Age Group Classification using an NHANES subset.

**Introduction:**

I wanted to introduce myself to working with health data, and thus decided to work with this simple subset of data that is used to predict which age group that a respondent to the NHANES survey belongs to. I undertook this project to work with this data because I was interested in the dataset. I wanted to use this data to see if my models could predict that adults exercise more than Seniors. If they do predict these accurately, then they could be used to apply to bigger and more valuable health data. So far, I have had some problems with the code, but I have fixed it, and it runs, not smoothly, but it runs.

**Data:**

NHANES stands for National Health and Nutrition Examination Survey. This was a survey administered by the CDC to collect extensive health and nutritional information from the US population. Of course, that dataset is too large, so this dataset was narrowed down to the point where it would use a subset of features from the larger NHANES dataset to predict age group. This dataset would classify respondents into 2 age groups: Adults (Younger than 65) and Seniors (65+). The following table is a description of the attributes/features of the dataset:

|  |  |  |
| --- | --- | --- |
| Attribute Name | Type | Description |
| SEQN | Continous | Respondent’s ID/SEquence Number |
| RIAGENDR | Continous | Gender of Respondent |
| age\_group | Categorical | Age Group (Senior - 65+/Non-Senior) |
| RIDAGEYR | Continous | Respondent’s Age |
| PAQ605 | Continous | If the respondent engages in moderate-vigorous physical activity in a normal week. |
| BMXBMI | Continous | Respondent’s BMI |
| LBXGLU | Continous | Blood Glucose after fasting. |
| DIQ010 | Continous | If Respondent is Diabetic? |
| LBXGLT | Continous | Respondent’s Oral Health |
| LBXIN | Continous | Blood Insulin Levels |

Table 1 – Description of input Attributes/Features

Only the age\_group variable is represented by strings like “Adult” and “Senior”. The RIAGENDR variable is represented by 1 (Male) or 2 (Female) for each respondent. PAQ605 is also represented by 1 (does takes part in weekly moderate or vigorous-intensity physical activity) or a 2 (does NOT takes part in weekly moderate or vigorous-intensity physical activity). However, those are only a few things that we need to worry about with regards to this data. Most of the respondents should be classified as under 65+, so based on that, we can estimate the count to be mostly adults, with a proportion of 0.8 being adults, and 0.2 being seniors.

**Methods:**

To access this data subset, we import it into Python using the ucimlrepo package, and then fetch the specific dataset and import X and y as separate Pandas dataframes. After that, we had to split the data into training data to train the models on and test data to test the models against. We do this by using the train\_test\_split function, to split everything into training and test sets.

I first trained my data against a simple Logistic Regression model, as it works best for linearly separable data. I then trained a KNN model with a k=5 fold. After training these models, I took the cross-validation scores of both of these models to determine which model would be better suited to the data and averaged them to find the average score for each model, and found that the Linear Regression model did better. My initial linear regression model used the training data and trained it on 200 iterations to avoid errors and better adjust to the data. I used a k=5 KNN model as I thought it would be better at this kind of classification problem, since it is used to adjusting centroids iteratively.

Since the Linear Regression model did have a better average cross-validated score than the KNN with k=5 model, I decided to use Linear Regression as the main model going forward. I first trained a new Linear Regression model using the training set over 200 iterations. I then evaluated this same model with the test set to get a linear regression score that was similar to the average cross-validated score of the initial linear regression model.

Once I had evaluated the model, I had to see which predictions were wrong, and which were right. To do this, I just did a simple for loop and an if statement inside that would compare the predicted class of a respondent with their actual class. However, I needed to extract that data from the test set. So, I used the .iloc function, which allows you to select specific rows or columns in a Pandas Dataframe by providing normal integer indices. I did this to extract the relevant data. However, this resulted in a valueError() whenever the if statement would execute. I forgot to check whether the .iloc function was extracting the whole row, or just the class. It turns out that it was extracting the whole record, with all the attributes like the respondent’s identification number, their age group (which is what we want) and the object type of the record. So, I needed to extract even further, so I just took the .iloc of the original .iloc to get the actual class data, or string. Once I had that string, I used it to compare classes and added the indices where the predicted class was wrong to a list named ‘wrong’, and the correct indices to a list named ‘right’.

A computer screen shot of a number

Description automatically generated

Fig. 1 – Comparison loop logic

Of course, this led to an out-of-bounds error and thus made me only compare the first 684 predictions, to avoid the out-of-bounds error. Once I did this, I tried to print out the actual and predicted classes for the first 5 wrong predictions, although they were mostly the same classes that the wrong predictions were wrong for. I then proceeded to print out the input attributes.

**Results:**

When I carried out my results, I received the averaged score, the predictions, the wrong and right indices, as well as the input attributes descriptions. Below is a picture of the output of 1 execution of my program.

A screenshot of a computer

Description automatically generated

Fig.2 – Beginning part of output of 1 execution.

A screen shot of a computer screen

Description automatically generated

Fig.3 – Output of wrong and right predictions via indices.

A screenshot of a computer

Description automatically generated

Fig. 4 – Predicted v. Actual class comparison, and input attribute description.

Obviously, this is a lot. As you can see in Fig.1, linear regression had a higher average score of 0.8356, whereas a KNN model with k=5 had a average score of 0.8067, thus making me choose Linear Regression as the better model. The list of wrong and right indices for our predictions is also shown. The actual and predicted classes are show below. Before, I encountered a logic error where both the actual and predicted classes were the same. I fixed that error when I realized that my comparison loop, (Fig.1) was appending the right indices to the wrong list and the wrong indices to the right list. Both figures are fixed now.

**Discussion:**

Based on these results, I can put much trust into the model shown above, as the average scores are very high, and it does show me where my predictions are wrong and which are right (for only the first 684 entries, but I can iterate through the other entries). I was not able to answer some of the questions I initially posed, but through further work, I could. The large dataset and multiple errors were some limitations I did encounter when working on this project, and my insufficient knowledge of Pandas was vastly improved because of this project. In the future, I may add more code to this project to better answer the questions that I initially posed. I also might use the dataset to answer future questions, like which gender exercises more? To further improve the project in the future, I might start it earlier to use more time, and also figure out a way to avoid an out-of-bounds error so I can compare all the predictions to all of the test set at once, instead of in just increments of 684, or only 684 predictions. I could try to use other models like Artificial Neural Networks, or Convolutional Neural Networks instead of linear regression to better classify data more quickly and accurately, but that is because ANNs are better at making good predictions as the width of the network decrease as it gets to the output layer.

**Conclusion:**

By carrying out this project, I have learned how to properly train and test machine learning models, how to work with data in the form of Pandas Dataframes, and how to filter out incorrect predictions and comparisons. I have learned that machine learning is a lot like the normal science/engineering process, where we create, train, test, then train again if it doesn’t work properly. Choosing a model requires lots of training and testing via cross-validation or other means, and that Python is sort of the best way to implement these kinds of projects, whether by custom packages or by importing local files/datasets to your local computer. Sometimes, when it comes to model choice, the simpler the model, the better.

**References:**

Kartikaybhutani. “Extracting Rows Using Pandas .Iloc[] in Python.” *GeeksforGeeks*, GeeksforGeeks, 4 Dec. 2023, www.geeksforgeeks.org/python-extracting-rows-using-pandas-iloc/.

“Linear Regression Example.” *Scikit*, scikit-learn.org/stable/auto\_examples/linear\_model/plot\_ols.html#sphx-glr-auto-examples-linear-model-plot-ols-py. Accessed 11 Dec. 2023.

“Sklearn.Linear\_model.Linearregression.” *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LinearRegression.html#sklearn.linear\_model.LinearRegression.score. Accessed 11 Dec. 2023.

“Sklearn.Linear\_model.Logisticregression.” *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LogisticRegression.html. Accessed 11 Dec. 2023.

“Sklearn.Neighbors.Kneighborsclassifier.” *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html. Accessed 11 Dec. 2023.

“Sklearn.Model\_selection.Cross\_val\_score.” *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.model\_selection.cross\_val\_score.html#sklearn.model\_selection.cross\_val\_score. Accessed 11 Dec. 2023.