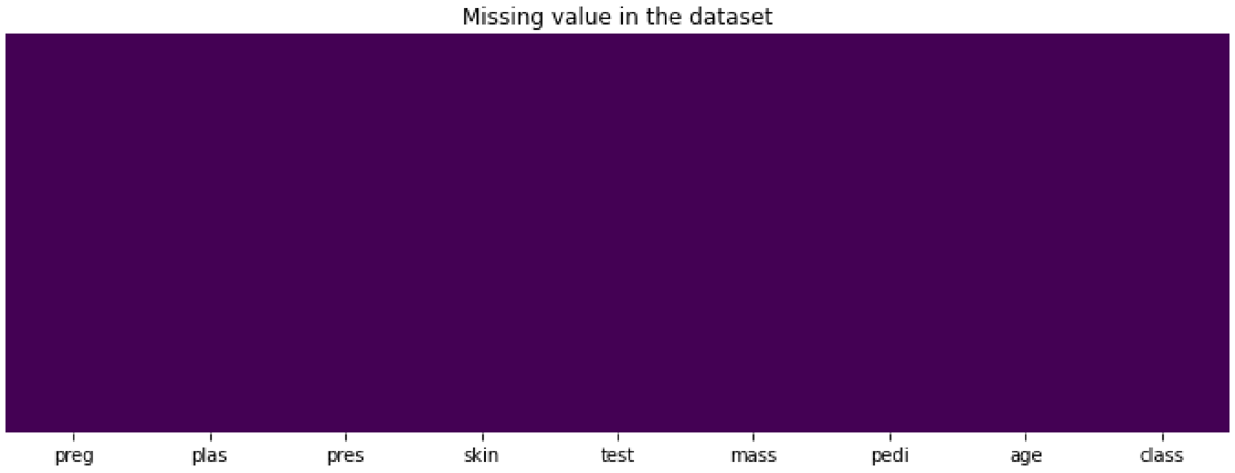
```
df.describe()
```

	preg	plas	pres	skin	test	mass	pedi	age	class
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

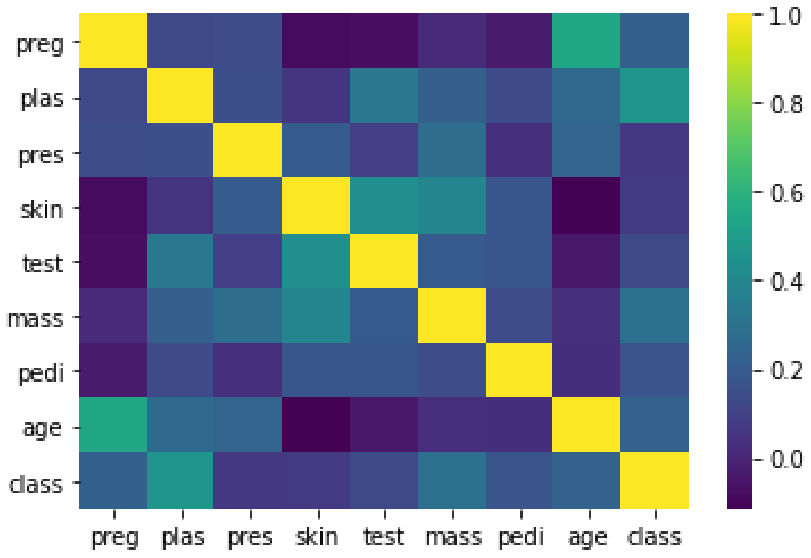
```
df.isnull().sum()
```

```
preg     0
plas     0
pres     0
skin     0
test     0
mass     0
pedi     0
age      0
class    0
dtype: int64
```

The plot for missing data:

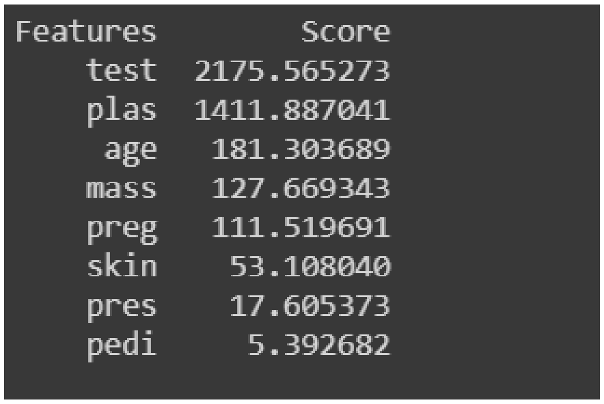


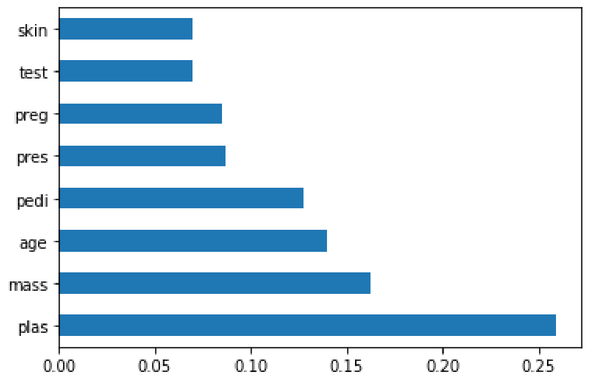
Correlation Plot:



- Features having high significance using both of the methods

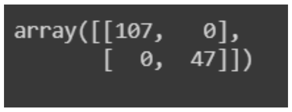
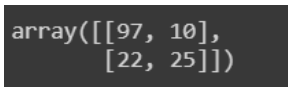
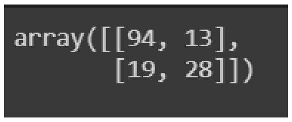
Model	Feature Scores	Highly significant

Chi-squared statistical test	
	```
	Features         Score
	    test  2175.565273
	    plas  1411.887041
	     age   181.303689
	    mass   127.669343
	    preg   111.519691
	    skin    53.108040
	    pres    17.605373
	    pedi     5.392682
	```
		['test', 'plas', 'age', 'mass']

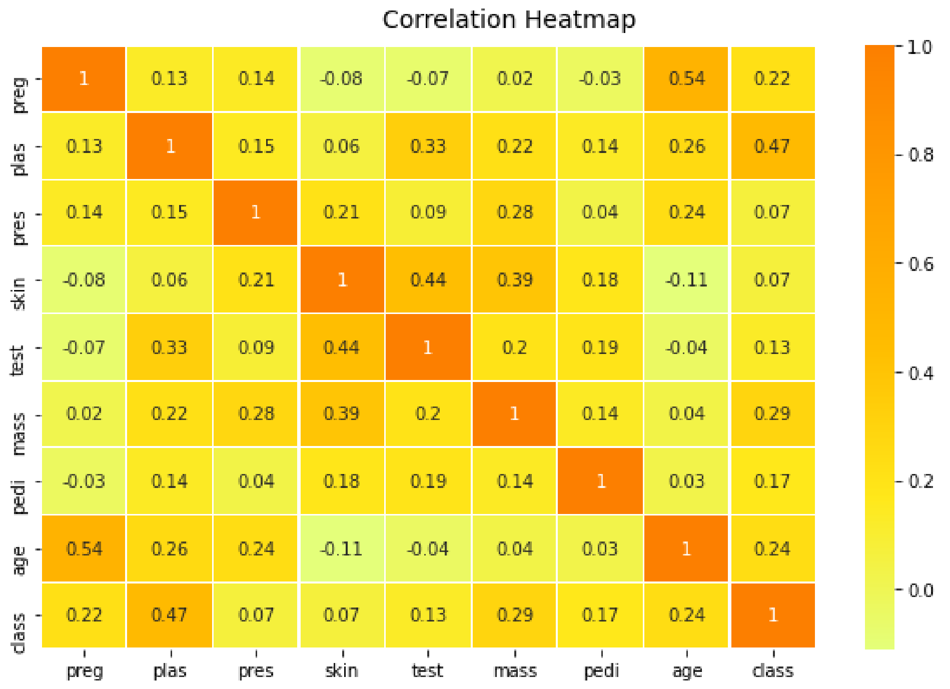
Random Generation		['plas', 'mass', 'pedi', 'age']

- Accuracies and F1 Scores based comparison

It turns out that the model trained with all features performed better than the other two while the models with selected features perform with similar efficiencies.

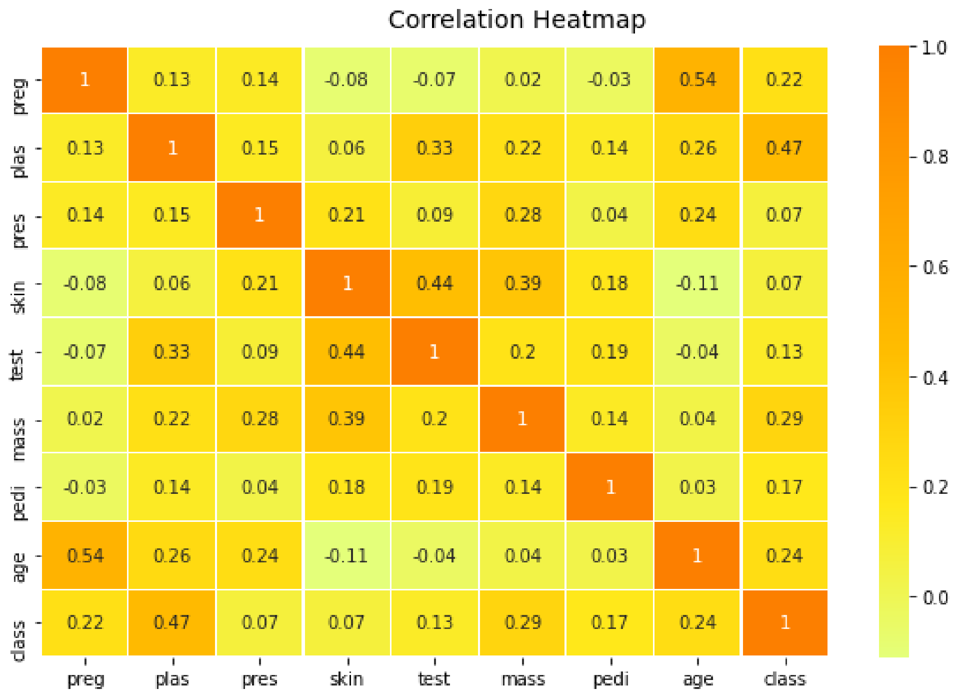
Model	F1 Score	Accuracy	
With all features	1.0	1.0	 `array([[107,   0],` `       [  0,  47]])`
Chi-squared statistical test	0.6363636363636364	0.7922077922077922	 `array([[97, 10],` `       [22, 25]])`
Random Generation	0.6097560975609756	0.7922077922077922	 `array([[94, 13],` `       [19, 28]])`

- Correlation Matrix



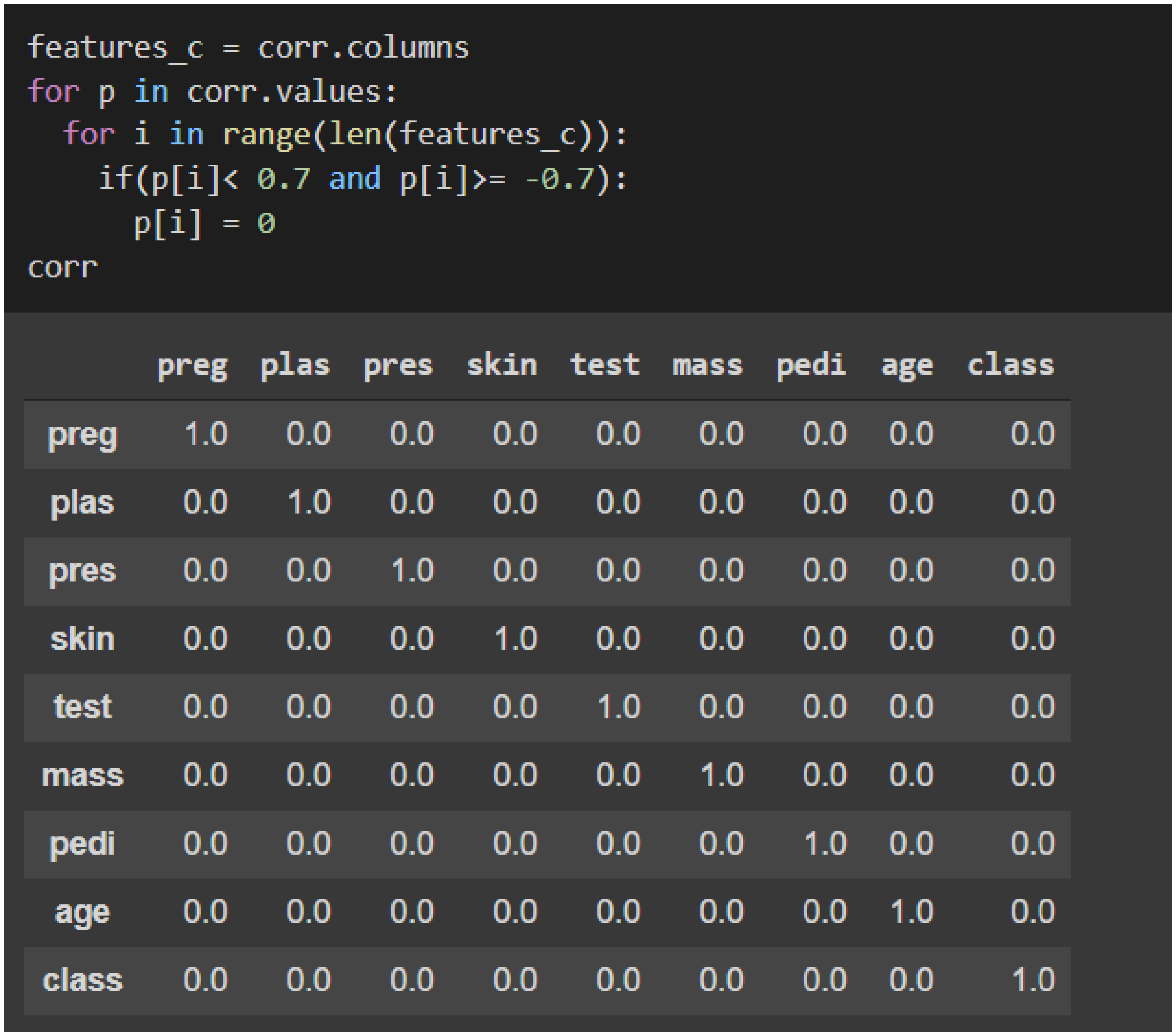
- Compute correlated features with a threshold of 70%.

We have to find out the correlation between the features and remove the features which have a correlation coefficient greater than a certain limit (here 70% is the threshold)
Plot after applying the threshold:



- Updating the correlation for features within the threshold:

This shows that all the features are correlated with a threshold of 70%



```
features_c = corr.columns
for p in corr.values:
  for i in range(len(features_c)):
    if(p[i]< 0.7 and p[i]>= -0.7):
      p[i] = 0
corr
```

	preg	plas	pres	skin	test	mass	pedi	age	class
preg	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
plas	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
pres	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
skin	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
test	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
mass	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
pedi	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
age	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
class	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0