CSCI 611 - LAB#5 SPECIFICATIONS

Version 2 (last updated 3-23-21)

OBJECTIVE:

- Familiarize with building and analyzing **Decision Trees, Random Forests, and Gradient Boosting** Classifiers
- Explore the ***Kyphosis*** dataset and classify cases where Kyphosis is *absent* or *present* after surgery
- Visualize *Kyphosis* dataset with **barplots** and **pairplots** using Matplotlib and/or Seaborn
- Assess impact of modifying multiple hyper-parameters
- Visualize model as a decision tree (using *graphviz*), and graph *feature importances*

PREPARATION & REFERENCES:

- Complete IMLP (Muller/Guido book) **Ch.2.35-2.36 working through .ipynb lecture notebook**.
- Reading csv files into dataframes and applying df methods
- Explore visualization tools for bar plots, pairplots, and trees
- What is *Kyphosis*? A deformity of the backbone (spine), when vertebrae in the upper back curve outward more than they should. A child with kyphosis has a back that is abnormally rounded or humpback.

TASKS:

GET KYPHOSIS DATA and GATHER INFORMATION

- Load the Kyphosis data from the .csv file provided (BBLearn, week 8) and read into a pandas Dataframe
- Explore the data using Pandas dataframe methods to display general information, statistical analysis, check for missing values, display some of the rows of values, get value counts for the two classifications (*absent, present*), etc.

VISUALIZE the data using MATPLOTLIB and/or SEABORN

- Create a labeled bar graph/histogram, to display the count of each classification, side by side.
- Create a labeled pairwise plot, to display classification (kyphosis) correlations between the numeric features

SPLIT the DATA[1]

- Reserve the *Kyphosis* attribute as your target label
- Use a 75/25% *train/test* split for **all** models, random_state=40
- Confirm your data types, shapes, columns, etc.

CREATE an EXEMPLARY DECISION TREE CLASSIFIER[1]

- Assess accuracy: display predictions, confusion matrix, classification report, and accuracy *score*
- Improve model by adjusting appropriate hyperparameters, seeking the best generalized model
  - Depth, min samples, max nodes, max features, random state, splitter, etc.

---

[1] For the sake of REPRODUCABILITY, please use the same 75/25% *train/test* split for **all** models, random_state=40

- Display model as a TREE, using sklearn.tree *graphviz*
- Display model *feature importances* as a bar graph

CREATE an EXEMPLARY RANDOM FOREST CLASSIFIER[1]

- Assess accuracy: display predictions, confusion matrix, classification report, and accuracy *score*
- Improve model by adjusting appropriate hyperparameters, seeking the best generalized model
  - Number of estimators, depth, min samples, max nodes, max features, etc.
- Display model *feature importances* as a bar graph

CREATE an EXEMPLARY GRADIENT BOOSTING CLASSIFIER[1]

- Assess accuracy: display predictions, confusion matrix, classification report, and accuracy *score*
- Improve model by adjusting appropriate hyperparameters, seeking the best generalized model
  - Learning Rate, number of estimators, depth, min samples, max nodes, max features, etc.
- Display model *feature importances* as a bar graph

CONCLUSIONS – Summarize your findings, in a final TEXT cell within your notebook. As part of your summary, consider your six visualizations. Mention at least one piece of useful information you were able to glean from each.

DELIVERABLES

- ***CLEAN Notebook:*** Before you save & print
  - Remove unnecessary code cells (e.g., troubleshooting, excessive testing, etc.)
  - Leave requested functionality & demonstrations, as well as exemplary models
  - Add documentation; expand code, comment, and output cells; close help windows
- ***xy***Lab5.ipynb - Python notebook, where ***xy*** are your initials; fully documented, including a general header & comments
- ***xy***Lab5.pdf – which SHOWS ALL code, documentation, and outputs generated from a full run of matching .ipynb file