DAV 6150 Module 5 Assignment

Understanding Classification Model Performance Metrics

***** You may work in small groups of no more than three (3) people for this Assignment *****

Evaluation of the performance of classification models can be facilitated through a combination of calculating certain types of performance metrics and generating model performance evaluation graphics. For this assignment you will be tasked with calculating a suite of classification model performance metrics via Python code functions that you create and then comparing your results to those of pre-built Python functions that automatically calculate those same metrics. You will also be creating graphical output via Python code that can be used to evaluate the performance of classification models.

The data set you will be using for this work is comprised of approximately 180 labeled binary observations (i.e., the classification is binary in nature). The classifications are either '0' or '1'. A classification model has been used to "predict" the actual classifications for each observation, and its algorithm works by estimating the probability that the correct classification for each observation is a '1'. The data set includes three attributes that you will make use of for your analysis:

1) **class**: the actual classification for the observation
2) **scored**.**class**: the predicted classification for the observation (can only be '0' or '1'; identified by comparing the classification model's probability score (i.e., the likelihood that the correct classification for an observation is a '1') for the observation against a 0.50 threshold))
3) **scored**.**probability**: the classification model's probability score (i.e., the likelihood that the correct classification for an observation is a '1') for the observation

Complete each of the following tasks as specified:

1) Load the provided M5_Data.csv file to your DAV 6150 Github Repository.
2) Then, using a Jupyter Notebook, read the data set from your Github repository and load it into a Pandas dataframe.
3) Use Pandas' **crosstab()** function to calculate the contents of a confusion matrix for the data. Make sure you closely examine the output e.g., do the rows represent the actual or the predicted classification? What about the columns? (HINT: A good way to ensure you understand the output of the **crosstab()** function is to check the **value_counts()** for the dataframe columns you've used as input to the **crosstab()** function. When observing the **value_counts()**, determine whether the rows or columns in the **crosstab()** output sum to the respective **value_counts()** figures).
4) Extract the individual confusion matrix values (i.e., True Positive, False Positive, True Negative, False Negative) from the output of the **crosstab()** function and save them for later use (e.g., save them each to individual variables or to the data structure of your choice). Knowing how to properly extract these values from the output of **crosstab()** will serve you well throughout the remainder of the assignment
5) Write a Python function that accepts as input the actual and predicted classifications for any binary classification data and then calculates and returns the **accuracy** metric for the predictions without utilizing any pre-built Python accuracy metric calculation functions.

6) Write a Python function that accepts as input the actual and predicted classifications for any binary classification data and then calculates and returns the **precision** metric for the predictions without utilizing any pre-built Python classification metric calculation functions.

7) Write a Python function that accepts as input the actual and predicted classifications for any binary classification data and then calculates and returns the **sensitivity** metric for the predictions without utilizing any pre-built Python classification metric calculation functions.

8) Write a Python function that accepts as input the actual and predicted classifications for any binary classification data and then calculates and returns the **specificity** metric for the predictions without utilizing any pre-built Python classification metric calculation functions.

9) Write a Python function that accepts as input the actual and predicted classifications for any binary classification data and then calculates and returns the **F1 Score** metric for the predictions without utilizing any pre-built Python classification metric calculation functions.

10) Write a Python function to **plot a ROC curve and also calculate AUC** for any binary classification data that contains both actual and predicted classifications as well as the associated scored probabilities without utilizing any pre-built Python classification metric calculation or ROC or AUC functions.

11) Apply the Python functions you've created to the relevant columns of the provided data set to produce the classification metrics specified in steps 5 through 9 of this assignment.

12) Now compare the output of your functions against the output of the pre-built functions available within the **scikit-learn** library. Specifically, apply the **confusion_matrix(), accuracy_score(), precision_score(), recall_score()** (remember: recall = sensitivity), **f1_score()**, and the **metrics.classification_report()** functions to the relevant columns of the provided data set. Discuss how well the results of these pre-built functions compare to those of the functions you've created for this assignment.

13) Using the **metrics.plot_roc_curve ()** and **metrics.auc**() functions from the scikit-learn package to generate a ROC plot and calculate AUC for the provided data set. How do the results compare with the ROC/AUC function you've created for this assignment?

Your deliverable for this assignment is your Jupyter Notebook. It should contain a combination of Python code cells and explanatory narratives contained within properly formatted Markdown cells.

Your Jupyter Notebook deliverable should be similar to that of a publication-quality / professional caliber document and should include clearly labeled graphics, high-quality formatting, clearly defined section and sub-section headers, and be free of spelling and grammar errors. Furthermore, your Pythion code should include succinct explanatory comments.

Upload your Jupyter Notebook within the provided M5 Assignment Canvas submission portal. Be sure to save your Notebook using the following nomenclature: **first initial_last name_M5_assn**" (e.g., J_Smith_M5_assn_). ***Small groups should identity all group members at the start of the Jupyter Notebook and each team member should submit their own copy of the team's work within Canvas.***

***Grading rubric**:* you'll receive **up to 90 points** if you successfully complete **tasks 1-9, 11,** and **12. To get the full 100 points** on the assignment, you will also need to successfully complete **tasks 10** and **13.**