

Predictions using Feature Reduction Techniques PCA and LDA

In [11]:

```
1 import pandas as pd
2 import numpy as np
3 from sklearn.decomposition import PCA
4 from sklearn.model_selection import train_test_split
5 from sklearn.naive_bayes import GaussianNB
6 from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, cohen_kappa_score
7
8 #csv file
9 url = 'C:/Users/Prerna/Desktop/ML_jupyter_notenooks/datasets/titanic.csv'
10
11 #Creating a dataframe
12 dataframe = pd.read_csv(url).fillna(0)
13
14 #DATA CLEANING
15 #Dropping columns
16 dataframe = dataframe.drop('Name',axis=1)
17 dataframe = dataframe.drop('SexCode',axis=1)
18
19 # Create mapper
20 pclass_mapper = {"1st":1,"2nd":2,"3rd":3}
21 gender_mapper = {"male":1,"female":2}
22
23 # Replace feature values with scale
24 dataframe["PClass"] = dataframe["PClass"].replace(pclass_mapper)
25 dataframe["Sex"] = dataframe["Sex"].replace(gender_mapper)
26
27 #Replacing missing values of Age with mean of age
28 dataframe["Age"] = np.where(dataframe['Age']==0,np.mean(dataframe['Age']),dataframe['Age'])
29
30 #print(dataframe)
31 # Input features
32 x = dataframe.iloc[:, :3].values
33 # Output class
34 y = dataframe.iloc[:, 3].values
35
36
37 '''
38 xtrain = training features
39 xtest = testing features
40 ytrain = classes of training data
41 ytest = classes of testing data
```

```

42  '''
43  #Splitting the data into training and test
44  xtrain, xtest, ytrain, ytest = train_test_split( x, y, test_size = 0.3, random_state = 0)
45
46  #Applying PCA
47  pca = PCA(n_components = 0.99)    #(variance = 0.99,0.95)
48
49  # apply feature Reduction
50  x_train = pca.fit_transform(xtrain)
51  x_test = pca.transform(xtest)
52
53  print("Original number of features:", x.shape[1])
54  print("Reduced number of features:", x_test.shape[1])
55
56  #Creating Model
57  gnb = GaussianNB()
58  #Fitting a model
59  gnb.fit(x_train, ytrain)
60
61  #Perform Predictions
62  y_pred = gnb.predict(x_test)
63
64  #Confusion Matrix
65  cm = confusion_matrix(ytest, y_pred)
66
67  print ("Confusion Matrix : \n", cm)
68  error_rate = 1 - accuracy_score(ytest, y_pred)
69  print ("Classification Report:\n")
70  print(classification_report(ytest,y_pred))
71  print("Accuracy: \n", accuracy_score(ytest, y_pred))
72  print("Error Rate: \n",error_rate )
73  print("Kappa Score: \n", cohen_kappa_score(ytest, y_pred))
74
75

```

Original number of features: 3

Reduced number of features: 1

Confusion Matrix :

```
[[229  35]
```

```
[111  19]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.67	0.87	0.76	264
1	0.35	0.15	0.21	130
accuracy			0.63	394
macro avg	0.51	0.51	0.48	394
weighted avg	0.57	0.63	0.58	394

Accuracy:

0.6294416243654822

Error Rate:

0.37055837563451777

Kappa Score:

0.015943615710962145

In [12]:

```
1 import pandas as pd
2 import numpy as np
3 from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
4 from sklearn.model_selection import train_test_split
5 from sklearn.ensemble import RandomForestClassifier
6 from sklearn.naive_bayes import GaussianNB
7 from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, cohen_kappa_score
8
9
10 #csv file
11 url = 'C:/Users/Prerna/Desktop/ML_jupyter_notenooks/datasets/titanic.csv'
12
13 #Creating a dataframe
14 dataframe = pd.read_csv(url).fillna(0)
15
16 #DATA CLEANING
17 #Dropping columns
18 dataframe = dataframe.drop('Name', axis=1)
19 dataframe = dataframe.drop('SexCode', axis=1)
20
21 # Create mapper
22 pclass_mapper = {"1st":1, "2nd":2, "3rd":3}
23 gender_mapper = {"male":1, "female":2}
24
25 # Replace feature values with scale
26 dataframe["PClass"] = dataframe["PClass"].replace(pclass_mapper)
27 dataframe["Sex"] = dataframe["Sex"].replace(gender_mapper)
28
29 #Replacing missing values of Age with mean of age
30 dataframe["Age"] = np.where(dataframe["Age"]==0, np.mean(dataframe["Age"]), dataframe["Age"])
31
32
33 # Input features
34 x = dataframe.iloc[:, :3].values
35 # Output class
36 y = dataframe.iloc[:, 3].values
37
38
39 ...
40 xtrain = training features
41 xtest = testing features
```

```

42 ytrain = classes of training data
43 ytest = classes of testing data
44 '''
45 #Splitting the data into training and test
46
47 xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.3, random_state = 0)
48
49 #Using LDA
50 lda = LDA(n_components=None)
51
52 # Transform the features
53 x_train = lda.fit_transform(xtrain,ytrain)
54 x_test = lda.transform(xtest)
55
56
57 print("Original number of features:", x.shape[1])
58 print("Reduced number of features:", x_test.shape[1])
59
60 #Create a model
61 gnb = GaussianNB()
62
63 #Fit a model
64 gnb.fit(x_train, ytrain)
65
66 #Perform Predictions
67 y_pred = gnb.predict(x_test)
68
69 #Confusion Matrix
70 cm = confusion_matrix(ytest, y_pred)
71
72 print ("Confusion Matrix : \n", cm)
73 error_rate = 1 - accuracy_score(ytest, y_pred)
74 print ("Classification Report:\n")
75 print(classification_report(ytest,y_pred))
76 print("Accuracy: \n", accuracy_score(ytest, y_pred))
77 print("Error Rate: \n",error_rate )
78 print("Kappa Score: \n", cohen_kappa_score(ytest, y_pred))
79

```

Original number of features: 3
Reduced number of features: 1

Confusion Matrix :

```
[[223  41]
```

```
[ 41  89]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.84	0.84	0.84	264
1	0.68	0.68	0.68	130
accuracy			0.79	394
macro avg	0.76	0.76	0.76	394
weighted avg	0.79	0.79	0.79	394

Accuracy:

0.7918781725888325

Error Rate:

0.20812182741116747

Kappa Score:

0.5293123543123543

In []: ▶

1