Machine Learning Lab Case Study

A LAB REPORT

Submitted by:-

MD SHAHIL 20010136

in partial fulfillment for the award of the

degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

Department of Computer Science & Engineering

C.V. RAMAN GLOBAL UNIVERSITY
BHUBANESWAR-ODISHA-752054

APRIL 2023

ACKNOWLEDGEMENT

I would like to articulate our deep gratitude to my project Guide **Dr. Debendra Muduli & Santosh Sharma, Professor, Department of Computer Science & Engineering,** who has always been source of motivation and firm support for carrying out the project.

I would also like to convey my sincerest gratitude and indebtedness to all other faculty members and staff of Department of **Computer Science & Engineering**, who bestowed their great effort and guidance at appropriate times without it would have been very difficult on my project work.

An assemblage of this nature could never have been attempted with our reference to and inspiration from the works of others whose details are mentioned in references section. I acknowledge our indebtedness to all of them. Further, I would like to express my feeling towards my parents and God who directly or indirectly encouraged and motivated us during Assertion.

CASE STUDY

Q1. Implement two dataset Exp1 and Exp2 using KNN, BPNN, Kernel SVM, Random Forest, Adaboost Random Forest, Adaboost SVM, XGboost with and without PCA. Then findout itsaccuracy, confusion matrix and ROC curve.

SOURCE CODE:-

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

In [2]:
```
data1 = pd.read_csv('Exp_1.csv')
data2 = pd.read_csv('Exp_2.csv')
```

In [3]:
```
data1.shape, data2.shape
```

Out[3]:
```
((1040, 97), (660, 97))
```

In [4]:
```
X1 = data1.iloc[: , :-1]

y1 = data1.iloc[: , -1]
```

In [5]:
```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
y1 = label_encoder.fit_transform(y1)
y1
```

Out[5]:
```
array([1, 1, 1, ..., 0, 0, 0], dtype=int64)
```

M,mm

In [6]:
```
X2 = data2.iloc[: , :-1]

y2 = data2.iloc[: , -1]
```

In [7]:
```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
y2 = label_encoder.fit_transform(y2)
```

In [8]:
```
from sklearn.preprocessing import Normalizer
norm = Normalizer()
columns1 = X1.columns
columns2 = X2.columns
X1 = norm.fit_transform(X1)
X1 = pd.DataFrame(X1, columns = columns1)
X2 = norm.fit_transform(X2)
X2 = pd.DataFrame(X2, columns = columns2)
```

In [9]:
```
X1.describe()
```

Out[9]:

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	...	F87	
count	1040.000000	1040.000000	1040.000000	1040.000000	1040.000000	1040.000000	1040.000000	1040.000000	1040.000000	1040.000000	...	1.040000e+03	1
mean	0.006010	0.017594	0.028439	0.172880	0.152109	0.165377	0.000954	0.000139	0.025268	0.009156	...	4.201056e-04	4
std	0.009270	0.011847	0.024708	0.045482	0.053191	0.051855	0.002727	0.000543	0.010903	0.004561	...	7.744505e-03	7
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	7.290779e-07	1
25%	0.000177	0.008713	0.005664	0.151358	0.128152	0.136601	0.000057	0.000013	0.017324	0.005661	...	2.576797e-06	4
50%	0.000682	0.015704	0.023690	0.164454	0.142248	0.159535	0.000082	0.000020	0.026512	0.008992	...	3.353329e-06	5
75%	0.008706	0.026260	0.045800	0.178801	0.159251	0.179050	0.000127	0.000030	0.032473	0.011983	...	4.778700e-06	6
max	0.052368	0.054929	0.133256	0.376966	0.384592	0.370908	0.028490	0.007428	0.088380	0.032303	...	1.443376e-01	1

8 rows × 96 columns

In [10]:
```
X2.replace(0, np.nan, inplace = True)
```

In [11]:
```
X2.dropna(axis = 1, inplace = True)
```

In [12]:
```
from sklearn.model_selection import train_test_split
X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, y1, test_size = 0.2, random_state = 0)
```

In [13]:
```
X_train1.shape, X_test1.shape, y_train1.shape, y_test1.shape
```

Out[13]:
```
((832, 96), (208, 96), (832,), (208,))
```

In [14]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [15]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
from xgboost import XGBClassifier
```

In [16]:
```
models = list()

models.append(('KNN', KNeighborsClassifier(n_neighbors=2)))
models.append(('BPNN', MLPClassifier()))
models.append(('SVC', SVC(kernel='poly', probability=True)))
models.append(('RF', RandomForestClassifier(n_estimators=100, max_features="auto",random_state=0)))
models.append(('AdaB', AdaBoostClassifier(n_estimators=100)))
models.append(('XGB', XGBClassifier()))
```

In [17]:
```
models = pd.DataFrame(models, columns = ['Name', 'Classifier'])
models.set_index('Name', inplace = True)
models
```

Out[17]:

Name	Classifier
KNN	KNeighborsClassifier(n_neighbors=2)
BPNN	MLPClassifier()
SVC	SVC(kernel='poly', probability=True)
RF	RandomForestClassifier(random_state=0)
AdaB	AdaBoostClassifier(n_estimators=100)
XGB	XGBClassifier(base_score=None, booster=None, c...

For Exp_1 DataSet WithOut PCA

In [18]:
```
from sklearn.metrics import classification_report,confusion_matrix
```

In [19]:
```
for classifier in models.Classifier:

    classifier.fit(X_train1,y_train1)
    pred = classifier.predict(X_test1)

    print(classifier)
    print('\n')
    print('CONFUSION MATRIX')
    print(confusion_matrix(y_test1,pred))
    print('\nCLASSIFICATION REPORT')
    print(classification_report(y_test1,pred))
```

KNeighborsClassifier(n_neighbors=2)

CONFUSION
MATRIX[[148 10]
[46 4]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.76	0.94	0.84	158
1	0.29	0.08	0.12	50
accuracy			0.73	208
macro avg	0.52	0.51	0.48	208
weighted avg	0.65	0.73	0.67	208

MLPClassifier()

CONFUSION MATRIX
[[158 0]
[50 0]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.76	1.00	0.86	158
1	0.00	0.00	0.00	50
accuracy			0.76	208
macro avg	0.38	0.50	0.43	208
weighted avg	0.58	0.76	0.66	208

SVC(kernel='poly', probability=True)

CONFUSION MATRIX
[[158 0]
[50 0]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.76	1.00	0.86	158
1	0.00	0.00	0.00	50
accuracy			0.76	208
macro avg	0.38	0.50	0.43	208
weighted avg	0.58	0.76	0.66	208

RandomForestClassifier(random_state=0)

CONFUSION MATRIX
[[138 20]
[45 5]]

CLASSIFICATION

REPORT

	precision	recall	f1-score	support
0	0.75	0.87	0.81	158
1	0.20	0.10	0.13	50
accuracy			0.69	208
macro avg	0.48	0.49	0.47	208
weighted avg	0.62	0.69	0.65	208

AdaBoostClassifier(n_estimators=100)

CONFUSION MATRIX[[136 22]
[46 4]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.75	0.86	0.80	158
1	0.15	0.08	0.11	50
accuracy			0.67	208
macro avg	0.45	0.47	0.45	208
weighted avg	0.60	0.67	0.63	208

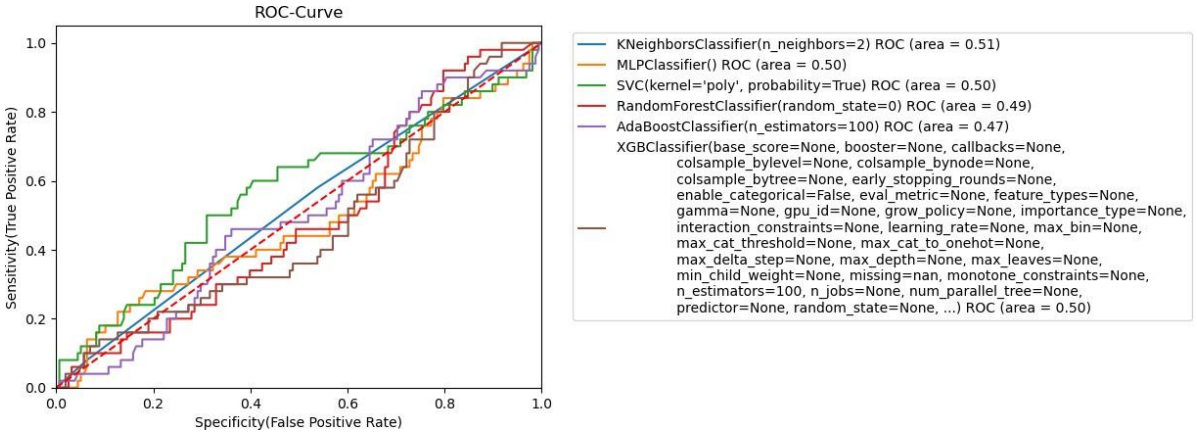
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None,
feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None,
importance_type=None,
              interaction_constraints=None, learning_rate=None,
max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan,
monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,predictor=None,
              random_state=None, ...)
```

CONFUSION MATRIX[[132 26]
[42 8]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.76	0.84	0.80	158
1	0.24	0.16	0.19	50
accuracy			0.67	208
macro avg	0.50	0.50	0.49	208
weighted avg	0.63	0.67	0.65	208

In [20]:
```
from sklearn import metrics
for model in models.Classifier:
    model.fit(X_train1, y_train1) # train the model
    y_pred=model.predict(X_test1) # predict the test data
# Compute False postive rate, and True positive rate
    fpr, tpr, thresholds = metrics.roc_curve(y_test1, model.predict_proba(X_test1)[:,1])
# Calculate Area under the curve to display on the plot
    auc = metrics.roc_auc_score(y_test1,model.predict(X_test1))
# Now, plot the computed values
    plt.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % (model, auc))
# Custom settings for the plot
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Specificity(False Positive Rate)')
plt.ylabel('Sensitivity(True Positive Rate)')
plt.title('ROC-Curve')
plt.legend(bbox_to_anchor = (1.05, 1), loc = 2)
plt.show()   # Display
```



For Exp_2 Dataset

In [21]:
```
models1 = pd.DataFrame(models)
models1
```

Out[21]:

Name	Classifier
KNN	KNeighborsClassifier(n_neighbors=2)
BPNN	MLPClassifier()
SVC	SVC(kernel='poly', probability=True)
RF	(DecisionTreeClassifier(max_features='auto', r...
AdaB	(DecisionTreeClassifier(max_depth=1, random_st...
XGB	XGBClassifier(base_score=None, booster=None, c...

In [22]:
```
from sklearn.model_selection import train_test_split
X_train2, X_test2, y_train2, y_test2 = train_test_split(X2, y2, test_size = 0.2, random_state = 0)
```

In [23]:
```
for classifier in models.Classifier:

    classifier.fit(X_train2,y_train2)
    pred = classifier.predict(X_test2)

    print(classifier)
    print('\n')
    print('CONFUSION MATRIX')
    print(confusion_matrix(y_test2,pred))
    print('\nCLASSIFICATION REPORT')
    print(classification_report(y_test2,pred))
```

KNeighborsClassifier(n_neighbors=2)

CONFUSION MATRIX[[98 0]
[0 34]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	1.00	1.00	1.00	98
1	1.00	1.00	1.00	34
accuracy			1.00	132
macro avg	1.00	1.00	1.00	132
weighted avg	1.00	1.00	1.00	132

MLPClassifier()

CONFUSION MATRIX[[98 0]
[34 0]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.74	1.00	0.85	98
1	0.00	0.00	0.00	34
accuracy			0.74	132
macro avg	0.37	0.50	0.43	132
weighted avg	0.55	0.74	0.63	132

SVC(kernel='poly', probability=True)

CONFUSION MATRIX
[[98 0]
[24 10]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.80	1.00	0.89	98
1	1.00	0.29	0.45	34
accuracy			0.82	132
macro avg	0.90	0.65	0.67	132
weighted avg	0.85	0.82	0.78	132

RandomForestClassifier(random_state=0)

CONFUSION MATRIX
[[98 0]
[0 34]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	1.00	1.00	1.00	98
1	1.00	1.00	1.00	34
accuracy			1.00	132
macro avg	1.00	1.00	1.00	132
weighted avg	1.00	1.00	1.00	132

AdaBoostClassifier(n_estimators=100)

CONFUSION MATRIX
[[98 0]
[0 34]] CLASSIFICATION

REPORT

	precision	recall	f1-score	support
0	1.00	1.00	1.00	98
1	1.00	1.00	1.00	34
accuracy			1.00	132
macro avg	1.00	1.00	1.00	132
weighted avg	1.00	1.00	1.00	132

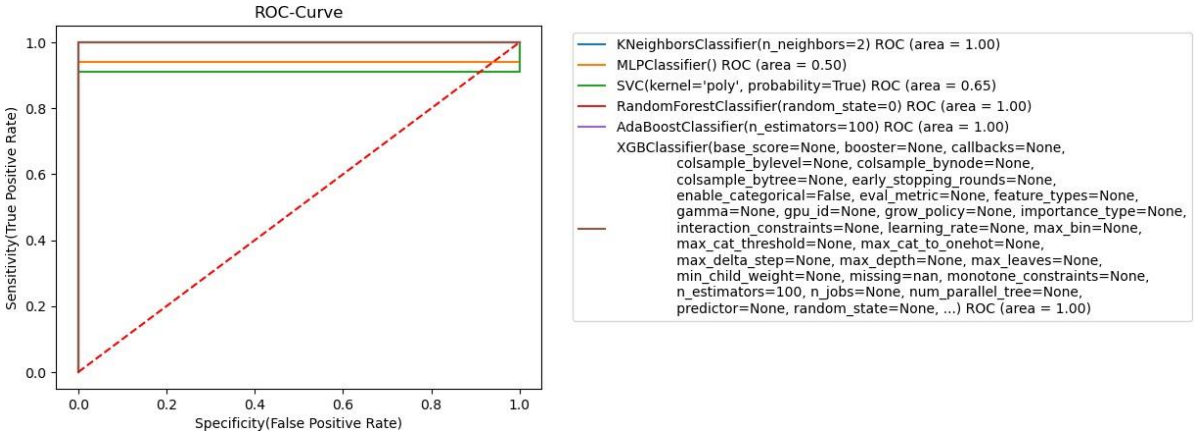
XGBClassifier(base_score=None, booster=None, callbacks=None,
colsample_bylevel=None, colsample_bynode=None,
colsample_bytree=None, early_stopping_rounds=None,
enable_categorical=False, eval_metric=None,
feature_types=None,
gamma=None, gpu_id=None, grow_policy=None,
importance_type=None,
interaction_constraints=None, learning_rate=None,
max_bin=None,
max_cat_threshold=None, max_cat_to_onehot=None,
max_delta_step=None, max_depth=None, max_leaves=None,
min_child_weight=None, missing=nan,
monotone_constraints=None,
n_estimators=100, n_jobs=None, num_parallel_tree=None,predictor=None,
random_state=None, ...)

CONFUSION
MATRIX[[98 0]
[0 34]] CLASSIFICATION

REPORT

	precision	recall	f1-score	support
0	1.00	1.00	1.00	98
1	1.00	1.00	1.00	34
accuracy			1.00	132
macro avg	1.00	1.00	1.00	132
weighted avg	1.00	1.00	1.00	132

In [24]:
```
for model in models.Classifier:
    model.fit(X_train2, y_train2) # train the model
    y_pred=model.predict(X_test2) # predict the test data
# Compute False postive rate, and True positive rate
    fpr, tpr, thresholds = metrics.roc_curve(y_test2, model.predict_proba(X_test2)[:,1])
# Calculate Area under the curve to display on the plot
    auc = metrics.roc_auc_score(y_test2,model.predict(X_test2))
# Now, plot the computed values
    plt.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % (model, auc))
# Custom settings for the plot
plt.plot([0, 1], [0, 1],'r--')
#plt.xlim([0.0, 1.0])
#plt.ylim([0.0, 1.05])
plt.xlabel('Specificity(False Positive Rate)')
plt.ylabel('Sensitivity(True Positive Rate)')
plt.title('ROC-Curve')
plt.legend(bbox_to_anchor = (1.05, 1), loc = 2)
plt.show()   # Display
```



Now Using PCA

For Exp_1 DataSet

In [25]:
```
from sklearn.decomposition import PCA

pca = PCA(n_components = 2)
pca.fit(X1)
X1_PCA = pca.transform(X1)
X1_PCA = pd.DataFrame(X1_PCA, columns = ['Feature_1', 'Feature_2'])
X1_PCA.head()
```

Out[25]:

	Feature_1	Feature_2
0	-0.001265	-0.007240
1	-0.047608	0.002488
2	-0.165532	0.039008
3	0.002476	-0.009847
4	0.005327	-0.009459

In [26]:
```
X1_PCA.shape, y1.shape
```

Out[26]:
```
((1040, 2), (1040,))
```

In [27]:
```
from sklearn.model_selection import train_test_split
XPCA_train1, XPCA_test1, y_train1, y_test1 = train_test_split(X1_PCA, y1, test_size = 0.2, random_state = 0)
```

```
In [28]: for classifier in models.Classifier:

             classifier.fit(XPCA_train1,y_train1)
             pred = classifier.predict(XPCA_test1)

             print(classifier)
             print('\n')
             print('CONFUSION MATRIX')
             print(confusion_matrix(y_test1,pred))
             print('\nCLASSIFICATION REPORT')
             print(classification_report(y_test1,pred))
```

KNeighborsClassifier(n_neighbors=2)

CONFUSION MATRIX[[143 15]
[44 6]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.76	0.91	0.83	158
1	0.29	0.12	0.17	50
accuracy			0.72	208
macro avg	0.53	0.51	0.50	208
weighted avg	0.65	0.72	0.67	208

MLPClassifier()

CONFUSION MATRIX

[[158 0]
[50 0]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.76	1.00	0.86	158
1	0.00	0.00	0.00	50
accuracy			0.76	208
macro avg	0.38	0.50	0.43	208
weighted avg	0.58	0.76	0.66	208

SVC(kernel='poly', probability=True)

CONFUSION MATRIX
[[157 1]
[50 0]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.76	0.99	0.86	158
1	0.00	0.00	0.00	50

accuracy			0.75	208
macro avg	0.38	0.50	0.43	208
weighted avg	0.58	0.75	0.65	208

RandomForestClassifier(random_state=0)

CONFUSION MATRIX[[131 27]
[42 8]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.76	0.83	0.79	158
1	0.23	0.16	0.19	50
accuracy			0.67	208
macro avg	0.49	0.49	0.49	208
weighted avg	0.63	0.67	0.65	208

AdaBoostClassifier(n_estimators=100)

CONFUSION MATRIX
[[151 7]
[46 4]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.77	0.96	0.85	158
1	0.36	0.08	0.13	50
accuracy			0.75	208
macro avg	0.57	0.52	0.49	208
weighted avg	0.67	0.75	0.68	208

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None,
feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None,
importance_type=None,
              interaction_constraints=None, learning_rate=None,
max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan,
monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,predictor=None,
              random_state=None, ...)
```

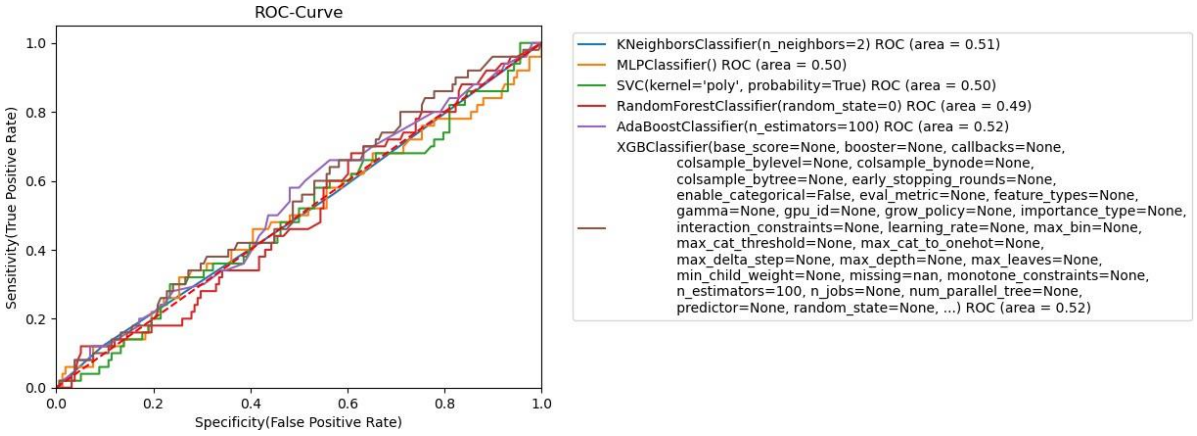
CONFUSION MATRIX[[125 33]

[38 12]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.77	0.79	0.78	158
1	0.27	0.24	0.25	50
accuracy			0.66	208
macro avg	0.52	0.52	0.52	208
weighted avg	0.65	0.66	0.65	208

```
In [29]: from sklearn import metrics
for model in models.Classifier:
    model.fit(XPCA_train1, y_train1) # train the model
    y_pred=model.predict(XPCA_test1) # predict the test data
# Compute False postive rate, and True positive rate
    fpr, tpr, thresholds = metrics.roc_curve(y_test1, model.predict_proba(XPCA_test1)[:,1])
# Calculate Area under the curve to display on the plot
    auc = metrics.roc_auc_score(y_test1,model.predict(XPCA_test1))
# Now, plot the computed values
    plt.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % (model, auc))
# Custom settings for the plot
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Specificity(False Positive Rate)')
plt.ylabel('Sensitivity(True Positive Rate)')
plt.title('ROC-Curve')
plt.legend(bbox_to_anchor = (1.05, 1), loc = 2)
plt.show()   # Display
```



For Exp_2 DataSet

In [30]:
```
pca.fit(X2)
X2_PCA = pca.transform(X2)
X2_PCA = pd.DataFrame(X2_PCA, columns = ['Feature_1', 'Feature_2'])
X2_PCA.head()
```

Out[30]:

	Feature_1	Feature_2
0	0.029888	0.002475
1	0.077907	-0.024651
2	0.112228	-0.042671
3	-0.032063	-0.003627
4	-0.045120	-0.038436

In [31]:
```
XPCA_train2, XPCA_test2, y_train2, y_test2 = train_test_split(X2_PCA, y2, test_size = 0.2, random_state = 0)
```

In [32]:
```
for classifier in models.Classifier:

    classifier.fit(XPCA_train2,y_train2)
    pred = classifier.predict(XPCA_test2)

    print(classifier)
    print('\n')
    print('CONFUSION MATRIX')
    print(confusion_matrix(y_test2,pred))
    print('\nCLASSIFICATION REPORT')
    print(classification_report(y_test2,pred))
```

KNeighborsClassifier(n_neighbors=2)

CONFUSION MATRIX[[98 0]
[0 34]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	1.00	1.00	1.00	98
1	1.00	1.00	1.00	34
accuracy			1.00	132
macro avg	1.00	1.00	1.00	132
weighted avg	1.00	1.00	1.00	132

MLPClassifier()

CONFUSION MATRIX[[98 0]
[33 1]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.75	1.00	0.86	98
1	1.00	0.03	0.06	34
accuracy			0.75	132
macro avg	0.87	0.51	0.46	132
weighted avg	0.81	0.75	0.65	132

SVC(kernel='poly', probability=True)

CONFUSION
MATRIX[[98 0]
 [14 20]] CLASSIFICATION

REPORT

	precision	recall	f1-score	support
0	0.88	1.00	0.93	98
1	1.00	0.59	0.74	34
accuracy			0.89	132
macro avg	0.94	0.79	0.84	132
weighted avg	0.91	0.89	0.88	132

RandomForestClassifier(random_state=0)

CONFUSION MATRIX
[[98 0]
 [0 34]] CLASSIFICATION

REPORT

	precision	recall	f1-score	support
0	1.00	1.00	1.00	98
1	1.00	1.00	1.00	34
accuracy			1.00	132
macro avg	1.00	1.00	1.00	132
weighted avg	1.00	1.00	1.00	132

AdaBoostClassifier(n_estimators=100)

CONFUSION MATRIX
[[98 0]
 [0 34]] CLASSIFICATION

REPORT

	precision	recall	f1-score	support
0	1.00	1.00	1.00	98
1	1.00	1.00	1.00	34
accuracy			1.00	132
macro avg	1.00	1.00	1.00	132
weighted avg	1.00	1.00	1.00	132

XGBClassifier(base_score=None, booster=None, callbacks=None,
colsample_bylevel=None, colsample_bynode=None,
colsample_bytree=None, early_stopping_rounds=None,
enable_categorical=False, eval_metric=None,
feature_types=None,
gamma=None, gpu_id=None, grow_policy=None,
importance_type=None,
interaction_constraints=None, learning_rate=None,
max_bin=None,

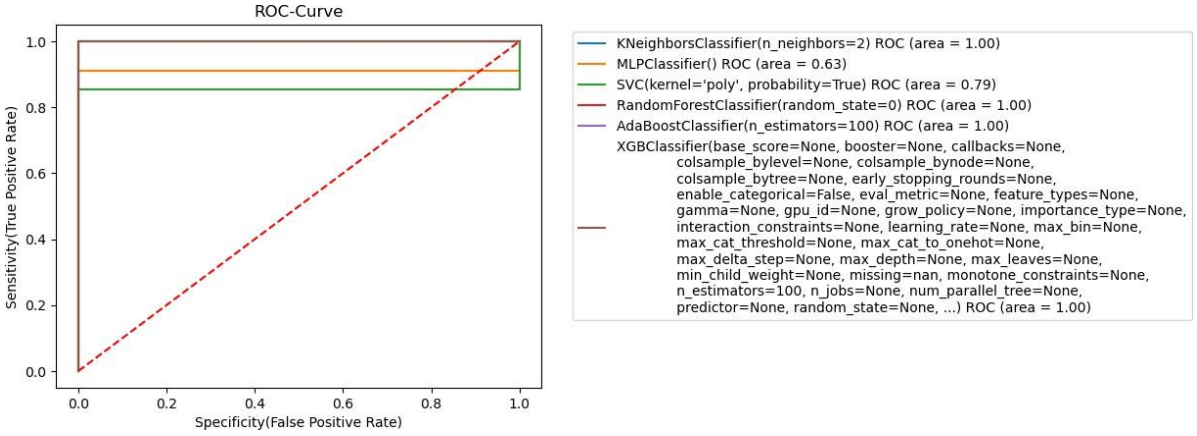
max_cat_threshold=None, max_cat_to_onehot=None,
max_delta_step=None, max_depth=None, max_leaves=None,
min_child_weight=None, missing=nan,
monotone_constraints=None,
n_estimators=100, n_jobs=None, num_parallel_tree=None,predictor=None,
random_state=None, ...)

CONFUSION
MATRIX[[98 0]
[0 34]] CLASSIFICATION

REPORT

	precision	recall	f1-score	support
0	1.00	1.00	1.00	98
1	1.00	1.00	1.00	34
accuracy			1.00	132
macro avg	1.00	1.00	1.00	132
weighted avg	1.00	1.00	1.00	132

```
In [33]: for model in models.Classifier:
             model.fit(XPCA_train2, y_train2) # train the model
             y_pred=model.predict(XPCA_test2) # predict the test data
         # Compute False postive rate, and True positive rate
             fpr, tpr, thresholds = metrics.roc_curve(y_test2, model.predict_proba(XPCA_test2)[:,1])
         # Calculate Area under the curve to display on the plot
             auc = metrics.roc_auc_score(y_test2,model.predict(XPCA_test2))
         # Now, plot the computed values
             plt.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % (model, auc))
         # Custom settings for the plot
         plt.plot([0, 1], [0, 1],'r--')
         #plt.xlim([0.0, 1.0])
         #plt.ylim([0.0, 1.05])
         plt.xlabel('Specificity(False Positive Rate)')
         plt.ylabel('Sensitivity(True Positive Rate)')
         plt.title('ROC-Curve')
         plt.legend(bbox_to_anchor = (1.05, 1), loc = 2)
         plt.show()   # Display
```



Q2.Implement two dataset Exp3 and Exp4 using KNN, BPNN, Kernel SVM, Random Forest, Ada boost Random Forest, Ada boost SVM, XG boost with and without PCA. Then find out its accuracy, confusion matrix and ROC curve.

SOURCE CODE:-

```
In [34]: data3 = pd.read_csv('Exp_3.csv')
         data4 = pd.read_csv('Exp_4.csv')

In [35]: X3 = data3.iloc[: , :-1]
         y3 = data3.iloc[: , -1]

         X4 = data4.iloc[: , :-1]
         y4 = data4.iloc[: , -1]

In [36]: y3 = label_encoder.fit_transform(y3)
         y3
Out[36]: array([1, 1, 1, ..., 0, 0, 0], dtype=int64)

In [37]: y4 = label_encoder.fit_transform(y4)

In [38]: from sklearn.preprocessing import Normalizer
         norm = Normalizer()
         columns3 = X3.columns
         columns4 = X4.columns
         X3 = norm.fit_transform(X3)
         X3 = pd.DataFrame(X3, columns = columns3)
         X4 = norm.fit_transform(X4)
         X4 = pd.DataFrame(X4, columns = columns4)

In [39]: X3, X4
```

Out[39]: (

	F1	F2	F3	F4	F5	F6	F7
0	0.000077	0.279858	0.000081	0.275569	0.000005	0.163158	0.000708
1	0.000047	0.257421	0.000043	0.260888	0.000004	0.143974	0.000449
2	0.000014	0.175988	0.000013	0.179417	0.000002	0.094923	0.000104
3	0.000091	0.279901	0.000090	0.278686	0.000005	0.154604	0.000967
4	0.000070	0.236117	0.000062	0.238359	0.000004	0.137780	0.000675
...	...	...	...	...	...	...	...
1015	0.000059	0.294545	0.000058	0.306874	0.000006	0.179523	0.000468
1016	0.000067	0.274129	0.000060	0.278582	0.000005	0.166746	0.000587
1017	0.000033	0.278288	0.000037	0.290640	0.000005	0.176258	0.000237
1018	0.000045	0.232080	0.000055	0.254176	0.000003	0.134147	0.000452
1019	0.000027	0.270228	0.000036	0.283793	0.000005	0.171788	0.000184

	F8	F9	F10	F11	F12	F13	F14
0	0.385331	0.000763	0.381087	0.000058	0.285942	0.665954	0.012411
1	0.365224	0.000457	0.370836	0.000032	0.253199	0.714079	0.004152
2	0.286618	0.000110	0.293055	0.000009	0.181810	0.852491	0.000000
3	0.386891	0.001036	0.388871	0.000066	0.275544	0.665557	0.017002
4	0.330216	0.000676	0.332392	0.000063	0.247662	0.766541	0.001066

```
1015  0.410887  0.000611  0.431315  0.000054  0.307120  0.580584  0.024105
1016  0.381756  0.000567  0.387581  0.000055  0.290544  0.662622  0.000000
1017  0.396699  0.000301  0.418698  0.000034  0.299863  0.620016  0.001293
1018  0.332405  0.000655  0.364276  0.000031  0.235047  0.751694  0.007532
1019  0.392802  0.000271  0.405182  0.000031  0.294815  0.641518  0.000000

[1020 rows x 14 columns],
          F1        F2        F3        F4        F5        F6        F7  \
0    0.000139  0.300399  0.000304  0.340019  0.000035  0.219361  0.000920
1    0.000136  0.286328  0.000317  0.354291  0.000031  0.209050  0.000870
2    0.000075  0.271109  0.000210  0.326915  0.000018  0.187492  0.000532
3    0.000156  0.313088  0.000313  0.349554  0.000042  0.231459  0.000880
4    0.000139  0.302087  0.000307  0.362791  0.000038  0.221890  0.000774
..        ...       ...       ...       ...       ...       ...       ...
645  0.000106  0.284841  0.000301  0.353247  0.000025  0.200072  0.000724
646  0.000106  0.284841  0.000301  0.353247  0.000025  0.200072  0.000724
647  0.000106  0.284841  0.000301  0.353247  0.000025  0.200072  0.000724
648  0.000106  0.284841  0.000301  0.353247  0.000025  0.200072  0.000724
649  0.000106  0.284841  0.000301  0.353247  0.000025  0.200072  0.000724

          F8        F9       F10       F11       F12       F13       F14
0    0.412152  0.002189  0.435811  0.000407  0.348067  0.396312  0.328646
1    0.390156  0.002643  0.467089  0.000365  0.329014  0.393115  0.340000
2    0.372274  0.001833  0.439804  0.000183  0.311544  0.503738  0.318813
3    0.418328  0.001997  0.443766  0.000450  0.358238  0.346119  0.325807
4    0.409635  0.002175  0.471938  0.000379  0.346702  0.331393  0.327620
..        ...       ...       ...       ...       ...       ...       ...
645  0.393132  0.002584  0.467301  0.000291  0.320495  0.401636  0.342191
646  0.393132  0.002584  0.467301  0.000291  0.320495  0.401636  0.342191
647  0.393132  0.002584  0.467301  0.000291  0.320495  0.401636  0.342191
648  0.393132  0.002584  0.467301  0.000291  0.320495  0.401636  0.342191
649  0.393132  0.002584  0.467301  0.000291  0.320495  0.401636  0.342191

[650 rows x 14 columns])
```

In [40]:
```
X_train3, X_test3, y_train3, y_test3 = train_test_split(X3, y3, test_size = 0.2, random_state = 0)
X_train4, X_test4, y_train4, y_test4 = train_test_split(X4, y4, test_size = 0.2, random_state = 0)
```

For Exp_3

In [41]:
```
for classifier in models.Classifier:

    classifier.fit(X_train3,y_train3)
    pred = classifier.predict(X_test3)

    print(classifier)
    print('\n')
    print('CONFUSION MATRIX')
    print(confusion_matrix(y_test3,pred))
    print('\nCLASSIFICATION REPORT')
    print(classification_report(y_test3,pred))
```

KNeighborsClassifier(n_neighbors=2)

CONFUSION MATRIX[[135 19]
[49 1]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.73	0.88	0.80	154
1	0.05	0.02	0.03	50
accuracy			0.67	204
macro avg	0.39	0.45	0.41	204

weighted avg	0.57	0.67	0.61	204

MLPClassifier()

CONFUSION MATRIX[[154 0]
[50 0]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.75	1.00	0.86	154
1	0.00	0.00	0.00	50
accuracy			0.75	204
macro avg	0.38	0.50	0.43	204
weighted avg	0.57	0.75	0.65	204

SVC(kernel='poly', probability=True)

CONFUSION MATRIX
[[154 0]
[50 0]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.75	1.00	0.86	154
1	0.00	0.00	0.00	50
accuracy			0.75	204
macro avg	0.38	0.50	0.43	204
weighted avg	0.57	0.75	0.65	204

RandomForestClassifier(random_state=0)

CONFUSION MATRIX
[[124 30]
[44 6]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.74	0.81	0.77	154
1	0.17	0.12	0.14	50
accuracy			0.64	204
macro avg	0.45	0.46	0.45	204
weighted avg	0.60	0.64	0.62	204

AdaBoostClassifier(n_estimators=100)

CONFUSION MATRIX
[[138 16]

[49 1]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.74	0.90	0.81	154
1	0.06	0.02	0.03	50
accuracy			0.68	204
macro avg	0.40	0.46	0.42	204
weighted avg	0.57	0.68	0.62	204

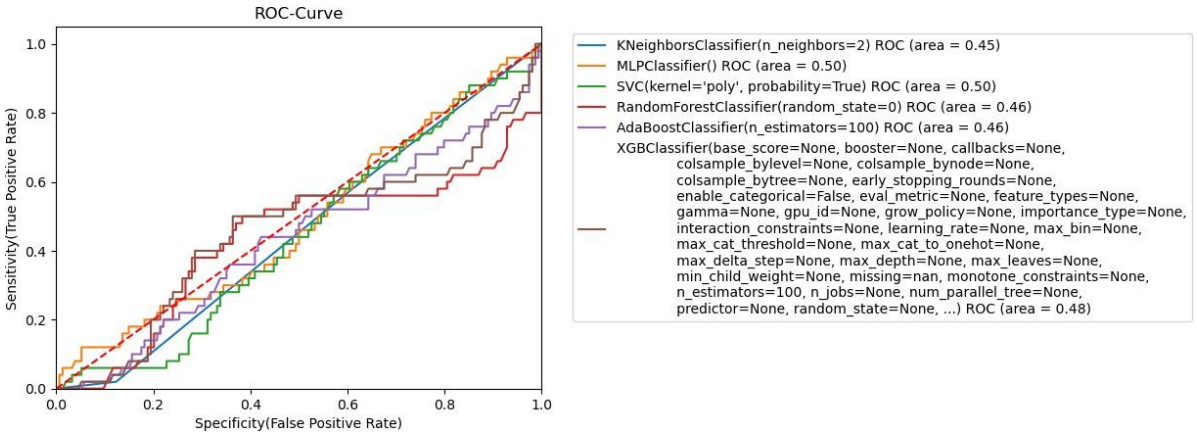
XGBClassifier(base_score=None, booster=None, callbacks=None,
colsample_bylevel=None, colsample_bynode=None,
colsample_bytree=None, early_stopping_rounds=None,
enable_categorical=False, eval_metric=None,
feature_types=None,
gamma=None, gpu_id=None, grow_policy=None,
importance_type=None,
interaction_constraints=None, learning_rate=None,
max_bin=None,
max_cat_threshold=None, max_cat_to_onehot=None,
max_delta_step=None, max_depth=None, max_leaves=None,
min_child_weight=None, missing=nan,
monotone_constraints=None,
n_estimators=100, n_jobs=None, num_parallel_tree=None,predictor=None,
random_state=None, ...)

CONFUSION MATRIX[[124 30]
[42 8]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.75	0.81	0.77	154
1	0.21	0.16	0.18	50
accuracy			0.65	204
macro avg	0.48	0.48	0.48	204
weighted avg	0.62	0.65	0.63	204

In [42]:
```
from sklearn import metrics
for model in models.Classifier:
    model.fit(X_train3, y_train3) # train the model
    y_pred=model.predict(X_test3) # predict the test data
# Compute False postive rate, and True positive rate
    fpr, tpr, thresholds = metrics.roc_curve(y_test3, model.predict_proba(X_test3)[:,1])
# Calculate Area under the curve to display on the plot
    auc = metrics.roc_auc_score(y_test3,model.predict(X_test3))
# Now, plot the computed values
    plt.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % (model, auc))
# Custom settings for the plot
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Specificity(False Positive Rate)')
plt.ylabel('Sensitivity(True Positive Rate)')
plt.title('ROC-Curve')
plt.legend(bbox_to_anchor = (1.05, 1), loc = 2)
plt.show()   # Display
```



For Exp_4

```
In [43]: for classifier in models.Classifier:

             classifier.fit(X_train4,y_train4)
             pred = classifier.predict(X_test4)

             print(classifier)
             print('\n')
             print('CONFUSION MATRIX')
             print(confusion_matrix(y_test4,pred))
             print('\nCLASSIFICATION REPORT')
             print(classification_report(y_test4,pred))
```

KNeighborsClassifier(n_neighbors=2)

CONFUSION MATRIX[[99 0]
[0 31]] CLASSIFICATION

REPORT

	precision	recall	f1-score	support
0	1.00	1.00	1.00	99
1	1.00	1.00	1.00	31
accuracy			1.00	130
macro avg	1.00	1.00	1.00	130
weighted avg	1.00	1.00	1.00	130

MLPClassifier()

CONFUSION MATRIX
[[99 0]
[31 0]] CLASSIFICATION

REPORT

	precision	recall	f1-score	support
0	0.76	1.00	0.86	99
1	0.00	0.00	0.00	31
accuracy			0.76	130

macro avg	0.38	0.50	0.43	130
weighted avg	0.58	0.76	0.66	130

SVC(kernel='poly', probability=True)

CONFUSION
MATRIX[[99 0]
[7 24]] CLASSIFICATION

REPORT

	precision	recall	f1-score	support
0	0.93	1.00	0.97	99
1	1.00	0.77	0.87	31
accuracy			0.95	130
macro avg	0.97	0.89	0.92	130
weighted avg	0.95	0.95	0.94	130

RandomForestClassifier(random_state=0)

CONFUSION MATRIX
[[99 0]
[0 31]] CLASSIFICATION

REPORT

	precision	recall	f1-score	support
0	1.00	1.00	1.00	99
1	1.00	1.00	1.00	31
accuracy			1.00	130
macro avg	1.00	1.00	1.00	130
weighted avg	1.00	1.00	1.00	130

AdaBoostClassifier(n_estimators=100)

CONFUSION MATRIX
[[99 0]
[0 31]] CLASSIFICATION

REPORT

	precision	recall	f1-score	support
0	1.00	1.00	1.00	99
1	1.00	1.00	1.00	31
accuracy			1.00	130
macro avg	1.00	1.00	1.00	130
weighted avg	1.00	1.00	1.00	130

XGBClassifier(base_score=None, booster=None, callbacks=None,
colsample_bylevel=None, colsample_bynode=None,
colsample_bytree=None, early_stopping_rounds=None,

enable_categorical=False, eval_metric=None,
feature_types=None,
gamma=None, gpu_id=None, grow_policy=None,
importance_type=None,
interaction_constraints=None, learning_rate=None,
max_bin=None,
max_cat_threshold=None, max_cat_to_onehot=None,
max_delta_step=None, max_depth=None, max_leaves=None,
min_child_weight=None, missing=nan,
monotone_constraints=None,
n_estimators=100, n_jobs=None, num_parallel_tree=None,predictor=None,
random_state=None, ...)

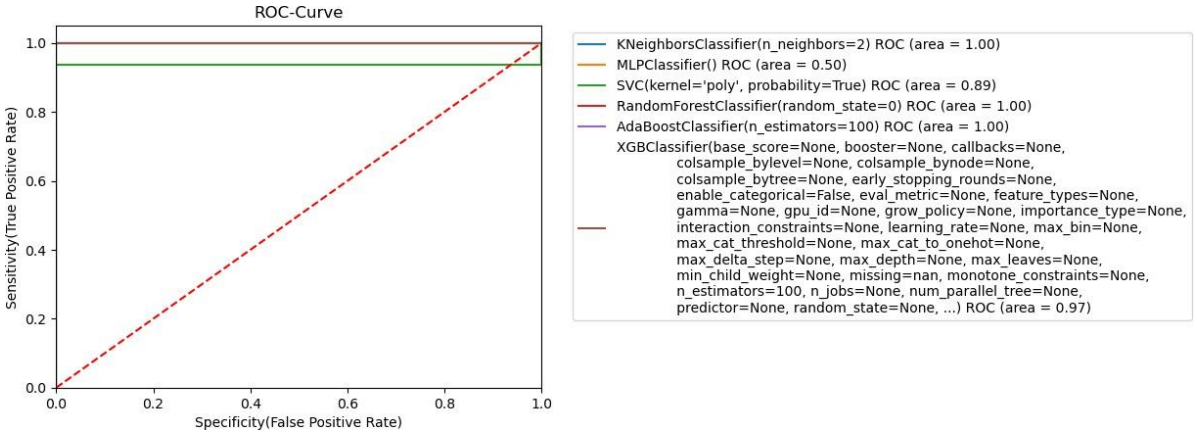
CONFUSION
MATRIX[[99 0]
[2 29]] CLASSIFICATION

REPORT

	precision	recall	f1-score	support
0	0.98	1.00	0.99	99
1	1.00	0.94	0.97	31
accuracy			0.98	130
macro avg	0.99	0.97	0.98	130
weighted avg	0.98	0.98	0.98	130

In [44]:
```
from sklearn import metrics
for model in models.Classifier:
    model.fit(X_train4, y_train4) # train the model
    y_pred=model.predict(X_test4) # predict the test data
# Compute False postive rate, and True positive rate
    fpr, tpr, thresholds = metrics.roc_curve(y_test4, model.predict_proba(X_test4)[:,1])
# Calculate Area under the curve to display on the plot
    auc = metrics.roc_auc_score(y_test4,model.predict(X_test4))
# Now, plot the computed values
    plt.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % (model, auc))
# Custom settings for the plot
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Specificity(False Positive Rate)')
plt.ylabel('Sensitivity(True Positive Rate)')
plt.title('ROC-Curve')
plt.legend(bbox_to_anchor = (1.05, 1), loc = 2)
plt.show()   # Display
```

In [44]:
```
from sklearn import metrics
for model in models.Classifier:
    model.fit(X_train4, y_train4) # train the model
    y_pred=model.predict(X_test4) # predict the test data
# Compute False postive rate, and True positive rate
    fpr, tpr, thresholds = metrics.roc_curve(y_test4, model.predict_proba(X_test4)[:,1])
# Calculate Area under the curve to display on the plot
    auc = metrics.roc_auc_score(y_test4,model.predict(X_test4))
# Now, plot the computed values
    plt.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % (model, auc))
# Custom settings for the plot
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Specificity(False Positive Rate)')
plt.ylabel('Sensitivity(True Positive Rate)')
plt.title('ROC-Curve')
plt.legend(bbox_to_anchor = (1.05, 1), loc = 2)
plt.show()   # Display
```



With PCA

In [45]:
```
pca.fit(X3)
X3_PCA = pca.transform(X3)
X3_PCA = pd.DataFrame(X3_PCA, columns = ['Feature_1', 'Feature_2'])


pca.fit(X4)
X4_PCA = pca.transform(X4)
X4_PCA = pd.DataFrame(X4_PCA, columns = ['Feature_1', 'Feature_2'])
X4_PCA.head()
```

Out[45]:

	Feature_1	Feature_2
0	-0.019326	0.037740
1	-0.002780	-0.005679
2	0.108383	0.048229
3	-0.071095	0.018291
4	-0.070151	-0.020981

In [46]:
```
XPCA_train3, XPCA_test3, y_train3, y_test3 = train_test_split(X3_PCA, y3, test_size = 0.2, random_state = 0)

XPCA_train4, XPCA_test4, y_train4, y_test4 = train_test_split(X4_PCA, y4, test_size = 0.2, random_state = 0)
```

In [47]:
```
for classifier in models.Classifier:

    classifier.fit(XPCA_train3,y_train3)
    pred = classifier.predict(XPCA_test3)

    print(classifier)
    print('\n')
    print('CONFUSION MATRIX')
    print(confusion_matrix(y_test3,pred))
    print('\nCLASSIFICATION REPORT')
    print(classification_report(y_test3,pred))
```

KNeighborsClassifier(n_neighbors=2)

CONFUSION MATRIX[[138 16]
[50 0]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.73	0.90	0.81	154
1	0.00	0.00	0.00	50
accuracy			0.68	204
macro avg	0.37	0.45	0.40	204
weighted avg	0.55	0.68	0.61	204

MLPClassifier()

CONFUSION MATRIX[[154 0]
[50 0]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.75	1.00	0.86	154
1	0.00	0.00	0.00	50
accuracy			0.75	204
macro avg	0.38	0.50	0.43	204
weighted avg	0.57	0.75	0.65	204

SVC(kernel='poly', probability=True)

CONFUSION MATRIX
[[154 0]
[50 0]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.75	1.00	0.86	154
1	0.00	0.00	0.00	50
accuracy			0.75	204
macro avg	0.38	0.50	0.43	204
weighted avg	0.57	0.75	0.65	204

RandomForestClassifier(random_state=0)

CONFUSION MATRIX
[[115 39]
[44 6]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.72	0.75	0.73	154
1	0.13	0.12	0.13	50
accuracy			0.59	204
macro avg	0.43	0.43	0.43	204
weighted avg	0.58	0.59	0.59	204

AdaBoostClassifier(n_estimators=100)

CONFUSION MATRIX
[[151 3]
[50 0]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.75	0.98	0.85	154
1	0.00	0.00	0.00	50
accuracy			0.74	204
macro avg	0.38	0.49	0.43	204
weighted avg	0.57	0.74	0.64	204

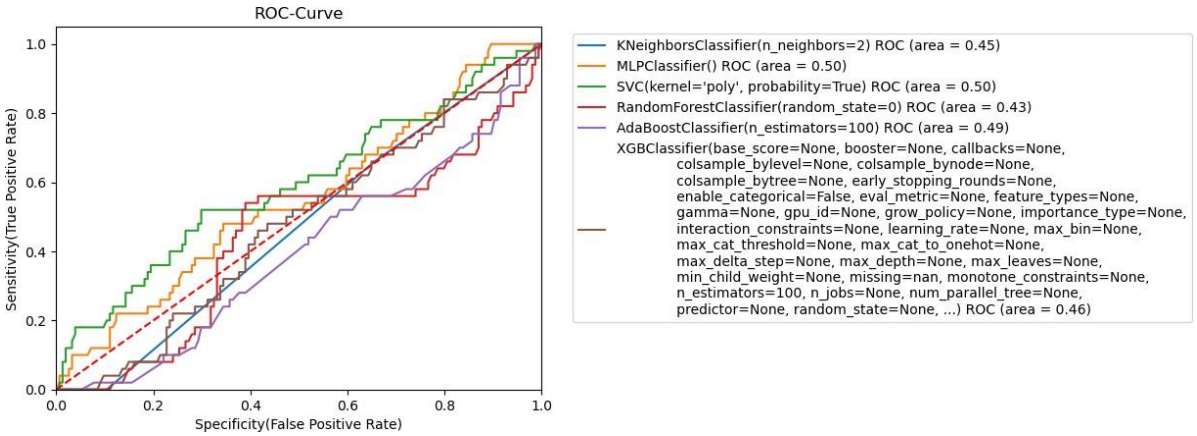
XGBClassifier(base_score=None, booster=None, callbacks=None,
colsample_bylevel=None, colsample_bynode=None, colsample_bytree=None,
early_stopping_rounds=None,enable_categorical=False, eval_metric=None,
feature_types=None,
gamma=None, gpu_id=None, grow_policy=None,
importance_type=None,
interaction_constraints=None, learning_rate=None,
max_bin=None,
max_cat_threshold=None, max_cat_to_onehot=None,
max_delta_step=None, max_depth=None, max_leaves=None,
min_child_weight=None, missing=nan,
monotone_constraints=None,
n_estimators=100, n_jobs=None, num_parallel_tree=None,predictor=None,
random_state=None, ...)

CONFUSION MATRIX[[119 35]
[43 7]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.73	0.77	0.75	154
1	0.17	0.14	0.15	50
accuracy			0.62	204
macro avg	0.45	0.46	0.45	204
weighted avg	0.60	0.62	0.61	204

In [48]:
```
from sklearn import metrics
for model in models.Classifier:
    model.fit(XPCA_train3, y_train3) # train the model
    y_pred=model.predict(XPCA_test3) # predict the test data
# Compute False postive rate, and True positive rate
    fpr, tpr, thresholds = metrics.roc_curve(y_test3, model.predict_proba(XPCA_test3)[:,1])
# Calculate Area under the curve to display on the plot
    auc = metrics.roc_auc_score(y_test3,model.predict(XPCA_test3))
# Now, plot the computed values
    plt.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % (model, auc))
# Custom settings for the plot
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Specificity(False Positive Rate)')
plt.ylabel('Sensitivity(True Positive Rate)')
plt.title('ROC-Curve')
plt.legend(bbox_to_anchor = (1.05, 1), loc = 2)
plt.show()   # Display
```



In [49]:
```
for classifier in models.Classifier:

    classifier.fit(XPCA_train4,y_train4)
    pred = classifier.predict(XPCA_test4)

    print(classifier)
    print('\n')
    print('CONFUSION MATRIX')
    print(confusion_matrix(y_test4,pred))
    print('\nCLASSIFICATION REPORT')
    print(classification_report(y_test4,pred))
```

KNeighborsClassifier(n_neighbors=2)

CONFUSION
MATRIX[[99 0]
 [0 31]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	1.00	1.00	1.00	99
1	1.00	1.00	1.00	31

accuracy			1.00	130
macro avg	1.00	1.00	1.00	130
weighted avg	1.00	1.00	1.00	130

MLPClassifier()

CONFUSION MATRIX
[[99 0]
[24 7]]

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.80	1.00	0.89	99
1	1.00	0.23	0.37	31
accuracy			0.82	130
macro avg	0.90	0.61	0.63	130
weighted avg	0.85	0.82	0.77	130

SVC(kernel='poly', probability=True)

CONFUSION
MATRIX[[99 0]
[14 17]]

CLASSIFICATION

	precision	recall	f1-score	support
REPORT				
0	0.88	1.00	0.93	99
1	1.00	0.55	0.71	31
accuracy			0.89	130
macro avg	0.94	0.77	0.82	130
weighted avg	0.91	0.89	0.88	130

RandomForestClassifier(random_state=0)

CONFUSION MATRIX
[[99 0]
[0 31]] CLASSIFICATION

REPORT

	precision	recall	f1-score	support
0	1.00	1.00	1.00	99
1	1.00	1.00	1.00	31
accuracy			1.00	130
macro avg	1.00	1.00	1.00	130
weighted avg	1.00	1.00	1.00	130

AdaBoostClassifier(n_estimators=100)

CONFUSION
MATRIX[[99 0]
 [1 30]] CLASSIFICATION

REPORT

	precision	recall	f1-score	support
0	0.99	1.00	0.99	99
1	1.00	0.97	0.98	31
accuracy			0.99	130
macro avg	0.99	0.98	0.99	130
weighted avg	0.99	0.99	0.99	130

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None,
feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None,
importance_type=None,
              interaction_constraints=None, learning_rate=None,
max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan,
monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,predictor=None,
              random_state=None, ...)
```

CONFUSION
MATRIX[[99 0]
 [1 30]] CLASSIFICATION

REPORT

	precision	recall	f1-score	support
0	0.99	1.00	0.99	99
1	1.00	0.97	0.98	31
accuracy			0.99	130
macro avg	0.99	0.98	0.99	130
weighted avg	0.99	0.99	0.99	130

In [50]:
```
from sklearn import metrics
for model in models.Classifier:
    model.fit(XPCA_train4, y_train4) # train the model
    y_pred=model.predict(XPCA_test4) # predict the test data
# Compute False postive rate, and True positive rate
    fpr, tpr, thresholds = metrics.roc_curve(y_test4, model.predict_proba(XPCA_test4)[:,1])
# Calculate Area under the curve to display on the plot
    auc = metrics.roc_auc_score(y_test4,model.predict(XPCA_test4))
# Now, plot the computed values
    plt.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % (model, auc))
# Custom settings for the plot
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Specificity(False Positive Rate)')
plt.ylabel('Sensitivity(True Positive Rate)')
plt.title('ROC-Curve')
plt.legend(bbox_to_anchor = (1.05, 1), loc = 2)
plt.show()   # Display
```