Feature_scaling

November 3, 2020

Coding Assignment 8

```
Feature Scaling
Dataset: Pima Indians Diabetes Database
Author: Sreejith S
```

Dataset: https://www.kaggle.com/uciml/pima-indians-diabetes-database

[11]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

%reload_ext autoreload
%matplotlib inline
%autoreload 2
%config InlineBackend.figure_format = 'retina'


#classifiers
from sklearn import svm
from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import train_test_split

#evaluation metrics
from sklearn.metrics import plot_confusion_matrix, confusion_matrix,␣
 ↪ConfusionMatrixDisplay
from sklearn.metrics import plot_roc_curve, roc_curve, roc_auc_score
from sklearn.metrics import f1_score
from sklearn.metrics import plot_precision_recall_curve, precision_recall_curve,␣
 ↪precision_score, recall_score
from sklearn.metrics import matthews_corrcoef, average_precision_score


#plotting utils
from utils import plot_table, plot_confusion

from sklearn.preprocessing import MinMaxScaler
```

```
from sklearn.preprocessing import StandardScaler
```

[29]:
```
# Importing the dataset
dataset = pd.read_csv('../datasets/pima-indians-diabetes.csv')

#selecting features
features = ['age', 'pregnant', 'plasma_glucose']
y = dataset[['Diab']].values.ravel()
X = dataset[features]
```

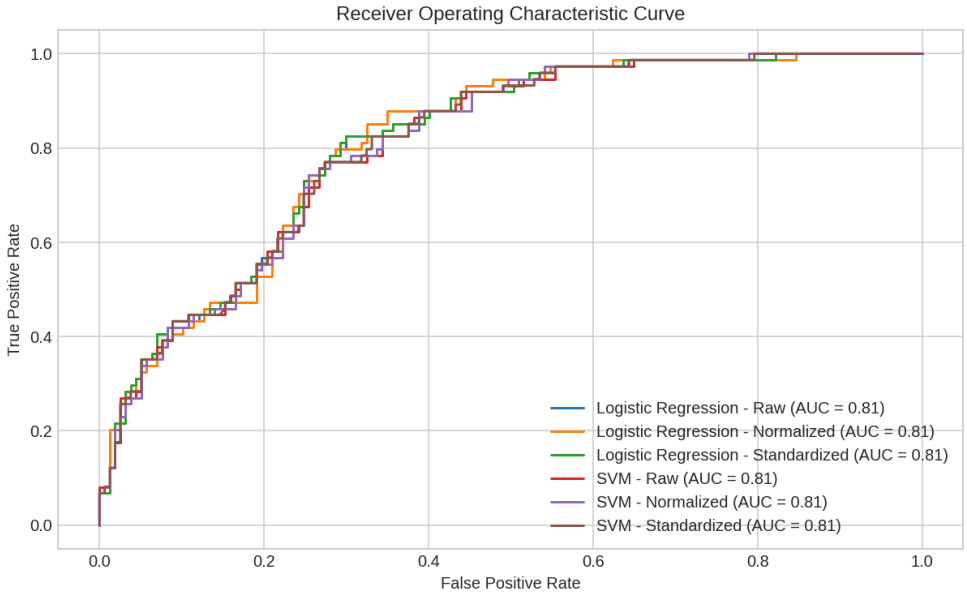
[25]:
```
def test_model(classifiers, scalers, short_titles, titles):
    """
    """

    confusion_matrices = []
    eval_metrics_list = []


    fig = plt.figure(figsize=(10, 20))
    plt.style.use("seaborn-whitegrid")
    ax_roc = fig.add_subplot(3, 1, 1)
    ax_roc.set_title("Receiver Operating Characteristic Curve")
    ax_pre = fig.add_subplot(3, 1, 2)
    ax_pre.set_title("Precision-Recall Curve")

    ax_table = fig.add_subplot(3, 1, 3)
    ax_table.set_title("Evaluation Metrices")


    for i, (clf, scaler) in enumerate(zip(classifiers, scalers)):
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                            random_state=0)

        #Scaling
        if scaler:
            s = scaler().fit(X_train)
            #Scale Train and Test seperately
            #scaled_train =  (train - train_mean) / train_std_deviation
            #scaled_test = (test - train_mean) / train_std_deviation
            X_train = pd.DataFrame(s.transform(X_train), columns=X_train.columns)
            X_test = pd.DataFrame(s.transform(X_test), columns=X_test.columns)

        clf.fit(X_train, y_train)
        y_preds = clf.predict(X_test)
        y_proba = clf.predict_proba(X_test)[:, 1]

        plot_roc_curve(clf, X_test, y_test, ax=ax_roc, name=titles[i])
```

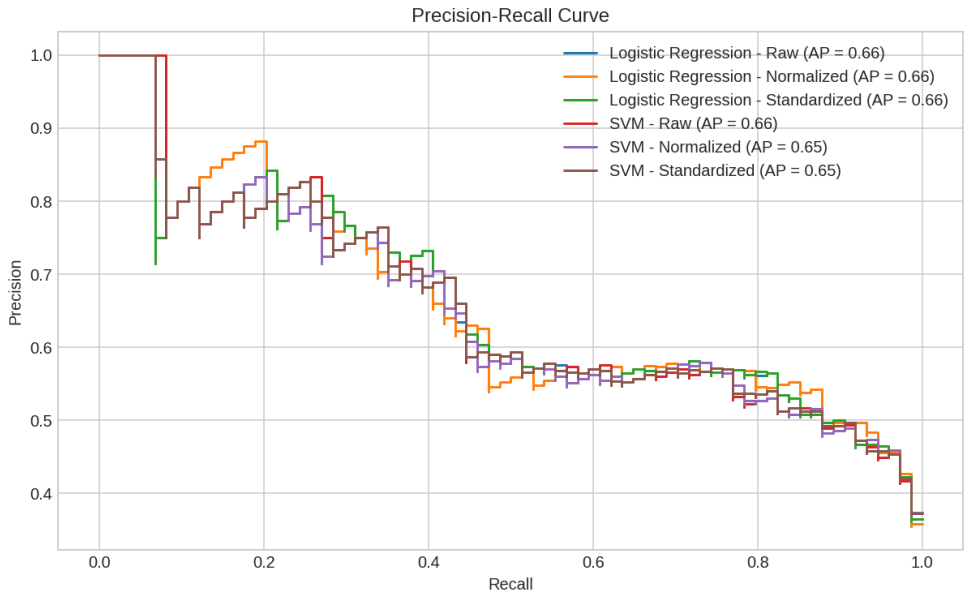
```
        plot_precision_recall_curve(clf, X_test, y_test, ax=ax_pre,␣
↪name=titles[i])

        confusion_matrices.append(confusion_matrix(y_test, y_preds))
        eval_metrics = [clf.score(X_test, y_test),
                        precision_score(y_test, y_preds, zero_division=0),
                        recall_score(y_test, y_preds),
                        f1_score(y_test, y_preds),
                        matthews_corrcoef(y_test, y_preds),
                        roc_auc_score(y_test, y_proba),
                        average_precision_score(y_test, y_proba)]
        eval_metrics = [f"{i:.2f}" for i in eval_metrics]
        eval_metrics_list.append(eval_metrics)

    ax_pre.legend(loc='upper right')
    plot_table(eval_metrics_list, ax_table, short_titles,
               ["Score", "Precison", "Recall", "F1 Score", "MCC", "AUROC", "Avg.␣
↪Precision"])
    plot_confusion(confusion_matrices, short_titles)
    plt.show()
```

8.1. Build logistic regression and SVM models with raw data, normalized data, and standardized data. Use only age, plasma_glucose, pregnancies as the three features for building these models. Compare their performances and report them below (using plots and tables). Please also mention your inferences regarding feature scaling and the reasons. Your primary goal is to present the steps, results and inferences in such a way that the reviewer is convinced whether or not to use feature scaling in this dataset

[31]:
```
short_titles = ["LR-Raw", "LR-Norm", "LR-Std",
                "SVM-Raw", "SVM-Norm", "SVM-Std"]
titles = ["Logistic Regression - Raw",
          "Logistic Regression - Normalized",
          "Logistic Regression - Standardized",
          "SVM - Raw",
          "SVM - Normalized",
          "SVM - Standardized"]
scalers = [None, MinMaxScaler, StandardScaler,
           None, MinMaxScaler, StandardScaler]

classifiers = [ LogisticRegression(max_iter=100),
                LogisticRegression(max_iter=100),
                LogisticRegression(max_iter=100),
                svm.SVC(kernel='linear', probability=True),
                svm.SVC(kernel='linear', probability=True),
                svm.SVC(kernel='linear', probability=True)
              ]
```

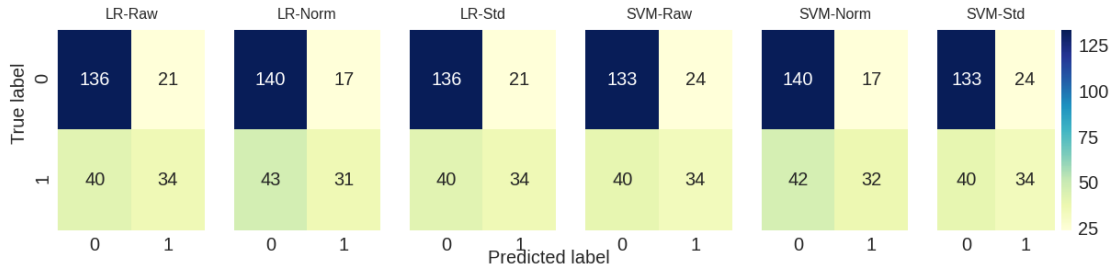
```
test_model(classifiers, scalers, short_titles, titles)
```





Evaluation Metrices

	Score	Precison	Recall	F1 Score	MCC	AUROC	Avg. Precision
LR-Raw	0.74	0.62	0.46	0.53	0.36	0.81	0.66
LR-Norm	0.74	0.65	0.42	0.51	0.36	0.81	0.66
LR-Std	0.74	0.62	0.46	0.53	0.36	0.81	0.66
SVM-Raw	0.72	0.59	0.46	0.52	0.33	0.81	0.66
SVM-Norm	0.74	0.65	0.43	0.52	0.37	0.81	0.65
SVM-Std	0.72	0.59	0.46	0.52	0.33	0.81	0.65



Observation

In this particular case, there isn't any significant change in efficiency by scaling the dataset.

Scaling can improve performance of the algorithm and enable faster convergence to an optimal minima, especially if there is a huge difference in scale between features.

source:https://medium.com/@947_34258/how-you-can-optimize-your-machine-learning-algorithms-d1c5ec9277bd

[ ]: