Matt Saperstein
DE300 Homework 3

Spark (due Feb 24 at 11:59pm; delivery by github)

General Instructions:

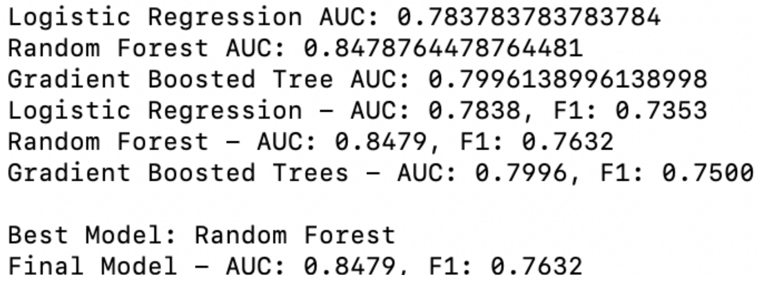
- You must perform all the data manipulation, filtering, and cleaning steps as well as training the models in Spark. You are NOT allowed to use any other ML libraries (e.g. scikit-learn or Pandas API in Python).
- Your submission must use the AWS EMR service for Spark.

In this module, you will implement a few cleaning steps from homework 2 in Spark on the original dataset and then train the models (using Spark ML API) on this transformed dataset to serve as a baseline.

1. Create an S3 bucket containing the original data. In the subsequent steps, you will be loading the data from S3.
2. Implement the cleaning steps from homework 2 in Spark.

Tasks 3-5 deal with training a heart disease prediction model on this data. It is desired that the model has low bias. Interpretability of the model is a bonus.

1. Split the data into training and test using a 90-10 training test split with stratification on the labels (i.e., both sets contain roughly the same proportion of positive labels).
    a. When doing this, I also removed the original smoke column from the dataframe because it still contains missing values and is no longer necessary since I created two new columns with updated smoke values.
2. Train a few binary classification models on the training data. If there are hyperparameters, tune them appropriately. Assess the performance using 5-fold cross-validation and report relevant performance metrics.
    a. I used Logistic Regression, Random Forest, and Gradient-Boosted Trees.
    b. Here are the best parameters, cross-validation scores, and final results:
        i.



```
Logistic Regression AUC: 0.783783783783784
Random Forest AUC: 0.8478764478764481
Gradient Boosted Tree AUC: 0.7996138996138998
Logistic Regression - AUC: 0.7838, F1: 0.7353
Random Forest - AUC: 0.8479, F1: 0.7632
Gradient Boosted Trees - AUC: 0.7996, F1: 0.7500

Best Model: Random Forest
Final Model - AUC: 0.8479, F1: 0.7632
```

3. Select your final model based on relevant criteria.
    a. The final chosen model based on relevant criteria is Random Forest because of the 3 models, it had the highest ROC-AUC score (0.8479). This means the model has a strong ability to differentiate between the two classes (heart disease or no heart disease). It also had the highest F1 score of the 3 models, meaning it is the closest to perfect precision and recall out of the 3 models.

Recurrent footnote: Use AWS resources sparingly; you have a budget. Test and debug locally and only make final runs on AWS.

Cleaning steps:

1. Retain only the following columns (apart from target):
   age, sex, painloc, painexer, cp, trestbps, smoke, Us, prop, nitr, pro, diuretic, thaldur, thalach, exang, oldpeak, slope
   a. Note: I kept the target column in my dataframe.
2. Cleaning and imputing steps for columns other than `smoke:'
   a. painloc, painexer
      i. I imputed missing values using the mode because they both contain binary data.
   b. trestbps: Replace values less than 100 mm Hg
      i. I imputed values less than 100 mm Hg using 100 mm Hg to create a lower bound.
   c. oldpeak: Replace values less than 0 and those greater than 4
      i. I imputed oldpeak values less than 0 with 0 and those greater than 4 with 4.
   d. thaldur, thalach: Replace the missing values
      i. I imputed thaldur and thalach using the mean because the two values are both pretty random.
   e. Us, prop, nitr, pro, diuretic: Replace the missing values and values greater than 1
      i. I imputed these missing values using the mode for each column and clipped values greater than 1 to 1.
   f. exang, slope: Replace the missing values
      i. I imputed missing exang values with the mode because exang data is binary.
      ii. I imputed slope values using the mode because there are only 4 possible values for slope.
   g. After completing these steps, I realized that some additional columns still had missing values that needed to be filled in to be able to use binary classification models on the training data. I did the following imputations:
      i. I imputed the missing age values with the mode.
      ii. I imputed the missing sex values with the mode.
      iii. I imputed the missing cp values with the mode.
      iv. I imputed the missing trestbps values with the mode.
      v. I imputed the missing oldpeak values with the mean because there were numerous possible values for oldpeak.
      vi. I imputed the missing target values with the mode.
         1. After imputing these missing values, the only column that still had missing values in it was the smoke column which I knew I would work on imputing the missing values later on.

h. The last step I took was I only kept the first 899 rows in the dataset because the rows that follow are not in the correct format. For example, some values have multiple numbers in them with no specified relation or order. In order to work with this dataset, I knew I needed to remove these rows.

3. Cleaning and imputing the smoke column:
You will impute the missing values in the “smoke” column with smoking rates by age or sex. You will use two different sources for obtaining these smoking rates. Create a separate column for each source. After imputing the missing values, apply an appropriate transform for each column.

a. Source 1: https://www.abs.gov.au/statistics/health/health-conditions-and-risks/smoking/latest-release
This source lists smoking rates (current daily smokers) by age group. Replace the missing values with the smoking rate in the corresponding age groups.

i. I scraped this website to grab the smoking data by age group from the 2011-12 column. Then, I created a dictionary with all of the smoking rates by age group in them. Next, I created a new column called smoke_source_1 to be an exact copy of the current smoke column so I could impute all missing values in a new column. I created a function to map age to the specified age group from the web scraped data. I then imputed all missing values using the web scraped data.

b. Source 2:
https://www.cdc.gov/tobacco/php/data-statistics/adult-data-cigarettes/index.html

c. This source lists smoking rates by age group and by sex.

i. For female patients, replace the missing values with the smoking rate in their corresponding age groups.

ii. For male patients, replace the missing values with *smoking rate in age group* × *smoking rate among men* / *smoking rate among women*

iii. You must write a python code to scrape this data (you are not allowed to manually download this data).

1. I found the specific male, female, and age group smoking rates on the following pdf:
2. https://www.cdc.gov/tobacco/media/pdfs/2024/09/cdc-osh-ncis-data-report-508.pdf
3. I utilized the tobacco product smoke rates by sex and by age that are shown on page 4. I hard-coded these values into my Python script to use in later calculations. Next, I created a new column called smoke_source_2 to be an exact copy of the current smoke column so I could impute all missing values in a new column. Finally, I imputed all missing values in the smoke_source_2 column using the specified formula for female and male patients respectively.

Sources: Used ChatGPT help setting up Docker Docker, AWS S3, imputing values using Pandas, web scraping, and using binary classification models.