Week 2.1
Using Python Tools for Data Cleaning and Preprocessing

Produced By: Abdullah Abdul Majid*(linkedin)*

Completion Date: July 8, 2024

Email: abdullahpisk@gmail.com

Jupyter Notebook Links:

GitHub

Google Drive

Table of Contents:

Problem Statement:

Implement and evaluate a logistic regression model for binary classification.
Dataset Used: Heart Disease Dataset on UCI

Purpose:

The purpose of this assignment is to inspect and preprocess the dataset. Then use the tools from the scikit-learn library to train a model and test it's performance.

Process:

Dataset Loading and Initial Inspection

- Imported all the necessary libraries, i.e pandas, matplotlib and sklearn.
- Downloaded the Heart Disease dataset from UCI.
- Loaded the dataset into a pandas dataframe (clv) using *read_csv* on the processed.cleveland.data file.
- Assigned column names according to the UCI website.
- Verified the dataset loading using print.

Inspecting and Preprocessing the Dataset

- Inspected the dataset usin *info* and *describe*.

- No null values were found but there was some missing data (represented by a '?') and not all the columns ('ca' and 'thal') were numerical.
- Used Boolean indexing on the problematic columns ('ca' and 'thal') to delete the rows that included the missing values as estimating them might mess up our model.
- Checked for *duplicated* rows, none were found.
- Visualized the data using *plot (line)* to check how far the values were from each other and evaluate whether to standardize the dataframe or not.
- The lines were all over the place and required scaling.
- Used sklearn.preprocessing's built in *StandardScaler* (which utilizes Z-scaling) to scale all the independent variable columns (i.e all columns except 'num' which is the Heart Disease diagnosis status dependent variable).
- Used *plot (line)* to again check the dataframe and this time all the lines were reasonably close together.

Building and Training the Model

- Identified dependent (y) and independent (x) variables.
- Split the dataset into test and train sets with a 20-80 split respectively and a random state of 42.
- Used the *OneVsRestClassifier* from sklearn to build the model with the *liblinear* solver since our dependent variable, 'num' is a multiclass variable.
- Next up, we train the model by fitting in the train values using *fit.*

Model Testing and Evaluation

- We *predict* the dependent 'y' variable using the independent 'x' test values to evaluate the model.
- Now, we compare the lengths of the predicted and test data and verify that they are equal.
- Next, we calculate the *accuracy*, *precision* (*weighted*) and *recall* (*weighted*) *scores*, and print the *confusion_matrix* which are as follows:

```
Accuracy: 0.6333333333333333
Precision: 0.5616666666666666
Recall: 0.6333333333333333
Confusion Matrix:
[[35  1  0  0  0]
 [ 5  1  0  2  1]
 [ 2  1  1  1  0]
 [ 1  2  1  1  2]
 [ 2  0  0  1  0]]
```

The results are not great but acceptable.

- Verifying the test and train scores also shows good results:

```
#test score
model.score(X_test, y_test)
```

```
0.6333333333333333
```

```
#train score
model.score(X_train, y_train)
```

```
0.6708860759493671
```

```
#the test and train scores are similar
```

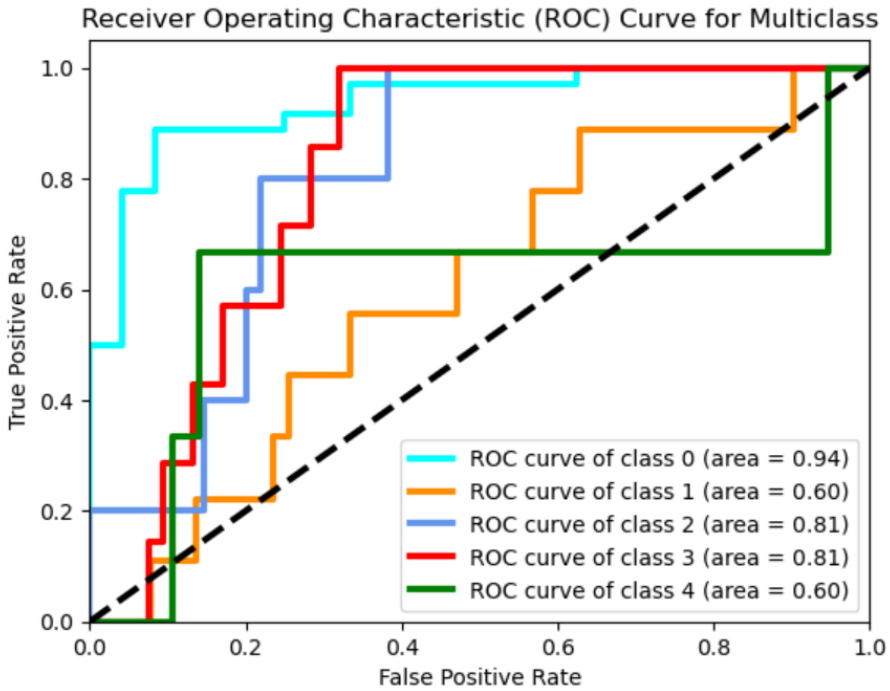
Plotting the ROC Curve

- Predicted the probability of the model.
- Used *dict()* to define *fpr, tpr* and *roc_auc* (Area Under Curve).
- Used *shape* to identify the number of classes (The dependent variable has 4 possible outcomes, and hence is multiclassed).
- Then used a *for* loop to calculate the above ROC values for every class.
- And finally plotted the ROC curve for each class as follows:

```
# Plot the ROC curve for each class
plt.figure()
colors = ['aqua', 'darkorange', 'cornflowerblue', 'red', 'green']

for i in range(n_classes):
    plt.plot(fpr[i], tpr[i], color=colors[i], lw=3,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=3)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve for Multiclass')
plt.legend(loc="lower right")
```



Conclusion:

I was able to preprocess and prepare a dataset for Logistic Regression model training and building using sklearn from scikit. The model's performance was also evaluated and was proved to be satisfactory. An ROC-Curve of the model has also been plotted.